Projet Qawi – Jeu de guerre galactique

Projet Annuel 3A Architecture de Logiciels

Ivan KLARMANQuentin TOUATIOUIWalson RENE

****

2013 - 2014

Table des matières

[1. Présentation 3](#_Toc392710824)

[1.1. Générale 3](#_Toc392710825)

[1.2. Objectif du jeu 3](#_Toc392710826)

[1.3. Description des fonctionnalités 3](#_Toc392710827)

[1.4. Limite du périmètre fonctionnel 4](#_Toc392710828)

[2. Gestion de projet 5](#_Toc392710829)

[2.1 Diagramme de Gantt 5](#_Toc392710830)

[2.2. Versioninng avec Git Hub 6](#_Toc392710831)

[2.3. Prévision des derniers travaux à réaliser 6](#_Toc392710832)

[2.4. Contraintes techniques 6](#_Toc392710833)

[3. Modélisation 7](#_Toc392710834)

[3.1 Interface Human Machine 7](#_Toc392710835)

[3.2.1. Diagramme de package 9](#_Toc392710836)

[3.2.2. Diagramme de cas d’utilisation (Cas Gestion planète) 10](#_Toc392710837)

[3.2.3. Diagramme de cas d'utilisation (Cas Gestion des joueurs) 10](#_Toc392710838)

[3.2.4. Diagramme de classes 11](#_Toc392710839)

[3.2.5. Diagramme de séquence (Cas Améliorer usines) 12](#_Toc392710840)

[3.3. Descriptifs cas d’utilisation 12](#_Toc392710841)

[3.4. Algorithmes Complexes Imaginés 17](#_Toc392710842)

[3.5. MCD 19](#_Toc392710843)

[4. Architecture Logiciel 20](#_Toc392710844)

[4.1. Structure client léger et clients lourds 20](#_Toc392710845)

[4.2. Architecture Logique 20](#_Toc392710846)

[4.3. Architecture de l’application lourde 20](#_Toc392710847)

[5. Conception 22](#_Toc392710848)

[5.1. Descriptifs des API / Framework utilisés 22](#_Toc392710849)

[5.1.1. JDBC 22](#_Toc392710850)

[5.1.2 JavaMail 22](#_Toc392710851)

[5.2. Premiers Programmes 23](#_Toc392710852)

[Envoi de mail 23](file:///C:\Users\Quentin\Documents\GitHub\QAWIProject\DOCUMENTATION_FINAL_GROUP.docx#_Toc392710853)

# 1. Présentation

## 1.1. Générale

|  |  |
| --- | --- |
| **Nom du Projet** | **QAWI** |
| **Logo** |  |

## 1.2. Objectif du jeu

Le jeu consiste à gagner des ressources pour construire des bâtiments et/ou usines. Grâce aux ressources produites en temps réels, les joueurs peuvent construire des vaisseaux pour attaquer une planète ennemie ou encore défendre sa propre planète.

## 1.3. Description des fonctionnalités

Un utilisateur est un joueur ou un administrateur.  
Un joueur est une personne ayant un nom de joueur, une adresse mail et un nombre de points initialisé à 0.  
Un utilisateur s’inscrit via une page d’inscription dédié et saisie son adresse mail, nom de joueur, mot de passe et choisie une Galaxie existante.  
Un utilisateur peut se connecter en sélectionnant une galaxie dans laquelle il a créée son compte.  
  
Un joueur peut modifier son adresse mail et son nom d’utilisateur.  
Un joueur peut voir son classement uniquement dans la galaxie dans laquelle il se trouve.  
  
Un administrateur peut voir le classement des utilisateurs en choisissant une galaxie.  
Un administrateur peut modifier le nom du jouer, sa galaxie et son mot de passe.

Une galaxie possède un identifiant et un nom.  
  
Un vaisseau peut être d’attaque ou de défense.

Un vaisseau possède un nom, un prix en ressources, une valeur défensive et une valeur d’attaque.  
  
Un regroupement de vaisseaux constitue une défense.  
Un regroupement de vaisseaux constitue une attaque.

Une usine contient un nom et un niveau.  
Un type d’usine produit un type de ressource.  
  
Le niveau d’une usine peut être amélioré par un joueur en échange d’un nombre de ressources dans le cas où le joueur n’a pas assez de ressources alors il ne peut pas améliorer l’usine et doit attendre d’avoir assez de ressources.  
  
Une ressource peut être de l’or, de l’argent, de la pierre ou de la nourriture.  
  
Un récapitulatif est un tableau récapitulant les deux dernières activités sur le compte d’un joueur (Amélioration et/ou attaques).  
  
Pour passer d’un niveau usine à un autre, l’on doit multiplier le niveau actuel par la somme du niveau actuel de l’usine plus 3.  
La création d’un vaisseau d’attaque ou de défense coûte des ressources. (Cela dépend des types de vaisseaux).

