Documentation des API

Chapus Louka

Table des matières

[1. Introduction 3](#_Toc135474968)

[2. Javascript 3](#_Toc135474969)

[2.1 Gestion\_utilisateur 3](#_Toc135474970)

[2.2 Gestion\_galaxie 3](#_Toc135474971)

[2.3 Planete\_nav 3](#_Toc135474972)

[3. Script PHP 3](#_Toc135474973)

[3.1 Selection\_univers 3](#_Toc135474974)

[3.2 Creer\_utilisateur 4](#_Toc135474975)

[3.3 Afficher\_planete 4](#_Toc135474976)

[3.4 Possession\_planete 4](#_Toc135474977)

[3.5 Attribuer\_planete 4](#_Toc135474978)

[3.6 Envoi\_ressources 4](#_Toc135474979)

[3.7 Recuperation\_ressources 4](#_Toc135474980)

[3.8 Afficher\_planete\_nav 4](#_Toc135474981)

# Introduction

Le but de ce document est de documenter les actions faites par les différents scripts sur les pages. Pour chaque script il y aura une description détaillée de son utilité dans le bon fonctionnement du jeu.

# Javascript

Pour chaque javascript on décrit le fonctionnement sans forcément rentrer dans les détails de l’implémentation, on explicite aussi quel autre script sont appelés dans un script.

## Gestion\_utilisateur

Ce script va ajouter les options de sélection dans la liste déroulante de la page index en appelant le script [Selection\_univers](#_3.1_Selection_univers). Puis il ajoute des EventListener sur le bouton de connexion et sur le bouton de création de compte. Lorsqu’on appui sur le bouton de connexion on appelle le script connexion.php et on met à jour les classes des différents messages d’erreurs si besoin. Pour le bouton de création de compte cette fois-ci on appelle le script [Créer\_utilisateur](#_3.2_Créer_utilisateur) et on met aussi à jour les messages d’erreurs si besoin.

## Gestion\_galaxie

Ce script est appelé dans la page Galaxie. Il commence par ajouter des EventListener sur les barres de sélection de galaxie et de système solaire, si les deux barres sont non vides en même temps alors on fait appel au script [Afficher\_planete](#_3.3_Afficher_planete) avec la valeur de la galaxie et celle du système solaire sélectionner en argument.

## Planete\_nav

Ce script est appelé dans toutes les pages (sauf la page index). Il permet d’attribuer une planète à un joueur et ensuite d’afficher ses « coordonnées » en haut à gauche de chaque page, pour que le joueur connaisse l’emplacement de sa planète. Il permet aussi de récupérer la quantité de ressource présente dans la base de données. Tout d’abord on regarde si le joueur possède une planète avec le script [Possession\_planete](#_3.4_Possession_planete), s’il n’en n’a pas alors on lui en attribue une grâce au script [Attribuer\_planete](#_3.5_Attribuer_planete). Puis on envoie les ressources de base à la base de données avec le script [Envoi\_ressources](#_3.6_Envoi_ressources). Ensuite si l’utilisateur possède déjà une planète alors on va simplement afficher ses « coordonnée » à l’aide su script [Afficher\_planete\_nav](#_3.7_Afficher_planete_nav), puis on récupère les ressources dans la base de données et on les affiche avec le script [Recuperation\_ressources](#_3.7_Recuperation_ressources).

# Script PHP

Tous les scripts php ou presque font appelle au script pdo qui permet de se connecter à la base de données avec l’utilisateur esirem et mot de passe esirem il crée donc le PDO et la variable db.

## 3.1 Selection\_univers

On fait une requête sql pour récupérer les noms de tous les univers stockés dans la table univers de la base de données, puis on ajoute toutes les options dans une variable texte que l’on renvoie à la fin du script.

## 3.2 Creer\_utilisateur

On commence par regarder la méthode utilisé (ici POST), puis on regarde si un utilisateur avec le même mail existe déjà, sinon on l’ajoute à la base de données en hachant le mot de passe pour pas qu’il soit visible en claire.

## 3.3 Afficher\_planete

On commence par regarder la méthode utilisé (ici GET), puis on récupère les différents id nécessaire pour faire l’affichage. Ensuite, on fait une boucle for pour chaque planète dans la base de données avec la bonne galaxie et le bon système solaire, s’il y a une planète alors on affiche ses informations sinon on affiche une case vide. On revoie le tableau des planètes rempli avec les bonnes informations.

## 3.4 Possession\_planete

On commence par regarder la méthode utilisé (ici GET), puis on récupère les id nécessaires pour faire la requête ou on compte le nombre de planète possède un joueur, si le joueur ne possède aucune planète on renvoie false, sinon on renvoie true.

## 3.5 Attribuer\_planete

On commence par regarder la méthode utilisé (ici GET), puis on récupère les id nécessaires. On récupère l’id de la première planète et de la dernière planète d’un univers et ensuite on tire aléatoirement un nombre entre les deux id tout en vérifiant que la planète n’est pas déjà attribuée à un autre joueur. Ensuite on récupère les « coordonnées » de la planète que l’on stocke dans la variable de session et on initialise les différentes valeurs pour un nouveau joueur comme les ressources ou le bâtiment niveau 0. On finit par renvoyer les « coordonnées » de la planète.

## 3.6 Envoi\_ressources

On récupère les id nécessaires, puis on regarde si l’utilisateur possède déjà des ressources. Si c’est le cas on met simplement à jour les données, sinon on crée la ligne dans le tableau ressources. Et à la fin on appelle le script [Recuperation\_ressources](#_3.7_Recuperation_ressources) pour faire l’affichage des ressources.

## 3.7 Recuperation\_ressources

On récupère les id nécessaires, puis on récupère les ressources stockées dans la base de données on va ensuite les mettre dans les différentes variables de la session. Puis on fait l’affichage avec les bonnes données et on renvoie le texte html.

## 3.8 Afficher\_planete\_nav

On récupère les id nécessaires, puis on tronque les noms en ne gardant que le numéro des différentes données et on les arrange pour que l’affichage soit fait correctement.