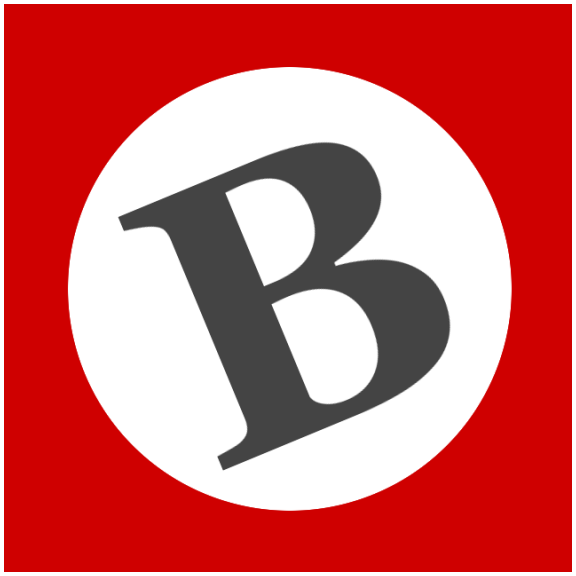

Projet d'AAE

Blitz

Clone du jeu Wazabi

Javier Lethé - Matteo Taroli - Przemyslaw Gasinski

19 décembre 2015

Table des matières

1 Introduction

Wazabi est un jeu de dés et de cartes dont le but est de se débarrasser de tous ses dés. Nous avons dû, dans le cadre du cours de mise en situation professionnelle, implémenter un clone de ce jeu sur une période d'une semaine. Le thème de notre clone est la 2eGuerre Mondiale et s'appelle Blitz, diminutif de « Blitzkrieg ».

Un des buts de ce projet étant de se rapprocher le plus possible de l'expérience professionnelle, certaines contraintes nous ont été imposées. Entre autres, le projet devait être développé en utilisant les technologies EJB et JSP, avec la possibilité d'utiliser d'autres technologies telles que Bootstrap, JavaScript et Ajax. Il nous a aussi été demandé de procéder de manière itérative, ce que nous avons fait en procédant par Usecase. Enfin, nous étions divisés en groupes de 4, qui ont été formés par les chefs d'équipe sur base de CV anonymes.

Nous avons divisé le travail de la manière suivante : deux personnes se sont concentrées sur le back-end (la partie EJB) et deux sur le front-end (JSP). Malheureusement, un membre du groupe a décidé d'arrêter le projet en cours de route. De ce fait, nous avons dû re-distribuer afin de pouvoir continuer à avancer.

Dans la suite de ce rapport, nous allons présenter les détails du programme développé.

2 Analyse

2.1 Diagramme de navigation

Le parcours de l'utilisateur dans Blitz se fait selon le diagramme suivant :