## 1.4. Limite du périmètre fonctionnel

Le jeu est limité par le fait que l’utilisateur n’a pas beaucoup de fonctionnalité à faire si ce n’est améliorer des usines, créer des vaisseaux et attaquer des joueurs avec une flotte.  
Le joueur est également limité aux temps d’attentes d’amélioration d’une usine.  
Le joueur ne peut pas faire plusieurs à la fois choses comme par exemple créer un vaisseau et en même temps améliorer une usine.

|  |
| --- |
|  |

# 2. Gestion de projet

## 2.1 Diagramme de Gantt

Tâches et sous tâches :

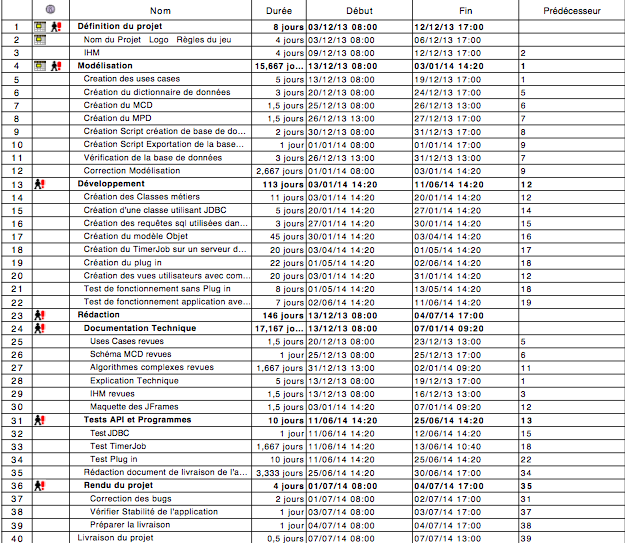
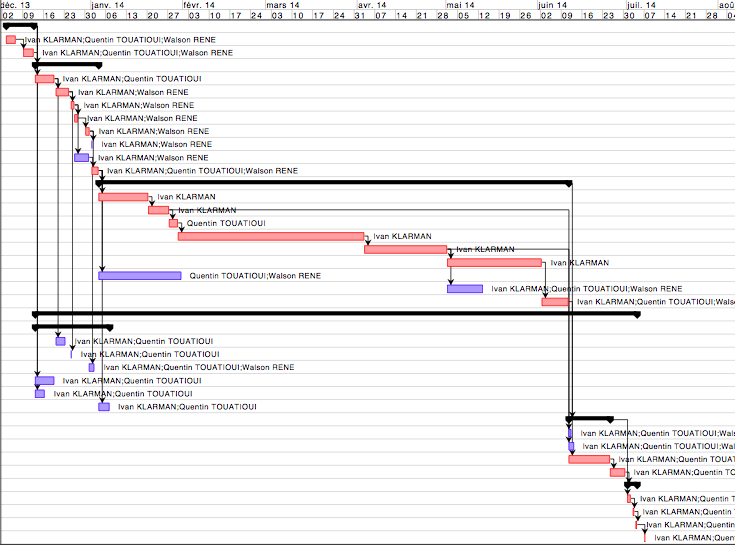


Diagramme de Gantt :



## 2.2. Versioninng avec Git Hub

Nous avons utilisé Git Hub pour la partie versionning, cela nous a permis de travailler sur nos fichiers de manières autonomes, nous avons pu partager également les fichiers mis par les différentes personnes du groupe pour s’assurer d’avoir les dernières versions des fichiers. De plus en terme de management, cela à permis de veiller à l’avancement du projet notamment sur ce qui a été fait.

Au 24/06/2014  **🡪 43 Commits**dont : **Ivan Klarman 🡪** 20 **Commits**

**Quentin Touatioui** 🡪 8 **Commits**

**Walson RENE 🡪** 15 **Commits**

## 2.3. Prévision des derniers travaux à réaliser

Les futurs rendues à faire par l’ensemble des collaborateurs de l’équipe projet sont :  
1 – Création des plug-in  
2 – Tests des plug-in  
3 – Implémentation des JFrames créaient avec les contrôleurs  
4 – Assemblage des différentes parties précédemment réalisées   
5– Le cahier de livraison  
6 – Préparation à la soutenance

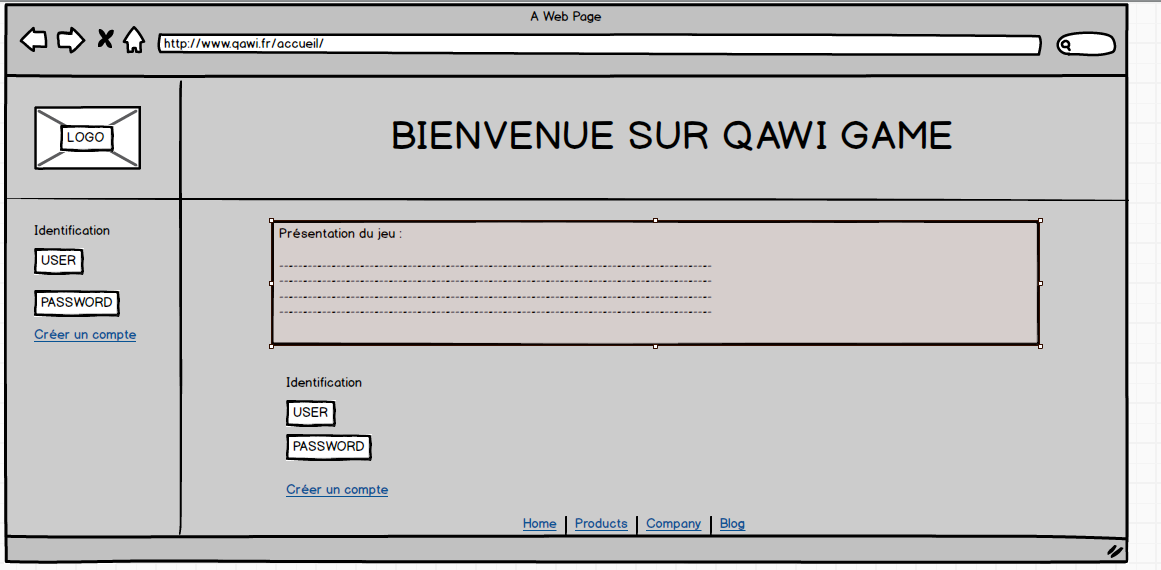
## 2.4. Contraintes techniques

Les contraintes techniques ont été identifiées dans le syllabus du projet annuel fournis par notre professeur Mr Sananes.

