

# OverwatchCollection V.1 : La documentation technique

---

*Sven Wikberg*

## Contenu

1	Introduction.....	3
1.1	Objectif.....	3
2	Analyse fonctionnelle .....	4
2.1	Généralités.....	4
2.1.1	Pourquoi ce sujet ? .....	4
2.1.2	Ce que ce projet à de plus !.....	4
2.2	Cahier des charges détaillé.....	4
2.2.1	Définition de l'audience .....	4
2.2.2	Définition du contenu et des fonctionnalités .....	4
2.2.3	Maquette préliminaire .....	6
2.3	Planning initial .....	7
3	Analyse organique .....	8
3.1	Analyse de l'environnement.....	8
3.2	Arborescence du site .....	8
3.3	Organigramme .....	9
3.4	Maquette Graphique .....	9
3.4.1	Page d'accueil.....	9
3.4.2	Page des héros .....	10
3.4.3	Page d'un héros.....	10
3.4.4	Page des évènements.....	11
3.4.5	Page d'un évènement.....	11
3.4.6	Page des autres objets.....	12
3.4.7	Page de connexion/inscription.....	12
3.4.8	Page du compte .....	13
3.4.9	Page administrateur.....	13
3.5	Modèle relationnel.....	14
3.6	Description des tables.....	14
3.6.1	Table users .....	14
3.6.2	Table reward_types.....	15
3.6.3	Table heroes.....	15
3.6.4	Table user_rewards .....	15
3.6.5	Table rewards.....	15
3.6.6	Table abilities .....	16

3.6.7	Table roles.....	16
3.6.8	Table currencies .....	16
3.6.9	Table qualities .....	16
3.6.10	Table events.....	16
3.7	Utilisation et explication des requêtes.....	17
3.7.1	Accueil.....	17
3.7.2	Les héros .....	17
3.7.3	Un héros .....	17
3.7.4	Les événements .....	17
3.7.5	Un événement.....	17
3.7.6	Les autres objets .....	18
3.7.7	Affichage d'objet héros/événement/autre objets .....	18
3.7.8	Connexion .....	18
3.7.9	Identification .....	18
3.7.10	Modification du compte .....	18
3.7.11	Mon compte, info et statistiques .....	18
3.7.12	Actions admin .....	19
3.8	Description détaillée de l'architecture du code.....	20
3.8.1	Séparation du code .....	20
3.8.2	Les classes.....	20
3.8.3	Les actions en GET .....	30
4	Tests.....	32
5	Conclusion .....	34
5.1	Problèmes rencontrées .....	34
5.2	Améliorations envisageables .....	34
5.3	Comparaison analyse et réalisation finale .....	34
5.4	Comparaison planning initial et final .....	34
5.5	Bilan personnel .....	35
6	Webographie.....	36
7	Table des illustrations.....	36

# OverwatchCollection :

## Documentation technique

---

### 1 Introduction

Le projet OverwatchCollection est basé sur le jeu Overwatch, Overwatch est un jeu de tir, très réactif, à la première personne, aux graphismes colorés et un peu cartoonés.

Dans Overwatch lorsqu'on commence une partie, il faut d'abord choisir un héros parmi les 23 disponibles, les héros ont des capacités et des statistiques différentes. Ils sont regroupés par rôles, les rôles définissent à quoi le héros sert dans l'équipe. Par exemple les « tanks » sont là pour faire en sorte de prendre les dégâts à la place des autres héros, car ils sont plus résistants.

Donc dans ce jeu on peut recevoir des objets cosmétiques à diverses occasions, ces objets permettent de customiser les héros. Par exemple il est possible de recevoir un skin, qui permet d'avoir un modèle différent de celui de base pour son héros (voir figure 1 et 2) ou alors on peut recevoir une phrase spéciale que le héros pourra dire pendant le jeu.



Figure 1, soldat 76 skin de base



Figure 2, soldat 76 skin d'halloween

#### 1.1 Objectif

Le but du projet OverwatchCollection est d'avoir une sorte d'inventaire de ses objets, car dans le jeu, la façon dont les objets sont ordonnés n'est pas pratique. Il permet donc de voir tous les objets triés de plusieurs façons différentes et de sélectionner ceux que l'on a obtenus dans le jeu afin de les reconnaître plus facilement ainsi que d'avoir les statistiques sur ceux-ci.

## 2 Analyse fonctionnelle

### 2.1 Généralités

#### 2.1.1 Pourquoi ce sujet ?

Ce sujet vaut la peine de se pencher dessus car Overwatch est un jeu très populaire et donc il y a une très grande communauté de joueurs autour et les joueurs sont en général très friands d'objets rares et exclusifs afin de les montrer aux autres joueurs. C'est donc normal que les objets cosmétiques dans Overwatch aient beaucoup de succès, surtout avec les événements spéciaux et la sortie de contenu régulière de la part des développeurs.

Mais le problème c'est qu'il y a vraiment beaucoup d'objets différents et dans le jeu, il n'y a pas de façon simple de voir où on en est avec tous ces objets, c'est là qu'Overwatch Collection entre en jeu. Il va pouvoir régler ce problème afin de rendre la vie des joueurs plus facile.

#### 2.1.2 Ce que ce projet a de plus !

Pour l'instant il n'y a pas vraiment de façon de faire un inventaire des objets existants ainsi que des objets obtenus dans le jeu, à part avec un papier et un crayon mais au niveau pratique ce n'est pas le moyen le plus avancé. C'est exactement l'utilité de ce projet, il permet de faire un inventaire de ses objets parmi tous les objets existant dans le jeu.

Il existe déjà des sites qui répertorie certaines informations du jeu, comme par exemple « <http://overwatch.gamepedia.com> » qui contient entre autres une liste des héros avec plein d'informations intéressantes sur chacun d'entre eux ou une liste des skins des héros, mais pas qui permettent d'avoir une liste complète des objets triés, un suivi, des statistiques détaillées par événement ou par héros sur ceux-ci, comme le propose ce projet.

### 2.2 Cahier des charges détaillé

#### 2.2.1 Définition de l'audience

OverwatchCollection est un projet qui a évidemment comme cibles les joueurs du jeu Overwatch, et plus particulièrement les joueurs collectionneurs, c'est-à-dire les joueurs aiment bien avoir un inventaire de ce qu'ils possèdent et surtout ce qu'ils n'ont pas encore.

#### 2.2.2 Définition du contenu et des fonctionnalités

Les fonctionnalités sont regroupées en quatre parties, premièrement le groupe de fonctionnalité accessible par les trois prochaines parties. Deuxièmement la partie « invité », c'est la partie qui contient les fonctionnalités accessibles en tant qu'invité. Ensuite la partie « utilisateur », c'est la partie contient les fonctionnalités accessibles lorsqu'on a créé un compte et que l'on est connecté sur ce compte. Et finalement la partie « administrateur » qui regroupe les fonctionnalités accessibles en tant qu'administrateur.

#### *Toutes catégories*

- affichage des utilisateurs du site
- affichage des objets cosmétiques d'Overwatch par héros
- affichage des objets cosmétiques d'Overwatch par événement
- affichage des objets cosmétiques d'Overwatch qui ne sont liés à aucun héros

- affichage des informations de base des héros
- affichage des informations de base des événements

### *Invité*

- création un compte
- identification

### *Utilisateur*

- possibilité de cliquer sur n'importe quel objet cosmétique non sélectionné afin de le sélectionner
- possibilité de cliquer sur n'importe quel objet cosmétique sélectionné afin de le désélectionner
- modification des informations de son compte
- affichage des objets sélectionné différent afin de les reconnaître
- affichage de statistiques sur les objets sélectionnés

### *Administrateur*

- bannissement d'un utilisateur
- dé- bannissement d'un utilisateur
- suppression d'un compte
- affichage des utilisateurs bannis
- affichage des utilisateurs non bannis

### 2.2.3 Maquette préliminaire

Voici les maquettes qui ont permis de se faire une idée de la forme ainsi que le concept général du site web :

#### Page d'accueil

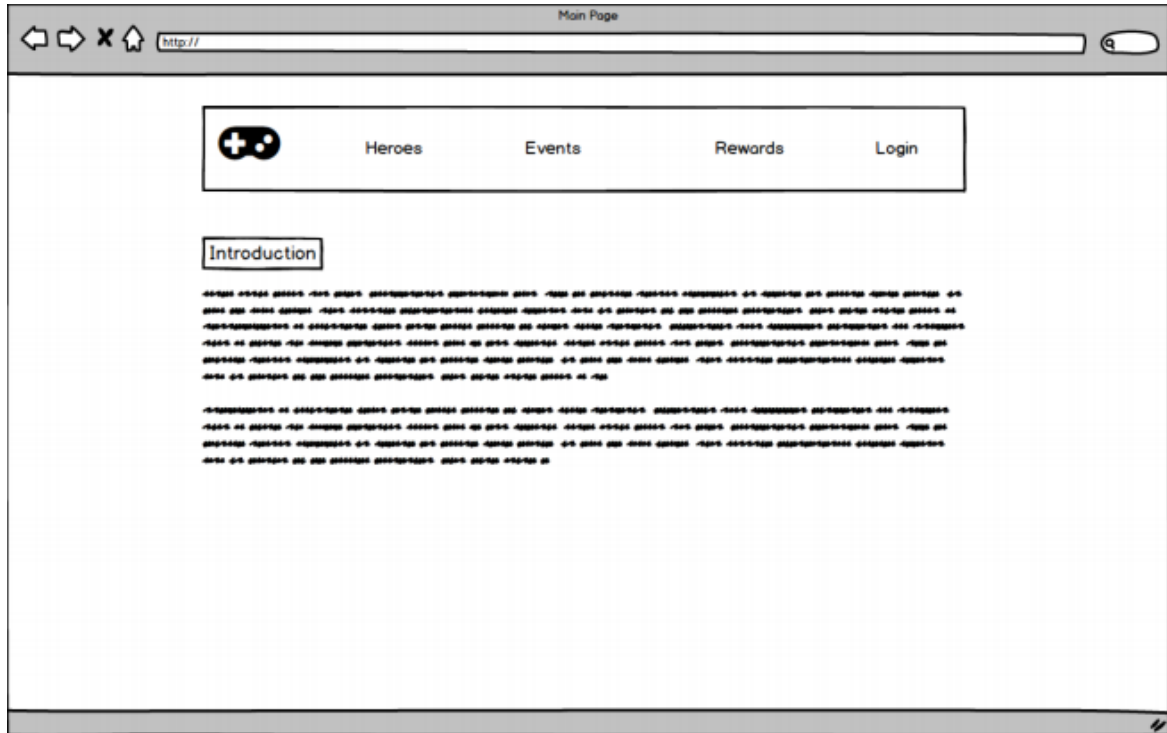


Figure 3, maquette préliminaire page d'accueil

#### Page des héros



Figure 4, maquette préliminaire page des héros



## Page d'un héros

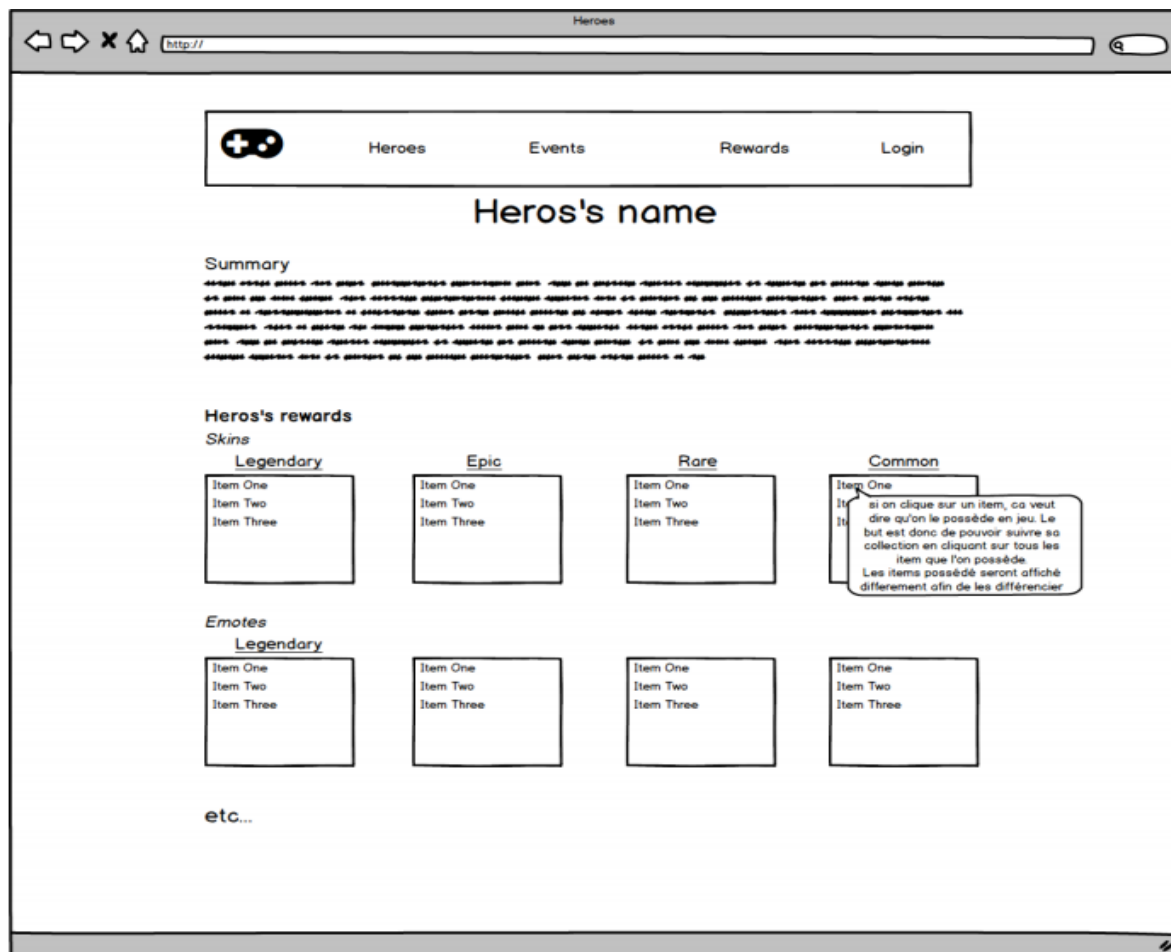


Figure 5, maquette préliminaire page d'un héros

## 2.3 Planning initial

Voici le planning initial proposé par mon enseignant, c'est un planning très simple mais quand même très réaliste au niveau des jours de développement sans pour autant oublier la documentation.

