**[Nom de la société]**

TPI - 2017

**OverwatchCollection : La documentation technique**

**Sven Wikberg**

Contenu

[1 Introduction 3](#_Toc485288314)

[1.1 Objectif 3](#_Toc485288315)

[2 Analyse fonctionnelle 4](#_Toc485288316)

[2.1 Généralités 4](#_Toc485288317)

[2.1.1 Pourquoi ce sujet ? 4](#_Toc485288318)

[2.1.2 Ce que ce projet à de plus ! 4](#_Toc485288319)

[2.2 Cahier des charges détaillé 4](#_Toc485288320)

[2.2.1 Définition de l’audience 4](#_Toc485288321)

[2.2.2 Définition du contenu et des fonctionnalités 4](#_Toc485288322)

[2.2.3 Maquette préliminaire 5](#_Toc485288323)

[2.3 Description de l’interface 7](#_Toc485288324)

[2.4 Planning initial 7](#_Toc485288325)

[3 Analyse organique 8](#_Toc485288326)

[3.1 Analyse de l’environnement 8](#_Toc485288327)

[3.2 Arborescence du site 8](#_Toc485288328)

[3.3 Organigramme 9](#_Toc485288329)

[3.4 Maquette Graphique 10](#_Toc485288330)

[3.4.1 Page d’accueil 10](#_Toc485288331)

[3.4.2 Page des héros 11](#_Toc485288332)

[3.4.3 Page d’un héros 12](#_Toc485288333)

[3.4.4 Page des évènements 13](#_Toc485288334)

[3.4.5 Page d’un évènement 14](#_Toc485288335)

[3.4.6 Page des autres objets 15](#_Toc485288336)

[3.4.7 Page de connexion/inscription 16](#_Toc485288337)

[3.4.8 Page du compte 17](#_Toc485288338)

[3.4.9 Page administrateur 18](#_Toc485288339)

[3.5 Modèle relationnel 19](#_Toc485288340)

[3.6 Description des tables 19](#_Toc485288341)

[3.6.1 Table users 19](#_Toc485288342)

[3.6.2 Table reward\_types 19](#_Toc485288343)

[3.6.3 Table heroes 20](#_Toc485288344)

[3.6.4 Table user\_rewards 20](#_Toc485288345)

[3.6.5 Table rewards 20](#_Toc485288346)

[3.6.6 Table abilities 20](#_Toc485288347)

[3.6.7 Table roles 21](#_Toc485288348)

[3.6.8 Table currencies 21](#_Toc485288349)

[3.6.9 Table qualities 21](#_Toc485288350)

[3.6.10 Table events 21](#_Toc485288351)

[3.7 Utilisation et explication des requêtes 21](#_Toc485288352)

[3.7.1 Accueil 21](#_Toc485288353)

[3.7.2 Les héros 21](#_Toc485288354)

[3.7.3 Un héros 22](#_Toc485288355)

[3.7.4 Les événements 22](#_Toc485288356)

[3.7.5 Un événement 22](#_Toc485288357)

[3.7.6 Les autres objets 22](#_Toc485288358)

[3.7.7 Affichage d’objet héros/événement/autre objets 22](#_Toc485288359)

[3.7.8 Connexion 23](#_Toc485288360)

[3.7.9 Identification 23](#_Toc485288361)

[3.7.10 Modification du compte 23](#_Toc485288362)

[3.7.11 Mon compte, info et statistiques 23](#_Toc485288363)

[3.7.12 Actions admin 24](#_Toc485288364)

[3.8 Description détaillée de l’architecture du code 24](#_Toc485288365)

[3.8.1 Séparation du code 24](#_Toc485288366)

[3.8.2 Les classes 24](#_Toc485288367)

[3.8.3 Les actions en GET 31](#_Toc485288368)

[4 Webographie 31](#_Toc485288369)

OverwatchCollection :

Documentation technique

# 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 cartoonesques.

Dans Overwatch lorsqu’on commence une partie, il faut d’abord choisir un héros parmi les 23 disponibles, les héros on des capacités et des statistiques différentes. Ils sont regroupé 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 diffèrent 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 , skin de base | Figure , skin d'halloween |

## 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 reconnaitre plus facilement ainsi que d’avoir les statistiques sur ceux-ci.

# Analyse fonctionnelle

## Généralités

### Pourquoi ce sujet ?

Ce sujet est vaut la peine de se pencher dessus car Overwatch est un jeu très populaire et donc il y a une très grande communauté de joueur autour et les joueurs sont en générale très friands d’objets rares et exclusif afin de les montrer aux autres joueurs. C’est donc normal que les objets cosmétiques dans Overwatch aient beaucoup de succès, surtout avec les évènements spéciaux et la sortie de contenue 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 ou 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.

### Ce que ce projet à de plus !

Pour l’instant il n’y a pas vraiment de façon de faire un inventaire des objets existants ainsi que des objets obtenu 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 fait un inventaire de ses objets parmi tous les objets existants dans le jeu.

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

## Cahier des charges détaillé

### 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.

### 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 connecter sur ce compte. Et finalement la partie « administrateur » qui regroupe les fonctionnalités accessible en tant qu’administrateur.

#### Toutes catégories

* affichage des utilisateurs du site
* affichage des objets cosmétiques d’Overwatch par héro
* affichage des objets cosmétiques d’Overwatch par événement
* affichage des objets cosmétiques d’Overwatch qui ne sont lié à aucun héro
* affichage des informations de base des héros
* affichage des informations de base des événements

#### Invité

* création un compte
* identification

#### Utilisateur

* possibilité cliquer sur n’importe quel objet cosmétique non sélectionné afin de le sélectionner
* possibilité 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 reconnaitre
* 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