# 3. Modélisation

## 3.1 Interface Human Machine

PAGE D'ACCUEIL



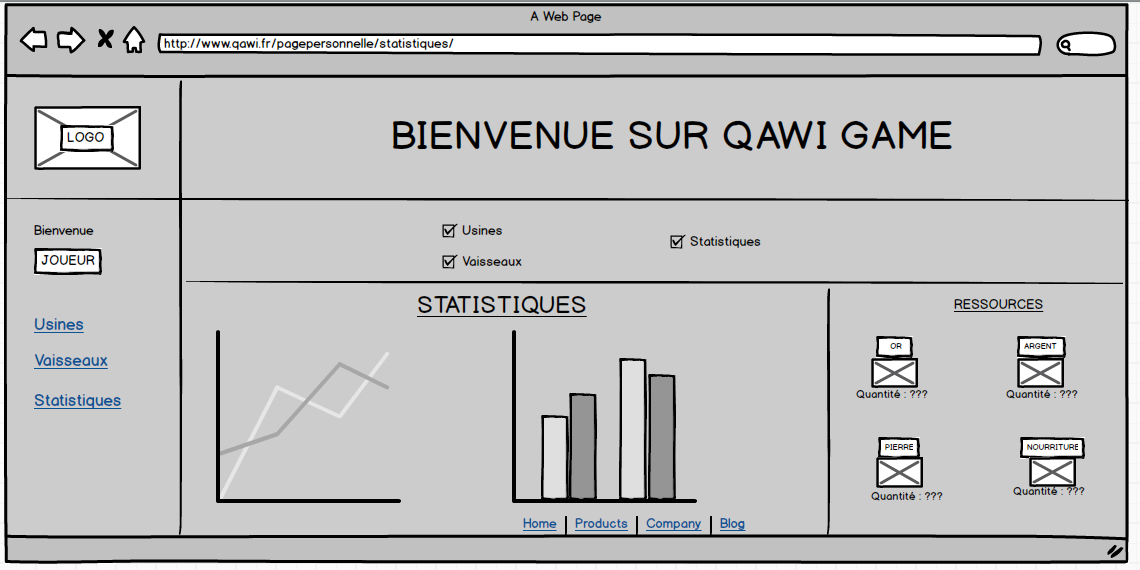
PAGE CREATION D'UN NOUVEAU COMPTE



PAGE "RESSOURCES"



CONSULTATION DES STATISTIQUES DU JOUEUR



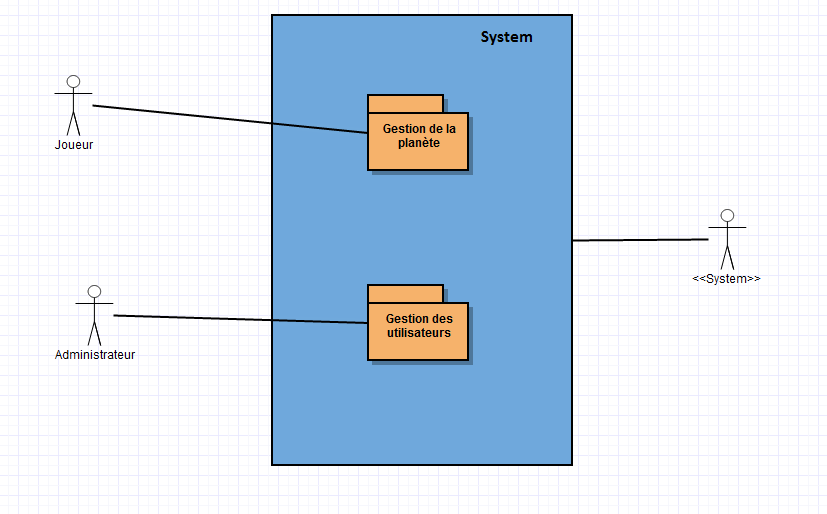
PAGE "USINES"



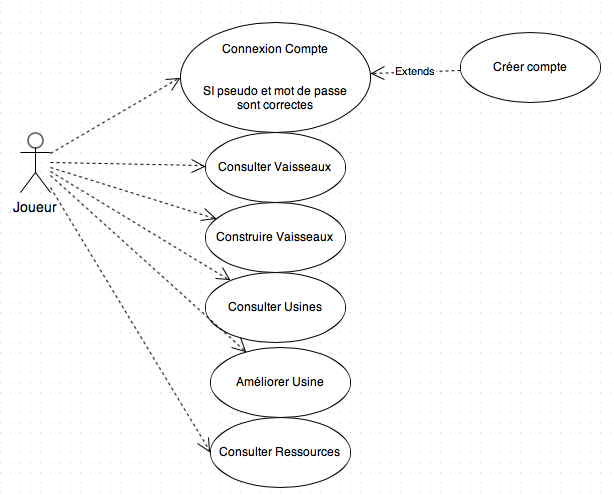
PAGE "VAISSEAUX"