Inventaire des étapes du projet										
Phases de réalisation, avec estimation du temps en heures de 60 minutes										
Le total des heures correspond à 80 heures (10 x 8h00)										
	Jour 1	Jour 2	Jour 3	Jour 4	Jour 5	Jour 6	Jour 7	Jour 8	Jour 9	Jour 10
Conception BDD	X	X								
Programmation			X	X	X	X	X			
Documentation	X	X	X	X	X	X	X			
Journal de bord	X	X	X	X	X	X	X			
Finalisation documentation								X	X	X

Figure 6, planning initial



### 3 Analyse organique

#### 3.1 Analyse de l'environnement

Ce projet a principalement été réalisé avec le logiciel « Visual Studio Code », en tout cas pour tout le HTML, le PHP ainsi que le CSS. Ce logiciel a été utilisé principalement pour des raisons d'habitude, depuis plus d'un an je ne fais mes projets web qu'avec ça.

Au niveau du serveur Apache, PHP ainsi que MySQL c'est le logiciel « EasyPHP » qui a été utilisé, pour les mêmes raisons d'habitude, la base de données a été gérée avec « phpMyAdmin ».

En ce qui concerne les sauvegardes du projet, c'est « GitHub Desktop » qui a été utilisé, car c'est vraiment facile à utiliser lorsqu'on fait un projet tout seul et il s'occupe de faire les versions de façon très claires et pratiques.

Pour la documentation j'ai utilisé l'application web « draw.io » pour le modèle relationnel ainsi que l'arborescence du site et « balsamiq mockup 3 » pour les maquettes graphiques. Et « Microsoft Word 2010 » pour la doc technique ainsi que le mode d'emploi.

#### 3.2 Arborescence du site

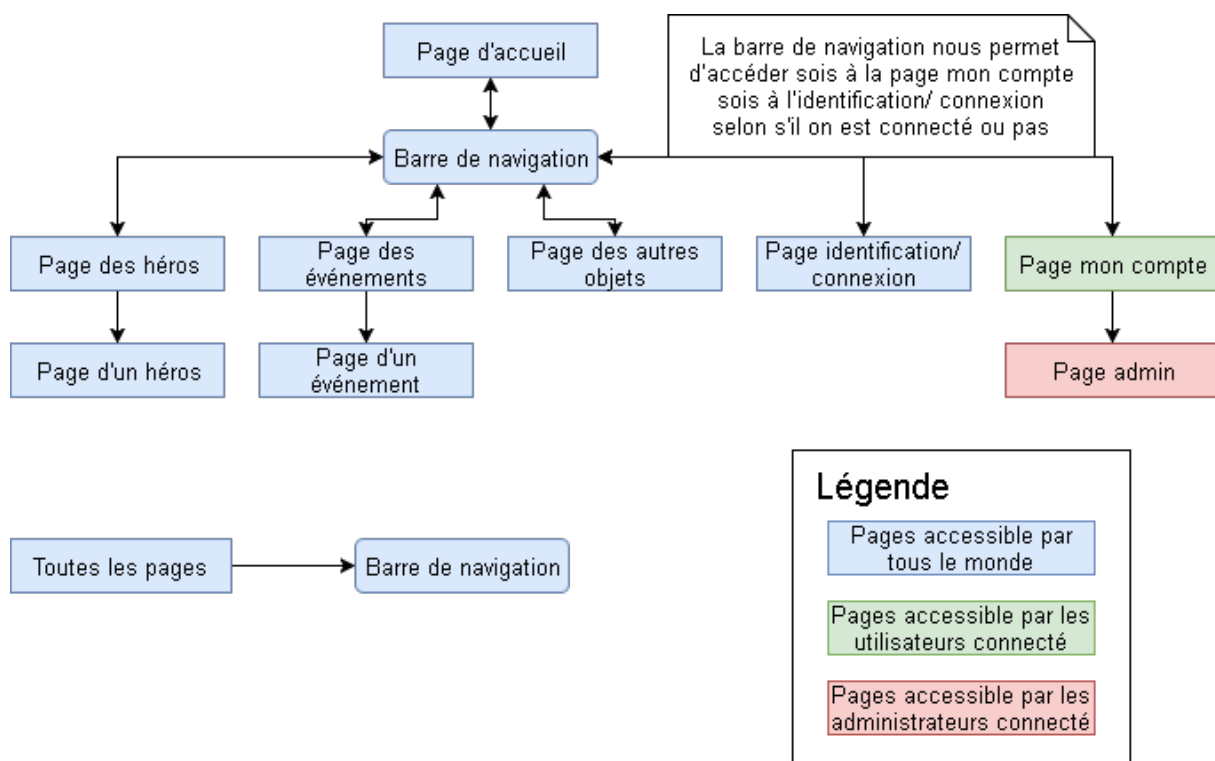


Figure 7, arborescence du site web

### 3.3 Organigramme

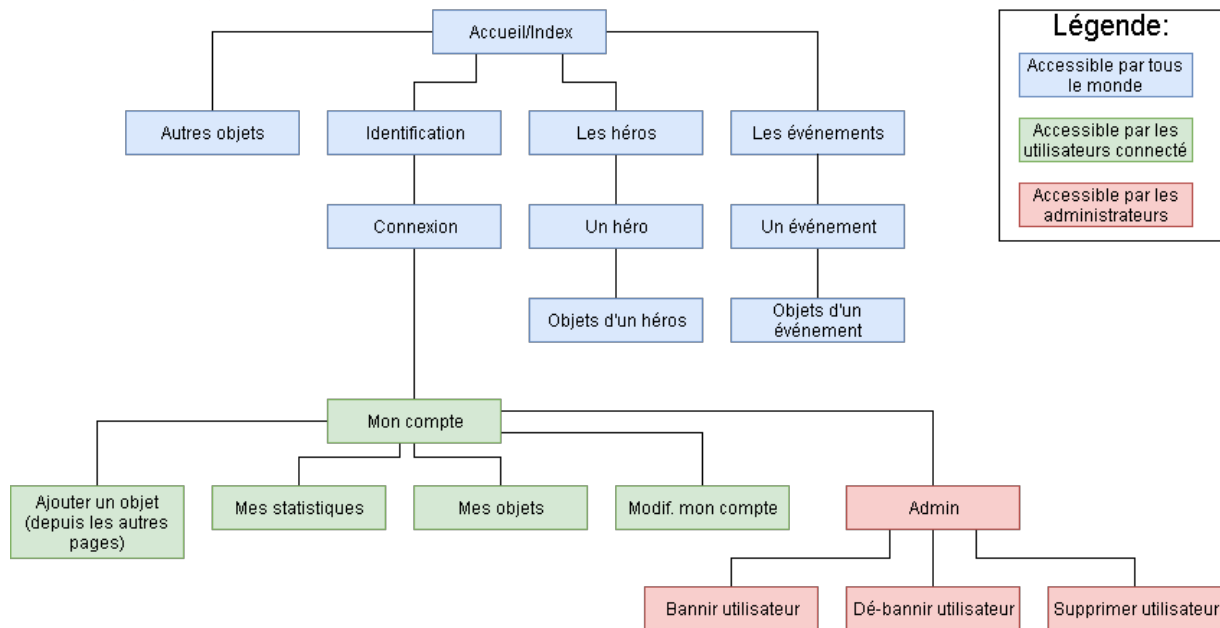


Figure 8, organigramme du site web

On peut donc remarquer dans ce schéma que les fonctionnalités sont bien réparties en trois parties distinctes, comme expliqué dans la légende.

### 3.4 Maquette Graphique

#### 3.4.1 Page d'accueil

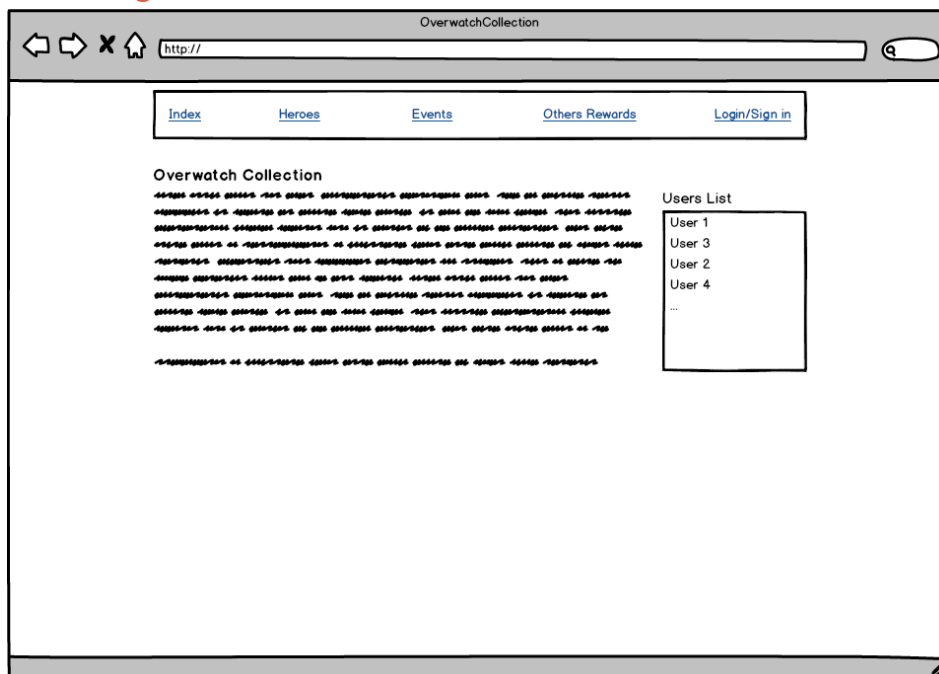


Figure 9, maquette page d'accueil

C'est la page de base, elle permet d'expliquer l'utilité du site web ainsi que sa manière de fonctionner. Elle contient également une liste des utilisateurs ayant un compte sur le site.

### 3.4.2 Page des héros

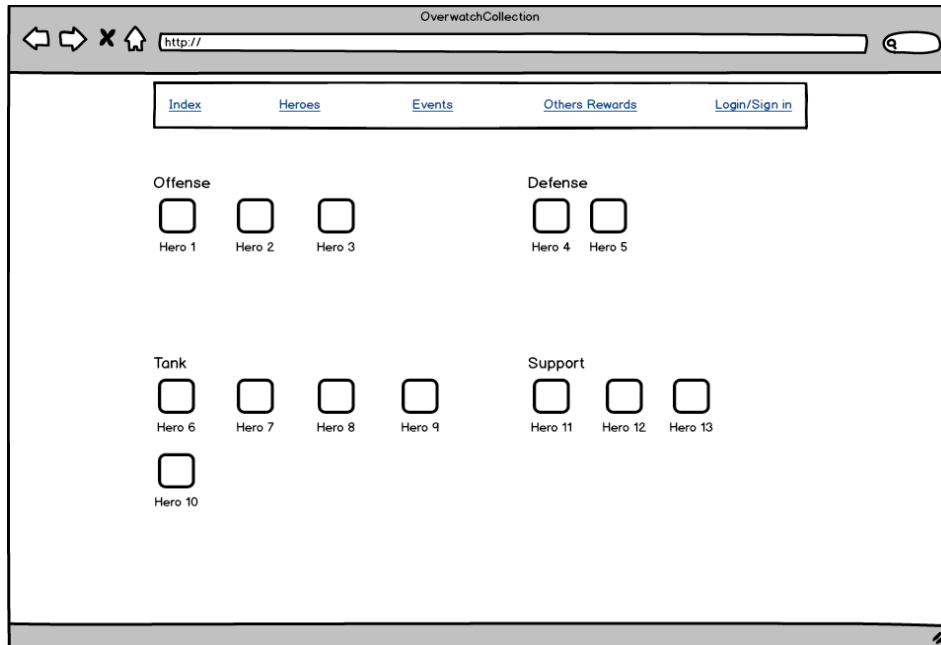


Figure 10, maquette page des héros

C'est la page qui contient un tableau des héros du jeu, regroupé par rôle afin de les trouver plus facilement, lorsqu'on clique sur un héros on est redirigé vers sa page respective.

