IG3

Projet Web

Organisation de tournois de jeux vidéo

http://projet-web-mahe.herokuapp.com/
http://github.com/mspaenle/website/

Mahé Spaenlé
2016/2017

Table des matières

1 PROJET

Ce site internet, mis en place dans le cadre du cours de web en IG 3 (2016/2017), a pour but de faciliter l'organisation et l'enregistrement des scores lors d'un tournois de jeux vidéo dans le cercle familiale ou des amis.

1.1 DESCRIPTIF ET ORIGINES DU PROJET INITIAL

Lors de l'organisation d'un tournois de jeux entre amis, il faut toujours quelqu'un qui prépare à l'avance les pools en fonction du nombre de participants pour savoir combien de manches organiser, et en fonction des points, qui passera à la manche suivante. C'est toujours une perte d'argent et il est toujours possible de contester les scores.

Le but de ce site est donc d'organiser un tournois « maison » de A à Z. Créer un tournois en commençant par entrer des données telles que les joueurs, le jeu et le mode de jeu. En fonction du jeu et du mode de jeu, des manches vont être faites de sorte que personne ne se retrouve à jouer seul. Au fur et à mesure de la partie, les joueurs pourront entrer leur score, et ainsi, en passant à la manche suivante, les nouveaux groupes de joueurs vont être calculés.

1.2 PROJET PRESENTE

Le site présenté est le stade initial de ce projet global. Il s'agit d'organiser un tournois avec différents paramètres : nombre de joueurs, jeu. Et à partir de là, avoir la possibilité de mettre à jour le score des joueurs dans le temps. Un administrateur à accès aux jeux, peut en ajouter ou en supprimer. Cette option est importante dans l'évolution du projet.

Il doit aussi permettre de voir les scores des joueurs enregistrés dans la base de données du site.

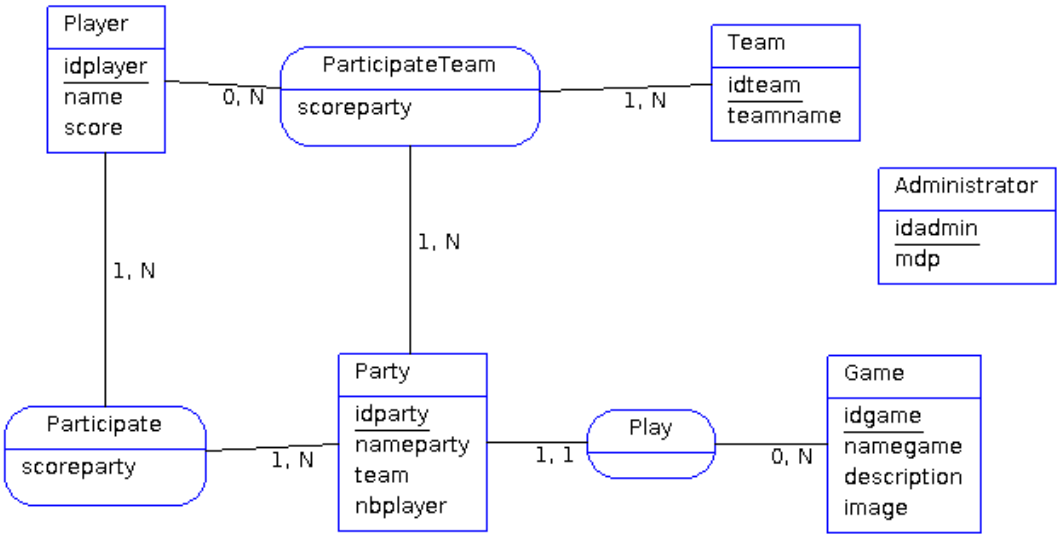
Les fonctionnalités présentées sont donc :

Non-administrateur :
- Création de tournois
- Ajout de joueurs
- Mise à jour des scores

Administrateur :
- Ajout de jeux
- Suppression de jeux

2 DESCRIPTION DES DONNEES

2.1 MODELE CONCEPTUEL DES DONNEES



2.2 MODELE LOGIQUE DES DONNEES

Administrator (idadmin, mdp)
Game (idgame, namegame, description, image)
Party (idparty, nameparty, team, nbplayer, #idgame)
Player (idplayer, name, score)
Team (idteam, teamname)
Participate (#idplayer, #idparty, scoreparty)
ParticipateTeam (#idteam, #idparty, #idplayer, scoreparty)

2.3 DESCRIPTIF

Administrator :

L'administrateur est décrit par son identifiant (*idadmin - integer*). Grâce à un mot de passe (*mdp - varchar*) il peut accéder à des options particulières. Ici il s'agit de supprimer ou d'ajouter un jeu à la base de données.