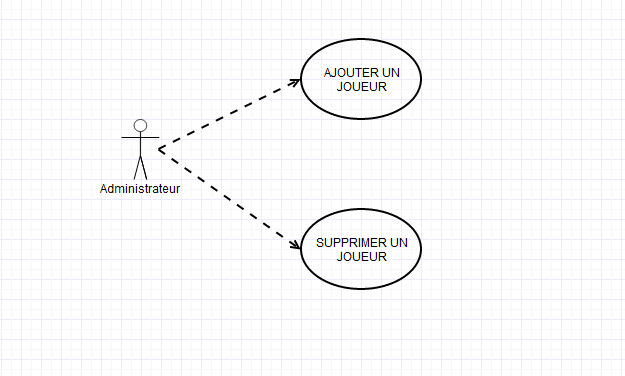
## 3.2.1. Diagramme de package



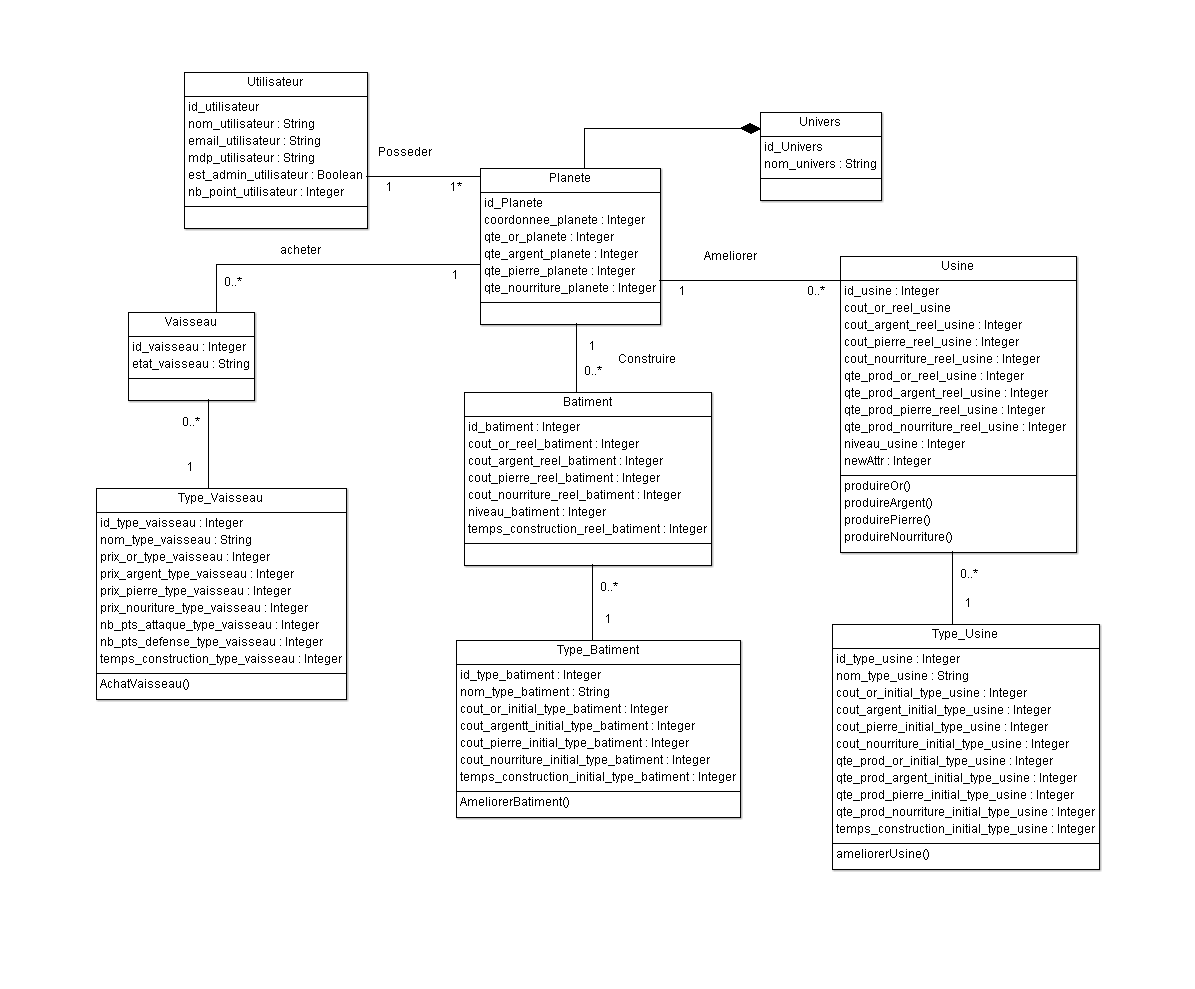
## 3.2.2. Diagramme de cas d’utilisation (Cas Gestion planète)



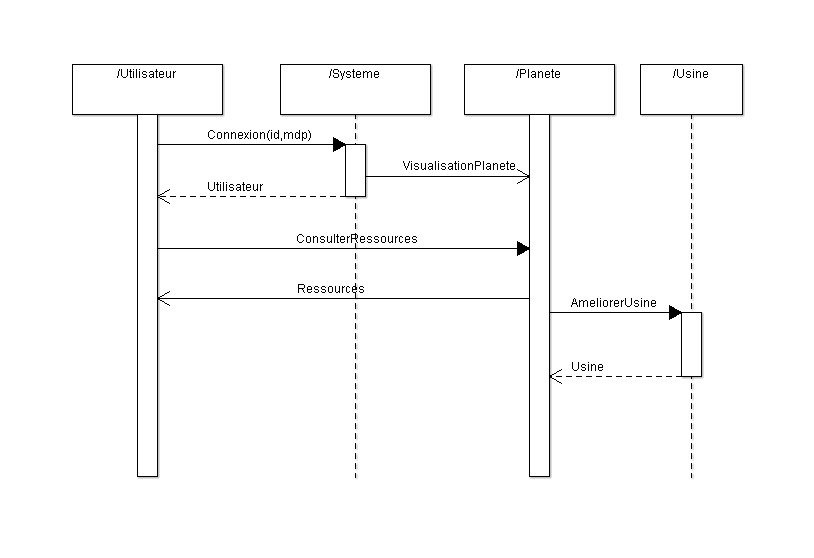
## 3.2.3. Diagramme de cas d'utilisation (Cas Gestion des joueurs)