### 3.4.3 Page d'un héros

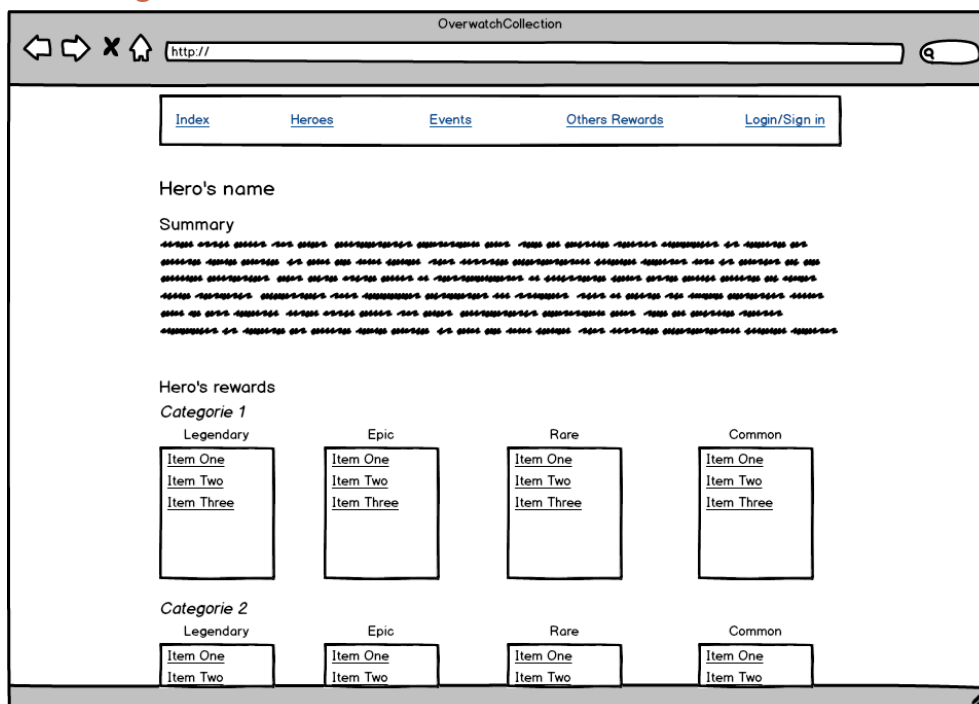


Figure 11, maquette page d'un héros

Cette page affiche quelques informations sur le héros en question ainsi que ses capacités, mais le plus important étant évidemment l'affichage des objets cosmétiques triés par catégorie et par

qualité afin de les trouver plus facilement. Si l'utilisateur est connecté les objets seront affichés comme des liens sur lesquels il pourra cliquer afin de marquer l'objet comme acquis.

### 3.4.4 Page des évènements

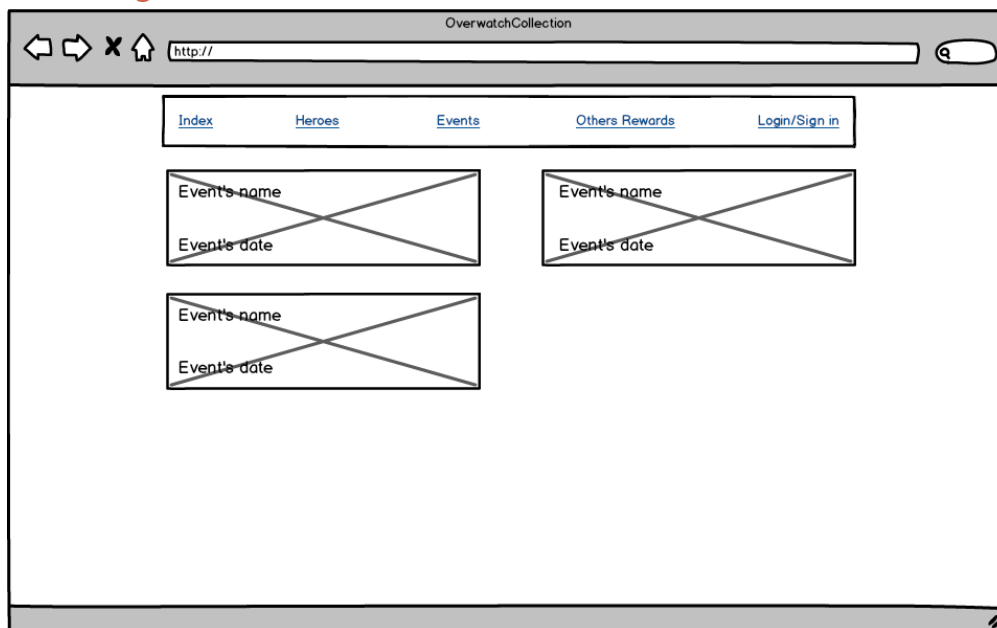


Figure 12, maquette page des événements

Cette page contient la liste des événements du jeu dans l'ordre de sortie de ceux-ci, en cliquant sur un événement un arrive sur la page de celui-ci.

### 3.4.5 Page d'un événement

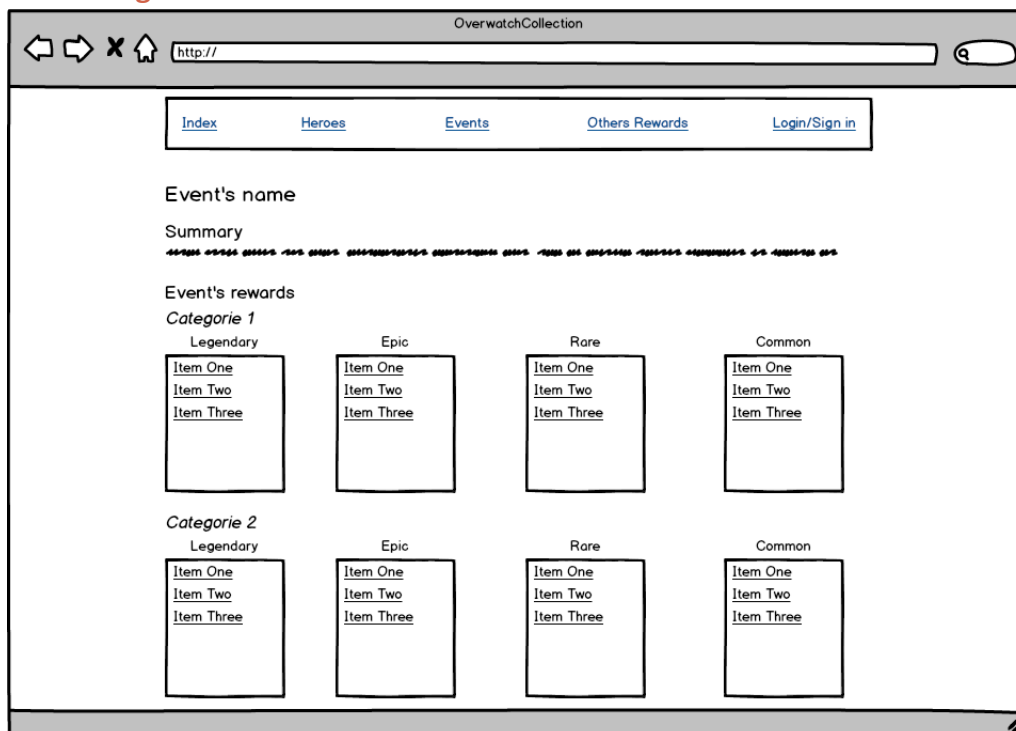


Figure 13, maquette page d'un événement

Cette page fonctionne de la même manière que la page « héros », d'abord un petit résumé et ensuite les objets triés.

### 3.4.6 Page des autres objets

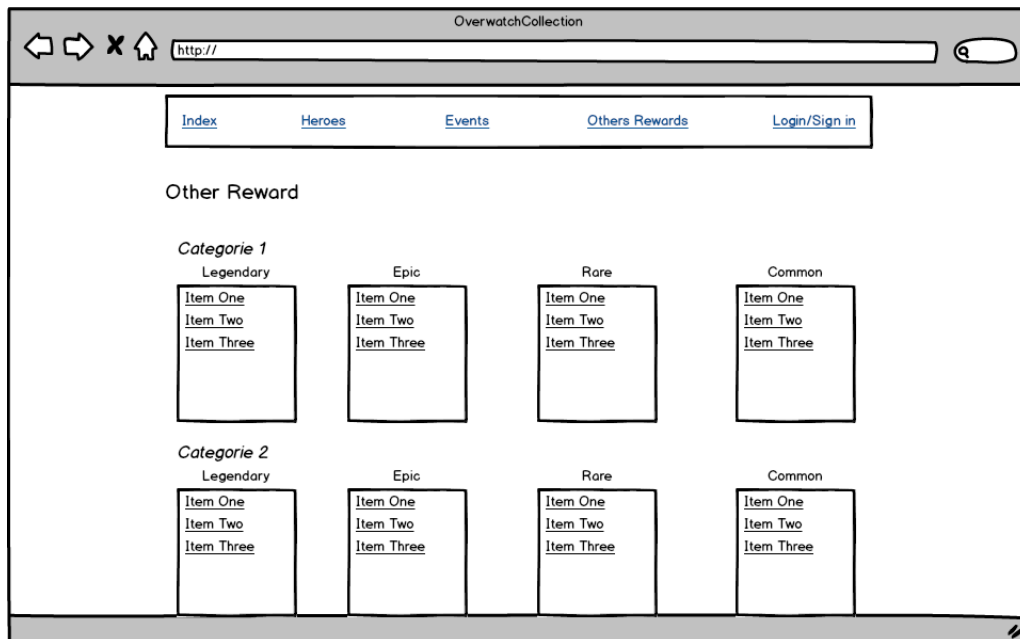


Figure 14, maquette page des autres objets

Cette page ne contient que des listes des objets qui ne sont liées à aucun héros. Certains de ces objets ne sont trouvable nulle part ailleurs, c'est la raison de l'existence de cette page.

### 3.4.7 Page de connexion/inscription

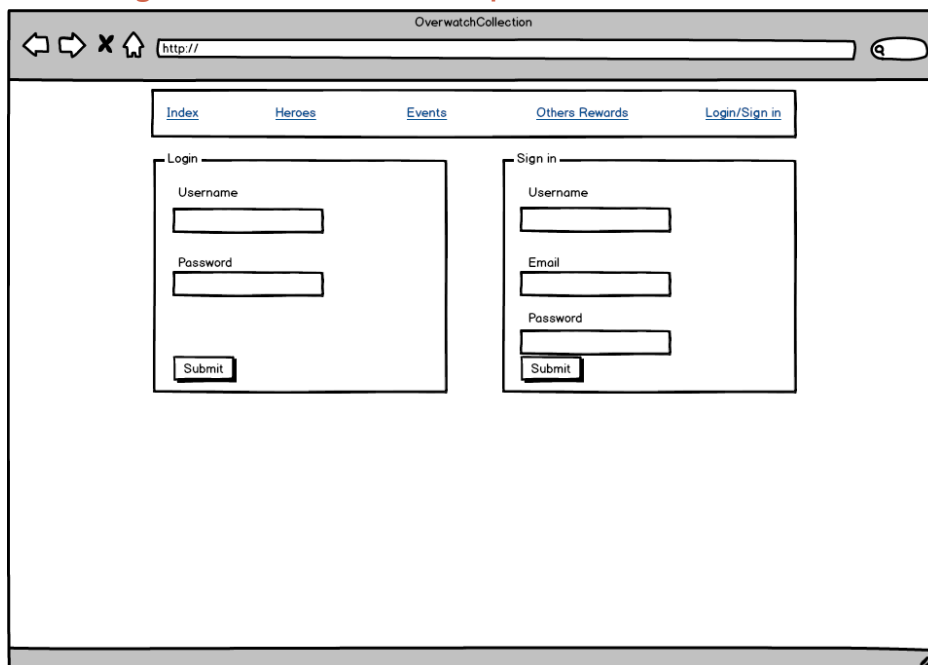


Figure 15, maquette page de connexion/inscription

Voici la page sur laquelle l'utilisateur va pouvoir soit se connecter, soit créer un compte pour se connecter. Cette page est en réalité la page utilisateur lorsqu'aucun utilisateur n'est connecté.

### 3.4.8 Page du compte

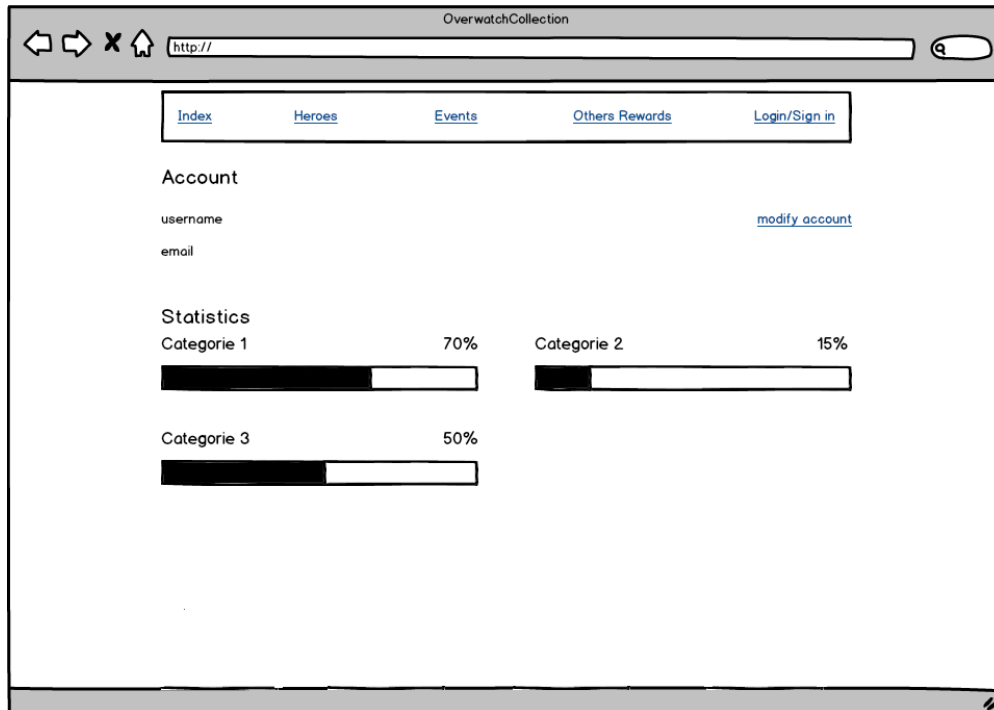


Figure 16, maquette page du compte

Voici la page « mon compte » de l'utilisateur, c'est ici qu'il pourra modifier ses informations, qu'il pourra voir toutes les statistiques sur ces objets du jeu. Cette page est en réalité la page utilisateur lorsqu'un utilisateur est connecté.