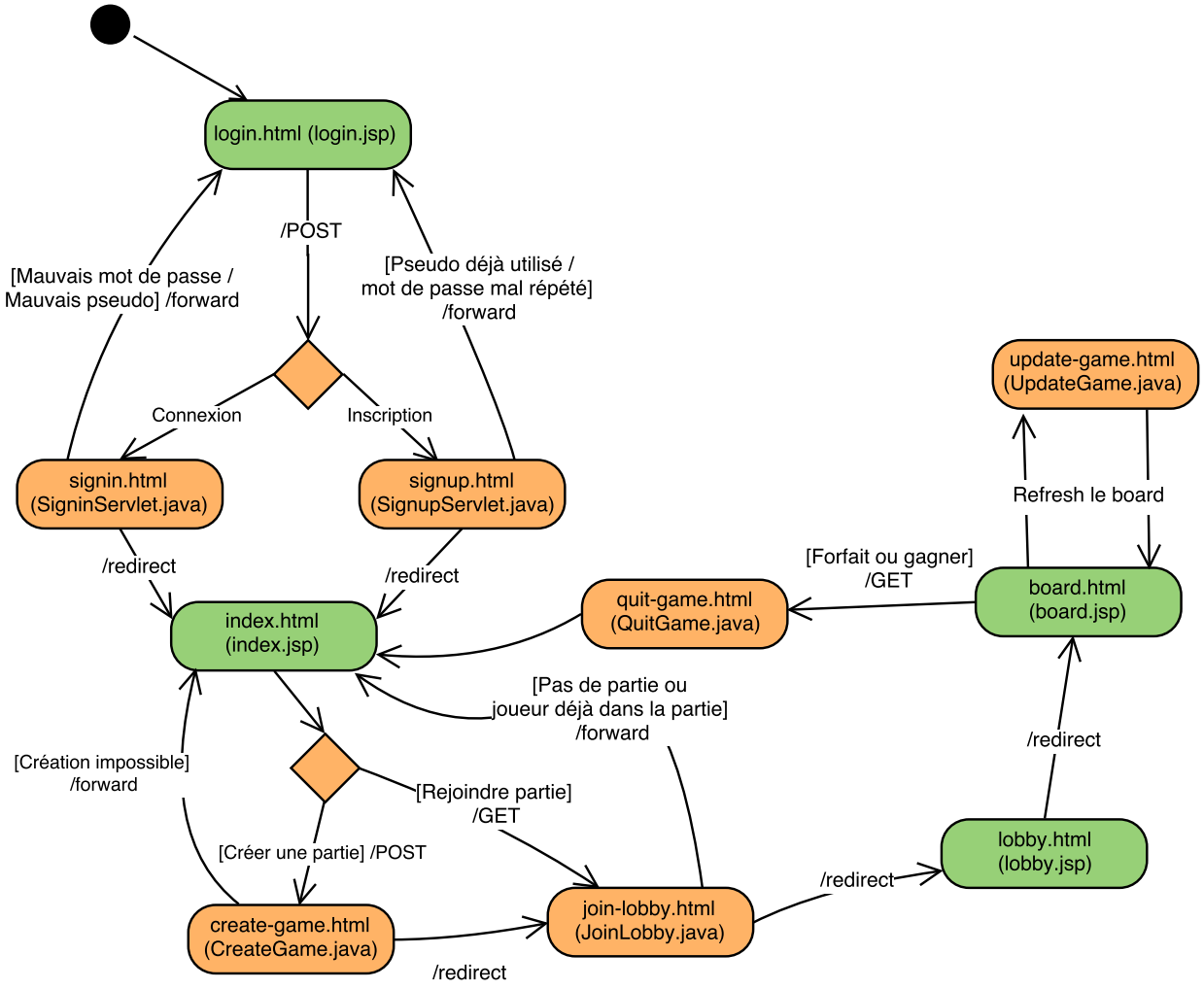

Table 1 – Diagramme de navigation

2.2 Scénario nominal « Rejoindre une partie »

L'utilisateur étant déjà connecté, il rejoint une partie selon le scénario suivant :

1. Le joueur clique sur le bouton « Rejoindre une partie »	
	2. Le système vérifie si une partie est déjà lancée
	a.1 Le système envoie le joueur vers le lobby.
	a.2 Le nombre de joueurs minimal étant atteint, le jeu est lancé

3 Tests

3.1 Tests unitaires

Les tests unitaires de Blitz se trouvent dans le projet « BlitzTests » et testent la classe GameUcc. Celà nous a permis de régler un nombre conséquent de bugs dans la partie EJB, ce qui n'empêche malheureusement pas certaines actions de mal fonctionner une fois appelées par l'utilisateur.

3.2 Test fonctionnels

3.2.1 Connexion

1. Cliquer sur le bouton « Connexion / Inscription » situé en haut à droite.	
	2. Un popup apparait proposant de s'inscrire ou de se connecter.
3. Dans la partie de droite, entrer « ol » comme nom d'utilisateur et « ol » comme mot de passe.	
	4. Le système vérifie que l'utilisateur existe et que le mot de passe est correct et renvoie le joueur sur la page d'accueil.
5. Cliquer sur le bouton « Déconnexion ».	
6. Cliquer à nouveau sur le bouton « Connexion / Inscription ».	
	7. Le popup réapparait
8. Dans la partie de droite, entrer « ol » comme nom d'utilisateur et « mauvaisMdp » comme mot de passe.	
	9. Le système vérifie que l'utilisateur existe et que le mot de passe est correct et affiche l'erreur « Nom d'utilisateur ou mot de passe incorrect » et reste sur le popup.

3.2.2 Création de partie

1. En partant de la page d'accueil, cliquer sur le bouton « Créer une partie ».	
	2. Un popup apparait permettant d'entrer le nom de la partie.
3. Entrer « test » en tant que nom de partie.	
4. Cliquer sur le bouton « OK ».	
	5. Le système renvoie vers le lobby de la partie.
6. Attendre d'autres joueurs et joueur.	