## 3.2.4. Diagramme de classes



## 3.2.5. Diagramme de séquence (Cas Améliorer usines)



## 3.3. Descriptifs cas d’utilisation

**CAS : « CREER UN COMPTE » :**Auteur : Quentin TOUATIOUI  
Date Modification : 03/04/2014  
Démarrage :   
 A l’initiative du visiteur  
Pré-Condition :

Le visiteur se trouve sur la page du jeu.  
Dialogue :   
Scénario nominal :

|  |  |
| --- | --- |
| **UTILISATEUR** | **SYSTEME** |
|  | 1) Affichage de la page "Création d'un compte" |
| 2) Saisi d'un identifiant et Mot de passe |  |
|  | 3) Récupération et vérification identifiant non existant dans BDD |
|  | 4) Nouveau compte utilisateur créé |

Scénarios Alternatifs :   
A1. Après 2) ◊ Identifiant déjà existant - Retour au point 1)

Post-condition : Le compte a été crée

**CAS : « CONNEXION COMPTE »**:   
Auteur : Ivan KLARMAN  
Date Modification : 02/04/2014  
Démarrage :   
 A l’initiative du visiteur  
Pré-Condition :   
 Le visiteur doit se connecter

Dialogue :   
Scénario nominal :

|  |  |
| --- | --- |
| **UTILISATEUR** | **SYSTEME** |
|  | 1) Affiche la page de connexion |
| 2) Saisir les identifiants |  |
|  | 3) Récupère les identifiants et vérifie si l’utilisateur est en BDD |
|  | 4) Renvois vers la page personnel |
|  | 5) Affichage message de bienvenue |

Scénarios Alternatifs :   
A1. Après 3) 🡪 Aucun utilisateur n’est trouvé – Afficher un message d’erreur, Retour au point 1)

**CAS : « CONSULTER UNIVERS»**:   
Auteur : Ivan KLARMAN  
Date Modification : 02/04/2014  
Démarrage :   
 A l’initiative de l’utilisateur au clic sur un lien « Galaxie »  
Pré-Condition :   
 L’utilisateur doit déjà être connecté  
Dialogue :   
Scénario nominal :

|  |  |
| --- | --- |
| **UTILISATEUR** | **SYSTEME** |
|  | 1) Récupérer toutes les Galaxies en BDD |
|  | 2) Affichage « Univers » avec les différentes galaxies |

Scénarios Alternatifs :   
A1. Aucune Galaxie après 1), Affichage message « Aucune Galaxie »

**CAS : « CONSULTER CLASSEMENT»**:   
Auteur : Ivan KLARMAN  
Date Modification : 02/04/2014  
Démarrage :   
 A l’initiative de l’utilisateur au clic sur le lien « Classement »  
Pré-Condition :   
 L’utilisateur doit se connecter.  
Dialogue :   
Scénario nominal :

|  |  |
| --- | --- |
| **UTILISATEUR** | **SYSTEME** |
|  | 1) Récupérer tous les utilisateurs en BDD avec leurs points gagnés. |
|  | 2) Affichage du Classement par ordre de nombre de points. |
| 3) Filtrer le Classement |  |
|  | 4) Affichage des Utilisateurs avec le filtre |

**CAS : « CONSULTER COMPTE UTILISATEUR»**:   
Auteur : Ivan KLARMAN  
Date Modification : 02/04/2014  
Démarrage :   
 A l’initiative de l’administrateur  
Pré-Condition :   
 L’administrateur doit se connecter  
Dialogue :   
Scénario nominal :

|  |  |
| --- | --- |
| **UTILISATEUR** | **SYSTEME** |
| 1) Cliquer sur le lien « Administration Comptes » |  |
|  | 2) Récupérer tous les utilisateurs en BDD. |
|  | 3) Renvoyer vers la page « Administration Comptes» |
|  | 4) Affichage des comptes utilisateurs avec leurs identifiants et mots de passe par ordre alphabétique du nom d’utilisateur. |
| 5) Filtrer par nom d’utilisateur ou id |  |
|  | 6) Récupérer le filtre en BDD |
|  | 7) Affichage des Utilisateurs avec le filtre |
| 8) Modification des données |  |
|  | 9) Changement des données en BDD |
|  | 10) Message de modification effectué |
| 10) Fermer le message de modification |  |
|  | 11) Retour page personnel |

**CAS : « CONSULTER RESSOURCES »**:   
Auteur : Quentin TOUATIOUI  
Date Modification : 03/04/2014  
Démarrage :   
 A l’initiative du visiteur  
Pré-Condition :   
 L'utilisateur doit être connecté.   
Dialogue :   
Scénario nominal :

|  |  |
| --- | --- |
| **UTILISATEUR** | **SYSTEME** |
|  | 1) Récupération ressources utilisateur dans BDD |
|  | 2) Affichage des ressources |
| 3) Consultation des ressources |  |

Scénarios Alternatifs :   
(Vide ◊ L'utilisateur peut toujours consulter ses ressources)

**CAS : « CONSULTER USINES »**:   
Auteur : Quentin TOUATIOUI  
Date Modification : 03/04/2014  
Démarrage :   
 A l’initiative du visiteur  
Pré-Condition :   
 L'utilisateur doit être connecté.

