Compte-rendu de projet

# DROP DA BOMB

par

VINCENT Pierre - DEUTSCH Rémi - LOIGNON Lucas

# Table des matières

# Introduction

## 1. Description

Ce rapport vise à présenter le travail réalisé pour l'UE Programmation Informatique Appliqué, sous la direction de Mme REN, par DEUTSCH Rémi, LOIGNON Lucas et VINCENT Pierre.

Nous avons imaginé un jeu de carte basé sur un système de tour par tour et de coût par carte. Nous avons décidé, pour cette UE, de mettre en place un système d'administration de base de données lié à notre concept de jeu.

## 2. Motivation et trame de fond

Pour les besoins de cette UE nous avons décidés de nous limiter à la mise en place d'une base de données cohérente avec notre concept de jeu ainsi qu'au développement d'une application web pouvant administrer notre base de données.

En effet la mise en place de ce type de jeu implique tout d'abord une gestion et une création de base de données, il faut que les joueurs puissent créer un compte, composer leurs deck, changer les informations liées à leurs comptes bref tout autant de fonctionnalités à implémentées avant de pouvoir commencer le développement du jeu à proprement parlé.

Ce projet est l'occasion pour nous de mettre à profil nos compétences, autant dans la conception de logiciel que dans l'implémentation. C'est aussi l'occasion de mettre en place des solutions de développement coopératif afin de pouvoir répondre aux objectifs de ce projet. Ce projet est également l'opportunité de se familiariser avec de nouvelles technologies ainsi que de nouvelles techniques de gestion de projet.

Pour la gestion de notre serveur, nous avions besoin d'une solution permettant la communication asynchrone indépendamment des requêtes classiques du protocole http dans un souci d'évolution. En effet nous avons le projet de continuer le développement de l'API de gestion et du jeu en lui-même, pour l'implémentation du jeu nous avions besoin de ce type de communication.

Nous avons donc choisi d'utiliser JAVA EE avec le conteneur de Servlets Tomcat car les sockets de java remplissaient nos spécifications, de plus JAVA EE est un langage souvent demandé dans le milieu professionnel ce qui n'a fait que renforcer notre choix de vouloir apprendre de nouvelles technologies.

## 3. Méthodologie

Pour permettre une organisation des taches optimale nous avons opté pour une méthode agile : SCRUM.

Nous avons découpé les taches en plusieurs sprints de 2 semaines chacun, nous avons progressé en implémentant des « couches » en commençant par mettre en place une base de données dans les

premiers temps, puis commencé l'implémentation de la première couche de l'API de gestion pour communiquer avec notre base de données puis nous avons implémenté la seconde couche de l'API qui se charge de faire les traitements des données et enfin nous avons terminé avec la mise en place de toutes les pages JSP et du CSS du coté client.

Le fichier de gestion avec la méthode SCRUM sera fournie en annexe de ce document.

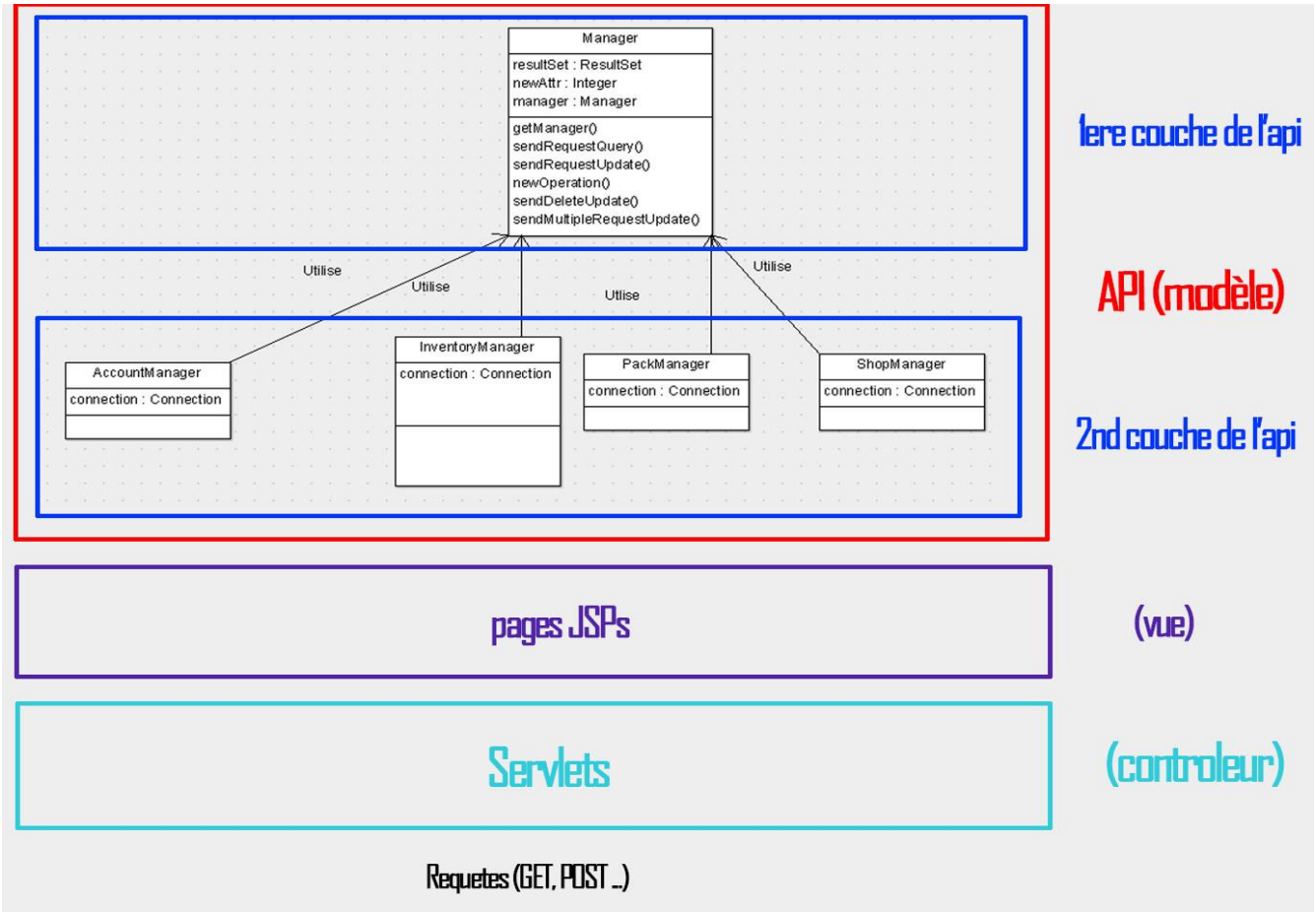
Afin de faciliter le développement et le partage du code nous avons opté pour une solution de type gestionnaire de version : git avec une solution en ligne : github.