### Maquette préliminaire

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

#### Page d’accueil

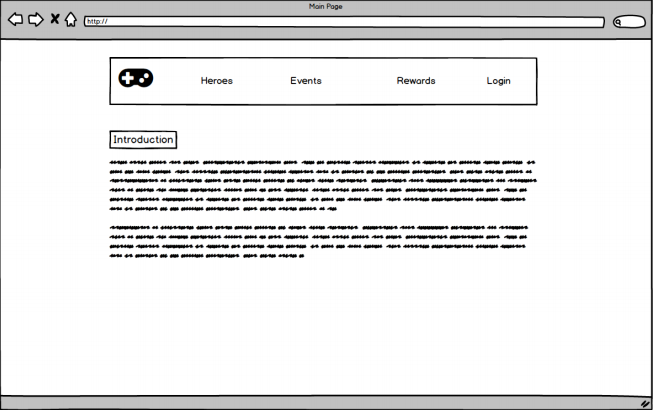


Figure , maquette préliminaire page d'accueil

#### Page des héros



Figure , maquette préliminaire page des héros

#### Page d’un héro

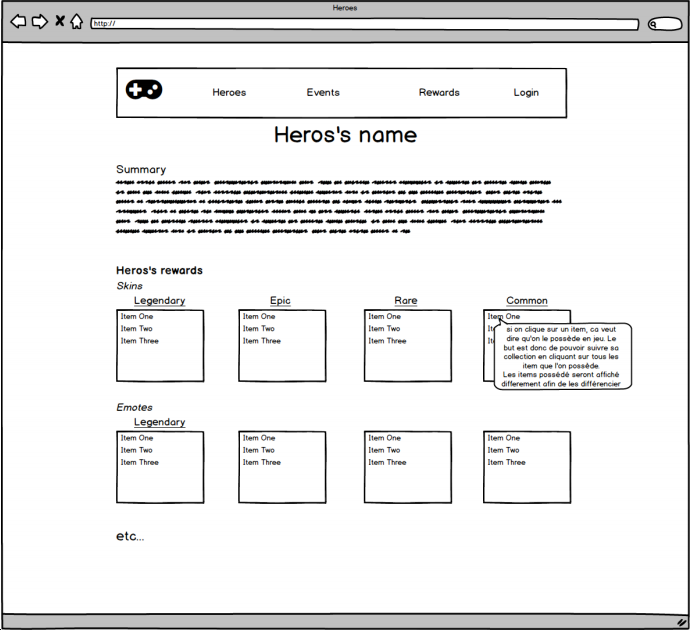


Figure , maquette préliminaire page d'un héro

## Description de l’interface

Comme on peut le voir sur les maquettes préliminaires de quelques pages, ci-dessus, l’idée est d’affiché les objets du jeu triés de différentes facon, par

## 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 oublié la doc.

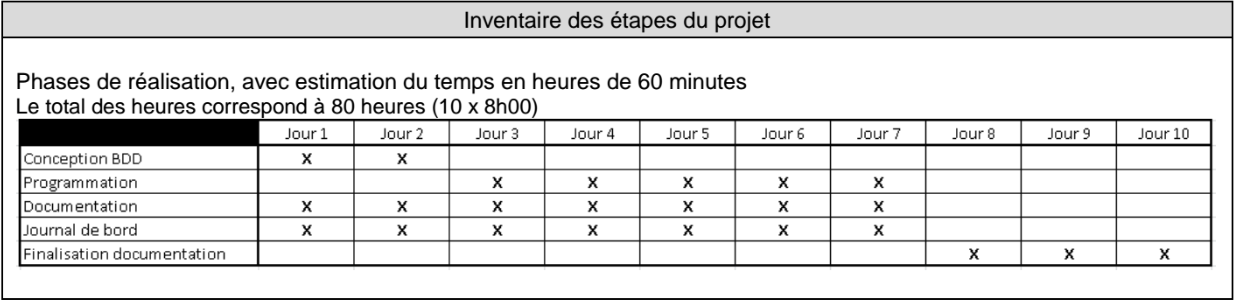


Figure , planning initial

# Analyse organique

## 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 elle 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 a utilisé 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.

## Arborescence du site

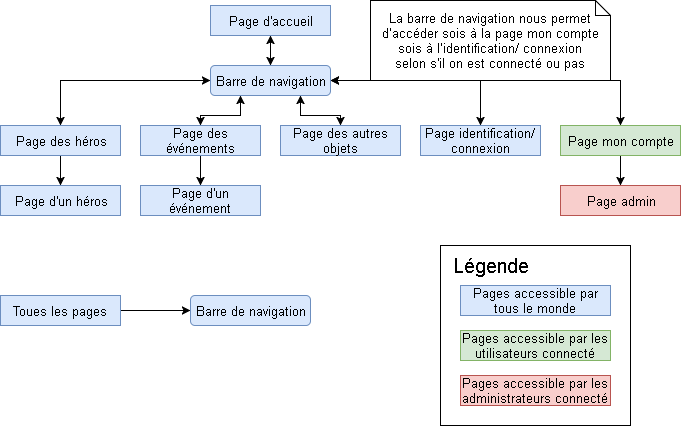


Figure , arborescence du site web

## Organigramme



Figure , organigramme du site web

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

## Maquette Graphique

### Page d’accueil

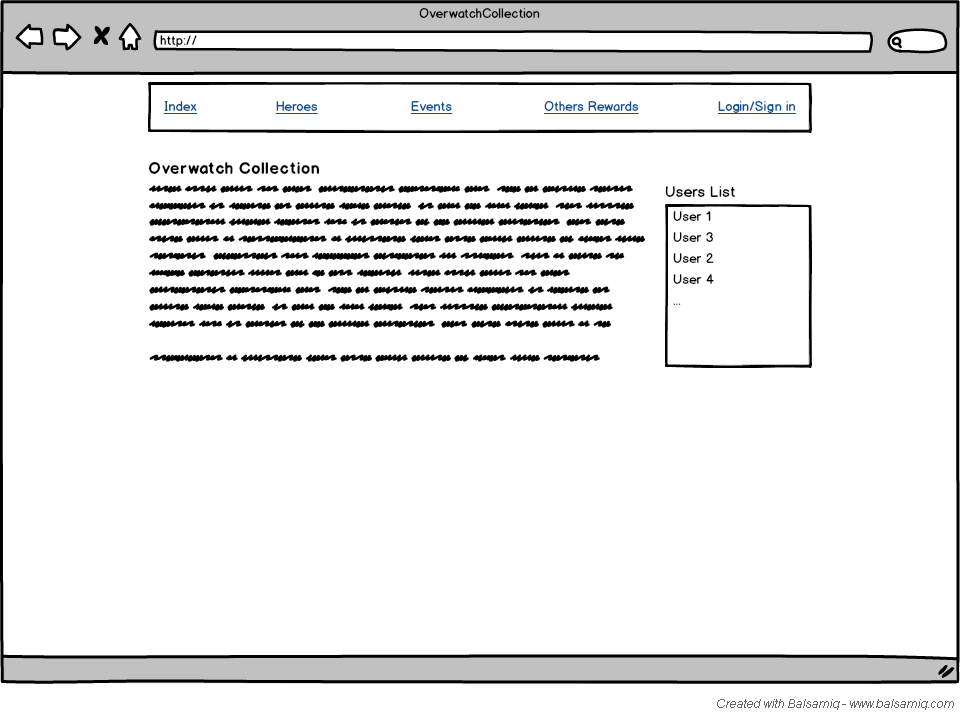


Figure , maquette page d'accueil

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.