Dialogue :   
Scénario nominal :

|  |  |
| --- | --- |
| **UTILISATEUR** | **SYSTEME** |
| 1) Choix : Visualisation "Usines" |  |
|  | 2) Récupérer niveaux des usines dans BDD |
|  | 3) Affichage des usines |
| 4) Consultation des usines |  |

Scénarios Alternatifs :

(Vide ◊ Il est toujours possible de consulter ses usines)

**CAS : « AMELIORER USINES »**:

Auteur : Quentin TOUATIOUI  
Date Modification : 03/04/2014  
Démarrage :   
 A l’initiative du visiteur  
Pré-Condition :

Le visiteur se trouve sur la page du jeu.  
Dialogue :   
Scénario nominal :

|  |  |
| --- | --- |
| **UTILISATEUR** | **SYSTEME** |
| 1) Choix : Visualisation "Usines" |  |
|  | 2) Récupération niveaux des usines dans BDD |
|  | 3) Affichage des usines |
| 4) Amélioration des usines |  |
|  | 5) Usines passent au niveau supérieur |

Scénarios Alternatifs :

A1. Après 4) ◊ Manque de ressources, Amélioration impossible - Retour au point 3)

**CAS : « CONSULTER VAISSEAUX »**:   
Auteur : Quentin TOUATIOUI  
Date Modification : 03/04/2014  
Démarrage :   
 A l’initiative du visiteur  
Pré-Condition :   
 L'utilisateur doit être connecté.

Dialogue :   
Scénario nominal :

|  |  |
| --- | --- |
| **UTILISATEUR** | **SYSTEME** |
| 1) Accéder à la page Vaisseaux |  |
|  | 2) Récupérer informations Vaisseaux |
|  | 3) Affichage des vaisseaux |

Scénarios Alternatifs :   
A1. Après 2) ◊ Aucun vaisseau disponible, affichage impossible - Retour au point 1)

**CAS : « ACHETER VAISSEAUX »**:   
Auteur : Quentin TOUATIOUI  
Date Modification : 03/04/2014  
Démarrage :   
 A l’initiative du visiteur  
Pré-Condition :   
 L'utilisateur doit être connecté.

Dialogue :   
Scénario nominal :

|  |  |
| --- | --- |
| **UTILISATEUR** | **SYSTEME** |
| 1) Choix : Visualisation "Vaisseaux" |  |
|  | 2) Récupération niveaux des vaisseaux dans BDD |
|  | 3) Affichage des vaisseaux |
| 4) Achat de nouveaux vaisseaux |  |
|  | 5) Obtention de nouveaux vaisseaux |

Scénarios Alternatifs :   
A1. Après 4) ◊ Ressources insuffisantes - Retour au point 3)

## 3.4. Algorithmes Complexes Imaginés

A) Temps d’attentes des améliorations/constructions des usines/vaisseaux :   
  
Nous étudierons un cas avec les Usines.  
  
// Deux constructeurs possibles   
 // Premier constructeur   
Classe Usine (String nom,,int tempsAttentes, int niveau, int nbRessourcesA, int RessourcesB)  
 // Deuxième constructeur   
Classe Usine (String nom, int tempsAttentes, int niveau, int nbRessourcesA, int RessourcesB, int nbRessourceC)  
// Calcul de l’attente en fonction du niveau de l’usine  
uneUsine.getTempsAttentes = uneUsine.getNiveau()\*(uneUsine.getNiveau() + 3);  
  
***Exemple :***  
Soit si une usine est de niveau 1 pour passer au niveau 2 en l’améliorant, le calcul sera :   
Attentes = 1\*(1 + 3)  
Attentes = 4 minutes  
Soit si une usine est de niveau 2 pour passer au niveau 3 en l’améliorant, le calcul sera :  
 Attentes = 2\*(2 + 3)  
Attentes = 10 minutes  
Soit si une usine est de niveau 2 pour passer au niveau 3 en l’améliorant, le calcul sera :  
Attentes = 3\*(3+ 3)  
Attentes = 18 minutes  
ETC …

// Vérifier possibilité d’amélioré ou non  
 Boolean faisable = false ;

int Compteur = uneUsine.getTempsAttentes() ;  
// Condition de faisabilité ou non ( améliorations ou constructions)   
 SI Compteur == 0  
 ALORS faisable = true ;  
 SINON faisable = false ;  
// Si faisable Que faire ?, si non faisable Que faire ?  
 SI faisable == true  
 ALORS uneUsine.upNiveau() ;  
 SINON alert(« Une amélioration n’est pas terminée »  ) ;