Nous évoquerons d'abord la structure de notre projet (MVC), puis nous donnerons une liste exhaustive de la description des fonctionnalités et leurs implémentations, nous nous intéresserons ensuite à la visualisation du site en donnant une description de l'intérêt de chaque page et enfin nous terminerons par une conclusion sur notre projet et ses objectifs suivi par une courte ouverture sur les éventuelles évolution du site et du jeu.

# Project description

## 4. Structure du projet

L'architecture du projet suit une approche objet avec un pattern de type MVC (Modèle Vue Contrôleur). Voici un schéma de l'architecture du serveur :



### a) Modèle

Le modèle est essentiellement l'API mise en place pour communiquer et administrer notre base de données, elle se charge de faire tous les traitements métiers de l'application. Cette API se décompose en deux couches.

### b) 1ere couche

La première couche de l'API va utiliser la classe JDBC (*Java DataBase Connectivity*) qui permet de dialoguer avec la base de données distante. Cette couche va implémenter des fonctions très basiques qui vont permettre d'envoyer des requêtes et de récupérer leurs retours sans traiter ces informations. La classe Manager qui implémente la première couche est un singleton, en effet nous ne voulions pas permettre la création de plusieurs connexions à la base de données.

### c) 2nd couche

La deuxième couche de l'API va permettre de traiter les informations récupérées avec la première couche. En effet les informations données par la première couche doivent être interprété afin de pouvoir faire un rendu graphique par exemple. Les classes qui se charge d'implémenter ce type de traitement sont dans le package nommé « Manager ».

La classe « AccountManager » va se charger de faire tous les traitements liés aux comptes joueurs dans la base de données. Par exemple créer un compte ou récupérer un email. La classe « InventoryManager » va se charger d'effectuer les traitements liés aux possessions d'un joueur (son inventaire). Par exemple récupérer son deck et créer les objets de vue qui lui sont associé. La classe « PackManager » va se charger d'effectuer les traitements liés à la création et l'administration de pack dans notre boutique. La classe « ShopManager » va se charger d'effectuer les traitements liés à la boutique.

Toutes ces classes de la 2$^{nd}$ couche de l'API vont utiliser un objet de type Manager à chaque fois qu'elles aurons besoin de communiquer avec la base de données.

### d) Vues

Afin de représenter au mieux ce qui se trouve dans notre base de données, nous avons mis en place toute une série d'objet de vue qui représente un objet de la base de données. Tous les objets de vues se trouve dans le package « View ». Par exemple la classe Carte se charge de représenter une carte, ses attributs sont remplis avec des entrés de la base de données tel que : son ID, son image, son cout, son nom, sa description etc.

La vue est assez simple, elle se contente d'afficher les objets de vue construit à partir de l'API. Cet affichage se fait via une série de page JSP (Java Server Page) qui permet un rendu dynamique d'un page HTML.

### e) Contrôleurs

Les contrôleurs sont les servlets, les servlets se chargent de recevoir les requêtes http puis de rediriger vers les pages JSP demandées. On peut donc, à la réception d'une requête, examiner les attributs de celle-ci afin d'appliquer un contrôle sur ce que les utilisateurs nous envoient et de rediriger la requête vers une page d'erreur si nécessaire. Les servlets sont routées grâce au fichier web.xml se trouvant dans le dossier WEB-INF qui définit toutes les routes et les URLs en associant une URL à une servlet.

## 5. Fonctionnalités et implémentation

### a) Création d'un compte

- **Description** : les joueurs doivent pouvoir créer un compte à partir de l'application web en renseignant leur pseudo, leur mot de passe et leur email.

- **Implémentation** : une page JSP permet à l'utilisateur de d'entrer ses identifiants, ceux-ci sont envoyé à la servlet « CreateAccountServletNavigator » qui se charge de vérifier les données saisies par l'utilisateur (par exemple elle vérifie que le pseudo n'existe pas déjà). Cette servlet appelle la fonction « createAccount() » si les données saisies sont conforme à ce qui est attendu, le cas échéant elle redirige la requête vers une page d'erreur.

### b) Gestion d'un compte

- **Description** : les joueurs doivent pouvoir administrer leur compte depuis l'application web, par exemple pouvoir modifier leur mot de passe ou leur adresse email.
- **Implémentation** : une page JSP permet à l'utilisateur de modifier les informations de son compte après s'être authentifié, des formulaires sont présent sur la page afin de permettre à l'utilisateur de rentrer ses nouvelles informations. Les données de ces formulaires sont envoyées à la servlet « AccountUpdaterServlet » qui se charge de contrôler les informations reçues. Si les informations sont conformes la servlet appelle la fonction de l'API correspondante au traitement demandé sinon elle redirige la requête vers une page d'erreur.