Game :

Il s'agit de jeux vidéo, répertoriés grâce à un identifiant (*idgame - integer*), ils possèdent un nom de jeu (*namegame - varchar*), une description, et une url d'image. Le nom de jeu est supposé unique, deux jeux n'existent pas sous le même nom.

Party :

Une party représente une partie de jeu. Elle possède donc en plus d'un identifiant (*idparty - integer*), la référence à une jeu (*idgame - integer*), un nom de partie (*nameparty - varchar*), un nombre de joueurs (*nbplayer - integer*) ainsi qu'un booléen *team* (par défaut faux) permettant de savoir si la partie se joue en équipes ou non.

Player :

Un joueur est identifié par *idplayer* (*integer*), il possède un nom unique (*name - varchar*) et un *score* initialisé à 0.

Participate / ParticipateTeam :

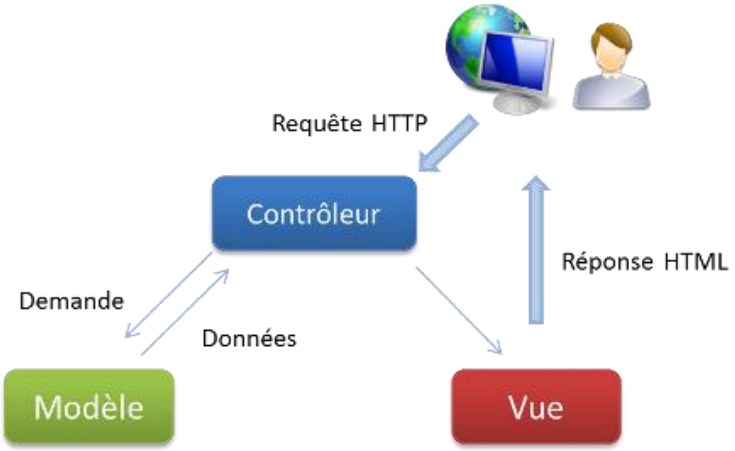
Cette table enregistre la participation d'un joueur à une partie, en prenant en compte s'il joue en équipe ou non. Elle prend donc en compte un *idplayer* et un *idparty* référencent la partie en cours, et le joueur en question, et un *scoreparty* permettant de connaitre et de mettre à jour le score du joueur sur cette partie.

2.4 LANGAGE UTILISE

La base de données mise en place est PostgreSQL. Il s'agit d'une solution open-source et prise en compte par l'hébergeur choisi. PostgreSQL est un système de gestion de base de données de modèle relationnel et objet. Il permet de mettre en place une base de données stable, robuste permettant de manipuler de gros volumes de données. Une autre option aurait pu être MySQL mais même s'il est facile d'utilisation et est adapté à des sites web de base, dans le vas d'une évolution de celui-ci, MySQL présente des limites notamment en termes de robustesse, d'utilisation simultanée et de migration des données.