### Page des héros

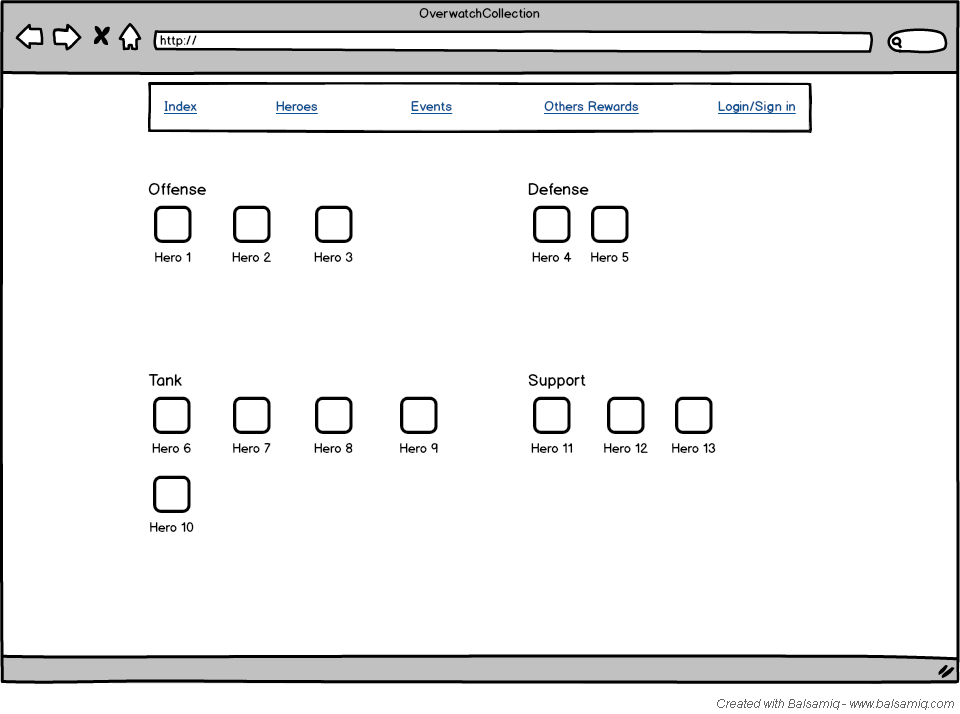


Figure , maquette page des héros

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

### Page d’un héros

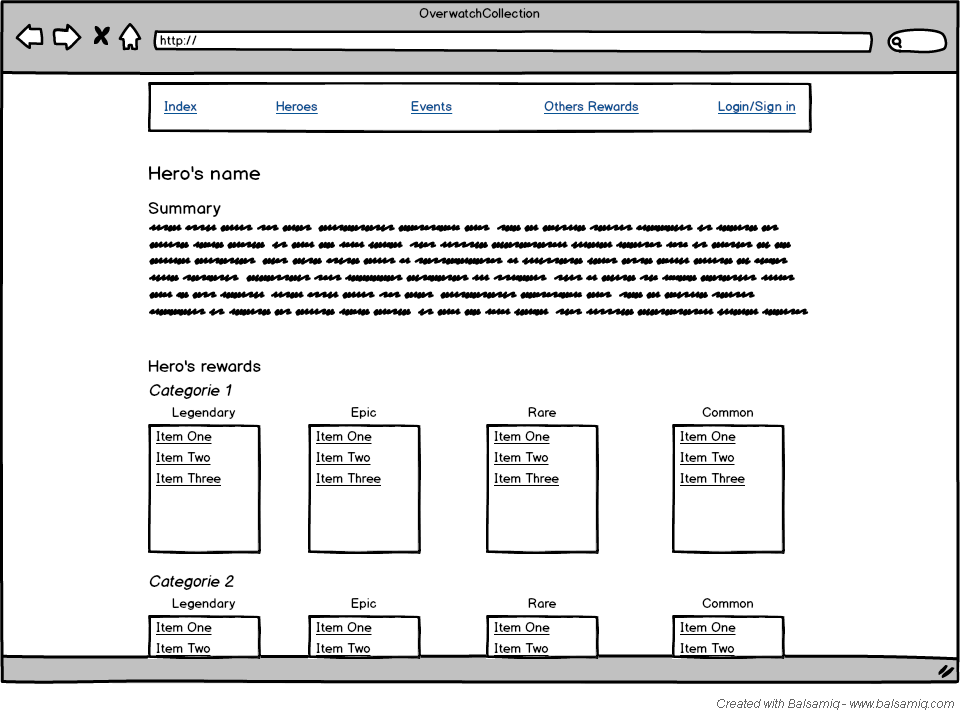


Figure , maquette page d'un héros

Cette page affiche quelques informations sur le héros en question ainsi que ces capacités, mais le plus important étant évidemment l’affichage des objets cosmétiques triées par catégorie et par qualité afin de les trouvé plus facilement. Si l’utilisateur est connecté les objets seront affiché comme des liens sur lesquels il pourra cliquer afin de marqué l’objet comme acquis.

### Page des évènements

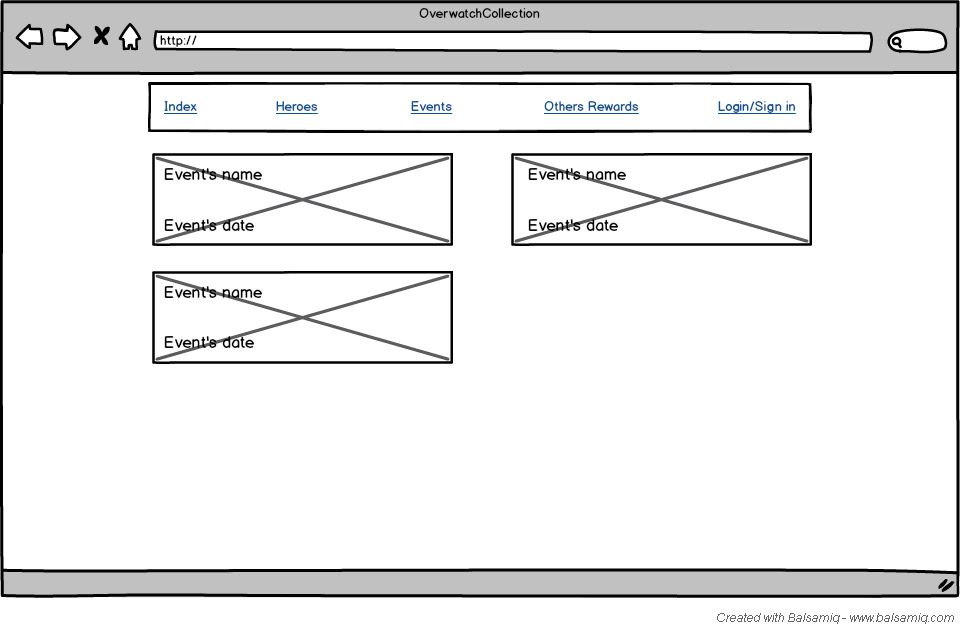


Figure , maquette page des événements

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.