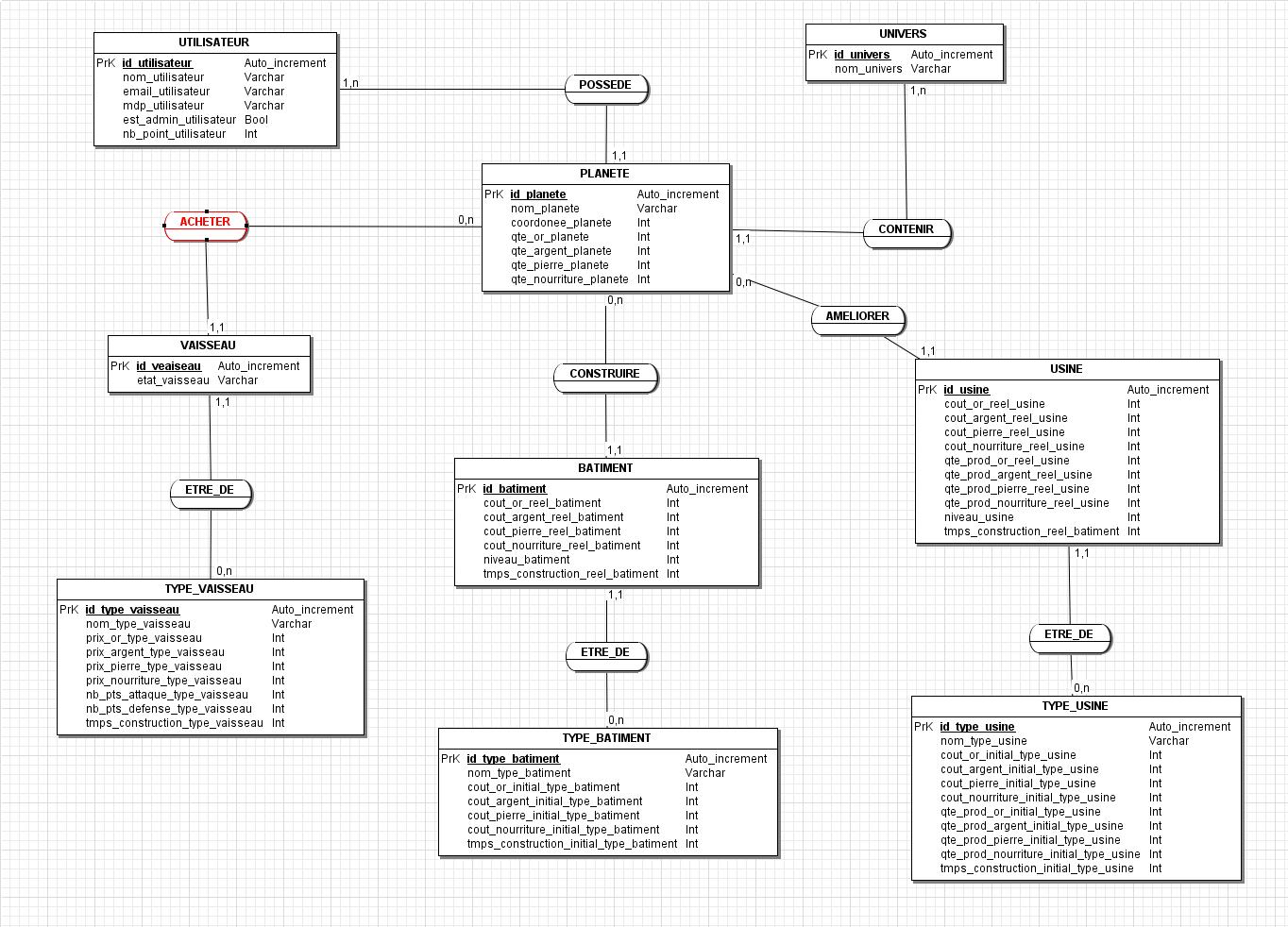
B) Algorithme d’attaque d’un joueur à un autre :   
// Méthode prévue pour l’attaque   
// @param : une liste de vaisseaux  
// @param : Un objet Joueur  
Classe Joueur avec une méthode attaquer (List<Vaisseaux> uneListeV, Joueur unJoueur) ;   
Classe Joueur avec une méthode estAttaquer (List<Vaisseaux> uneListeV, Joueur unJoueur) ;

// Dans cette méthode un calcul décrit la valeurd’attaque de chaques vaisseaux envoyé au combat et on a la valeur d’attaque final.  
 int valeurAttaqueFinal = 0 ;  
 foreach(Vaisseaux unVaisseau as uneListeV){

valeurAttaqueFinal += unVaisseau.getValeurAttaque() ;  
 }  
// Pour savoir si la personne est gagnante, on vérifie dans une autre méthode la valeur défensive de l’adversaire représenté par des vaisseaux également mais défensif  
 int valeurDéfensiveFinal = 0 ;  
 foreach(Vaisseaux unVaisseauDéfensif as uneListeV){

valeurDéfensiveFinal += unVaisseauDéfensif.getValeurDéfense() ;  
 }

## 3.5. MCD

****

# 4. Architecture Logiciel

## 4.1. Structure client léger et clients lourds

PARTIE APPLICATION LOURDE

PARTIE WEB

Application Lourde   
JSE

Plug In  
JSE

TimerJob  
JSE

Site Web   
PHP

## 4.2. Architecture Logique

DB

Site Web  
PHP

Serveur Web  
Apache  
  
SGBD  
MySQL

Timer Job  
JSE

Exécuter côté serveur

Plug In  
JSE

Application lourde   
JSE

Plug In  
JSE

Chargés par l’application lourde

## 4.3. Architecture de l’application lourde

🡪 Package Data  
 🡪 Connexion.java //Chargement JDBC  
 🡪 InsertData.java // Insertion en base de données  
 🡪 SelectData.java //Sélection en base de données

🡪 UpdateData.java //Modification en base de données

🡪 Package Business Layer // Classes métiers  
 🡪 Planete.java  
 🡪 User.java  
 🡪 Vaisseau.java  
 🡪 Usine.java  
 🡪 Flotte.java

🡪 Package Model  
 🡪 ModelLayer.java //Toutes les fonctions de traitements et requêtes base de données

🡪 Package Plug In

?