### c) Authentification d'un joueur

- **Description** : les joueurs doivent pouvoir se connecter avec leurs comptes afin d'effectuer diverse action d'administration dessus.
- **Implémentation** : un formulaire de connexion est mis à disposition des utilisateurs ce qui leur permet de renseigner leur pseudo ainsi que leur mot de passe, ensuite ces informations sont transmises à la servlet « AuthentificationAccountServlet » qui va se charger d'appeler la fonction « authentification » (du package Manager). Cette

fonction retourne vrai si l'utilisateur a renseigné le bon mot de passe et le bon pseudo, si c'est le cas la servlet va mettre les informations du client en variable de session, si ce n'est pas le cas la servlet va rediriger la requête vers une page d'erreur.

### d) Gestion des decks

- **Description** : les joueurs possèdent deux decks, l'application web doit leur fournir un moyen de modifier leur decks en fonction des cartes qu'ils possèdent.
- **Implémentation** : une page d'administration des decks est mise à la disposition de l'utilisateur, elle est disponible une fois que celui-ci s'est identifié. Cette page propose une interface de gestion très intuitive, sur le haut de la page se trouve le deck actuel du joueur et en dessous se trouve toutes les cartes qu'il possède. La constitution du deck se fait sous la forme d'échange de carte entre le deck actuel et l'inventaire, une fois l'échange réalisé un script sur la page client se charge de récupéré les informations de cet échange et d'envoyer les données à une servlet via une requête de type AJAX (afin de ne pas faire recharger la page à chaque modification du deck). Une fois que la servlet a reçu les informations sur la modification du deck, cette servlet fait appel à une fonction de l'API afin de modifier le deck du joueur dans la base de données.

### e) Déconnexion d'un joueur

- **Description :** un joueur doit pouvoir se déconnecter, afin de pouvoir potentiellement se reconnecter avec un autre compte par exemple.
- **Implémentation :** une fois qu'un utilisateur est connecté, un bouton de déconnexion est disponible sur la barre de navigation. Ce bouton envoie une requête à une servlet qui se charge tous simplement de supprimer les informations du joueur qui en fait la demande des variables de sessions. Cet action va en effet déconnecter le client, il pourra donc se reconnecter avec un autre compte si il le désire.

### f) Achat en Boutique

- **Description :** les joueurs doivent pouvoir utiliser de la monnaie virtuelle pour effectuer diverses transactions, notamment l'achat d'offres proposées par les administrateurs.
- **Implémentation :**

  Une page JSP, générée à partir d'une vue, créée par le serveur en fonction du pseudo de l'utilisateur, et d'un traitement JavaScript associé, permet à l'utilisateur de visualiser les différentes offres à sa disposition.

  Chacune des offres peut être sélectionnées, affichant une description de l'offre, et proposant deux options d'achat, en fonction de la monnaie à utiliser. Un dernier prompt de vérification permet de s'assurer que l'achat est intentionnel.

Dès lors, la fonction JS d'achat envoie une requête POST en AJAX au serveur, récupérée par la servlet de contrôle ShopAchatServlet, qui lance l'exécution de l'achat par la fonction doAchat du modèle ShopManager.

La fonction récupère l'offre, vérifie que le joueur a assez de monnaie pour effectuer la transaction, récupère le type d'offre (Pack/Boost/etc), puis exécute les traitements relatifs au type :

- Si l'offre est un Pack, ouvre le Pack, récupère aléatoirement les cartes selon la structure du Pack, puis les ajoute à l'inventaire de l'utilisateur.
- Si l'offre est un Boost, vérifie qu'il n'existe pas déjà un boost du même type activé, auquel cas ajoute la durée du boost acheté à l'actif, sinon ajoute un nouveau Boost.
- Si l'offre est un Skin ou une Icone, l'ajoute à l'inventaire de l'utilisateur. (L'utilisateur ne peut pas acheter de Skin ou d'icone qu'il possède déjà).

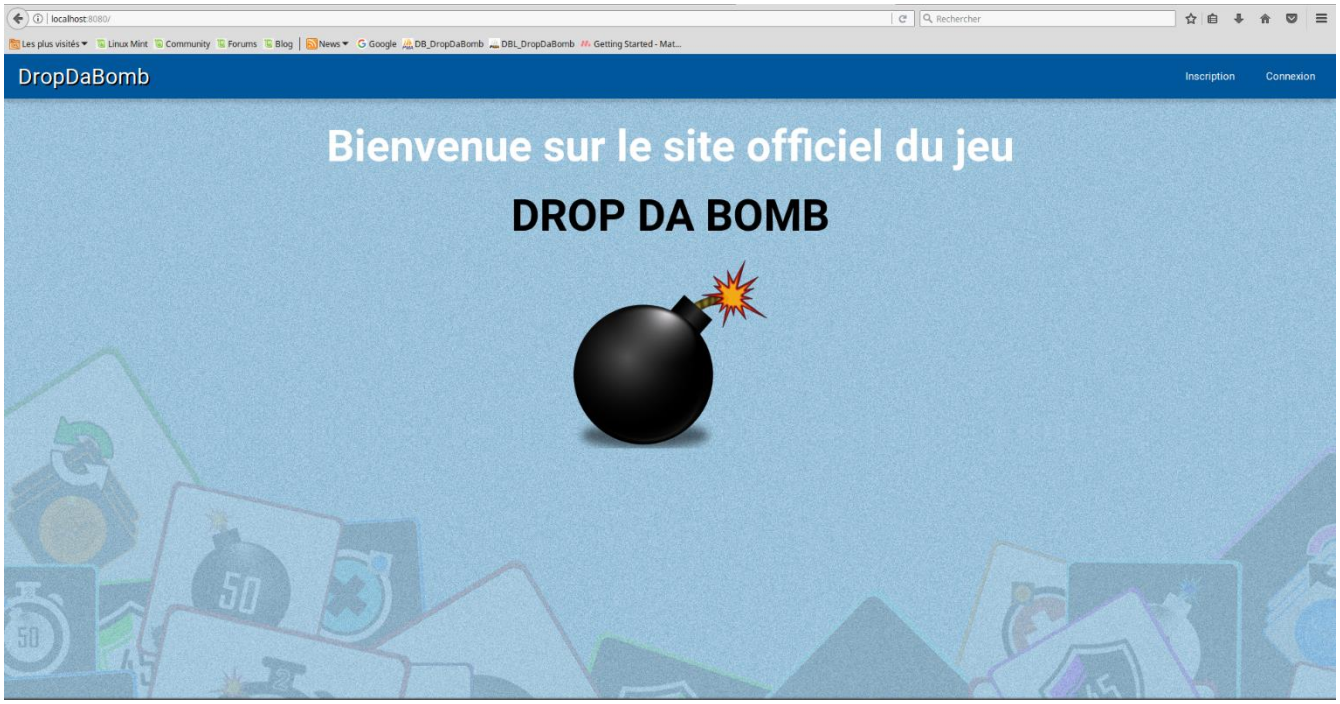
Si l'exécution se termine correctement, doAchat retire une quantité de monnaie correspondante au prix à la besace du joueur, puis renvoie un JSON de ce que l'achat a généré au contrôleur, qui le renvoie à la JSP qui l'affichera à l'utilisateur.

