Rapport personnel projet web mapping

Baptiste Rivière

27 février 2022

1 Objectif principal et résultat

Mon rôle au sein de ce travail d'équipe a été de mettre en place toute la partie serveur du jeu, comportant notamment l'arbitrage du jeu. C'est à dire traiter la demande des joueurs en ne leur renvoyant que les informations disponibles en mode brouillard de guerre, pour cela on réalise de nombreuses requête sur la table 'parties' d'une base de données.

Outre le résultat final sur la plateforme principale disponible sur https://chesswihou.herokuapp.com/, des tests unitaires sont réalisés sur une base de données à part, sans modification, sur https://chesswihou.herokuapp.com/partie/tests_unitaires/test_unit.html

Le code est disponible sur github à l'adresse https://github.com/baptisteRiviere/projet-web-mapping-echecs

2 Contribution personnelle

2.1 Résumé

En résumé, j'ai programmé les fichiers suivants entièrement :

- maj.json.php
- cr.json.php
- nouvelle_partie.json.php
- nul_abandon.json.php
- recharger.php
- test_unit.html
- test_unit.js

Et j'ai mis en place les tables parties et parties_test_unitaire de la base de données.

2.2 Explications générales

Ces fichiers gèrent donc tout le déroulement du jeu et l'arbitrage permettant de communiquer avec la base de données tout en envoyant les bonnes informations au bon joueur.

Tous les comportements concernant le déroulement du jeu ont été programmés et testés, notamment :

- les comportements de chaque pièce selon sa nature (R,D,T,F,P,C)
- les coups possibles et les vues
- le fait de prendre un pion
- le petit et le grand roque (les règles ont ici été simplifiées : la tour ou le roi peut avoir bougé auparavant et les cases intermédiaires peuvent être mises en échec)
- la prise en passant
- la promotion

La règle du pat n'a pas pu être développée (manque de temps), ainsi le jeu peut bloquer dans ce cas.

Toutes les explications concernant les entrées et les sorties des fichiers PHP sont dans le README du dossier partie, à lire de préférence sous format markdown. (disponible via l'URL https://chesswihou.herokuapp.com/partie/README)

3 Explication pratique du résultat

3.1 maj.json.php

Il s'agit du fichier central gérant l'arbitrage du jeu, le joueur envoie en entrée les divers identifiants (coté, partie, joueur) et la situation de son côté (trait, coup, tour). Selon le cas, on va renvoyer un message d'erreur ou renvoyer vers les fonctions appropriées à partir de la fonction repartition_cas. Il peut s'agir des fonctions suivantes :

— il_joue (le joueur adverse a joué, on renvoie les coups disponibles et les pièces vues)
— je_joue (le joueur joue le coup souhaité, on va mettre à jour le plateau et la BDD si le coup est bien valide)
— premier_coup (le joueur blanc veut débutter la partie, il doit recevoir les coups disponibles)

Les entrées et sorties détaillées sont dans le README dans le dossier partie. Un diagramme résumant les liens entre chaque fonction se trouve en annexe.

3.2 cr.json.php

renvoie le compte rendu d'un joueur, c'est à dire son historique complet, le tour et le trait de la partie. Ne prend en entrée que l'identifiant de la partie et le nom du joueur.

Il faut noter que la commande sql prend directement en compte l'auto incrémentation de l'identifiant de la partie.

3.3 nouvelle_partie.json.php

Ce fichier permet de créer ou de rejoindre une nouvelle partie :
— Si on envoie un nom de joueur tout seul, on créée nouvelle partie avec le joueur en côté 2
— Si on envoie nom de joueur et une partie on met le joueur dans la partie en côté 1

3.4 nul_abandon.json.php

Ce fichier permet de gérer les demandes/acceptations de nul et les abandons.

On lui envoie en entrée l'identifiant du joueur, celui de la partie et le côté. On envoie ensuite nul = 1 ou abandon = 1, dans le premier cas on souhaite faire une demande (nul = 0 dans la base de données) ou une acceptation (nul = 1 dans la base de données) de nul. Le deuxième cas implique l'abandon du joueur.

Un schéma expliquant le processus pour arriver à un nul est présent en annexe.