### 3.4.9 Page administrateur

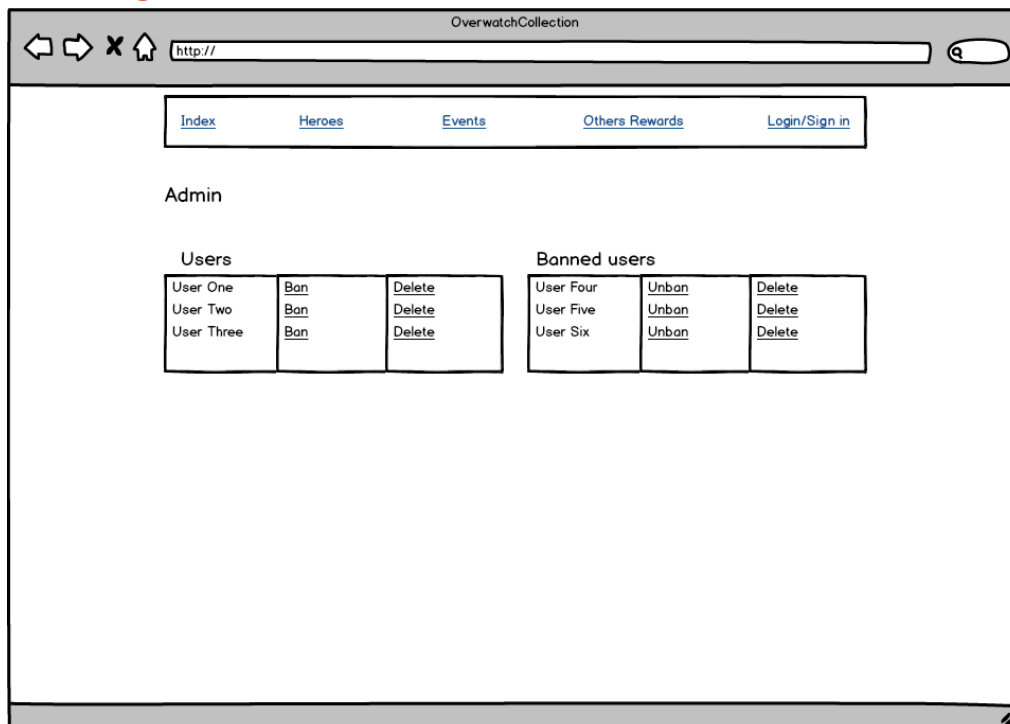


Figure 17, maquette page administrateur

Finalement la page administrateur, c'est une page uniquement accessible par les administrateurs qui leur servira à bannir, dé-bannir et supprimer des utilisateurs.

### 3.5 Modèle relationnel

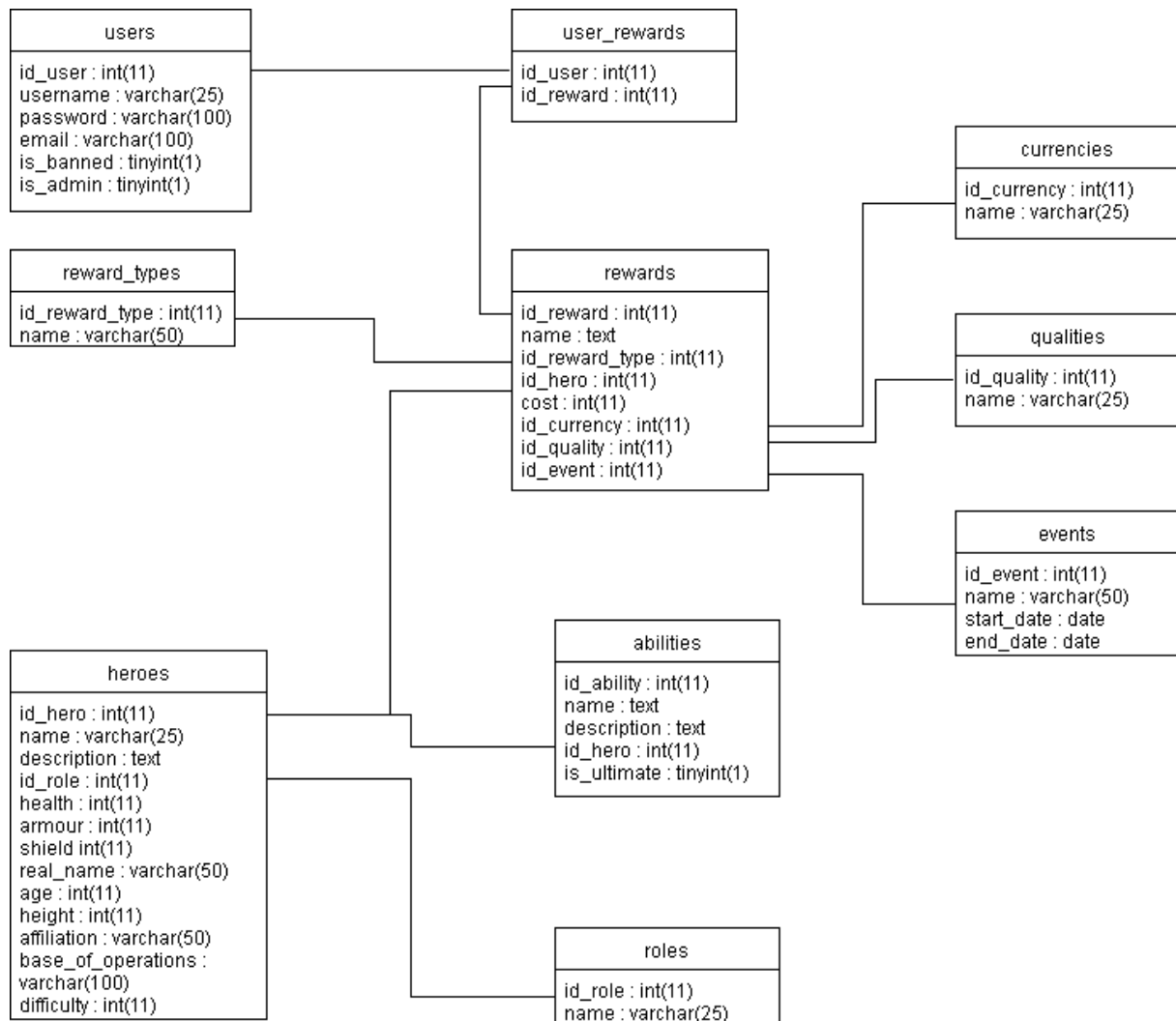


Figure 18, modèle relationnel

### 3.6 Description des tables

#### 3.6.1 Table users

Nom du champ	Type	Description
<b>id_user</b>	int(11), Non null, AI, PK	Identifiant de l'utilisateur
<b>username</b>	varchar(25), Non null	Nom de l'utilisateur
<b>password</b>	varchar(100), Non null	Mot de passe de l'utilisateur
<b>email</b>	varchar(100), Non null	Email de l'utilisateur
<b>is_banned</b>	tinyint(1), Non null	Définit si l'utilisateur est banni ou pas
<b>is_admin</b>	tinyint(1), Non null	Définit si l'utilisateur est admin ou pas



### 3.6.2 Table reward\_types

Nom du champ	Type	Description
<b>id_reward_type</b>	int(11), Non null, AI, PK	Identifiant de la catégorie d'objets cosmétique
<b>name</b>	varchar(50), Non null	Nom de la catégorie de cosmétique

Cette table est la table qui contient les catégories d'objets cosmétiques.

### 3.6.3 Table heroes

Nom du champ	Type	Description
<b>id_hero</b>	int(11), Non null, AI, PK	Identifiant du héros
<b>name</b>	varchar(25), Non null	Nom/pseudo du héros
<b>description</b>	text, Non null	Description du héros
<b>id_role</b>	int(11), Non null, FK	Identifiant du rôle du héros
<b>health</b>	int(11), Non null	Points de vie du héros
<b>armour</b>	int(11), Non null	Points d'armure du héros
<b>shield</b>	int(11), Non null	Points de bouclier du héros
<b>real_name</b>	varchar(50), Non null	Vrai nom (et prénom) du héros
<b>age</b>	int(11), Non null	Age du héros
<b>height</b>	int(11), Null	Taille, en cm, du héros
<b>affiliation</b>	varchar(50), Null	Organisation du héros
<b>base_of_operations</b>	varchar(100), Null	Base d'opérations/ Lieu d'habitation du héros
<b>difficulty</b>	int(11), Non null	Difficulté du héros, en jeu

Cette table contient les héros avec des informations de bases sur eux, il y a une clé étrangère de la table rôle.

### 3.6.4 Table user\_rewards

Nom du champ	Type	Description
<b>id_user</b>	int(11), Non null, FK	Index de l'utilisateur
<b>id_reward</b>	int(11), Non null, FK	Index de l'objet cosmétique

Voici la table de liaison entre la table utilisateur et la table d'objets cosmétiques elle permet de savoir quel utilisateur à quel objet. Car chaque utilisateur ne peut avoir qu'une seule fois chaque objet.

### 3.6.5 Table rewards

Nom du champ	Type	Description
<b>id_reward</b>	int(11), Non null, AI, PK	Identifiant de l'objet cosmétique
<b>name</b>	text, Non null	Nom de l'objet cosmétique
<b>id_reward_type</b>	int(11), Non null, FK	Identifiant de la catégorie de l'objet cosmétique
<b>id_hero</b>	int(11), Null, FK	Identifiant du héros de l'objet cosmétique
<b>cost</b>	int(11), Null	Coût de l'objet cosmétique
<b>id_currency</b>	int(11), Null, FK	Identifiant de la monnaie de l'objet cosmétique
<b>id_quality</b>	int(11), Non null, FK	Identifiant de la qualité de l'objet cosmétique
<b>id_event</b>	int(11), Null, FK	Identifiant de l'événement de l'objet cosmétique

Cette table contient les objets cosmétiques, il y a une clé étrangère de la table des catégories d'objets, qui définit de quelle catégorie fait partie cet objet. Il y a une clé étrangère de la table héros, pour savoir si l'objet est un objet qui modifie un héros, elle peut être nulle car certains objets ne sont pour aucun héros. Ensuite il y a une clé étrangère de la table des monnaies, qui définit de quelle monnaie faut utiliser pour acheter l'objet, elle peut être nulle car certains objets ne peuvent pas être achetés. Après la clé étrangère de la table des qualités d'objets, qui définit de quelle qualité est l'objet. Et finalement Il y a la clé étrangère de la table événement, pour savoir si l'objet est un objet qui est apparu avec un événement, elle peut être nulle car certains objets ne sont arrivés dans le jeu lors d'aucun événement.

### 3.6.6 Table abilities

Nom du champ	Type	Description
<b>id_abilities</b>	int(11), Non null, AI, PK	Identifiant de la capacité
<b>name</b>	text, Non null	Nom de la capacité
<b>description</b>	text, Non null	Description de la capacité
<b>id_hero</b>	int(11), Non null, FK	Identifiant du héros à qui appartient la capacité
<b>is_ultimate</b>	tinyint(1), Non null	Définit si la capacité est une capacité ultime ou pas

C'est la table qui contient toutes les capacités de tous les héros, il y a une clé étrangère de la table héros afin de déterminer à quel héros la capacité appartient-elle.

### 3.6.7 Table roles

Nom du champ	Type	Description
<b>id_role</b>	int(11), Non null, AI, PK	Identifiant du rôle
<b>name</b>	varchar(25), Non null	Nom du rôle

Cette table contient les différents rôles possibles des héros.

### 3.6.8 Table currencies

Nom du champ	Type	Description
<b>id_currency</b>	int(11), Non null, AI, PK	Identifiant de la monnaie
<b>name</b>	varchar(25), Non null	Nom de la monnaie

Cette table contient les différentes monnaies du jeu.

### 3.6.9 Table qualities

Nom du champ	Type	Description
<b>id_quality</b>	int(11), Non null, AI, PK	Identifiant de la rareté/qualité
<b>name</b>	varchar(25), Non null	Nom de la rareté/qualité

Cette table contient les différentes qualités d'objets.

### 3.6.10 Table events

Nom du champ	Type	Description
<b>id_event</b>	int(11), Non null, AI, PK	Identifiant de l'événement
<b>name</b>	varchar(50), Non null	Nom de l'événement
<b>start_date</b>	date, Non null	Date de début de l'événement
<b>end_date</b>	date, Non null	Date de fin de l'événement

Cette table contient les différents événements du jeu.

### 3.7 Utilisation et explication des requêtes

Afin que le site soit dynamique, il faut obligatoirement que les données soient récupérées depuis une base de données. Base de données qui va aussi évidemment recevoir des ajouts ainsi que des modifications depuis le site web. Voici donc quelles requêtes sont utilisées sur les différentes pages du site web.

*Toutes les requêtes « SELECT » retournent des tableaux qui sont ensuite traités dans la partie affichage.*

#### 3.7.1 Accueil

Sur l'accueil du site, il n'y a que la liste des utilisateurs qui est récupérés depuis la base, il y a donc un « SELECT » des utilisateurs sans les utilisateurs bannis, car on ne veut pas les afficher.

#### 3.7.2 Les héros

Sur la page héros on récupère d'abord les rôles existant avec un « SELECT » de la table des rôles car on veut trier les héros par rôle justement. Une fois qu'on a les rôles on récupère tous les héros correspondant à chaque rôle, grâce à un « SELECT » de la table héros et à une condition qui est que l'identifiant du rôle du héros doit correspondre à celui donné, afin de les afficher sous leurs rôles respectifs.

#### 3.7.3 Un héros

Sur cette page on affiche des informations de base du héros, les capacités du héros et finalement les objets cosmétiques du héros trié par catégorie et ensuite par rareté.

On récupère les objets triés par catégories et par raretés. Donc il faut d'abord récupérer toutes les catégories existantes (« SELECT » de la table catégorie d'objets) et ensuite toutes les qualités existantes (« SELECT » de la table qualité). À partir de là, on récupère, pour chaque combinaison de catégorie/rareté, tous les objets dont l'identifiant de la catégorie ainsi que de la rareté correspond (« SELECT » de la table objets, avec les conditions citées) afin de les mettre dans un tableau pour qu'ils soient triés par catégorie et rareté (vu qu'on veut récupérer les objets d'un héros, il faut aussi que l'id du héros corresponde).