3.2.3 Jeu

N'importe quand dans le jeu :

1. Un joueur clique sur le bouton « Abandonner »	
	2. Le serveur le redirige vers la page d'accueuil.
	3. Si il ne reste qu'un seul joueur dans la partie, le serveur affiche un message de victoire au dernier joueur.

Quand c'est votre tour :

1. Le joueur clique sur le bouton de lancer les dés.	
	2. Le serveur affiche les faces des dés lancés.
3. Le joueur clique sur un dé	
[Dé à donner]	
	3.a.1 Le serveur affiche un pop-up pour demander à qui le donner.
3.a.2 Le joueur clique sur l'adversaire choisit.	
	3.a.3 Le serveur met a jour la case de l'adversaire choisit en incrémentant ses dés, affiche un message indiquant que l'opération s'est bien déroulée et grise le dé joué.
[Dé de pioche]	
	3.b.1 Le serveur rafraichit la liste des cartes du joueur qui en aura alors une de plus, affiche un message indiquant que l'opération s'est bien déroulée et grise le dé joué.

4. Le joueur clique sur une carte	
[La carte ne peut pas être jouée]	
	Le serveur affiche un mesage d'échec
[Si la carte nécéssite une cible]	
	4.a.1 Le serveur affiche un pop-up pour demander à qui sera la cible.
4.a.2 Le joueur clique sur l'adversaire choisit.	
	4.a.3 Le serveur met a jour la case de l'adversaire choisit et affiche un message indiquant que l'opération s'est bien déroulée.
[Si la carte nécéssite un sens]	
	4.b.1 Le serveur affiche un pop-up pour demander quel sera le sens.
4.b.2 Le joueur clique sur le sens choisit.	
	4.b.3 Le serveur met a jour la case de l'adversaire choisit et affiche un message indiquant que l'opération s'est bien déroulée.
[La carte ne nécessite aucune autre information]	
	4.c.1 Le serveur effectue l'action indiquée sur la carte et rafraichit le plateau de jeu. Il affiche un message indiquant que l'opération s'est bien déroulée.
5. Le joueur clique sur une deuxième carte.	
	6. Le serveur affiche une message indiquant à l'utilisateur qu'il ne peut pas jouer une deuxième fois.
7. Le joueur clique sur le bouton « finir le tour »	
	8. Le serveur rend inopérant boutons de l'interface et passe au joueur suivant.

4 Architecture

4.1 Partie EJB

dao
Contient le DAO permettant l'accès aux tables de la base de données.

daoImpl
Contient les DAO implémentés permettant l'accès à une table précise selon l'entité.

domaine
Contient les différents objets formant l'application.

usecases
Contient les interfaces correspondantes aux usecases. Ces interfaces sont le seul lien entre la partie EJB et la partie JSP.

usecasesImpl
Contient les implémentations des interfaces Usecases.

utils
Contient les classes offrant des méthodes de vérification, appelées par les autres classes.

4.2 Partie JSP

- Partie Java

  filters
  Contient les filtres gérant les requêtes utilisateurs. Permet de rediriger l'utilisateur en cas d'erreur et de vérifier l'utilisation des cookies.

  game
  Contient les servlets Java gérant une partie.

  servlets
  Contient les servlets gérant le reste de l'application.

- Partie Web

  css
  Contient le css régissant le style de l'interface du programme.

  images
  Contient les images utilisées dans l'interface.

  js
  Contient les fichiers gérant la partie JavaScript de l'interface.

  libs
  Contient les librairies Bootstrap, jQuery et Toastr

WEB-INF

Contient les fichiers décrivant le HTML utilisé par les pages web ainsi que la configuration du server permettant d'afficher la page correspondant à la requête.

5 Notes

Il est demandé dans l'énoncé de charger les données du jeu à partir du fichier XML. Après l'avoir fait, nous n'avons pas compris pourquoi nous devions faire persister les dés dans la base de donnée. C'est pourquoi vous ne trouverez que 5 tables dans la DB et pas 7.

Nous avons aussi implémenté un patron de conception de type « State » dans la classe domain.Game.java de la partie EJB, ainsi qu'un timer de 30 secondes qui est lancé dès que le premier joueur rejoint la partie.

Enfin, une amélioration possible serait de gerer le cas ou la pioche est vide. Il faudrait alors proposer au joueur de voler une carte à un de ses adversaires.

6 Manuels

6.1 Manuel d'installation

Les constantes sont dans un fichier xml, situé a Blitz/BlitzEJB/ejbModule/blitz.xml.