### Page d’un évènement

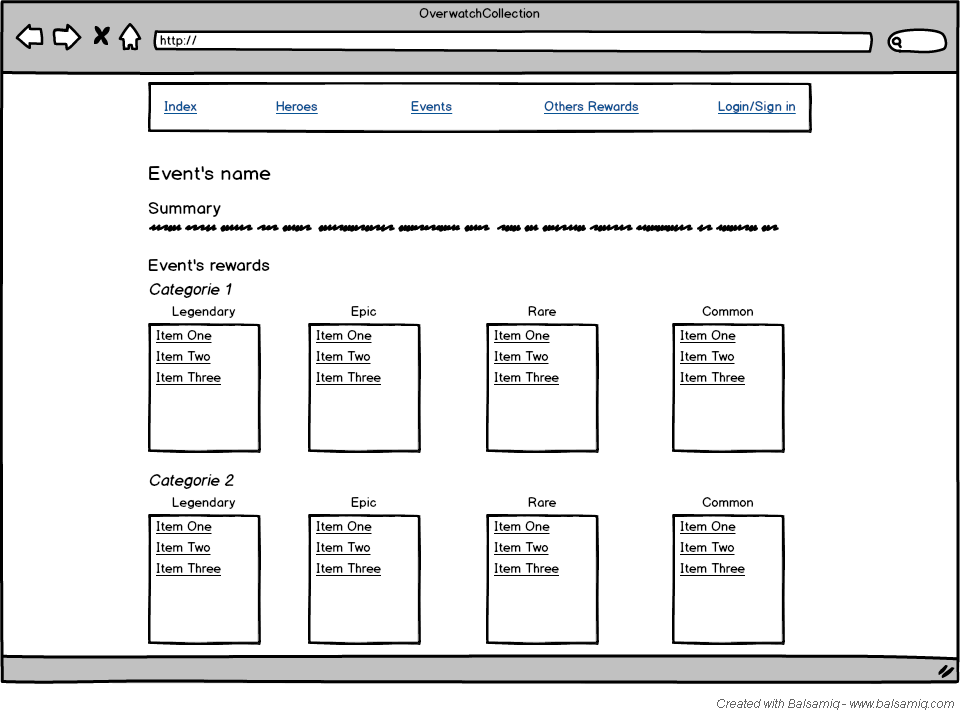


Figure , maquette page d'un événement

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

### Page des autres objets

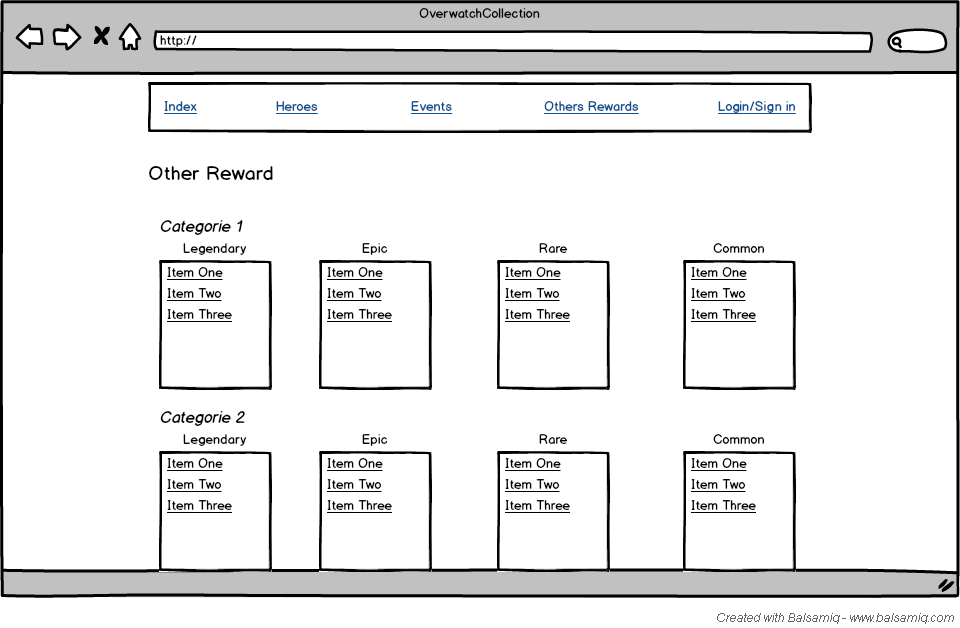


Figure , maquette page des autres objets

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

### Page de connexion/inscription

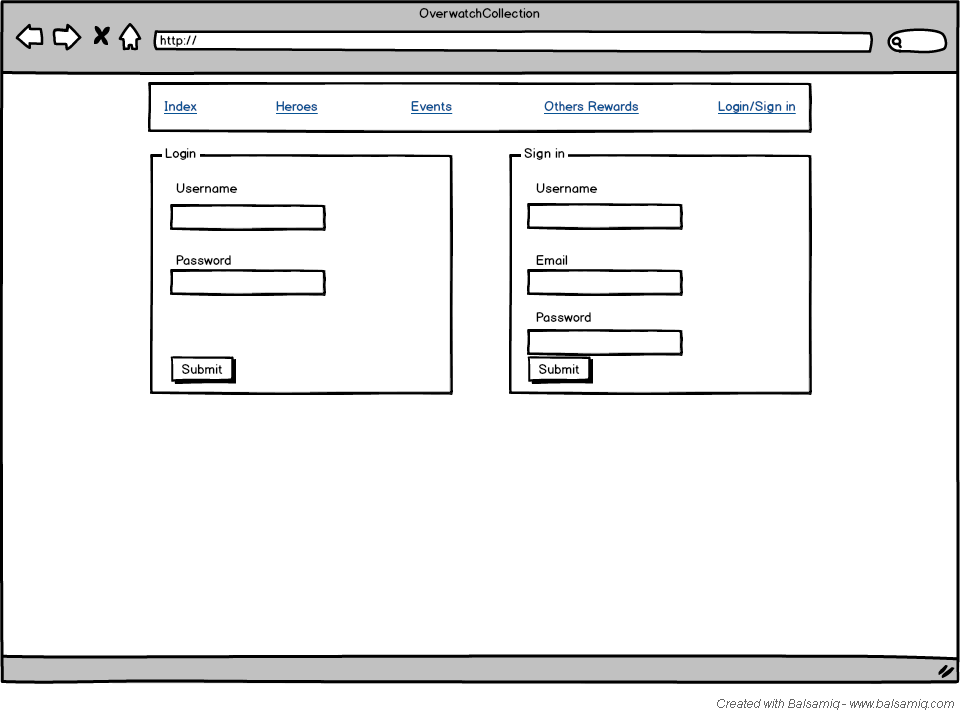


Figure , maquette page de connexion/inscription

Voici la page sur laquelle l’utilisateur va pouvoir soit se connecter, soit créé un compte pour se connecter.