🡪 Package GUI //Vue et Contrôleurs

🡪 JFrame Connexion  
 🡪 JFrame Inscription  
 🡪 JFrame Jeux [.]

🡪 Dossier « pict »  
🡪 Dossier « conf »

🡪 Dossier « plug »

[…]

# 5. Conception

## 5.1. Descriptifs des API / Framework utilisés

### 5.1.1. JDBC

JDBC est l’API utilisée pour l’accès aux bases de données en Java. Grâce à cette dernière, il est possible d’utiliser des références d’interface comme par exemple « Connection », « Statement » ou encore « ResultSet ».  
Parmi ses interfaces, l’on utilise « Driver » pour récupérer un objet de type « Connection » qui nous donnera le droit grâce à une connexion avec un user et un mot de passe accès à notre base de données.

### 5.1.2 JavaMail

JavaMail a été utilisé pour l’envoi de mail. En effet on utilise des objets de type « Session », « Properties », « Message » ou encore « Transport ».

## 5.2. Premiers Programmes

## Envoi de mail

**public** **void** **generationRapport**(**int** annonce) **throws** IOException{

**final** String username = "klarman.ivan@gmail.com";

**final** String password = "Vet1felor";

Properties props = **new** Properties();

props.put("mail.smtp.auth", "true");

props.put("mail.smtp.starttls.enable", "true");

props.put("mail.smtp.host", "smtp.gmail.com");

props.put("mail.smtp.port", "587");

Session session = Session.*getInstance*(props,

**new** javax.mail.Authenticator() {

**protected** PasswordAuthentication **getPasswordAuthentication**() {

**return** **new** PasswordAuthentication(username, password);

}

});

**if**(annonce == 0){

**try** {

Message message = **new** MimeMessage(session);

message.setFrom(**new** InternetAddress("klarman.ivan@gmail.com"));

message.setRecipients(Message.RecipientType.TO,

InternetAddress.*parse*("klarman.ivan@gmail.com"));

message.setSubject("Qawi - Rapport de Combat");

DataSource fds = **new** FileDataSource("images\\Background.jpg");

message.setDataHandler(**new** DataHandler(fds));

message.setContent("<h2><b>Salut combattant des etoiles <br /><br />"

+ " <p style='color:red;'>La flotte attaquante a echouer !</p></b></h2>"

+ "<img height='500' width='500' src=\"http://nsa34.casimages.com/img/2014/06/23/140623010637705974.jpg\">",

"text/html");

Transport.*send*(message);

System.out.println("Done");

} **catch** (MessagingException e) {

**throw** **new** RuntimeException(e);

}

}

}

## 

**Deux extraits de méthodes de la partie ModelLayer**

**1) Améliorer usine**  
**public** **void** **ameliorerUsine**(Usine usine){

**int** newNiveau = Integer.*parseInt*(usine.getNiveau()) + 1;

**int** newProdUsine = Integer.*parseInt*(usine.getProd\_usine()) \* newNiveau;

UpdateData upUsine = **new** UpdateData(con.getConnection()," "

+"UPDATE usine "

+"SET usine.prod\_usine = " + newProdUsine + " ,"

+"usine.niveau\_usine = "+ Integer.*toString*(newNiveau) +" ,"

+"usine.cout\_or = "+ Integer.*parseInt*(usine.getCout\_or())\*2 + " ,"

+"usine.cout\_argent = "+ Integer.*parseInt*(usine.getCout\_argent())\*2 + " ,"

+"usine.cout\_pierre = "+ Integer.*parseInt*(usine.getCout\_pierre())\*2 + " ,"

+"usine.cout\_nourriture = "+ Integer.*parseInt*(usine.getCout\_nourriture())\*2 + " "

+"WHERE usine.id\_usine = "+ usine.getId\_usine());

}

**2) Ajouter les ressources aux planètes**

**public** **void** **addRessourcesAllUsine**(){

List<Usine> listeUsines = **new** ArrayList<Usine>();

List<Planete> listePlanetes = **new** ArrayList<Planete>();

listeUsines = getAllUsine();

listePlanetes = getAllPlanete();

**for**(Planete pl : listePlanetes){

**for** (Usine farm : listeUsines){

**switch**(farm.getId\_type\_usine()){

**case** "0" : // AJOUT DE L'OR

UpdateData updOr = **new** UpdateData(con.getConnection(),

" UPDATE planete JOIN usine ON planete.id\_planete = usine.id\_planete "

+"SET planete.qte\_or = "+pl.getQte\_or()+" + "+farm.getProd\_usine()+" "

+"WHERE usine.id\_planete = planete.id\_planete "

+"AND usine.id\_type\_usine = 0 ");

**break**;

**case** "1" : //AJOUT DE L'ARGENT

UpdateData updArgent = **new** UpdateData(con.getConnection(),

" UPDATE planete JOIN usine ON planete.id\_planete = usine.id\_planete "

+"SET planete.qte\_argent = "+pl.getQte\_argent()+" + "+farm.getProd\_usine()+" "

+"WHERE usine.id\_planete = planete.id\_planete "

+"AND usine.id\_type\_usine = 1 ");

**break**;

**[…]**

Réalisé par :  
  
Ivan KLARMAN  
Quentin TOUATIOUI  
Walson RENE