### g) Page d'administration : gestion des Packs :

- **Description :** les administrateurs doivent pouvoir gérer les Packs déjà crées, et donc les modifier dans leur intégralité, en créer de nouveau, et définir quels sont les Packs mis en vente ou non.
- **Implémentation :** une page JSP, accessible uniquement par les administrateurs, est disponible afin de gérer les Packs et toutes leurs sous-couches. Celle-ci récupère tous les Packs et les LootPacks et Ensembles, attachés à un Pack ou non, présent en base de données. Toutes les opérations suivent le même schéma :
    - L'administrateur effectue une opération de gestion sur la page JSP ;
    - Une requête POST en AJAX est envoyée au contrôleur AdminServlet, constituée d'une idRequest correspondant à l'opération, ainsi que des informations complémentaires nécessaires ;
    - Le contrôleur récupère la requête et détermine quelle partie de la structure, représentée par la vue AdminPackView, est affectée, afin d'appeler à la modification de la vue ;
    - La vue récupère la demande, fait appel au modèle PackManager pour effectuer les traitements et envoyer les requêtes à la base de données lorsque nécessaire, puis modifie la vue en fonction de la demande de l'utilisateur ;
    - Le contrôleur retourne un JSON de la partie de la vue potentiellement modifiée par les traitements à la JSP ;
    - La fonction de callback de la requête AJAX récupère le JSON puis regénère la partie de la page modifiée.

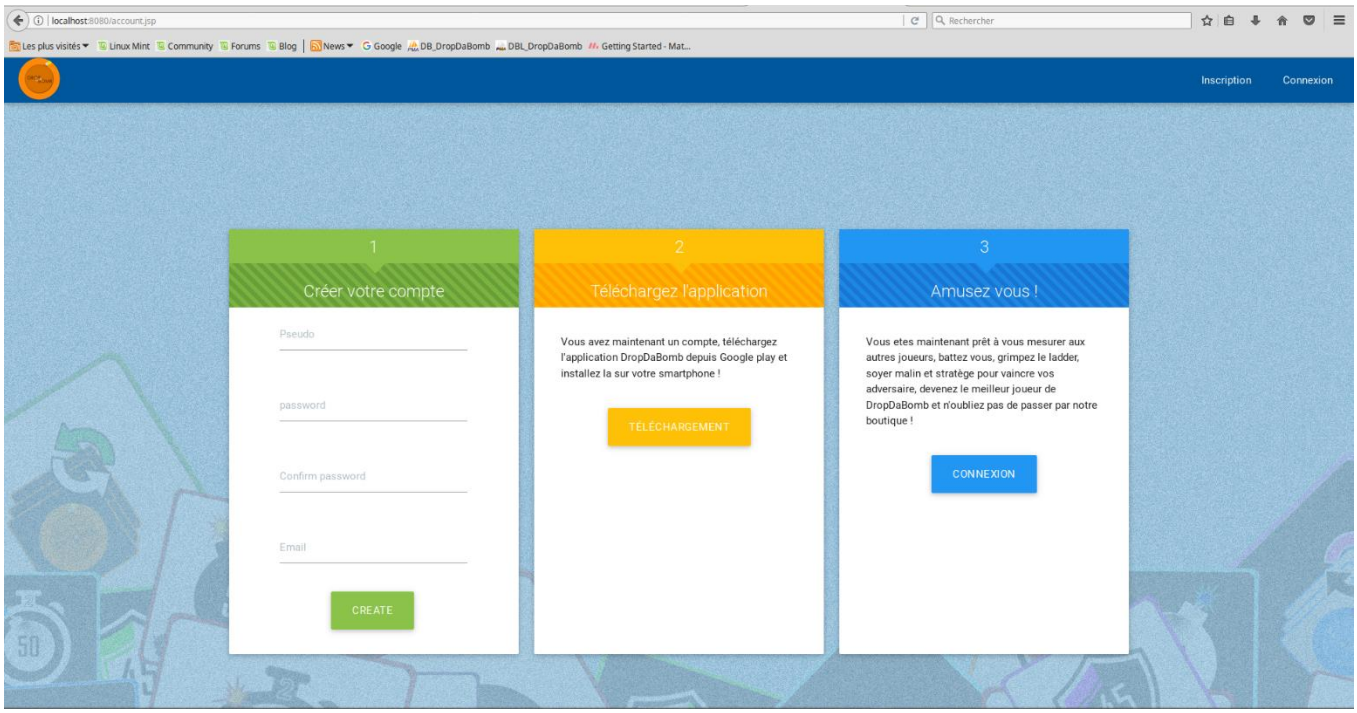
# Visualisation du site :

## a) Index :




L'arrivée sur le site se fait sur cette page, l'index. Comme l'utilisateur n'est pas connecté, la barre de navigation lui propose de s'inscrire ou de se connecter. Si l'utilisateur ne s'est jamais inscrit, il peut pour cela cliquer sur le bouton d'Inscription et être redirectionné vers la page de création de compte.