3.5 recharger.php

Ce fichier permet seulement de pouvoir recharger une partie dans la base de données afin d'accélérer et de simplifier les tests du jeu en mode développement.

3.6 test_unit.html et test_unit.js

Il s'agit des fichiers permettant de contrôler mon travail, la page html est disponible à l'url https://chesswihou.herokuapp.com/partie/tests_unitaires/test_unit.html. Elle permet, en ouvrant la console, de recevoir la réponse des fichiers en texte (au lieu de json) dans différents cas intéressants.

3.7 Base de données

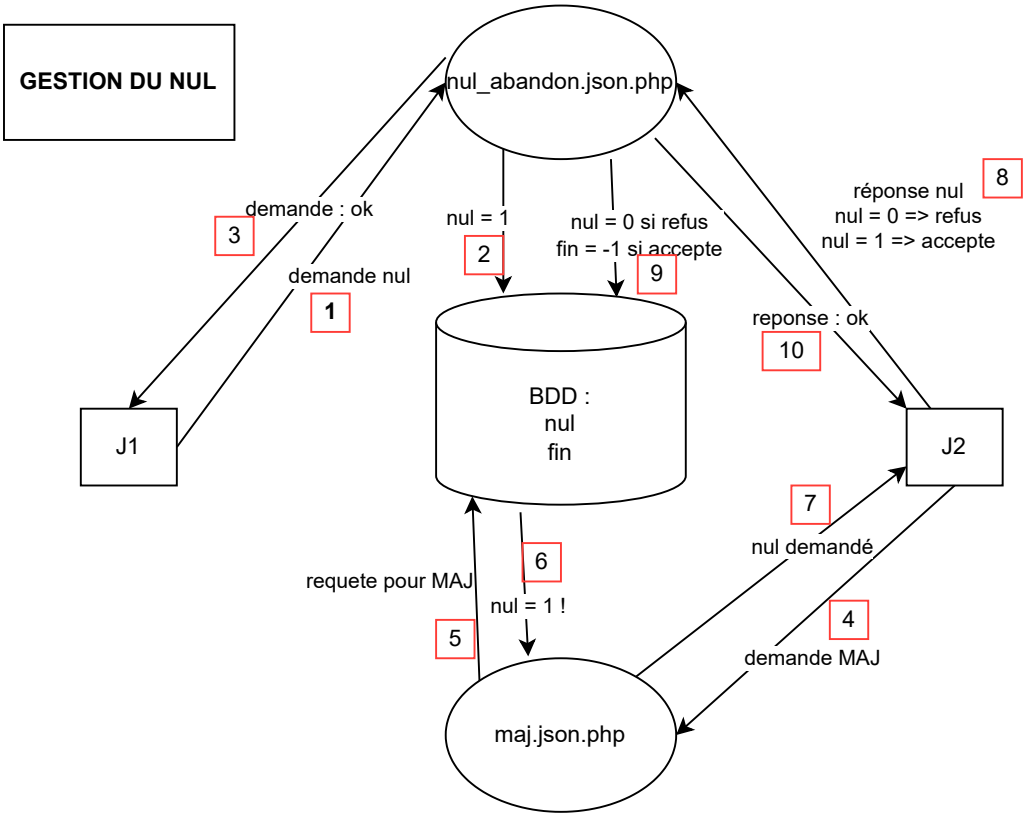
Les tables parties et parties_test_unitaire de la base de données permettent de sotcker les différentes variables relatives à une partie, elles sont construites de la même façon (parties_test_unitaire permettant de séparer les tests unitaires de la table réellement utilisée).

La structure est présentée ci-dessous :
— id (clé primaire, int(11)) : identifiant de la partie
— nom (varchar(15)) : nom de la partie, dans les faits utile que dans les tests unitaires
— tour (int(4)) : tour courant de la partie
— trait (int(1)) : 1 ou 2 selon le côté du joueur

— j1 (varchar (1)) : id du joueur 1 (nom)
— j2 (varchar (1)) : id du joueur 2 (nom)
— histo1 (mediumtext) : historique du joueur 1 sous format json
— histo2 (mediumtext) : historique du joueur 2 sous format json
— plateau (text) : plateau sous format json
— nul (int (1)) : 1 s'il y a une demande de nul en cours, 0 sinon
— fin (int(1)) : 0 si la partie n'est pas finie, -1 si on a un nul, côté gagnant dans les autres cas (1 ou 2)