#### 3.7.4 Les événements

Sur la page événement, il n'y a qu'un tableau des événements du jeu, on récupère donc juste tous les événements avec un « SELECT » sur la table événement.

#### 3.7.5 Un événement

Cette page fonctionne assez similaire à la page d'un héros, il y a d'abord des informations de base sur l'événement et ensuite tous les objets de cet événement.

Donc pour les informations de base on fait un « SELECT » dans la table événement avec l'id de l'évènement en question.

Et pour les objets cosmétiques, ça marche comme pour la page héros à la différence qu'on récupère tous les objets correspondant avec l'identifiant de l'événement plutôt qu'avec celui héros. Il y a aussi une autre différence qui est que les événements ont parfois des objets avec le même nom mais qui correspondent à différents héros, il a fallu donc récupérer donc le nom du héros en liant la table héros avec la table objets cosmétiques, afin de savoir de quel objet il est question.

Mais il y avait un problème qui était que pas tous les objets ont des héros associés et du coup ça ne récupérait que les objets qui étaient associés à des héros, et ce n'est pas ce que l'on veut, on veut que tous les objets de l'événement soient récupérés, la solution était de faire une jointure externe sur la table héros lorsqu'on la lie avec la table objets cosmétiques.

### 3.7.6 Les autres objets

Sur cette page on affiche seulement les objets qui ne correspondent à aucun héros, paradoxalement on récupère les données de la même façon que pour la page héros à la seule différence qu'au lieu de récupérer les objets avec un identifiant héros, on récupère les objets dont le champ identifiant héros est égal à zéro ou est nul.

### 3.7.7 Affichage d'objet héros/événement/autre objets

Lorsqu'on affiche un objet sur une de ces trois pages et qu'un utilisateur est connecté il peut cliquer sur cet objet pour le sélectionner ou le désélectionner et s'il l'a sélectionné il doit pouvoir le savoir (affichage en vert).

Donc lorsqu'il clique sur l'objet pour l'ajouter à ses objets, dans la base on traduit ça par l'ajout d'un enregistrement dans la table de liaison entre la table utilisateur et la table objets cosmétiques qui contient l'identifiant de l'utilisateur ainsi que l'identifiant de l'objet en question. Il faut donc faire un « INSERT » dans cette table de liaison.

Afin de savoir quel objet l'utilisateur a sélectionné il faut avoir une liste des objets en question (ou seulement de l'identifiant de ceux-ci en l'occurrence) donc on fait un « SELECT » dans la table de liaison entre la table utilisateur et la table objets avec comme condition l'identifiant de l'utilisateur.

### 3.7.8 Connexion

Au niveau de la connexion, l'utilisateur se connecte avec son nom d'utilisateur et son mot de passe donc afin de vérifier si le nom d'utilisateur correspond avec le mot de passe, on récupère l'enregistrement de l'utilisateur avec son nom d'utilisateur (« SELECT » dans la table utilisateur avec comme condition le nom d'utilisateur) et on vérifie que le mot de passe de la base (haché) correspond à celui que l'utilisateur a entré.

### 3.7.9 Identification

Pour l'identification, il suffit de faire un « INSERT » des données entré par l'utilisateur dans la table utilisateur.

### 3.7.10 Modification du compte

Pour la modification des données, la requête n'est qu'un « UPDATE » avec les données entrées par l'utilisateur.

### 3.7.11 Mon compte, info et statistiques

Lorsqu'on va sur son compte utilisateur il y a d'abord les informations du compte, ensuite si on est administrateur il y a un lien vers la page admin et ensuite les barres de progression.

Pour les informations il faut faire un « SELECT » dans la table utilisateur avec l'identifiant de l'utilisateur et c'est là aussi qu'on récupère l'information qui détermine si l'utilisateur est administrateur ou pas.

Pour chaque barre de progression il faut avoir comme information, le nombre d'objets qu'il y a en tout dans la catégorie choisie ainsi que le nombre d'objets que l'utilisateur a pour cette catégorie, par exemple si on veut faire une barre de progression de l'évènement 1 il faut le nombre d'objets de l'évènement 1 en tout et le nombre d'objets de l'évènement 1 que l'utilisateur a sélectionné. Ces données sont récupérées en faisant un « SELECT » de tous les objets de l'évènement où tous les objets de l'évènement sélectionné par l'utilisateur et un « COUNT » sur l'identifiant des objets afin de compter le nombre qu'il y en a.

Malheureusement il avait un problème lorsqu'on récupérait le nombre d'objets par héros ou par évènement pour un utilisateur, si l'utilisateur n'avait aucun objet d'un héros ou d'un évènement la requête ne récupérait pas zéro dans le count mais rien, même pas le nom du héros ou évènement. Et c'est embêtant car dans les barres de progression, on veut quand même avoir tous les héros et évènement, même si l'utilisateur n'a aucun de leurs objets. Afin de résoudre ce problème plusieurs solutions ont été essayées, d'abord j'ai essayé de mettre un « DISTINCT » sur le champ du nom des héros (dans le cas des barres sur les héros) en espérant que ça forcerait l'affichage de tous les héros, ça n'a pas marché. Ensuite j'ai essayé de faire une jointure externe sur la table users\_rewards (à la place d'une jointure normale) vu qu'on compte le nombre d'id\_reward de cette table je me suis dit qu'avec une jointure externe il compterait même s'il n'y a rien, mais non. Finalement j'ai trouvé la solution suivante : d'abord on récupère les héros pour lesquels l'utilisateur a des objets et on fait une « UNION » avec une autre requête qui récupère les héros qui ne font pas partie des héros de la première requête.

### 3.7.12 Actions admin

Les administrateurs peuvent faire trois actions : bannir, dé-bannir et supprimer un utilisateur dans les trois cas il faut faire quelque chose dans la base de données.

Dans le cas du bannissement ou du dé-bannissement on met à jour le champ « is\_banned » avec un simple « UPDATE » et dans le cas de la suppression d'utilisateur on fait un « DELETE » avec l'identifiant de l'utilisateur qu'on veut supprimer.

## 3.8 Description détaillée de l'architecture du code

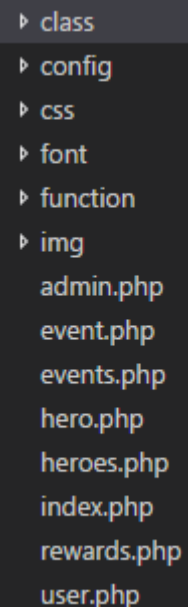
### 3.8.1 Séparation du code

Dans ce projet le code se sépare en cinq parties, d'abord il y a le HTML basique, le squelette du site, cette partie se trouve dans chacune des pages affichée (par exemple dans «index.php»). Dans les pages affichées se trouvent aussi toutes les actions de redirections qui ne concernent que la page en question, par exemple l'action de bannissement d'utilisateur pour la page administrateur. (Voir 3.8.3)

Ensuite il y a la partie style du site web, cette partie se trouve dans le dossier « CSS », nommé ainsi car le style est fait en CSS. Ce dossier contient donc le fichier de style du site qui est récupéré depuis les différentes pages du site.

Après ça, il y a le dossier « function », ce dossier contient les différentes fonctions sur lesquelles on peut être redirigé afin d'exécuter un script. Par exemple la fonction qui permet d'ajouter ou d'enlever un objet à un utilisateur, cette fonction étant utilisée sur plusieurs pages, elle se trouve dans un fichier séparé.

Et finalement il y a le dossier « class », qui contient les classes utilisées sur le site web. C'est là que ce trouvent les deux plus grosses parties du projet, les classes de dialogue avec la base de données (« OcDao » qui signifie OverwatchCollection Data Access Object) et d'affichage des données (« OcDisplay » pour OverwatchCollection Display). Ces deux classes sont utilisées comme des sortes de répertoires afin de regrouper les fonctions du même type.



```
▸ class
▸ config
▸ css
▸ font
▸ function
▸ img
  admin.php
  event.php
  events.php
  hero.php
  heroes.php
  index.php
  rewards.php
  user.php
```

### 3.8.2 Les classes

#### 3.8.2.1 MyPdo

MyPdo est la classe qui sert à récupérer une instance de l'objet PDO. L'objet PDO étant un objet qui permet d'accéder à une base de données depuis PHP, avec évidemment l'IP de la base, le nom de la base, le nom d'un utilisateur qui a accès à la base et finalement le mot de passe de cet utilisateur.

Cette classe ne contient donc qu'une variable ainsi qu'une seule fonction. La variable est privée, donc on ne peut pas y accéder.

La fonction (GetMyPdo()) retourne donc l'instance de cet objet PDO, mais étant donné que cet objet PDO ne change pas pour chaque requête, ou selon l'utilisateur qui est connecté, on fait en sorte de n'avoir qu'une instance de cet objet. Donc cette fonction retourne l'instance de cet objet PDO, mais par contre elle le crée seulement s'il n'existe pas déjà, ce qui permet de n'avoir qu'une instance de cet objet.

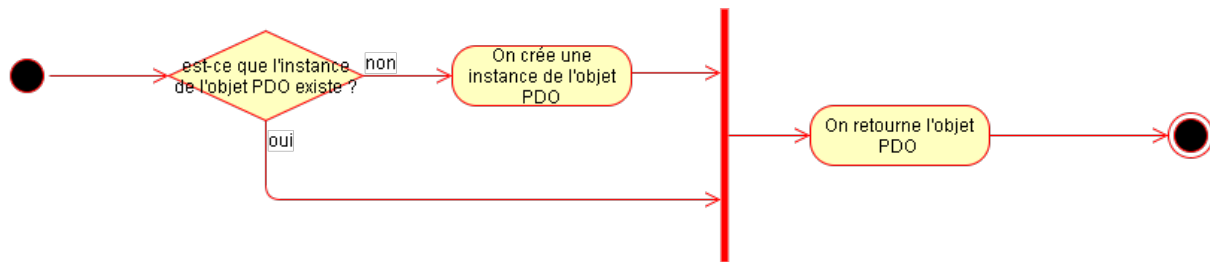


Figure 19, algorithme GetMyPdo

### 3.8.2.2 OcDao

La classe « OcDao » contient toutes les fonctions de récupération de données, d'insertion de données, de modification de données et de suppression de données.

Les fonctions de récupération basiques. Il s'agit des fonctions de récupérations de données qui ne récupèrent que des données de requête simple, les données sont retournées comme on les reçoit.

Nom	Description
<b>SelectHeroes()</b>	Retourne tous les héros
<b>SelectHeroById(\$id)</b>	Retourne un héros, sélectionné avec son id
<b>SelectRoles()</b>	Retourne tous les rôles
<b>SelectQualities()</b>	Retourne toutes les qualités d'objets
<b>SelectRewardTypes()</b>	Retourne tous les types d'objets
<b>SelectEvents()</b>	Retourne tous les événements
<b>SelectEventById(\$id)</b>	Retourne un événement sélectionné avec son id
<b>SelectUsers()</b>	Retourne tous les utilisateurs
<b>SelectCleanUsers()</b>	Retourne tous les utilisateurs dont le champ is_banned est false
<b>SelectCleanUsersNoAdmin()</b>	Retourne tous les utilisateurs dont les champs is_banned et is_admin sont false
<b>SelectBannedUsers()</b>	Retourne tous les utilisateurs dont le champ is_banned est true et is_admin est false
<b>SelectUserByUsername(\$username)</b>	Retourne un utilisateur, sélectionné avec son nom
<b>SelectUserById(\$id)</b>	Retourne un utilisateur, sélectionné avec son id
<b>SelectAbilitiesByIdHero(\$id)</b>	Retourne les capacités d'un héros. Elles sont sélectionnées avec l'id du héros.
<b>SelectIdRewardsByIdHeroAndIdUser(\$id_hero, \$id_user)</b>	Retourne les id d'objets cosmétiques. Ils sont sélectionnés avec l'id d'un héros et l'id d'un utilisateur testé sur la table de liaison users_rewards (liée dans la requête à la table rewards)
<b>SelectIdRewardsByIdEventAndIdUser(\$id_event, \$id_user)</b>	Retourne les id d'objets cosmétiques. Ils sont sélectionnés avec l'id d'un événement et l'id d'un utilisateur testé sur la table de liaison users_rewards (liée dans la requête à la table rewards)
<b>SelectIdRewardsByNoIdHeroAndIdUser(\$id</b>	Retourne les id d'objets cosmétiques. Ils sont



<b>_user)</b>	sélectionnés s'il n'y a pas d'id héro ou s'il vaut 0 et l'id d'un utilisateur testé sur la table de liaison users_rewards (liée dans la requête à la table rewards)
<b>SelectCountReward()</b>	Retourne le nombre d'id d'objets cosmétique dans la table rewards.
<b>SelectCountRewardByIdUser(\$id)</b>	Retourne le nombre d'id d'objets cosmétique dans la table users_rewards, pour un id d'utilisateur donné
<b>SelectCountRewardEvents()</b>	Retourne les noms de tous les événements ainsi que le nombre d'id d'objets que chaque événement a. Le nom de l'événement est récupéré en liant la table objets cosmétiques avec la table événement sur l'id des événements.
<b>SelectCountRewardHeroes()</b>	Retourne les noms de tous les héros ainsi que le nombre d'id d'objets que chaque héro a. Le nom du héros est récupéré en liant la table objets cosmétiques avec la table héros sur l'id des héros.

Les fonctions d'insertions, de modifications ou de suppression. Il s'agit des fonctions qui servent à insérer, modifier et supprimer de la base.