## b) Page de création de compte :

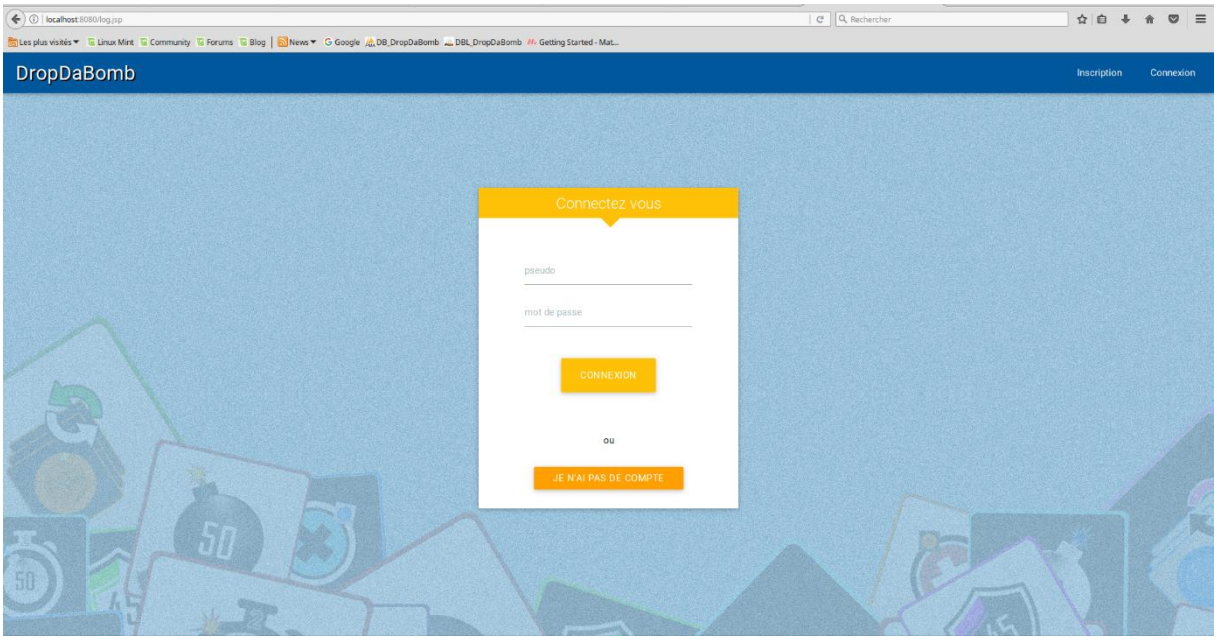


L'utilisateur se trouve désormais sur la page de création de compte. Il est invité à saisir ses informations de connexion et son adresse email. Si son email n'est pas conforme, ou que les deux mots de passe sont différents, une erreur s'affiche en rouge. Sinon, le compte est créé, et l'utilisateur peut alors se connecter en cliquant sur le bouton Connexion, ou le bouton de la barre de navigation.

(NB : Le Téléchargement est présent en prévision d'une publication prochaine de l'application)

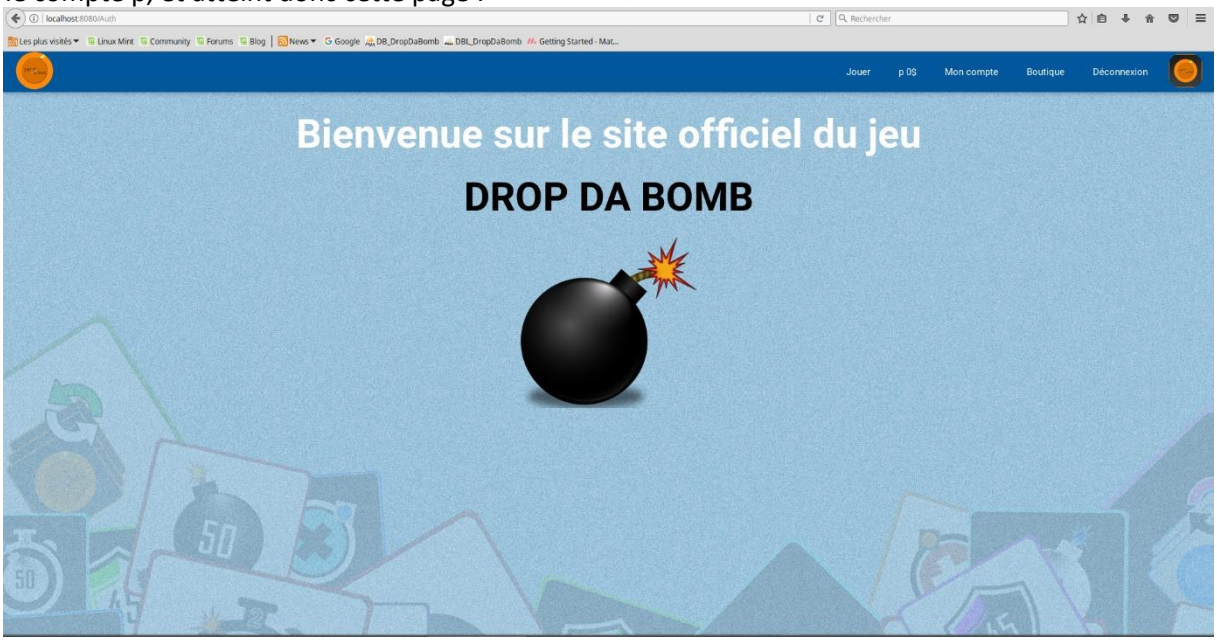
### c) Page de connexion :

