Projet PHP – POO

Jeux d’échec en ligne

Html – CSS – Bootstrap – JS – PHP - MySQL

Connexion à la base de données

La base «onlinechess » contient 3 tables :

|  |  |
| --- | --- |
| Game | contient les parties en cours |
| Gamedata | contient l’historique de toutes les parties |
| Player | contient les login et mot de passe des joueurs inscrits |

L’historique des mouvements est stocké pour chaque mouvement sous la forme d’une chaine de 4 caractères correspondant aux coordonnées de la position de départ et d’arrivée.

Ex :

Une reine noire se déplace de la case D8 à F6, on stocke en base ‘4866’

Un pion blanc se déplace de la case D2 à D3, on stocke en base ‘7273’

Dans le fichier database.php, réaliser une classe Database qui hérite de la classe native PDO native de PHP. Les propriétés sont les paramètres de connexion PDO à la base. La classe ne possède qu’une seule méthode : son constructeur dans lequel la connexion est réalisée.

Les autres fichiers pour lesquels une connexion à la base est nécessaire incluent le fichier database.php

include("database.php");

La connexion à la base se fait en instanciant la classe Database

$dbh = new DataBase();

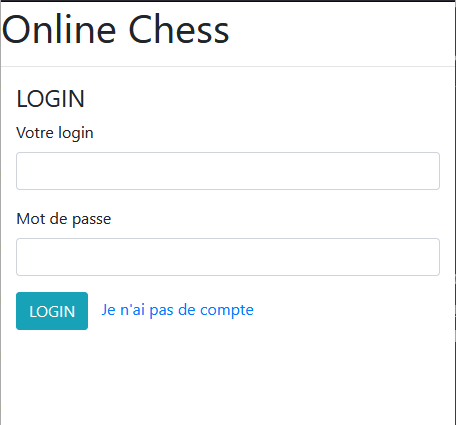
Page d’accueil

La page d’accueil (index.php) du site affiche un formulaire d’identification de l’utilisateur comprenant :

* Un champ pour login
* Un champ pour le mot de passe
* Un bouton de soumission
* Un lien pour créer un compte.

Utiliser bootstrap pour rendre la page adaptable aux différentes tailles d’écran.

Les paramètres de connexion existants sont (login/mdp) : player1/player1, player2/player2, player3/player3

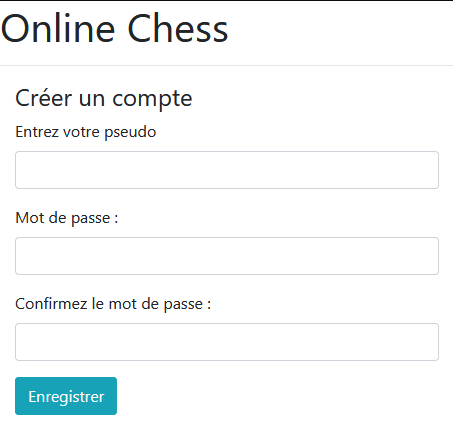


La page signup.php

La page de création de compte affiche un formulaire comprenant :

* Un champ de saisie du pseudo
* Un champ de saisie du mot de passe
* Un champ de confirmation du mot de passe
* Un bouton de soumission

Le mot de passe est encodé en base de données (md5). Réaliser un appel AJAX vers une fonction vérifiant l’unicité du login en base et un autre appel AJAX pour vérifier la concordance des deux mots de passe.



Une fois logué, le joueur est dirigé vers la page list.php

La page list.php

Indiquer sous le titre le login du joueur « Vous êtes connecté en tant que xxxxxx »

Cette page contient deux zones :

* Un formulaire permettant la création d’une partie comprenant :

Un champ liste déroulante (blanc/noir) permettant le choix de la couleur.

Un champ liste déroulante (logins inscrits) permettant le choix de l’adversaire.