6.1.1 Instalation avec Eclipse

1. Créer un schéma nommé « blitz » dans la base de données.
2. Importer le projet dans éclipse. [1]
3. Click droit sur le serveur
4. Cliquer sur « Add and Remove »
5. Ajouter « BlitzEAR »
6. Cliquer sur « Full Publish »
7. Lancer le serveur

6.1.2 Installation sans Eclipse

1. Créer un schéma nommé « blitz » dans la base de données.
2. Mettre Blitz.EAR dans le dossier suivant JBOSS_HOME/server/your config>/deploy folder.

6.2 Manuel d'utilisation de Toastr

Pour afficher les notifications aux joueurs, nous utilisons Toastr.js (`https://codeseven.github.io/toastr`), qui est un plugin de jQuery. L'utilisation que nous en faisons est assez basique, les seule méthode à connaitre sont les suivantes : toastr.info(), toastr.warning() et toaster.success().

Ces méthodes fonctionnent de la façon suivante :

- toastr.info(« Message »)
  Affiche un Toast contenant un message à l'utilisateur.
- toastr.info(« Titre », « Message »)
  Affiche un Toast contenant un message et un titre à l'utilisateur

1. Il est supposé que Wildfly est déjà configuré, ainsi que la JDK.

Toastr peut aussi prendre certaines options à son instanciation. Les plus importantes pour notre projet ont été les suivantes :

positionClass : toast-top-center
Permet de placer le toast au centre de l'écran

onClick : null
Empêche de cliquer sur le toast (car nous n'avosn aucune action liée à un clic)

preventDuplicate : false
Permet les messages dupliqués (car un joueur pourrait faire plusieurs fois une action identique)

D'autres options existent pour personaliser les animations et durées d'affichage.

6.3 Manuel d'utilisation

6.3.1 Comment accéder au jeu ?

Pour jouer sur une machine locale, tapez dans le browser « `localhost:8080/Blitz` » (sans les guillemets).

Une fois connecté, vous arrivez sur l'écran suivant :

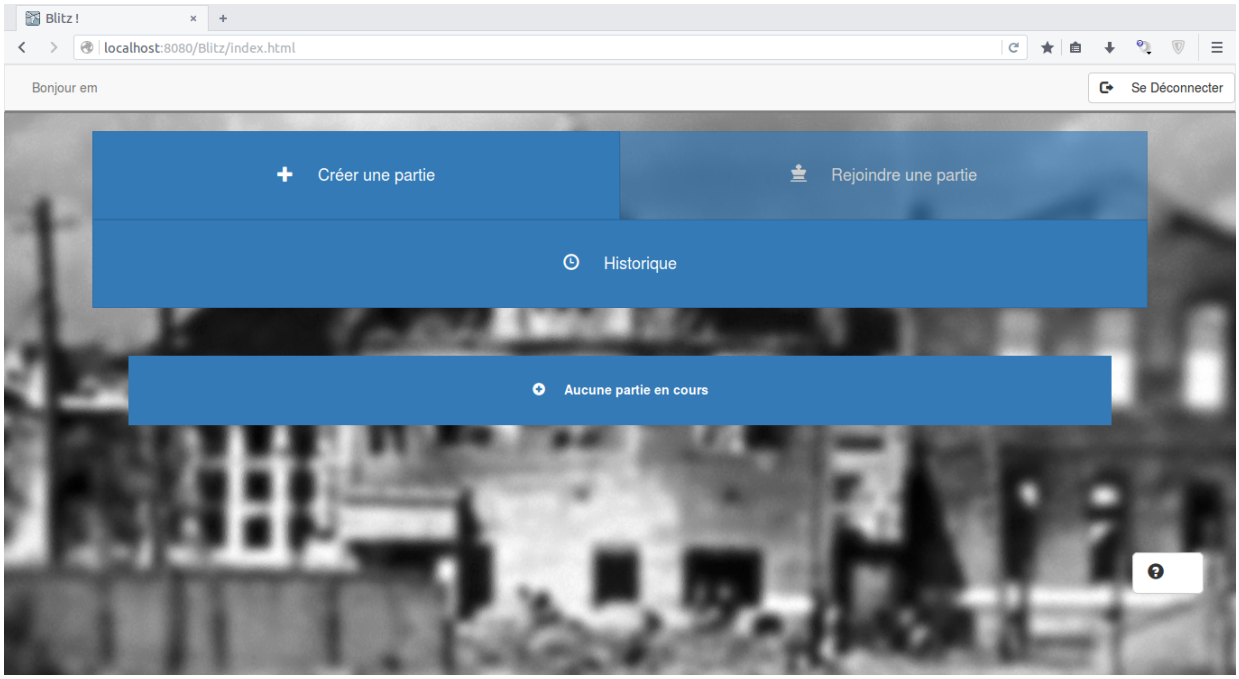

Table 2 – Diagramme de navigation

6.3.2 Comment créer une partie ?

Pour lancer une partie, il vous suffit de cliquer sur créer une partie. Si **aucune partie n'a été créée**, une pop-up apparaitra alors pour que vous donniez un nom à la partie. Après ça, vous voilà redirigé vers le lobby.