L'utilisateur est alors sur la page de connexion :




Il est invité à rentrer ses identifiants de connexion, puis à cliquer sur Connexion. S'il n'a pas encore créé de compte, il peut le préciser et être renvoyé vers la page de création de compte.

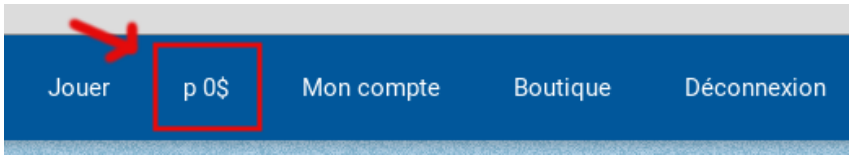
Si les identifiants sont bons, l'utilisateur est alors connecté. Ici, l'utilisateur s'est connecté sur le compte p, et atteint donc cette page :



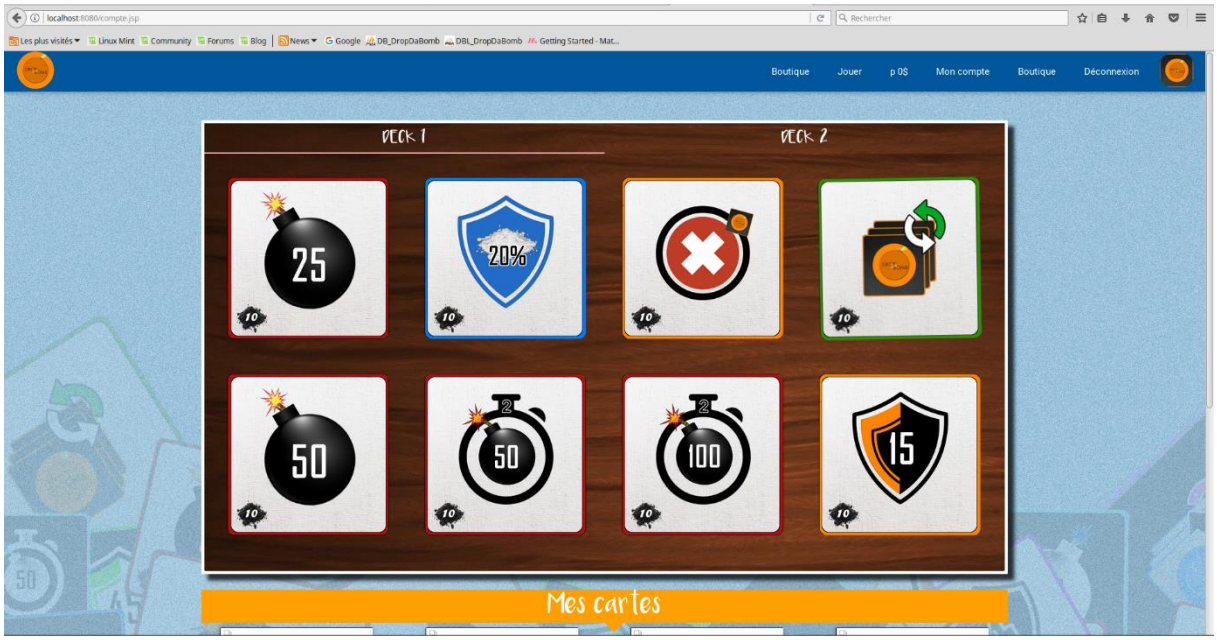
La barre de navigation est alors modifiée, et l'utilisateur a accès à d'autres sections. A savoir que si l'utilisateur se connecte sur un compte Administrateur, il a accès à une section de plus, sa barre de navigation ayant alors pour forme :


d)

e) Inventaire :

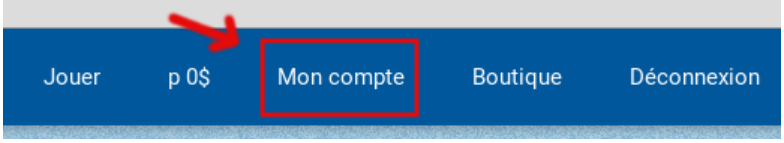



La première section est l'inventaire de l'utilisateur : dès lors que l'utilisateur clique sur le bouton représenté par son pseudo et la quantité de monnaieIG qu'il possède, il est redirectionné vers son inventaire :