Un bouton de soumission « Créer une partie »

Lorsque la partie est créée, le joueur est redirigé vers la page « game.php »

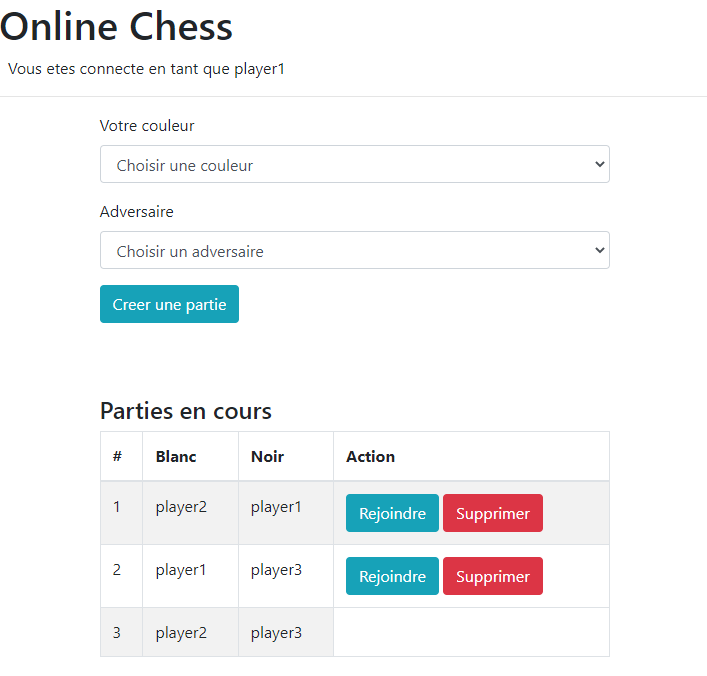
* Un tableau affichant les parties en cours

Ce tableau comprend :

le numéro d’ordre de la partie et pour chaque partie, le pseudo du joueur blanc, le pseudo du joueur noir et les actions possibles sur les parties, rejoindre ou supprimer. Ces boutons sont visibles uniquement pour les parties concernant le joueur.

Pour créer une nouvelle partie, il faut insérer dans la table « game » les identifiants du joueur blanc et noir.

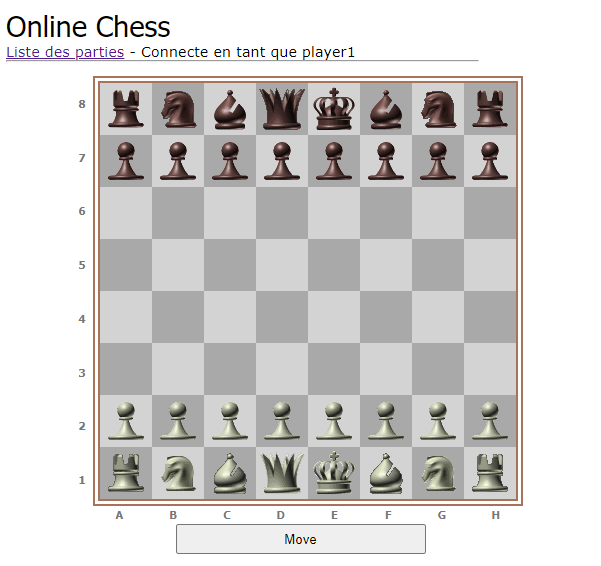
Le joueur rejoignant une partie est redirigé vers la page game.php



La page game.php

La page initie le tableau $\_SESSION dans le cas d’une nouvelle partie ou d’un changement de partie. Les index du tableau $\_SESSION sont :

* Login contient le login du joueur
* Id contient l’identifiant du joueur
* white contient le login qui joue les blancs
* black contient le login qui joue les noirs
* idgame contient l’identifiant de la partie