### Page du compte

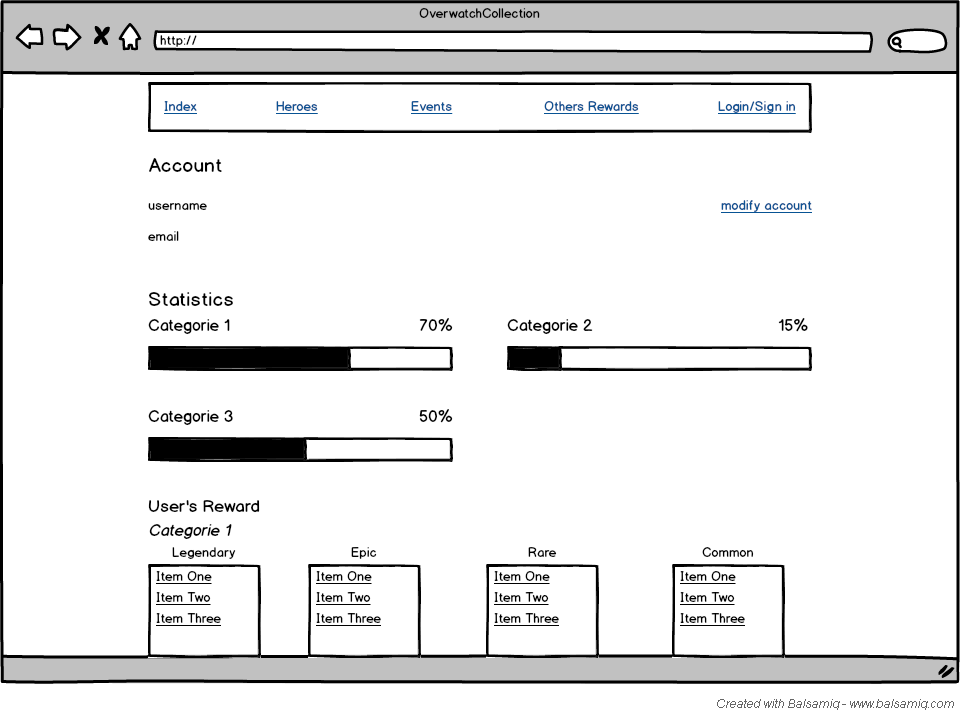


Figure , maquette page du compte

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 et qu’il pourra à voir son inventaire, c’est-à-dire tous les objets qu’il a sélectionné.

### Page administrateur

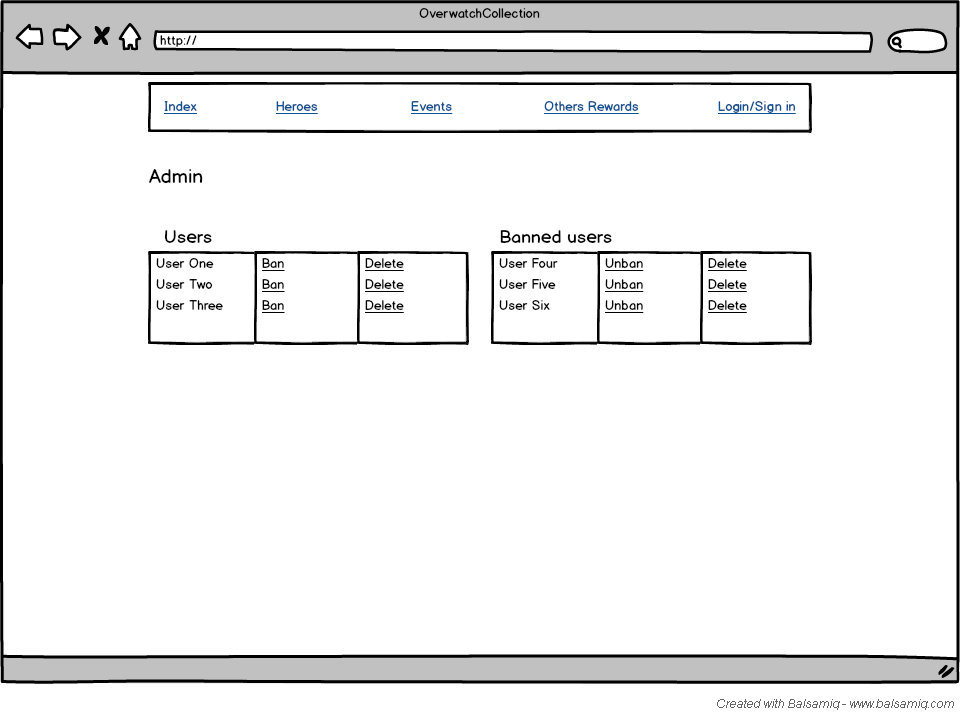


Figure , maquette page administrateur

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

## Modèle relationnel

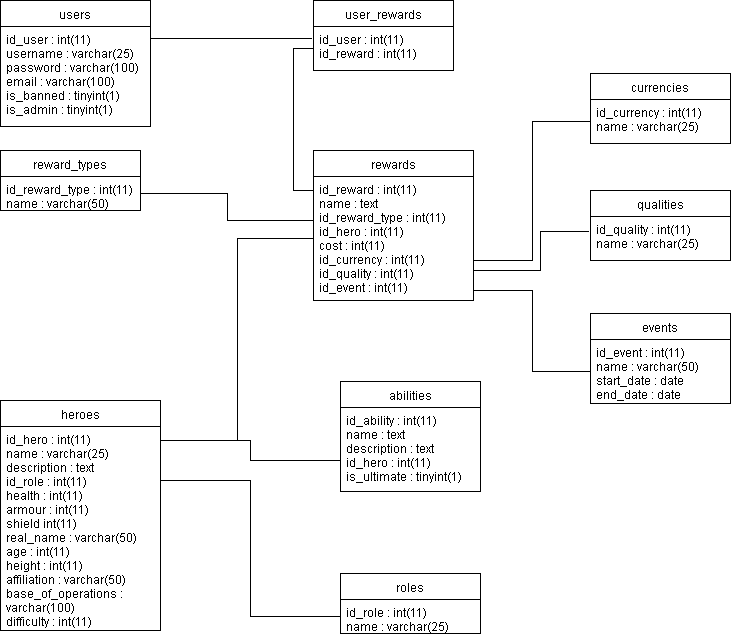


Figure , modèle relationnel

## Description des tables

### Table users

|  |  |  |
| --- | --- | --- |
| Nom du champ | Type | Description |
| id\_user | int(11), Non null, AI, PK | Identifiant de l’utilisateur |
| username | varchar(25), Non null | Nom de l’utilisateur |
| password | varchar(100), Non null | Mot de passe de l’utilisateur |
| email | varchar(100), Non null | Email de l’utilisateur |
| is\_banned | tinyint(1), Non null | Définit si l’utilisateur est banni ou pas |
| is\_admin | tinyint(1), Non null | Définit si l’utilisateur est admin ou pas |

### Table reward\_types

|  |  |  |
| --- | --- | --- |
| Nom du champ | Type | Description |
| id\_reward\_type | int(11), Non null, AI, PK | Identifiant de la catégorie d’objet cosmétique |
| name | varchar(50), Non null | Nom de la catégorie de cosmétique |

### Table heroes

|  |  |  |
| --- | --- | --- |
| Nom du champ | Type | Description |
| id\_hero | int(11), Non null, AI, PK | Identifiant du héros |
| name | varchar(25), Non null | Nom/Pseudo du héros |
| description | text, Non null | Description du héros |
| id\_role | int(11), Non null, FK | Identifiant du rôle du héros |
| health | int(11), Non null | Points de vie du héros |
| armour | int(11), Non null | Points d’armure du héros |
| shield | int(11), Non null | Points de bouclier du héros |
| real\_name | varchar(50), Non null | Vrai nom (et prénom) du héros |
| age | int(11), Non null | Age du héros |
| height | int(11), Null | Taille, en cm, du héros |
| affiliation | varchar(50), Null | Organisation du héros |
| base\_of\_operations | varchar(100), Null | Base d’opération/ Lieu d’habitation du héros |
| difficulty | int(11), Non null | Difficulté du héros, en jeu |