Nom	Description
<b>BanUserById(\$id)</b>	Modifie la valeur du champ is_banned et la passe à true. L'utilisateur est sélectionné avec son id.
<b>UnbanUserById(\$id)</b>	Modifie la valeur du champ is_banned et la passe à false. L'utilisateur est sélectionné avec son id.
<b>DeleteUserById(\$id)</b>	Supprime un utilisateur. L'utilisateur est sélectionné avec son id.
<b>UpdateUserByIdNoPwd(\$id, \$username, \$email)</b>	Modifie le nom et l'email d'un utilisateur. L'utilisateur est sélectionné avec son id.
<b>InsertUserReward(\$id_user, \$id_reward)</b>	Ajoute un champ à la table de liaison users_rewards.
<b>DeleteUserReward(\$id_user, \$id_reward)</b>	Supprime un champ à la table de liaison users_rewards.
<b>InsertUser(\$username, \$email, \$password)</b>	Ajoute un utilisateur.

Les fonctions de récupération complexes, qui ont du traitement de données avant de retourner les données. En résumé c'est les fonctions de récupération qui méritent d'être expliquées plus précisément.

### SelectHeroesInArrayOfRole()

Retourne un tableau de héros trié par rôle sous cette forme :

```
Array
(
    [Role1] => Array
        (
            [Hero1] => Array
                (
                    [Name] => ...
                    [Description] => ...
                    etc
                )
            [Hero2] => Array
                (
                    [Name] => ...
                    [Description] => ...
                    etc...
                )
            [Hero1] => Array
                (
                    etc...
                )
        )
    [Role2] => Array
        (
            etc...
        )
)
```

Fonctionnement de la fonction :

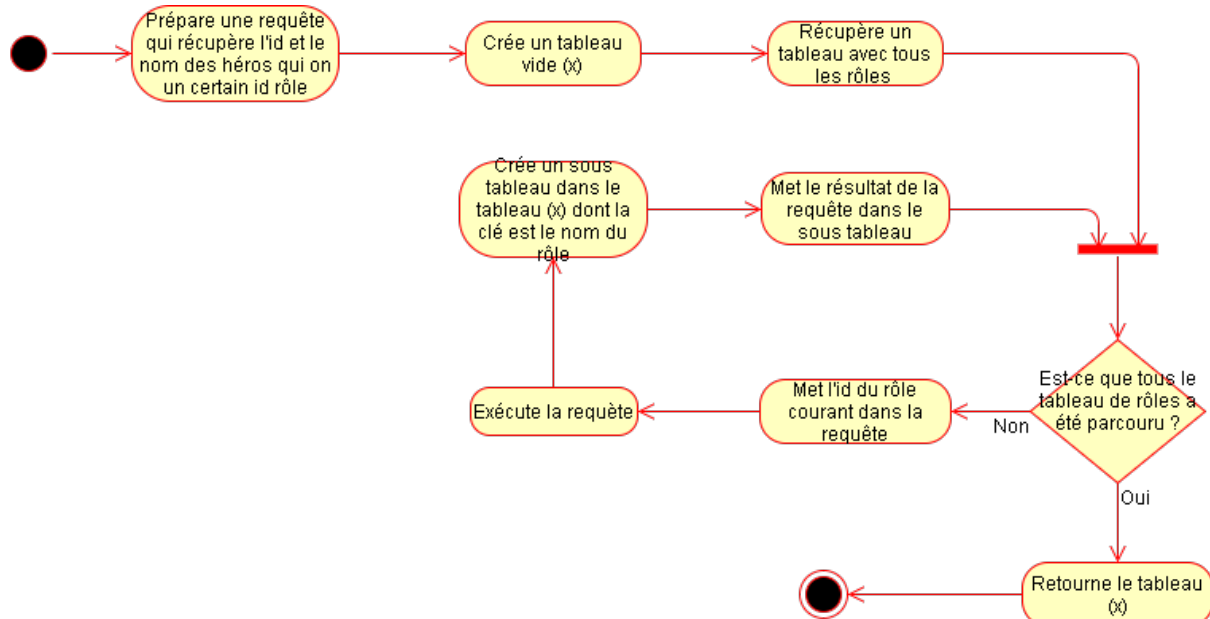


Figure 20, algorithme SelectHeroesInArrayOfRole

### IsCreatedUserReward(\$id\_user, \$id\_reward)

Cette fonction retourne un booléen qui détermine si un certain enregistrement de la table de liaison users\_rewards existé ou pas.

Fonctionnement de la fonction :

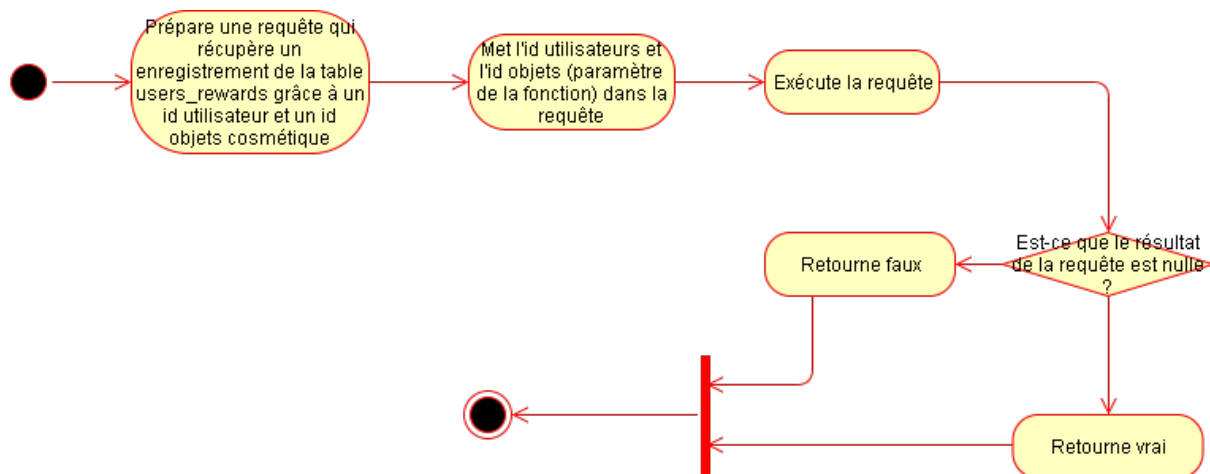


Figure 21, algorithme IsCreatedUserReward

### *SelectIdRewardsByIdHeroAndIdUser(\$id\_hero, \$id\_user)*

Cette fonction sert à récupérer une liste d'identifiants d'objets cosmétique qui sont utilisés afin de savoir quel objet est acquis par l'utilisateur et quel objet ne l'est pas. La liste retournée étant la liste des id des objets que l'utilisateur a.

Cette fonction contient donc un petit peu de traitement des données après la récupération depuis la base car le format récupéré n'était pas idéal, il est donc transformé et un tableau beaucoup plus simple :

```

Array
(
    [0] => Array
        (
            [id_reward] => 221
        )
    [1] => Array
        (
            [id_reward] => 224
        )
    [2] => Array
        (
            [id_reward] => 673
        )
    etc...
)

```

Figure 22, SelectIdReward ancien format

Figure 23, SelectIdReward nouveau format

Cette fonction marche exactement de la même manière que les deux fonctions suivante : SelectIdRewardsByIdEventAndIdUser et SelectIdRewardsByNoldHeroAndIdUser.

*SelectRewardsInArrayOfQualityAndTypeByIdHero(\$id)*

Voici sans doute une des fonctions les plus importantes du site, c'est la fonction qui permet de récupérer les objets cosmétiques d'un héros trié dans un tableau par catégorie et qualité :

```

Array
(
    [Catégorie1] => Array
        (
            [Qualité1] => Array
                (
                    [Objet1] => Array
                        (
                            [id_reward] => 221
                            [Name] => ...
                        )
                    [Objet2] => Array
                        (
                            [id_reward] => 221
                            [Name] => ...
                        )
                    etc...
                )
            [Qualité4] => Array
                (
                    [Objet1] => Array
                        (
                            [id_reward] => 221
                            [Name] => ...
                        )
                    [Objet2] => Array
                        (
                            [id_reward] => 221
                            [Name] => ...
                        )
                    etc...
                )
        )
    [Catégorie2] => Array
        (
            [Qualité3] => Array
                (
                    [Objet1] => Array
                        etc...
                )
        )
    etc...
)

```

Les objets ne sont évidemment pas récupéré tel quel depuis la base, il y a un algorithme qui s'en charge et voici son fonctionnement :

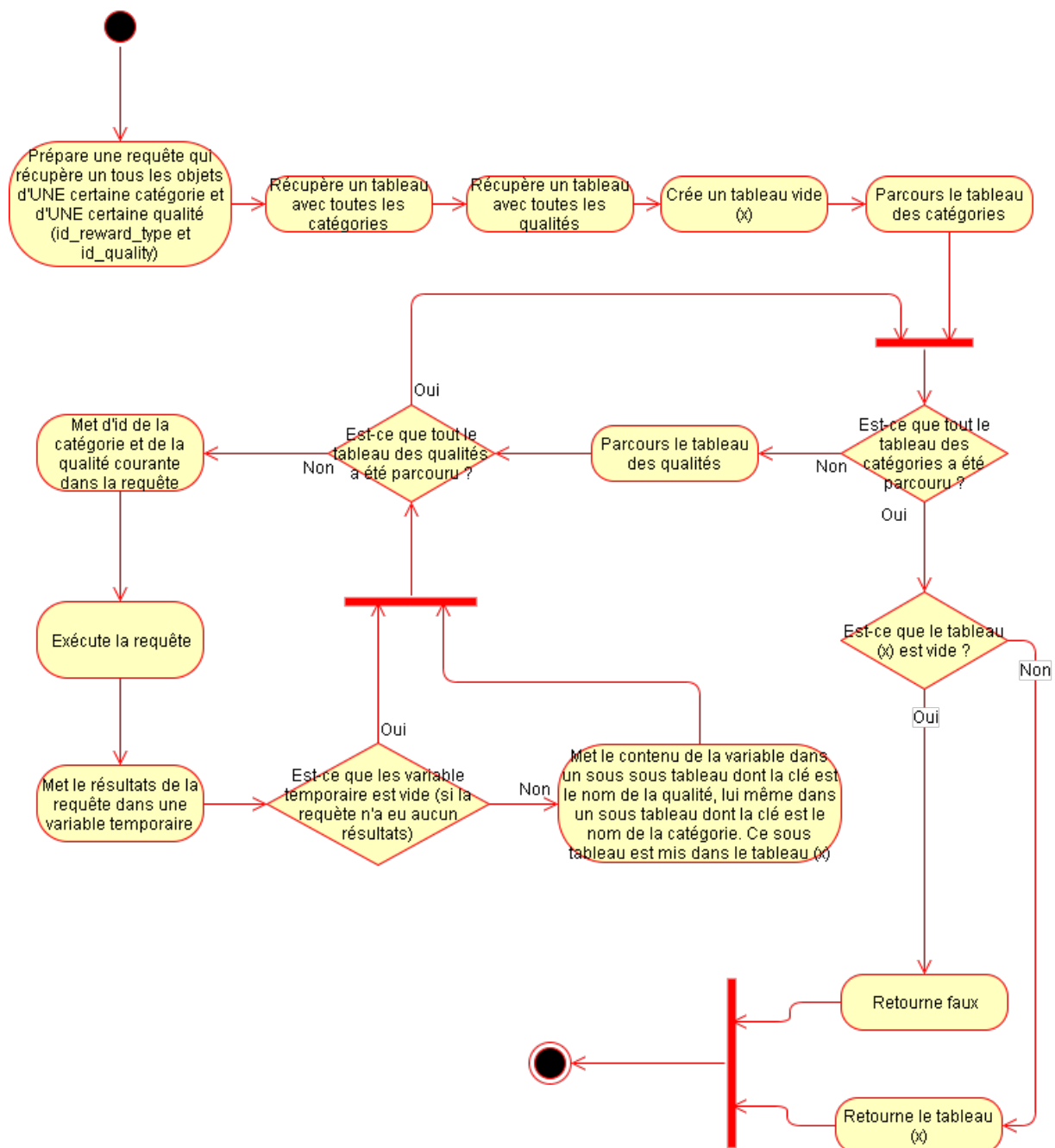


Figure 24, algorithme SelectRewardsInArrayOfQualityAndTypeByHero

Cette fonction a le même fonctionnement que les fonctions SelectRewardsInArrayOfQualityAndTypeByNoHero (qui récupère les objets qui n'ont pas de héros) et SelectRewardsInArrayOfQualityAndTypeByIdEvent (qui récupère les objets d'un évènement).

### *3.8.2.3OcDisplay*

La classe « OcDisplay » contient toutes les fonctions qui traitent les données récupérées et les affichent, elle contient aussi les fonctions d'affichage des formulaires avec les différents messages d'erreurs ou l'affichage de la barre de navigation. En gros toutes les fonctions qui affichent des éléments HTML et qui contiennent du PHP :

#### *DisplayNavbar()*

Comme son nom l'indique, c'est la fonction qui affiche la barre de navigation. Cette fonction est dans toutes les pages du site. Selon les situations la barre n'affiche pas toujours la même chose, lorsque personne n'est connecté, le dernier lien est « Login/Sign in », et si quelqu'un se connecte ça n'a pas de sens de garder le login, donc on le remplace par « My Account », d'ailleurs la fonction détecte si l'utilisateur se trouve sur la page de son compte, si c'est le cas, elle affiche « Log out » à la place de « My Account ».

#### *DisplayListUsers()*

Cette fonction affiche une liste des utilisateurs du site, elle trie et n'affiche pas les utilisateurs bannis et affiche les administrateurs en premier avec une indication de leurs statuts.

#### *DisplayListCleanUserWithBan() et DisplayListBannedUserWithUnbanDelete()*

Ces fonctions sont utilisées dans la page admin. La première sert à afficher une liste des utilisateurs non bannis avec un bouton à côté de leur nom d'utilisateur qui permet à l'administrateur de les bannir, la deuxième affiche une liste des utilisateurs bannis avec deux boutons par utilisateur, un pour dé-bannir et l'autre pour supprimer définitivement l'utilisateur.