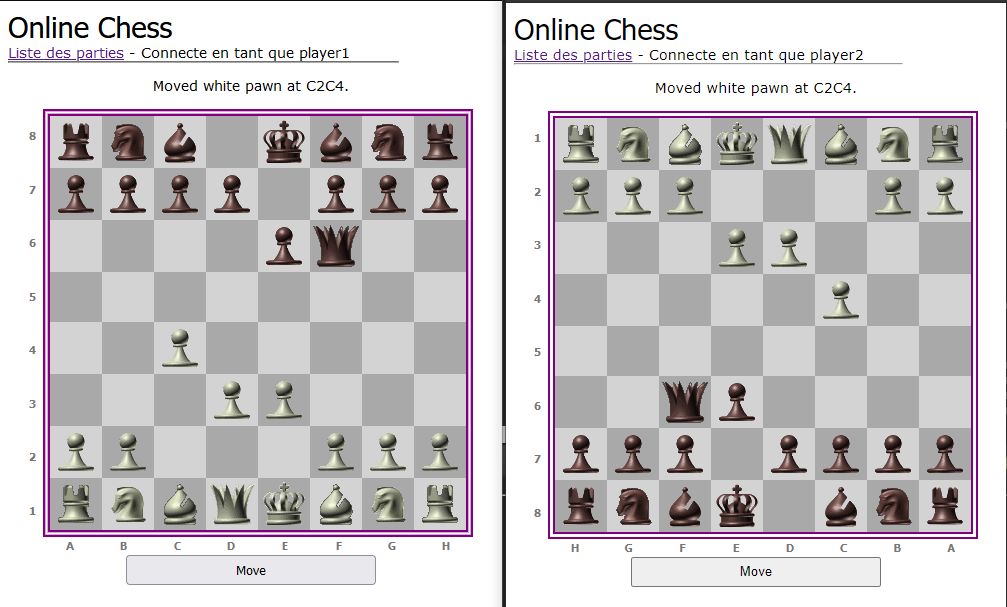
Lorsque c’est son tour, le joueur déplace une pièce de sa couleur en cliquant sur la pièce, puis en cliquant sur la case d’arrivée. Le joueur clique sur le bouton ‘déplacer’ pour valider son mouvement. Il attend son tour, la page est rafraichie toutes les 5 secondes au moyen de la balise meta :

<meta http-equiv="Refresh" content="5;url=game.php">

NB : la page du joueur dont c’est le tour ne doit pas être rafraichie.

Lorsqu’une page est rafraichie :

* On dessine le plateau (le joueur voit sa couleur au bas du plateau : le joueur noir voit le plateau inversé par rapport au joueur blanc.)
* On ajoute les pièces dans leur position de départ
* On interroge la table gamedata pour récupérer l’historique complet de la partie
* On rejoue la partie en déplaçant les pièces jusqu’au dernier mouvement enregistré
* On affiche dans la zone de texte le dernier mouvement réalisé (Par ex : ‘Le fou blanc s’est déplacé en F1B5’)
* Le joueur attend que l’adversaire ait joué pour reprendre la main



Pour afficher le plateau en html, on utilise une table. Pour les blancs, on commence par la cellule en haut à gauche, pour les noirs on commence par la cellule en bas à droite.

Déterminer $x\_start, $y\_start, $x\_increment et $y\_increment pour les deux cas.

Utiliser cette boucle pour parcourir la table :

for ($y = $y\_start; (($y-1 & 8) == 0); $y += $y\_increment) {

for ($x = $x\_start; (($x-1 & 8) == 0); $x += $x\_increment) {

Les classes du jeux

La classe OnlineChess\_Piece possède les propriétés suivantes :

|  |  |
| --- | --- |
| $color | (string) |
| $name | (string) |
| $x | (int) |
| $y | (int) |
| $num\_moves | (int) Nombre de mouvement depuis le début de la partie |
| $controlled\_squares | (array) cases que cette pièce contrôle |

Les pièces du jeu sont représentées par des classes spécialisées qui hérite de OnlineChess\_Piece :

class OnlineChess\_King extends OnlineChess\_Piece // Le Roi

class OnlineChess\_Queen extends OnlineChess\_Piece // La Reine

class OnlineChess\_Bishop extends OnlineChess\_Piece // le fou

class OnlineChess\_Knight extends OnlineChess\_Piece // le cavalier

class OnlineChess\_Rook extends OnlineChess\_Piece // la tour

class OnlineChess\_Pawn extends OnlineChess\_Piece // le pion

Ces classes contiennent 3 méthodes chacune :

Le constructeur qui fait un simple appel au constructeur parent

function \_\_construct($color, $x, $y) {

parent::\_\_construct($color, $x, $y);