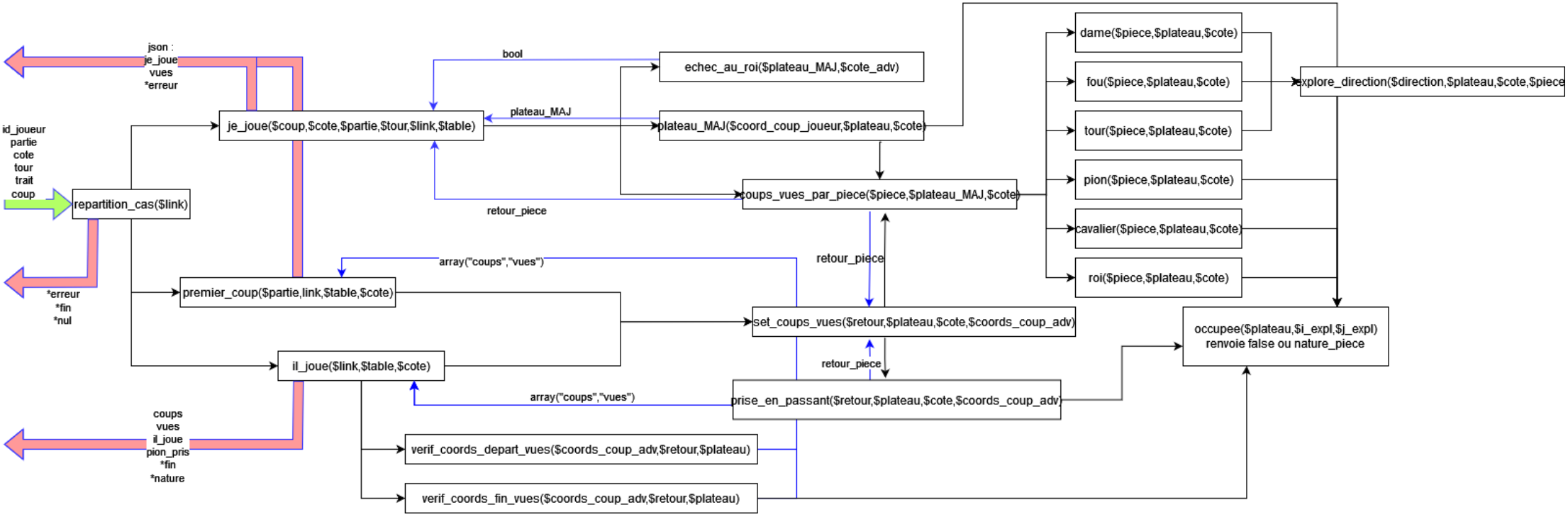
Date	détail
13/12	Mise en place du repository github Schématisation de l'architecture générale du code à partir d'un diagramme pseudo UML formation de Florian et Clément aux bases de github
14/12	mise en fonction du code cr.json.php préparation de la base du code maj.json.php
19/12	développement de la fonction MAJ dans maj.json.php, permettant de récupérer le plateau en json
21/12	Mise en place des comportements de base de tous les pions (une fonction par pion)
22/12	Programmation de verif_coords_depart_vues et verif_coords_fin_vues vérifiant si la position de départ ou de fin d'une pièce jouée par l'adversaire est visible par le joueur courant Mise en place du retour vers la BDD pour mettre à jour l'histogramme du joueur après une mise à jour par fonction set_coups_vues permettant de renvoyer les listes de coordonnées vues et les coups possibles
27/12	Récupération des coordonnées du coup joué lorsque le joueur joue : fonction je_joue Mise en place du contrôle de validité du coup dans la phase je_joue : le coup met-t-il le roi en échec ? Fonctions échec_au_roi, plateau_MAJ
29/12	Mise en place de la réponse je_joue avec l'ajout des cases vues Amélioration de la structure du code avec la fonction coups_vues_par_piece renvoyant les coups possibles par la pièce et les cases qu'elle peut voir
30/12	Mise en place de la fonction repartition_cas permettant de fluidifier la lecture du code, il s'agit de la porte d'entrée de l'algorithme complet début de débuggage du code
01/01	Suite débuggage code
Rentrée	Mise en pause du projet pour finir le PIR
11/01	Rassemblement pour fusionner tous nos codes Récupération de la partie de Clément et de l'interface graphique, ce qui est plus simple pour le débuggage et la finalisation du code (roque, prise en passant...)
23/01	début de programmation d'une fonction permettant de recharger la partie sans passer par la BDD
07/02	Fin de la programmation de recharger.php, permettant de recharger une partie pour faciliter le débuggage de maj.json.php Schématisation du comportement de maj.json.php, ce fichier comportant un nombre très important de lignes
15/02	mise en place d'un fichier html, d'un fichier js et d'une table dans la base de données spécialement conçus pour les tests unitaires Résolution d'un problème sur le comportement des pions
16/02	Mise en place de tests unitaires sur les coups reçus lorsqu'on reçoit la confirmation que l'adversaire a joué
18/02	Mise en place de tests unitaires impliquant la prise d'un pion
20/02	Ajout du roque, de la mise en échec ainsi que les test unitaires associés Mise en place de la gestion de demande et d'acceptation du nul
21/02	Mise en place de la promotion, roque et prise en passant dans la fonction plateau_MAJ Tests unitaires et README associé à maj.json.php
22/02	Mise en place du fichier nouvelle_partie.json.php permettant de créer de nouvelles parties dans la BDD : amélioration de la base de données avec auto_increment. fin des tests unitaires et du README
23/02	Débuggage du code qui comportait des faiblesses malgré les tests unitaires, Clément les mettant en lumière en développant le javascript gérant la partie