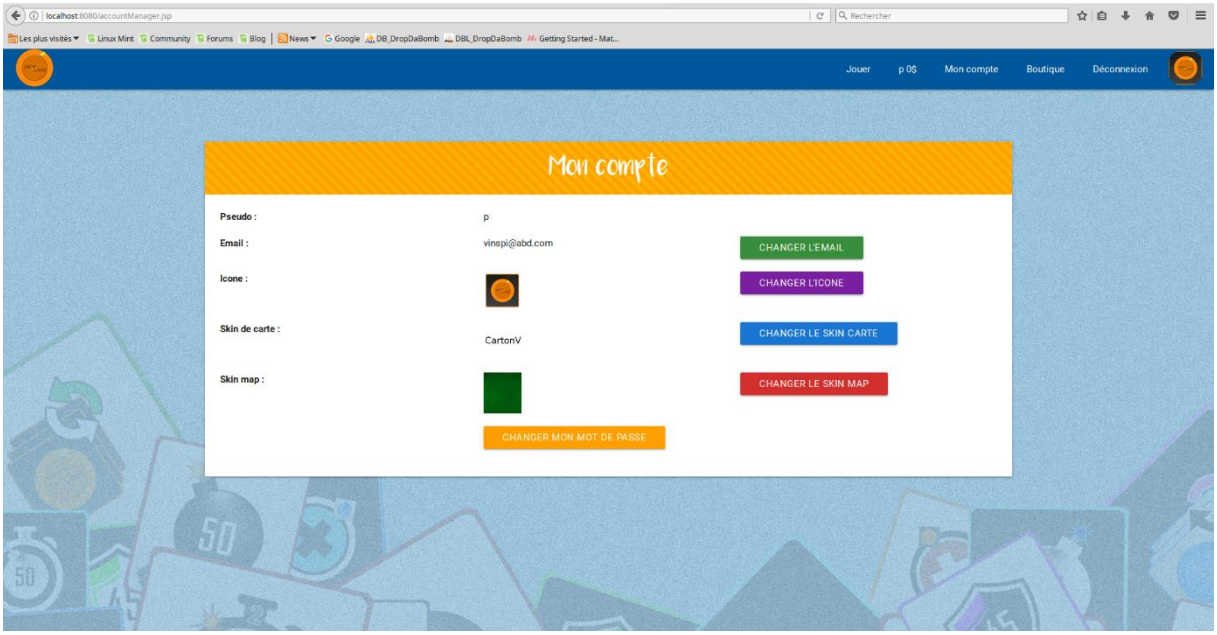



Dans celui-ci, l'utilisateur peut composer ses Decks et choisir son deck actif. De même, il peut visualiser les cartes qu'il possède.

## f) Gestion de compte :



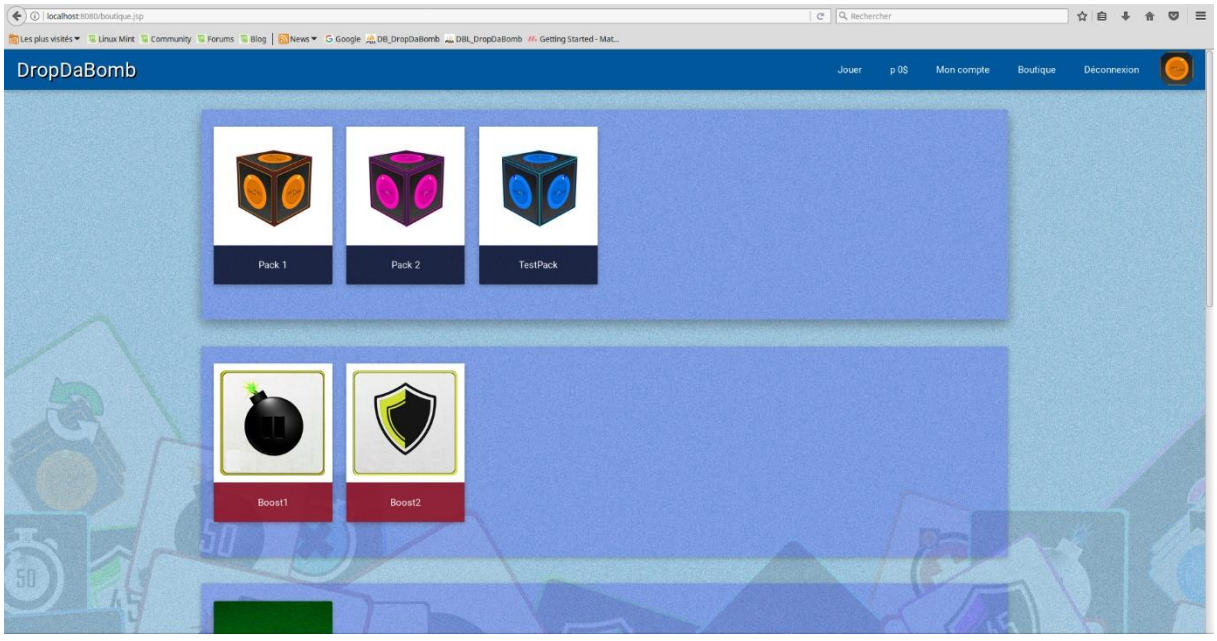

La deuxième section est la gestion du compte de l'utilisateur, accessible via le bouton « Mon compte ».




Elle permet à l'utilisateur de modifier son email, son icone, son skin de carte, son skin map et son mot de passe.

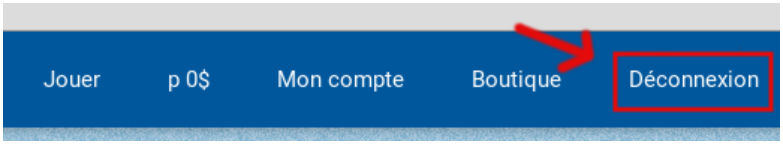
## g) Boutique :


La troisième section est la Boutique :



Elle permet de d'accéder aux différentes offres disponibles à la vente, et de les acheter. Les offres sont classées par type : les Packs, les Boosts, les Icones, etc. Sélectionner une image permet d'afficher un descriptif et d'accéder aux boutons d'achats, indiquant ainsi les prix.