### Table user\_rewards

|  |  |  |
| --- | --- | --- |
| Nom du champ | Type | Description |
| id\_user | int(11), Non null, FK |  |
| id\_reward | int(11), Non null, FK |  |

### Table rewards

|  |  |  |
| --- | --- | --- |
| Nom du champ | Type | Description |
| id\_reward | int(11), Non null, AI, PK | Identifiant de l’objet cosmétique |
| name | text, Non null | Nom de l’objet cosmétique |
| id\_reward\_type | int(11), Non null, FK | Identifiant de la catégorie de l’objet cosmétique |
| id\_hero | int(11), Null, FK | Identifiant du héros de l’objet cométique |
| cost | int(11), Null | Cout de l’objet cosmétique |
| id\_currency | int(11), Null, FK | Identifiant de la monnaie de l’objet cosmétique |
| id\_quality | int(11), Non null, FK | Identifiant de la qualité de l’objet cosmétique |
| id\_event | int(11), Null, FK | Identifiant de l’événement de l’objet cosmétique |

### Table abilities

|  |  |  |
| --- | --- | --- |
| Nom du champ | Type | Description |
| id\_abilities | int(11), Non null, AI, PK | Identifiant de la capacité |
| name | text, Non null | Nom de la capacité |
| description | text, Non null | Description de la capacité |
| id\_hero | int(11), Non null, FK | Identifiant du héros à qui appartient la capacité |
| is\_ultimate | tinyint(1), Non null | Définit si la capacité est une capacité ultime ou pas |

### Table roles

|  |  |  |
| --- | --- | --- |
| Nom du champ | Type | Description |
| id\_role | int(11), Non null, AI, PK | Identifiant du rôle |
| name | varchar(25), Non null | Nom du rôle |

### Table currencies

|  |  |  |
| --- | --- | --- |
| Nom du champ | Type | Description |
| id\_currency | int(11), Non null, AI, PK | Identifiant de la monnaie |
| name | varchar(25), Non null | Nom de la monnaie |

### Table qualities

|  |  |  |
| --- | --- | --- |
| Nom du champ | Type | Description |
| id\_quality | int(11), Non null, AI, PK | Identifiant de la rareté/qualité |
| name | varchar(25), Non null | Nom de la rareté/qualité |

### Table events

|  |  |  |
| --- | --- | --- |
| Nom du champ | Type | Description |
| id\_event | int(11), Non null, AI, PK | Identifiant de l’événement |
| name | varchar(50), Non null | Nom de l’événement |
| start\_date | date, Non null | Date de début de l’événement |
| end\_date | date, Non null | Date de fin de l’événement |

## Utilisation et explication des requêtes

Afin que le site soit dynamique, il faut obligatoirement que les données soient récupérer 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 traité dans la partie affichage.*

### Accueil

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

### Les héros

Sur la page héros on récupère d’abord les rôles existants 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 correspondants 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 leur rôle respectifs.

### 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). A partie de là, on récupère, pour chaque combinaison de catégorie/rareté, tous les objets dont l’identifiant de la catégorie ainsi que de la rareté correspondent (« SELECT » de la table objets, avec les conditions citées) afin de les mettre dans un tableau pour qu’ils soient trié par catégorie et rareté (vu qu’on veut récupérer les objets d’un héros, il faut aussi que l’id du héros corresponde).

### Les événements

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

### 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 cette évènement.

Donc pour les informations de base on fait un « SELECT » dans la table évènement 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 évènements ont parfois des objets avec le même nom mais qui correspondent à différent 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 quelle objet il est question.

Mais il y avait un problème qui était que pas tous les objets ont des héros associé 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 sois récupérer, la solution était de faire une jointure externe sur la table héros lorsqu’on la lie avec la table objets cosmétiques.

### Les autres objets

Sur cette page on affiche seulement les objets qui ne correspondent à aucun héro, paradoxalement on récupère les données de la même façon que pour la page héro à la seule différence que aux lieux de récupérer les objets avec un identifiant héros, on récupère les objets dont le champ identifiant héro et égal à zéro ou est nulle.

### Affichage d’objet héros/événement/autre objets

Lorsqu’on affiche un objet sur une de ses trois page et qu’un utilisateur est connecter 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 questions (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.

### 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 récupére 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.

### Identification

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

### Modification du compte

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

### 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’objet 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’objet de l’évènement 1 en tout et le nombre d’objet de l’évènement 1 que l’utilisateur a sélectionné. Ces données sont récupérer en faisant un « SELECT » de tous les objets de l’évènement ou 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’objet par héro ou par événement pour un utilisateur, si l’utilisateur n’avait aucun objet d’un héros ou d’un événement 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 événement, même si l’utilisateur n’a aucun de leurs objets. Afin de résoudre ce problème plusieurs solution ont été essayé, 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 partis des héros de la première requête.

### 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 donnée.

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.

## Description détaillée de l’architecture du code

### 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 trouve 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.

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.

Apres 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’ajouté ou d’enlever un objet a un utilisateur, cette fonction étant utilisé sur plusieurs pages, elle se trouve dans un fichier séparé.

Et finalement il y a le dossier « class », qui contient les classes utilisé sur le site web. C’est là que ce trouve 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 regroupé les fonctions du même type.

### Les classes

#### MyPdo

MyPdo est la classe qui sert à récupérer une instance e l’objet PDO. L’objet PDO étant un objet qui permet d’accéder a une base de données de puis 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 objets 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 cette 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.