Table 1 – Calendrier des réalisation



1. le joueur 1 demande le nul en envoyant nul à nul_abandon.json.php
2. le fichier php va envoyer une requête à la BDD : la colonne nul de la BDD est mise à jour (nul = 1)
3. La mise à jour de la BDD a bien été réalisée
4. le joueur 2 fait une requête de mise à jour à maj.json.php
5. le fichier php fait une requête à la BDD
6. la BDD indique que la colonne nul est égale à 1
7. le php indique au joueur que le nul est demandé
8. le joueur 2 accepte ou refuse le nul en envoyant respectivement nul = 1 ou nul = 0 à nul_abandon.json.php
9. le fichier php mets à jour la BDD, s'il refuse la colonne nul = 0, s'il accepte on a une fin de partie : fin = -1
10. la réponse a bien été prise en compte dans la BDD

S'il a accepté le nul, le j2 a un affichage fin de partie, sinon il continue de demander une maj
A partir du moment où J1 a fait sa demande d'abandon il demande des maj, (nul = 1 lui donnera {ras : 1}), si le j2 refuse il recevra alors une réponse avec ses coups possibles (ou attente coup adv), si le j1 accepte il recevra une fin de partie


json :
je_joue
vues
*erreur
bool
echec_au_roi($plateau_MAJ,$cote_adv)
plateau_MAJ
je_joue($coup,$cote,$partie,$tour,$link,$table)
plateau_MAJ($coord_coup_joueur,$plateau,$cote)
dame($piece,$plateau,$cote)
fou($piece,$plateau,$cote)
tour($piece,$plateau,$cote)
pion($piece,$plateau,$cote)
cavalier($piece,$plateau,$cote)
roi($piece,$plateau,$cote)
explore_direction($direction,$plateau,$cote,$piece
id_joueur
partie
cote
tour
trait
coup
repartition_cas($link)
retour_piece
coups_vues_par_piece($piece,$plateau_MAJ,$cote)
retour_piece
array("coups","vues")
*erreur
*fin
*nul
premier_coup($partie,link,$table,$cote)
set_coups_vues($retour,$plateau,$cote,$coords_coup_adv)
occupee($plateau,$i_expl,$j_expl)
renvoie false ou nature_piece
retour_piece
il_joue($link,$table,$cote)
array("coups","vues")
prise_en_passant($retour,$plateau,$cote,$coords_coup_adv)
coups
vues
il_joue
pion_pris
*fin
*nature
verif_coords_depart_vues($coords_coup_adv,$retour,$plateau)
verif_coords_fin_vues($coords_coup_adv,$retour,$plateau)