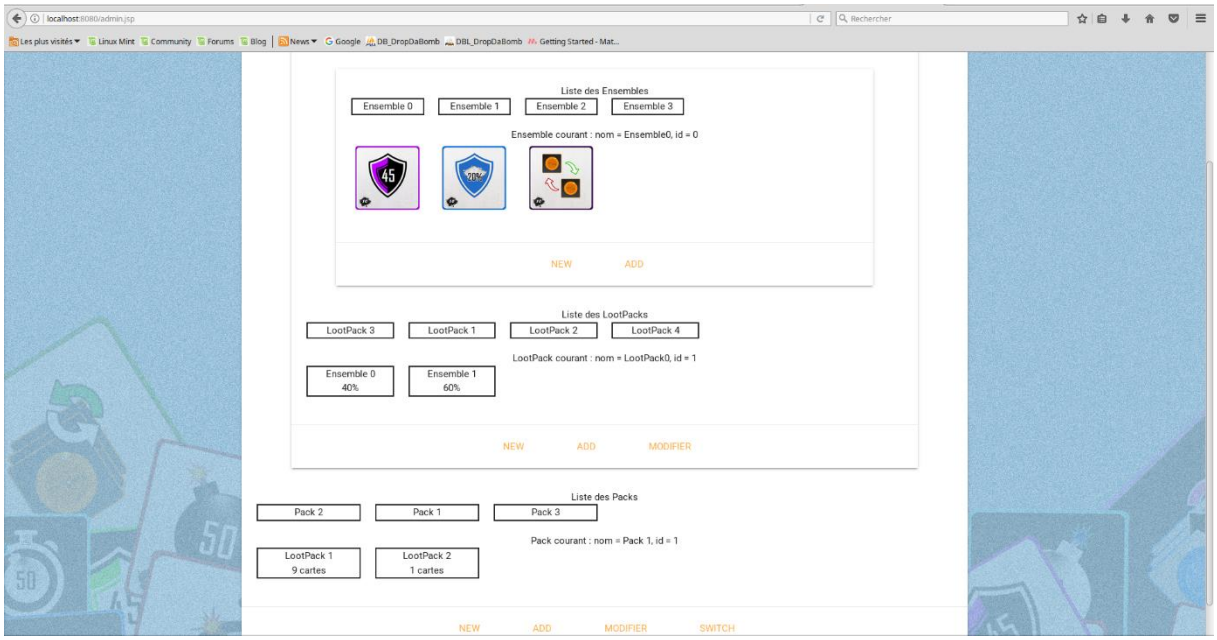
### h) Déconnexion :




Le bouton Déconnexion permet de se déconnecter : il déconnecte l'utilisateur, puis le redirige vers l'index.

### i) Panneau d'administration :

Cette section, réservée aux utilisateurs Administrateurs permet, pour le moment, d'administrer les Packs présents en base de données, et d'en créer de nouveau :




Le panneau est décomposé en 3 parties : les Ensembles, inclus dans les LootPacks, inclus dans les Packs. Dans chacune des parties, l'utilisateur a accès à la liste des éléments inclus en base de données, ainsi que la composition de l'élément courant, c'est-à-dire choisi en dernier. Il est possible de modifier leurs structures en utilisant les différents boutons disponibles (ajout, modification, création).

# Conclusion

Pour conclure sur ce projet, nous pouvons dire que cela a été une expérience très enrichissante, en effet nous avons pu travailler avec des technologies qui nous étaient encore inconnues, nous pensons notamment à JAVA EE et Tomcat, ce qui nous a donc permis de les découvrir et d'acquérir les bases de fonctionnement de ces technologies, ce qui enrichit donc notre expérience.

Ce projet nous a également poussé à acquérir plus d'expérience dans les langages déjà connus, nous avons fait énormément de progrès avec le langage javascript et les technologies qui le compose comme AJAX par exemple.

Ce projet a aussi été l'occasion de mettre en pratique nos connaissances en conception de base de données et en architecture logicielle avec l'approche objet que nous avons mis en place pour réaliser notre server d'application.

Et enfin, cette application a été l'occasion pour nous d'apprendre à gérer un projet conséquent avec une équipe. Nous avons pu apprendre des méthodes de gestion de projet qui mettent en avant l'équipe tout en respectant une chronologie efficace dans la réalisation des tâches. Nous avons pu réaliser que la mise en place d'un projet informatique ne se réalise pas seulement sur le plan technique, il faut également mettre en place un cahier des charges étudié ainsi qu'une planification correcte afin de pouvoir tenir les délais.

Après avoir travaillé avec la technologie JAVA EE couplé à git, nous avons eu énormément de problèmes liés à l'association de IntelliJ (notre éditeur de code) et git. Le fait qu'Intellij se repose sur énormément de fichiers de configurations qu'il ne faut en aucun cas partager sur le git (car les autres développeurs n'ont probablement pas la même architecture de dossiers), a été problématique dans le sens ou git a réécrit à plusieurs reprises des fichiers de configuration de l'éditeur ce qui nous a obligé à reprendre des sauvegardes et ce qui nous a fait perdre beaucoup de temps. Ces problèmes sont sûrement liés à notre méconnaissance de d'Intellij et de git. Cela dit nous souhaiterions tout de même, pour l'avenir de notre projet, changer de technologie et travailler avec une nouvelle technologie émergente qui permet de répondre à nos besoins : Node.js.

## Perspective d'évolution

A l'avenir, nous aimerions continuer le développement de notre application avec Node.js et non JAVA EE car c'est une technologie qui nous correspond plus. Bien évidemment, la prochaine étape dans le développement de l'application est le développement d'une interface de jeu jouable, Nous avons déjà mis au point un prototype de jeu fonctionnel et multijoueur qui se sert de la base de données mise en place précédemment. Il reste néanmoins beaucoup de travail à fournir afin d'obtenir un résultat correct, par exemple, nous aimerions ajouter des fonctionnalités au site web : gestion d'une liste d'amis, service de messagerie instantanée, accès à l'interface de jeu via le site web, ajout de nouvelles cartes, équilibrage du jeu, amélioration du panneau d'administration etc.