Si une **partie est déjà créée**, et que vous ne **pouvez pas la rejoindre** (ce qui arrive si la partie est déjà en cours ou si le nombre de joueurs maximum est déjà atteint), le bouton est désactivé et vous empêche donc de cliquer dessus.

6.3.3 Comment rejoindre une partie ?

Pour rejoindre une partie, cliquez sur le bouton correspondant sur la page d'accueil.

Si une partie est **en attente de joueurs supplémentaires**, vous serez redirigé vers le lobby de cette partie.

Si vous ne pouvez pas rejoindre la partie créée (**aucune partie**, partie **en cours** ou le **nombre de joueurs maximum est atteint**), le bouton est désactivé.

6.3.4 Comment lancer une partie ?

Une fois dans la lobby, la partie se lance automatiquement une fois le nombre minimum de joueur atteint. Une fois ce nombre atteint, tous les joueurs sont redirigés vers la page de jeu.

6.3.5 Comment se déroule le tour de jeu ?

Dans la partie supérieure du plateau de jeu se trouvent vos adversaires. Dans la partie inférieure se trouve votre jeu.Vous pouvez y voir vos cartes ainsi que le nombre de dés que vous possédez. Sur la droite se trouve aussi un bouton vous permettant de lancer les dés.

Si un ennemie est en surbrillance, celà signifie que c'est à son tour de jouer. Vous pouvez aussi voir son nom en haut à gauche, après « Tour de : ».

Quand vient votre tour de jouer, commencez par lancer vos dés. Ce lancé amène à 3 types de dé possible :

- Rond jaune : Pièce de monnaie, permet de payer une carte à jouer.
- Carré avec un + : Pioche une carte.
- Flèche vers le haut : Donne un dé à un ennemi.

Une fois les dés lancés, vous pouvez maintenant cliquer sur une carte pour la jouer ou encore cliquer sur un dé de pioche ou de don de dé pour l'utiliser.

Si une action demande de viser un joueur, une fenêtre modale apparait pour vous permettre de choisir un ennemie. Une fenêtre apparait aussi pour permettre de choisir le sens dans lequel passer les dés.

Une fois que vous avez fini votre tour, vous pouvez cliquer sur « Passer le tour », se trouvant en haut à droite.

Durant le jeu, des notifications de type Toast apparaissent selon vos actions.

7 Conclusion

Durant ce projet, nous avons beaucoup appris. Nous avons en effet approfondis notre connaissance des technologies utilisées et amélioré notre façon de travailler ensemble grâce, entre-autre, à l'utilisation de Github. Cet outil nous a permis de travailler en parallèle tout en restant tous sur la même version de notre code. Nous avons cependant préféré l'utilisation de la ligne de commande pour accéder aux commandes Git, l'utilisation du plugin d'Eclipse ne permettant pas un contrôle total de l'outil. Nous avons aussi pu nous rendre compte de l'importance des « outils développeur » permettant un débug plus aisé de la partie web et principalement la partie JavaScript.

Ce projet nous aura aussi montré certaines facettes moins sympathiques du déroulement d'un projet. La durée du projet étant très courte, et nous retrouvant à 3 après le départ d'un membre, nous avons dû travailler chaque jour plus d'une dizaine d'heures. Ce travail intensif nous a tous fait ressentir une grande pression et une importante fatigue.

Ce délai assez court pour réaliser le projet nous aura aussi empechés de rendre un travail aussi propre que nous aurions voulu. Nous nous sommes en effet rendu compte que certaines parties de notre code, telles que la gestion des effets des cartes, auraient pu être améliorées via l'utilisation de patterns que nous n'avons pas eu le temps d'implémenter. De plus, la sécurité de l'application peut aussi être améliorée.

Malgré celà, nous sommes fiers de ne pas avoir abandonné et d'avoir pu implémenter toutes les fonctions nécessaires à un programme fonctionnel.