}

public function validateMove(&$board, $x, $y, $piece\_turn = true) :bool

public function getControlledSquares($board) :array

La classe OnlineChess\_Square représente les cases du plateau. Elle possède les propriétés suivantes :

|  |  |
| --- | --- |
| $x | coordonnées x de la case |
| $y | coordonnées y de la case |
| $piece | vaut false si la cas e est vide (par défaut), sinon renvoie l’objet pièce instance de OnlineChess\_piece |
| $controlled\_by | tableau des pièces de l’adversaire qui contrôlent la case |

function \_\_construct($x, $y)

public function hasPiece() :bool

La classe PHPChess\_Board contient les méthodes nécessaires à la manipulation des pièces

|  |  |
| --- | --- |
| $square | (array) Tableau d'objets OnlineChess\_Square |
| $white | (object) Joueur blanc (type: OnlineChess\_Player) |
| $black | (object) Joueur noirr (type: OnlineChess\_Player) |
| $user | (object) Dépend du tour du joueur. Alias de white et black |
| $opponent | (object) Dépend du tour du joueur opposant. Alias de white et black |
| $history | (array) Tableau stocke tous les mouvements de la partie |
| $removed\_piece | (object) La dernière pièce retirée du plateau, ou FALSE si la case est vide |
| $has\_moved | (bool) "true" si le joueur a déplace une piece, "false" sinon. |
| $is\_check | (bool) "true" si le Roi du joueur est attaque |
| $is\_check\_mate | (bool) "true" si le joueur capture le Roi oppose |
| $is\_stalemate | (bool) "true" si le joueur ne peut plus faire de mouvement valide |
| $dbh | Connexion a la base de données |

public function \_\_construct()

public function createGame(string $white, string $black)

public function turn(bool $this\_user = true) :string

public function movePiece($x\_start, $y\_start, $x\_end, $y\_end, $really\_move = true) :bool

public function removePiece(int $x, int $y)

public function verifyCheckMate() : bool

private function addKing($color, $x, $y, $num\_moves = 0)

private function addQueen($color, $x, $y, $num\_moves = 0)

private function addBishop($color, $x, $y, $num\_moves = 0)

private function addKnight($color, $x, $y, $num\_moves = 0)

private function addRook($color, $x, $y, $num\_moves = 0)

private function addPawn($color, $x, $y, $num\_moves = 0)

public function updateData()

public function unpackData()

La classe OnlineChess\_Interface permet l’affichage du plateau

|  |  |
| --- | --- |
| $board | (object) Un alias à la propriété "OnlineChess->board" |
| $script\_file | (string) |
| $javascript\_loaded | (bool) |
| $message | (string) |
| $image\_dir | (string) |
| $square\_size | (int) |

function \_\_construct( OnlineChess\_Board $board)

public function board()

Cette méthode affiche le tableau suivant (le code html est une chaine de caractères contenue dans la variable $html)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Le plateau de 8 x 8 cases (variable $html\_board) | | | | | | | |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
|  | A | B | C | D | E | F | G | H |

La classe OnlineChess\_Player contient les propriétés suivantes :

|  |  |
| --- | --- |
| $color | (string) couleur du joueur |
| $pieces | (array) Tableau d’objets. Chaque élément du tableau est une pièce possédée par le joueur |
| $player | (string) le login |

public function \_\_construct($color, $player)

public function addPiece($piece)

La classe OnlineChess est la classe principale.

Elle possède les propriétés :

|  |  |
| --- | --- |
| $board | (objet OnlineChess\_Board) Contient toutes les informations des cases et des pièces |
| $interface | (objet OnlineChess\_Interface) Contient les methodes pour afficher le plateau |
| $dbh | (objet Database) Handler de connexion a la base de donnees |

Elle possède les méthodes :

public function \_\_construct()

public function setImageDir(string $dir) :bool

public function board() :string

public function form() :string

public function message(bool $bool\_return = false) :string

private function confirmMove($x\_start, $y\_start, $x\_end, $y\_end) :bool

public function getPlayer() :string

Méthodologie

Créer une base de données ‘onlinechess’. Importer la base de données (fichier onlinechess.sql)

Coder le script database.php

Coder la page index.php

Coder le script check\_availabity.php

Coder la page list.php

Coder la page game.php

Dans le fichier class.pieces.php : coder le constructeur et la méthode getName()

Dans le fichier class.square.php : coder le constructeur et la méthode hasPiece()

Dans le fichier class.player.php : coder le constructeur et la méthode addPiece()

Dans le fichier class.board.php : coder le constructeur et les méthodes createGame, turn, addKing, addQueen, addBishop, addKnight, addRook, addPawn, unpackData

Dans le fichier class.chess.php : coder le constructeur, les méthodes setImageDir, getPlayer