Les boutons sont en réalité des liens vers la page admin avec le nom d'une action en GET (voir les actions en GET).

#### *DisplayEventRewards(\$id\_event), DisplayHeroRewards(\$id\_hero) et DisplayOtherRewards()*

Ces fonctions sont les fonctions qui affichent les objets cosmétiques dans des listes triées. La première s'occupe des objets d'un événement, la deuxième des objets d'un héros et la troisième des objets liés à aucun héros.

Elles récupèrent d'abord un tableau trié des objets (voir le point 3.8.2.2.4), ensuite si un utilisateur est connecté, elles récupèrent les id des objets de l'utilisateur. Ensuite en parcourant le tableau et les sous-tableaux elles regroupent les objets par qualités et les qualités par catégories d'objets afin que les utilisateurs puissent trouver les objets plus facilement.

Lorsqu'on arrive au moment où on affiche le nom de l'objet, on regarde d'abord si l'id de l'objet fait partie de la liste d'id d'objet de l'utilisateur, s'il n'y a pas d'utilisateur connecté, la liste sera vide donc pas besoin de tester si un utilisateur est connecté. S'il fait partie de la liste de l'utilisateur on met

dans une variable le code CSS pour afficher du texte en vert, sinon on ne met rien. Cette variable est ensuite concaténée dans la balise HTML qui affiche le nom de l'objet.

Juste avant d'afficher le nom de l'objet, il faut aussi tester si un utilisateur est connecté afin d'entourer le nom de l'objet avec une balise « <a> » qui contient un lien vers la fonction qui ajoute ou enlève un enregistrement à la table `users_rewards`, car si personne n'est connecté on affiche uniquement du texte.

### *DisplayEvents(\$nb\_col\_max)*

Voici la fonction qui permet d'afficher un tableau contenant les événements sur la page événements. Le paramètre `$nb_col_max` sert à définir les nombres de colonnes que l'on veut pour le tableau. Fonctionnement :

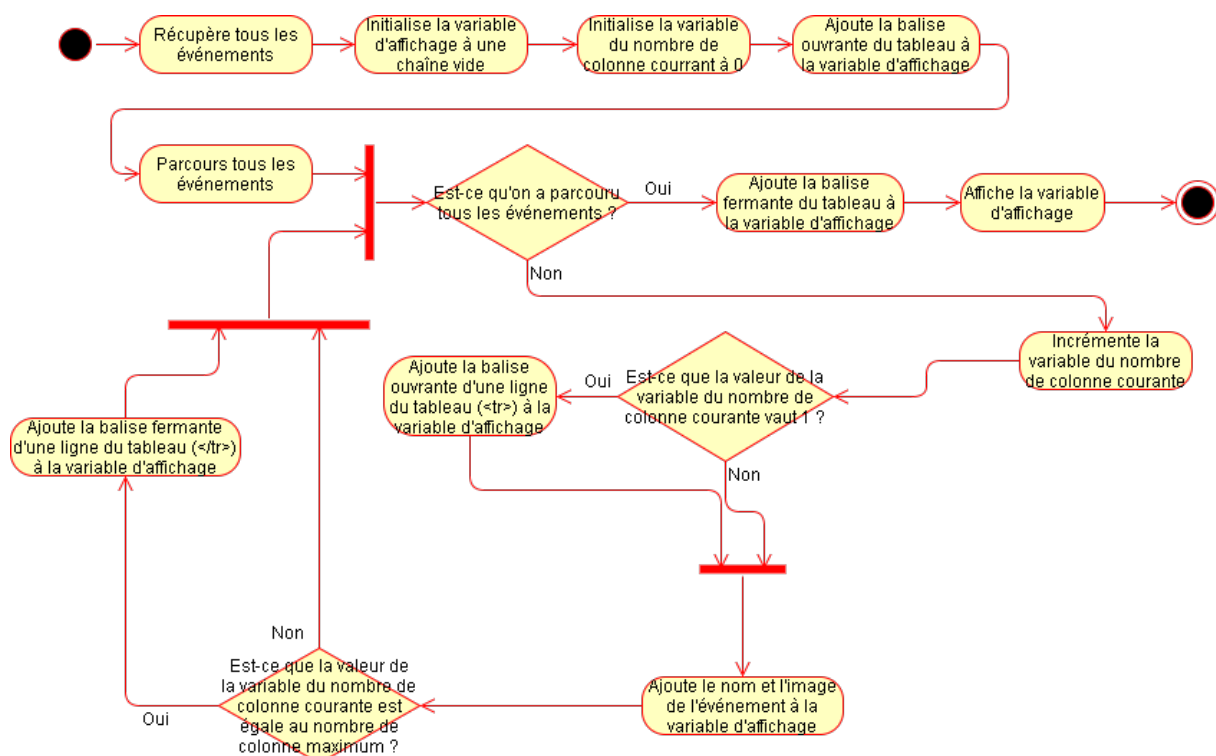


Figure 25, algorithme `DisplayEvents`

### *DisplayHeroesByRole(\$nb\_col\_max)*

Cette fonction permet d'afficher tableaux des héros, chaque tableau correspond à un rôle, cette fonction marche de la même manière que la fonction précédente à la différence qu'on exécute le code autant de fois qu'il y a de rôles. Pour chaque rôle, on traite le tableau de héros comme le tableau d'événements dans la fonction précédente.

### *DisplayHeroAbilities(\$id\_hero)*

Cette fonction permet d'afficher un tableau avec les capacités d'un héros, elle a le même fonctionnement que les deux fonctions précédentes pour les colonnes du tableau, mais cette fois le nombre de colonnes maximum n'est pas en paramètre car les héros auront toujours à peu près le même nombre de capacités, donc pas besoin de la changer.



### *DisplayHeroInfo(\$id\_hero), DisplayAccountInfo(\$id\_user) et DisplayEventInfo(\$id\_event)*

Ces fonctions sont des fonctions d'affichage basiques, elle récupère un enregistrement et affiche les informations. DisplayHeroInfo et DisplayEventInfo ont toutes les deux une petite sécurité qui fait que si les fonctions de récupération de données envoient false alors l'id donnée n'était pas bon (l'id est récupéré en GET, donc n'importe qui peut mettre un id inexistant), dans ce cas-là on redirige la page vers les pages de sélection de héros ou d'événements.

Dans DisplayAccountInfo il y a aussi un bouton qui redirige vers la page de modification de compte.

### *DisplayLogin(), DisplaySignIn() et DisplayAccountInfoUpdating(\$id\_user)*

Ces fonctions affichent respectivement le formulaire pour se connecter et le formulaire pour s'enregistrer, et le formulaire pour modifier ses informations de compte. S'il y a des messages d'erreurs (dans ces formulaires), c'est aussi elles qui les affichent.

### *GetProgressBar(\$max\_value, \$current\_value, \$width\_percent, \$height\_px)*

Cette fonction retourne une chaîne de caractères qui contient le code HTML pour afficher une barre de progression.

\$max\_value est la valeur maximum de la barre de progression, \$current\_value est la valeur actuelle de la barre de progression, \$width\_percent est la largeur, en pourcentage du conteneur parent, de la barre de progression et \$height\_px est la hauteur, en pixel, de la barre de progression.

Fonctionnement :

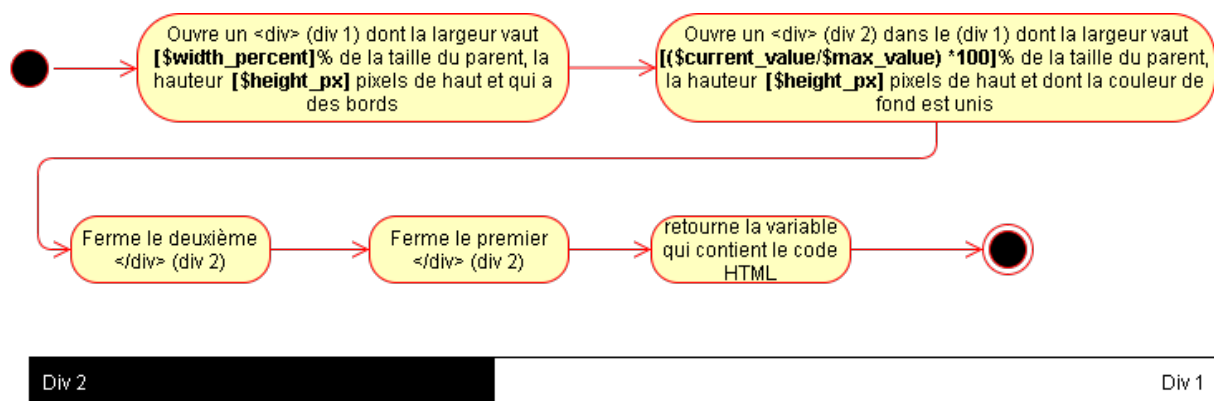


Figure 26, algorithme `GetProgressBar`

### *DisplayMainProgressBar(\$id\_user), DisplayEventsProgressBar(\$id\_user) et DisplayHeroesProgressBar(\$id\_user)*

Ces fonctions affichent les barres de progression de leurs catégories respectives, la première affiche une barre pour tous les objets, la deuxième affiche une barre pour chaque événement du jeu et la troisième affiche une barre pour chaque héros. Elles affichent aussi les titres des barres et le nombre d'objets de l'utilisateur sur le nombre d'objets en tout pour chaque barre.

### *DisplayAccountStats(\$id\_user)*

Cette fonction regroupe les fonctions d'affichage de statistiques (par exemple les barres de progression) et les met toutes dans une section.

#### 3.8.3 Les actions en GET

Certaines fonctions de ce site web se font seulement lorsque des évènements particuliers sont enclenchés, par exemple lorsqu'on clique sur un objet pour le sélectionner. Il a fallu donc trouver une méthode pour que tout le site fonctionne de la même manière à ce niveau-là, la méthode que j'ai appliquée est de mettre les fonctions sur les pages qui les enclenchent, et de les enclencher seulement si la variable en GET « action » non seulement existe mais contient le mot-clé de la fonction.

Par exemple, lorsqu'on veut se connecter, on va sur la page « user.php », on entre nos identifiants et on clique sur le bouton « submit ». On est ensuite redirigé sur la même page mais avec du GET en plus : « user.php?action=login ». Dès qu'on a une variable « action » en GET, on entre dans un « if » qui nous redirigera, à la fin, vers la page sans cette variable en GET. Dans ce « if », on vérifie si la variable action contient le mot-clé d'une fonction connue, en l'occurrence « login » en est une donc on exécute le script de la fonction et on est ensuite redirigé comme expliqué plus tôt.

##### 3.8.3.1 hero.php, event.php et rewards.php

Sur ces trois pages se trouvent les objets du jeu, et donc c'est sur ces pages, lorsqu'on est connecté, qu'on peut cliquer sur les objets afin de les sélectionner, fonctionnement :

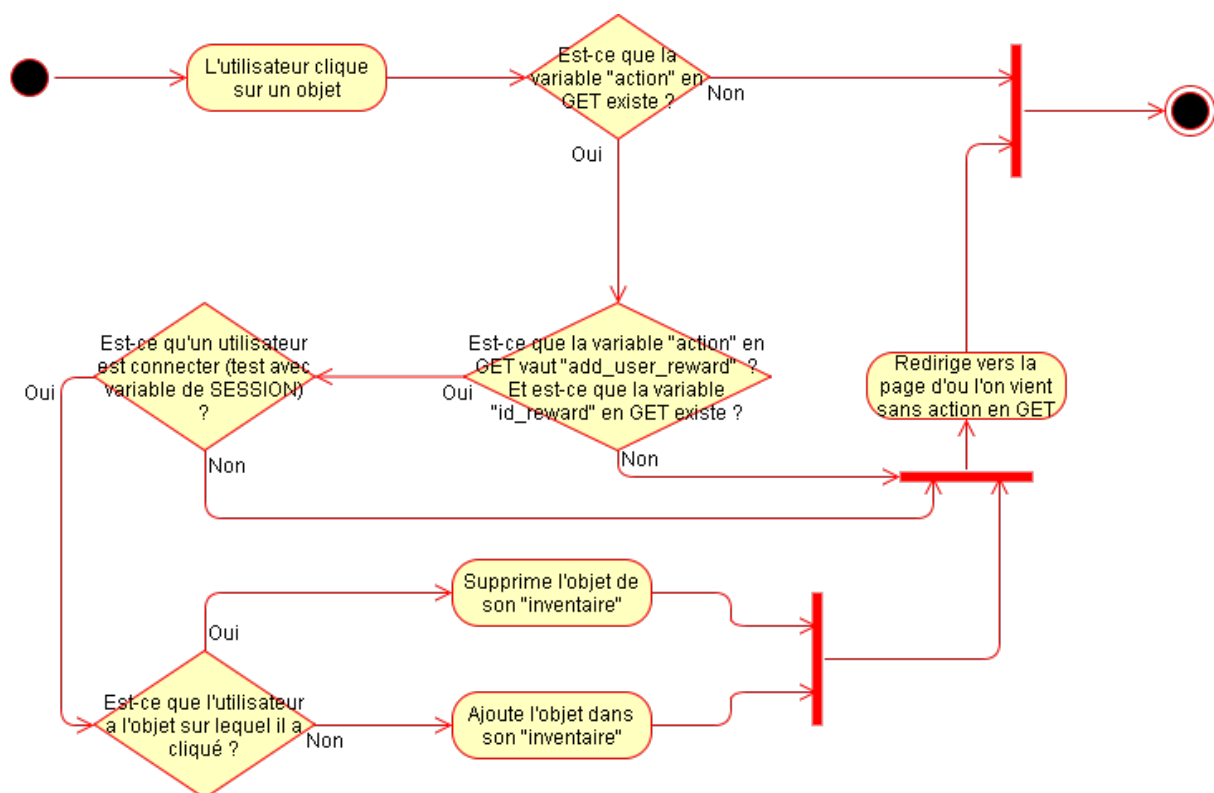


Figure 27, algorithme clique objets

### 3.8.3.2user.php

Voici la page utilisateur, elle est utilisée pour une grande partie des fonctionnalités des utilisateurs, donc elle ne contient pas mal d'action en GET :

#### Action Login

Lorsqu'on complète le formulaire de connexion et qu'on appuie sur le bouton « submit », on est redirigé sur la même page avec en GET l'action « login ». Ce script commence par hacher le mot de passe en sha1, ensuite il récupère, dans la base de données l'utilisateur qui a le même nom que celui entré, s'il n'y en a pas, on met dans une variable le nom du message d'erreur (par exemple « wrongUsername ») et il sera ensuite passé en GET pour que lorsqu'on recharge la page il s'affiche grâce aux fonctions d'affichages. Si l'utilisateur existe on teste si le mot de passe est bon et s'il n'est pas banni et si c'est tout ok, on met l'id de l'utilisateur dans une variable de SESSION, si ce n'est pas le cas on recharge la page avec les messages d'erreurs.