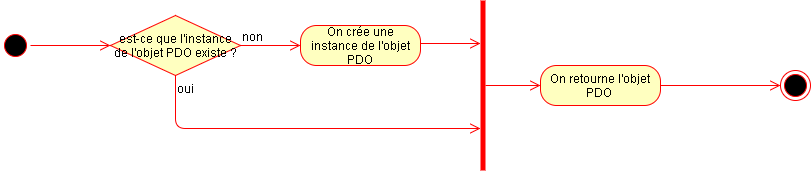


Figure , algorigramm GetMyPdo

#### 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.

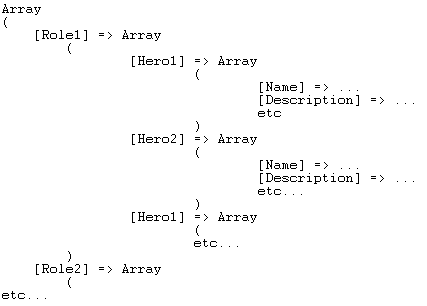
|  |  |
| --- | --- |
| Nom | Description |
| SelectHeroes() | Retourne tous les héros |
| SelectHeroById($id) | Retourne un héros, sélectionné avec son id |
| SelectRoles() | Retourne tous les rôles |
| SelectQualities() | Retourne toutes les qualités d’objets |
| SelectRewardTypes() | Retourne tous les types d’objets |
| SelectEvents() | Retourne tous les évènements |
| SelectEventById($id) | Retourne un événement sélectionné avec son id |
| SelectUsers() | Retourne tous les utilisateurs |
| SelectCleanUsers() | Retourne tous les utilisateurs dont le champ is\_banned est false |
| SelectCleanUsersNoAdmin() | Retourne tous les utilisateurs dont les champs is\_banned et is\_admin sont false |
| SelectBannedUsers() | Retourne tous les utilisateurs dont le champ is\_banned est true et is\_admin et false |
| SelectUserByUsername($username) | Retourne un utilisateur, sélectionné avec son nom |
| SelectUserById($id) | Retourne un utilisateur, sélectionné avec son id |
| SelectAbilitiesByIdHero($id) | Retourne les capacités d’un héros. Elles sont sélectionnées avec l’id du héros. |
| SelectIdRewardsByIdHeroAndIdUser($id\_hero, $id\_user) | Retourne les id d’objets cosmétiques. Ils sont sélectionné avec l’id d’un héros et l’id d’un utilisateur testé sur la table de liaison users\_rewards (liée dans la requête a la table rewards) |
| SelectIdRewardsByIdEventAndIdUser($id\_event, $id\_user) | Retourne les id d’objets cosmétiques. Ils sont sélectionné avec l’id d’un événement et l’id d’un utilisateur testé sur la table de liaison users\_rewards (liée dans la requête a la table rewards) |
| SelectIdRewardsByNoIdHeroAndIdUser($id\_user) | Retourne les id d’objets cosmétiques. Ils sont sélectionné 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 a la table rewards) |
| SelectCountReward() | Retourne le nombre d’id d’objets cosmétique dans la table rewards. |
| SelectCountRewardByIdUser($id) | Retourne le nombre d’id d’objets cosmétique dans la table users\_rewards, pour un id d’utilisateur donné |
| SelectCountRewardEvents() | Retourne le nom de tous les événements ainsi que le nombre d’id d’objets que chaque évènement a. Le nom de l’événement est récupérer en liant la table objets cosmétiques avec la table évènement sur l’id des évènements. |
| SelectCountRewardHeroes() | Retourne le nom de tous les héros ainsi que le nombre d’id d’objets que chaque héro a. Le nom du héros est récupérer 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.

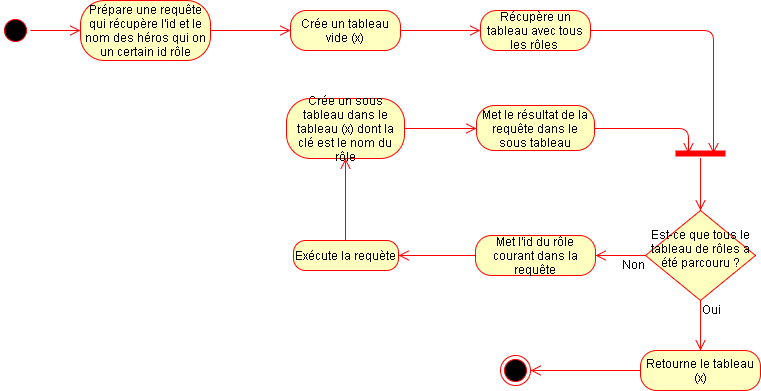
|  |  |
| --- | --- |
| Nom | Description |
| BanUserById($id) | Modifie la valeur du champ is\_banned et la passe à true. L’utilisateur est sélectionné avec son id. |
| UnbanUserById($id) | Modifie la valeur du champ is\_banned et la passe à false. L’utilisateur est sélectionné avec son id. |
| DeleteUserById($id) | Supprime un utilisateur. L’utilisateur est sélectionné avec son id. |
| UpdateUserByIdNoPwd($id, $username, $email) | Modifie le nom et l’email d’un utilisateur. L’utilisateur est sélectionné avec son id. |
| InsertUserReward($id\_user, $id\_reward) | Ajoute un champ à la table de liaison users\_rewards. |
| DeleteUserReward($id\_user, $id\_reward) | Supprime un champ à la table de liaison users\_rewards. |
| InsertUser($username, $email, $password) | 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 : 

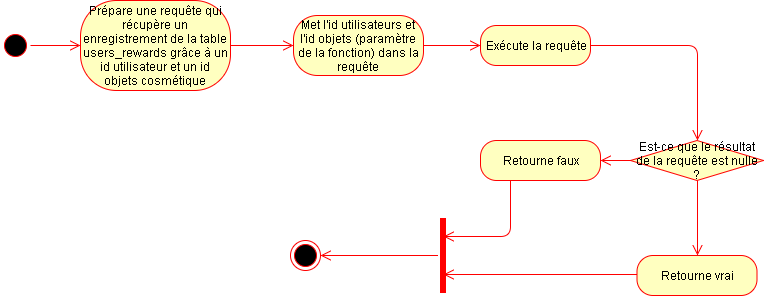
Fonctionnement de la fonction :



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

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

Fonctionnement de la fonction :



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

Cette fonction sert à récupérer une liste d’identifiants d’objets cosmétique qui sont utilisé 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 de puis la base car le format récupéré n’était pas idéal, il est donc transformé et un tableau beaucoup plus simple :

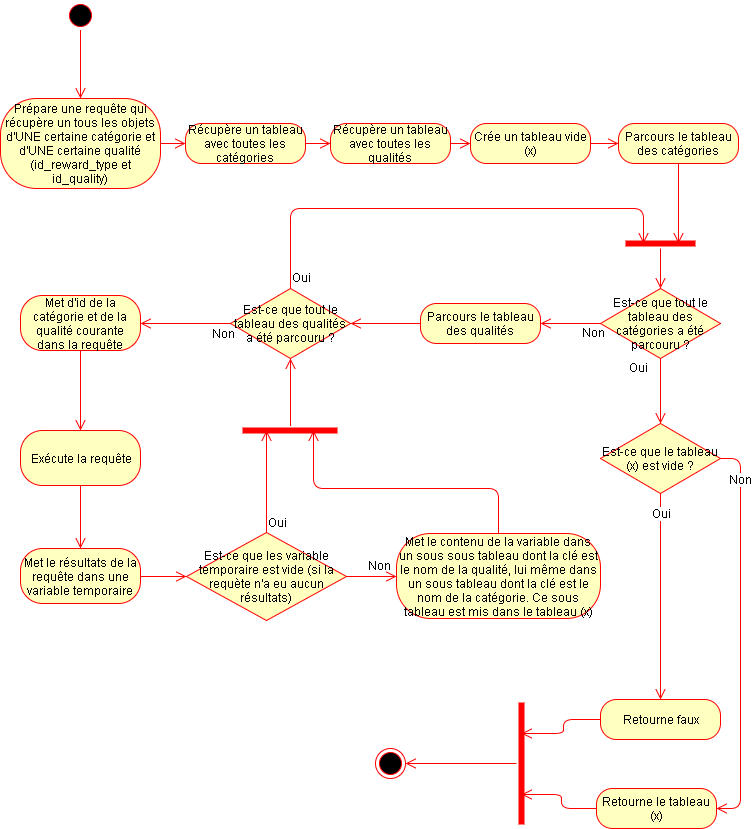
|  |  |
| --- | --- |
| C:\Users\wikbergs_info\Desktop\OverwatchCollection\doc\OverwatchCollection_DocTechnique\ressources\ExempleTableau2.PNG  Figure , SelectIdReward ancien format | C:\Users\wikbergs_info\Desktop\OverwatchCollection\doc\OverwatchCollection_DocTechnique\ressources\ExempleTableau3.PNG  Figure , SelectIdReward nouveau format |

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

##### SelectRewardsInArrayOfQualityAndTypeByIdHero($id)

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

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

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).