3 DESCRIPTIF DU CODE

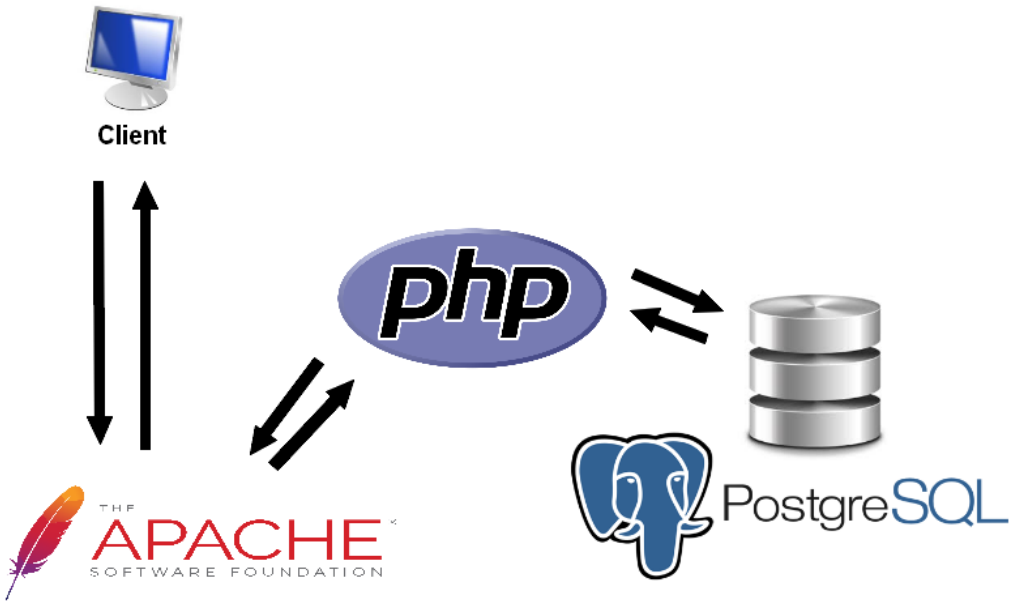
3.1 PATTERN MVC



L'architecture Modèle-Vue-Contrôleur permet de séparer ses données (*Modèle*), de l'affichage (*Vue*) et des actions (*Contrôleur)*, et ainsi les manipulations propres au client (*Vue*) et au serveur (*Modèle*). Cette séparation permet une conception claire. Sur le long terme, lors de la maintenance et de l'évolution du site, cela permet d'être plus efficace. En effet si une modification à lieu dans la base de donnée, il peut être suffisant de modifier le modèle concerné. Si le projet se fait à plusieurs, cela permet aussi de séparer les tâches.

Cependant, ce modèle d'architecture n'est adaptable qu'à des projets de petite envergure. Il est souvent nécessaire de créer plusieurs fichiers lorsque l'on veut ajouter une fonctionnalité et sur des projets simples cela peut être une perte de temps.

3.2 LANGAGES



Pour ce projet, j'ai décidé de choisir comme langage back-end. N'ayant jamais réalisé d'application web, le PHP m'a semblé être un langage facile à apprendre. Il permet d'avoir des pages dynamiques côté serveur même si elles apparaissent comme statiques coté client et gère les requêtes SQL. De plus PHP est stable.
En termes de maintenance, un bug de PHP n'est pas très bloquant, il va juste empêcher l'affichage de la page, alors qu'en java par exemple, un bug dans un traitement va juste bloquer ce traitement. Mais ce bug sera plus difficile à gérer en PHP qu'en Java, et on ne peut pas compiler pour appréhender des erreurs simples (oublis de parenthèses et de ;).

PHP, qui est un langage interprété par le client, peut aussi être complété par d'autres langages comme du JavaScript (par exemple pour rendre la page dynamique côté client).

3.3 BOOTSTRAP

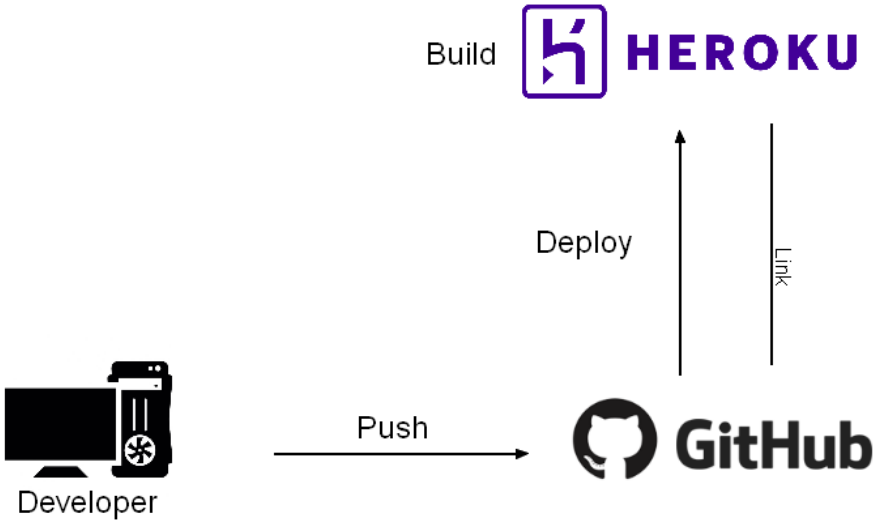
Au niveau du visuel, j'ai décidé d'utiliser bootstrap. Il s'agit d'une solution mise en place par des développeurs de Twitter répondant aux besoins lors du développement d'une application web. Bootstrap propose des codes HTML, des frameworks CSS prêts à l'emploi et des composants JavaScript utilisant la bibliothèque jQuery.