#### Action signin

Lorsqu'on complète le formulaire d'enregistrement et qu'on appuie sur le bouton « submit » on est redirigé sur la même page avec en GET l'action « signin ». Ce script filtre le texte entré par l'utilisateur et hache le mot de passe en sha1, ensuite on insère l'utilisateur dans la base, s'il n'y a aucun message de la base c'est tout bon on est redirigé, sinon c'est que le nom ou l'email existe déjà, dans ce cas-là on est redirigé avec le message d'erreur de duplication.

#### Action deco

Lorsqu'on est connecté et qu'on clique sur le lien pour se déconnecter, redirigé sur la même page avec en GET l'action « deco ». Ce script ne fait que détruire la SESSION et redirige sur la page utilisateur.

#### Action update

Lorsqu'on complète le formulaire de modification de compte et qu'on appuie sur le bouton « submit » on est redirigé sur la même page avec en GET l'action « update ». Ce script commence par tester si un utilisateur est connecté, si ce n'est pas le cas il redirige sur la page user sans action en GET. Si un utilisateur est connecté on teste si les valeurs en POST existent (si l'utilisateur a rempli le formulaire) si c'est bon on filtre les valeurs et on modifie la base avec. S'il n'y a aucun message de la base c'est tout bon on est redirigé, sinon c'est que le nom ou l'email existe déjà, dans ce cas-là on est redirigé avec le message d'erreur de duplication.

### 3.8.3.3admin.php

#### Actions ban, unban et delete

Ces actions testent d'abord si un id utilisateur a été envoyé, si c'est le cas elles exécutent leurs fonctions du dao respectives de bannissement, de dé-bannissement et de suppression. Il n'y a pas vraiment de sécurité pour les id envoyés car c'est la page admin et l'administrateur n'a pas dans le but de faire planter le site web.

## 4 Tests

N°	Date	Testeur	Description	Résultat obtenu	OK/ KO
1	16/06/2017	Sven W.	Affichage de la liste des utilisateurs sur la page d'accueil.	Les utilisateurs sont listés.	OK
1.1	16/06/2017	Sven W.	Affichage des différents héros triés par rôle sur la page des héros.	Les héros sont affichés et triés.	OK
1.2	16/06/2017	Sven W.	Affichage des informations de bases de n'importe quel héros.	Les 3 héros testés ont leurs informations affichées.	OK
1.3	16/06/2017	Sven W.	Affichage des capacités de n'importe quel héros.	Les 3 héros testés ont leurs capacités affichées.	OK
1.4	16/06/2017	Sven W.	Affichage des objets triés de n'importe quel héros.	Les 3 héros testés ont leurs objets triés affichés.	OK
1.5	16/06/2017	Sven W.	Affichage du tableau d'événement.	Les événements sont affichés	OK
1.6	16/06/2017	Sven W.	Affichage des informations de bases de n'importe quel événement.	Les 2 événements testés ont leurs informations affichées.	OK
1.7	16/06/2017	Sven W.	Affichage des objets triés de n'importe quel événement.	Les 2 événements testés ont leurs objets triés affichés.	OK
1.8	16/06/2017	Sven W.	Affichage des autres objets triés.	Les autres objets sont affichés et triés	OK
1.9	16/06/2017	Sven W.	Affichage du formulaire de connexion lorsqu'on n'est pas connecté	Le formulaire est affiché	OK
1.10	16/06/2017	Sven W.	Affichage du formulaire d'inscription lorsqu'on n'est pas connecté	Le formulaire est affiché	OK
2	16/06/2017	Sven W.	Inscription sans erreur	Retour sur la même page avec un message « account created »	OK
2.1	16/06/2017	Sven W.	Connexion sans erreur	Redirection sur la page du compte	OK
2.2	16/06/2017	Sven W.	Inscription/ connexion avec un ou plusieurs champs vides.	Message d'erreur, le formulaire ne peut pas être envoyé	OK
2.3	16/06/2017	Sven W.	Inscription avec une chaîne qui n'est pas un email valide	Message d'erreur, le formulaire ne peut pas être envoyé	OK
2.4	16/06/2017	Sven W.	Inscription avec un mot de passe de moins de 8 caractères	Message d'erreur, le formulaire ne peut pas être envoyé	OK
2.5	16/06/2017	Sven W.	Inscription avec un nom déjà existant	Retour sur la même page avec le message d'erreur : « Username or email already used »	OK

2.6	16/06/2017	Sven W.	Inscription avec un email déjà existant	Retour sur la même page avec le message d'erreur : « Username or email already used »	OK
2.7	16/06/2017	Sven W.	Connexion avec un nom d'utilisateur non valide	Retour sur la même page avec le message d'erreur : « Username not valid »	OK
2.8	16/06/2017	Sven W.	Connexion avec un nom d'utilisateur valide et un mot de passe non valide	Retour sur la même page avec le message d'erreur : « Wrong password »	OK
2.9	16/06/2017	Sven W.	Connexion avec un utilisateur banni	Retour sur la même page avec le message d'erreur : « Your account has been banned »	OK
3	16/06/2017	Sven W.	Affichage de la page du compte lorsqu'on est connecté	La page du compte est affichée	OK
3.1	16/06/2017	Sven W.	Affichage des informations de base du compte	Les infos de bases sont affichées	OK
3.2	16/06/2017	Sven W.	Affichage du lien vers la page admin si on est admin	Le lien est affiché seulement si l'utilisateur est admin	OK
3.3	16/06/2017	Sven W.	Affichage des barres de progression avec les bons montants	Les barres sont affichées avec les bons montants	OK
3.4	16/06/2017	Sven W.	Affichage du formulaire de modification lorsqu'on clique sur l'image de modification	On est redirigé sur le formulaire	OK
3.5	16/06/2017	Sven W.	Modification du compte sans erreur	La modification marche	OK
3.6	16/06/2017	Sven W.	Modification avec un nom déjà utilisé	Retour sur la page du compte avec un message d'erreur : « Username/Email already used »	OK
3.7	16/06/2017	Sven W.	Modification avec un email déjà utilisé	Retour sur la page du compte avec un message d'erreur : « Username/Email already used »	OK
3.8	16/06/2017	Sven W.	Clique sur un objet qui n'est pas en vert, lorsqu'on est connecté	L'objet est maintenant affiché en vert	OK
3.9	16/06/2017	Sven W.	Clique sur un objet qui est en vert, lorsqu'on est connecté	L'objet n'est plus affiché en vert	OK
3.10	16/06/2017	Sven W.	Affichage des barres de progression avec les montants mis à jour.	Les montants des barres sont mis à jour	OK

3.11	16/06/2017	Sven W.	Déconnexion	Redirection sur la page de connexion	OK
4	16/06/2017	Sven W.	Accès à la page admin pour un admin	Seulement un admin y a accès.	OK
4.1	16/06/2017	Sven W.	Affichage des utilisateurs non bannis	Les utilisateurs non bannis sont affichés	OK
4.2	16/06/2017	Sven W.	Affichage des utilisateurs bannis	Les utilisateurs bannis sont affichés	OK
4.3	16/06/2017	Sven W.	Clique sur le bouton de bannissement	L'utilisateur est maintenant affiché dans la liste des bannis	OK
4.4	16/06/2017	Sven W.	Clique sur le bouton de débannissement	L'utilisateur est maintenant affiché dans la liste des non bannis	OK
4.5	16/06/2017	Sven W.	Clique sur le bouton de suppression	L'utilisateur est supprimé	OK

## 5 Conclusion

### 5.1 Problèmes rencontrés

Étant donné que le but de ce projet était de montrer mes capacités, il n'y avait pas vraiment de nouvelles choses, je savais déjà faire la majorité du projet. Après ça ne m'a pas empêché de rencontrer des problèmes. Mais ce n'était pas des problèmes donc j'ai pu m'en sortir avec l'aide d'internet ou de mes collègues.

### 5.2 Améliorations envisageables

Durant toute la réalisation du projet j'ai eu pas mal d'idées de fonctionnalités qu'on pourrait ajouter, et étant donné que je les aie toutes notées, voici une liste :

- faire un classement des utilisateurs sur la valeur de leurs objets
- faire un système qui permet de voir les statistiques des autres
- faire un système d'achievement
- ajout/suppression/modification des objets par l'admin
- connexion avec le nom d'utilisateur ou le mail
- faire des menus sur les côtés afin de mieux naviguer sur les pages
- afficher le cout des objets manquant sur les pages héros et événements

### 5.3 Comparaison analyse et réalisation finale

Il n'y a pas vraiment de différence entre ce que je voulais faire et le résultat, un peu au niveau des vues, par exemple la page des héros qui est sous forme de tableau plutôt qu'une simple liste. Et c'est plutôt logique s'il n'y a pas de différence étant donné que c'est l'analyse qui m'a permis d'avoir les idées claires sur mon projet et la façon dont j'allais le faire.

### 5.4 Comparaison planning initial et final

Même si le planning initial était simpliste, j'ai bien été dans les temps tout le long, et la répartition entre la programmation et la documentation c'est faites très naturellement, plus facilement que ce

que je pensais. Donc je n'ai pas eu de mal à suivre le planning et on peut le voir dans le planning courant que j'ai fait tous le long du projet :

	Jour 1	Jour 2	Jour 3	Jour 4	Jour 5	Jour 6	Jour 7	Jour 8	Jour 9	Jour 10
	06/06/2017	07/06/2017	08/06/2017	09/06/2017	12/06/2017	13/06/2017	14/06/2017	15/06/2017	16/06/2017	19/06/2017
Programmation										
<i>Web/PHP</i>										
<i>Base de données/Requêtes</i>										
<i>Rangement de code</i>										
<i>Style</i>										
Documentation										
<i>Documentation technique</i>										
<i>Mode d'emploi</i>										
<i>Finalisation documentation</i>										
Journal de bord										
Planning courant										

Figure 28, planning courant

Comme dans le planning initial, j'ai réussi à faire de la documentation un peu tous le long du projet, évidemment sur la fin il y en avait beaucoup plus étant donné qu'il y a certaines parties de la doc qui peuvent être faite uniquement si le travail est terminé. Au niveau de la programmation j'ai réussi à finir le 7eme jour, comme sur le planning initial.

## 5.5 Bilan personnel

Je suis très content d'avoir pu faire ce projet car c'est un projet important pour mon avenir ce qui fait que j'étais stressé pendant tout le long du travail, chose dont je n'ai absolument pas l'habitude, c'était intéressant de sortir un peu de sa zone de confort. Le fait que la gestion du temps soit si importante va, je pense, beaucoup m'aider pour mes projets futurs. C'est aussi grâce à ce projet que j'ai compris l'utilité du journal de bord, c'est très plaisant d'avoir toutes ses idées et tous ses problèmes sur papier, c'est beaucoup plus facile à gérer.



## 6 Webographie

<https://www.tutorialspoint.com/sql/sql-syntax.htm> - Syntaxe sql

<https://stackoverflow.com/questions/36760/sql-query-count-with-o-count> - aide requête count o  
(post de David M Goodwin) – 13/06/2017

<https://playoverwatch.com/fr-fr/> - site du jeu overwatch

<http://eu.blizzard.com/fr-fr/company/about/legal-faq.html> - Faq de blizzard (entreprise  
d'Overwatch) sur les copyrights (pour les images)

## 7 Table des illustrations

Figure 1, soldat 76 skin de base.....	3
Figure 2, soldat 76 skin d'halloween .....	3
Figure 3, maquette préliminaire page d'accueil .....	6
Figure 4, maquette préliminaire page des héros .....	6
Figure 5, maquette préliminaire page d'un héro .....	7
Figure 6, planning initial .....	7
Figure 7, arborescence du site web .....	8
Figure 8, organigramme du site web .....	9
Figure 9, maquette page d'accueil.....	9
Figure 10, maquette page des héros.....	10
Figure 11, maquette page d'un héros.....	10
Figure 12, maquette page des événements .....	11
Figure 13, maquette page d'un événement.....	11
Figure 14, maquette page des autres objets .....	12
Figure 15, maquette page de connexion/inscription .....	12
Figure 16, maquette page du compte .....	13
Figure 17, maquette page administrateur.....	13
Figure 18, modèle relationnel .....	14
Figure 19, algorithme GetMyPdo.....	21
Figure 20, algorithme SelectHeroesInArrayOfRole.....	23
Figure 21, algorithme IsCreatedUserReward .....	24
Figure 22, SelectIdReward ancien format .....	24
Figure 23, SelectIdReward nouveau format.....	24
Figure 24, algorithme SelectRewardsInArrayOfQualityAndTypeByldHero .....	26
Figure 25, algorithme DisplayEvents .....	28
Figure 26, algorithme GetProgressBar .....	29
Figure 27, algorithme clique objets.....	30
Figure 28, planning courant.....	35