#### OcDisplay

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 affiche 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 gardé le login, donc on le remplace par « My Account », d’ailleur la fonction detecter si l’utilisateur se trouve sur la page de son compte, elle afficher « Log out » a 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 leur statuts.

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

Ces fonctions sont utilisées dans la page admin. La première sert à affiché une liste des utilisateurs non-bannis avec un bouton à coté de leur nom d’utilisateur qui permet à l’administrateur de les bannir, la deuxième affiche une liste des utilisateurs bannis avec deux bouton 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()

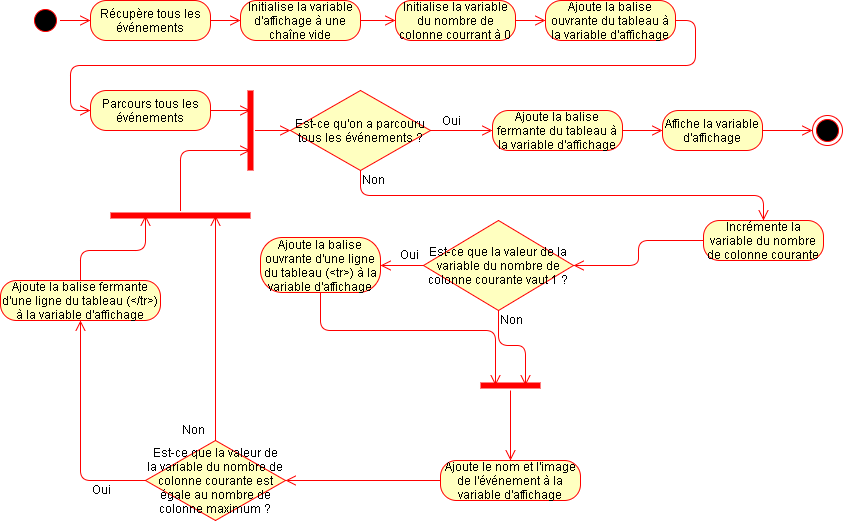
Ces fonctions sont les fonctions qui affichent les objets cosmétiques dans des listes triées. La première s’occupe des objets d’un événement, la deuxième des objets d’un héros et la troisième des objets liés n’a aucun héros.

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, si il n’y a pas d’utilisateur connecter, la liste sera vide donc pas besoin de tester si un utilisateur est connecter. S’il fait partie de la liste de l’utilisateur on met dans un variable le code CSS pour affiche 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’affiché le nom de l’objet, il faut aussi tester si un utilisateur est connecter 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 connecter on affiche uniquement du texte.

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

Voici la fonction qui permet d’afficher un tableau contenant les évènements sur la page évènements. Le paramètre $nb\_col\_max sert à définir les nombre de colonnes que l’on veut pour le tableau. Fonctionnement :

##### 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’évènements 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 précédente pour les colonnes du tableau, mais cette fois le nombre de colonne maximum n’est pas en paramètre car les héros auront toujours à peu près le même nombre de capacité, donc pas besoin de la changer.

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

Ces fonctions sont des fonctions d’affichage basique, 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 envoie 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 d’héros ou d’événements.

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

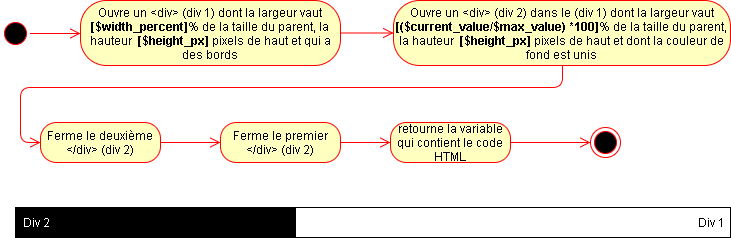
##### DisplayLogin(),DisplaySignIn() et DisplayAccountInfoUpdating($id\_user)

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 chaine 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, du la barre de progression et $height\_px est la hauteur, en pixel, de la barre de progression.

Fonctionnement : 

##### 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’objet 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 progressions) et les met toutes dans une section.

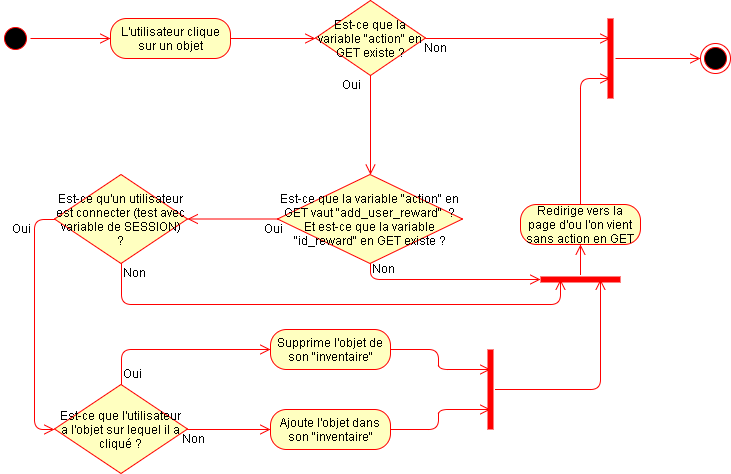
### Les actions en GET

Certaines fonctions de ce site web se font seulement lorsque des évènements particuliers sont enclenché, 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 rediriger 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.

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

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



#### user.php

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

##### Action login

Lorsqu’on complète le formulaire de connexion et qu’on appuis sur le bouton « submit », on est redirigé sur la même page avec en GET l’action « login ». Ce script commence par haché 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 test 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 un 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 appuis sur le bouton « submit » on est redirigé sur la même page avec en GET l’action « signin ». Ce script filtre le texte entré pas l’utilisateur et hache le mot de passe en sha1, ensuite on insère l’utilisateur dans la base, s’il n’y aucun message de la base c’est tous 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 connecter et qu’on clique sur le lien pour ce 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 appuis 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 existe (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 aucun message de la base c’est tous 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.

#### admin.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 envoyé car c’est la page admin et l’administrateur n’a pas dans le but de faire planté le site web.

# Webographie

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

<https://stackoverflow.com/questions/36760/sql-query-count-with-0-count> - aide requête count 0 (post de David M Goodwin) – 13/06/2017