Bootstrap permet aussi de rendre son site sensible au changement de taille d'écrans (devices ?) sur lequel va être consulté le site.

4 DEPLOIEMENT

La méthode de déploiement a été choisie selon plusieurs critères, notamment les langages supportés et le prix de la plateforme. J'ai donc choisi Heroku.

Déployer mon application grâce à heroku est une solution gratuite. Il supporte les langages que j'ai pu utiliser. Par rapport à PGSQL, Heroku impose des contraintes sur le nombre de requêtes effectuées sur une même page, mais vu l'évolution de mon projet, cela ne pose pas de problèmes. Ne voulant pas prolonger la vie de cette application dans le temps pour l'instant, aucun nom de domaine n'a été acheté.
Il est possible d'ajouter des extensions à son application, là encore, à ce stade du projet je n'en ai pas besoin.
Le serveur d'Heroku va directement chercher les codes de l'application dans un dépôt GitHub. Lors de la mise à jour du dépôt GitHub, il est alors très facile de déployer son site, directement sur le site de Heroku ou en ligne de commandes. Les erreurs lors du déploiement sont alors données par Heroku.



5 POST MORTEM

5.1 DIFFICULTES RENCONTREES

Le lancement du projet a été difficile dans le sens où le sujet n'était pas imposé. Une fois le sujet vraiment délimité, là encore il a fallu faire le choix des fonctionnalités qui allaient être mises en application dans le temps impartis. Là encore ces ambitions ont été revues à la baisse.

En effet, au fur et à mesure du projet, j'ai pu me rendre compte que des soucis aussi bien techniques que le debug prenaient énormément de temps. Alors là encore, au milieu du projet j'ai dû me faire une raison et abandonner certaines fonctionnalités, ainsi que l'utilisation d'AngularJS.

Ne m'étant pas assez renseignée avant, j'ai aussi perdu beaucoup de temps à voir (ou revoir) les particularités des langages utilisés. Je n'ai donc pas pu mettre en place de routage, et ainsi je n'ai pas pu utiliser les verbes http adéquats pour la mise à jour et la suppression de données. Les fonctionnalités par équipes n'ont pas non plus abouti.

Concernant la base de données, je ne m'étais pas assez posé de questions et ait dû l'adapter au fur et à mesure du projet, ce qui est là encore une perte d'efficacité. Si je considérais continuer le projet, je devrais d'ailleurs adapter la base de données en conséquence.

5.2 CE QUE CE PROJET A PU M'APPRENDRE

Malgré tout ce qui a pu être dis plus haut, concernant la prise en compte du temps, j'ai pu me rendre compte que la mise en place d'un projet sur un temps si court est possible. Il m'a permis d'en apprendre plus sur les différents langages utilisés, ainsi que sur la programmation web et le protocole http.

Avec plus de préparation en amont, il est certain que d'autres fonctionnalités auraient pu être mises en place, et le projet aurait pu être plus propre. L'organisation est vraiment importante, de même que de savoir où est-ce qu'on veut aller. Juste préparer les différentes applications utilisées lors du projet m'aurait fait gagner une demi-journée.

6 CONCLUSION

L'application répond aux premières attentes fixées, mais pas encore à certaines règles. Un utilisateur peut donc créer un tournois en mode individuel, ajouter des joueurs et entrer au fur et à mesure de la partie des scores. Et l'administrateur peut ajouter et supprimer des jeux.

Vous pouvez trouver si joint, un plan chronologique d'une utilisation de l'application aussi bien du côté utilisateur qu'administrateur. Les codes relatifs à la base de données, ainsi qu'une arborescence des différents fichiers.

Sources :

- OpenClassroom
- W3school