# Projet Web : Morpiien

## Groupe 12

Reda Aoutem, Lucas Baran, Clément Bisson, Etienne Gaucher

## Problématique

En ces temps de confinement, les élèves de l'ENSIIE sont à la recherche de sources de distraction. Nous avons donc pensé à créer un site où ils pourraient jouer au morpion, et ceci de manière illimitée.

## Projet initial

Initialement, notre idée était de créer un morpion en ligne, où les iiens pourraient jouer les uns contre les autres. En accord avec le sujet donné, l'utilisateur devrait d'abord s'identifier ou créer un profil, ce qui lui permettrait ensuite d'accéder au site. Une fois connecté, l'utilisateur voit la liste des autres utilisateurs connectés, puis a la possibilité de jouer une partie de morpion contre une de ces personnes. A chaque partie jouée, le joueur accumule des points en fonction de son score, et tente de rejoindre la tête du classement. Les utilisateurs pourraient également rejoindre une équipe et participer à un tournoi.

## Répartition des rôles

Avant de commencer l'implémentation du projet, nous avons défini des tâches personnelles à réaliser. Chaque personne a reçu plusieurs tâches, en tenant compte de ses préférences et compétences techniques. Voici une répartition globale du travail :

- **Lucas Baran** : comme il avait déjà une expérience importante dans le développement Web, et avait donc des connaissances, c'est lui qui a défini les bases (routeur, contrôleurs, architecture de fichiers, style général, etc) et implémenté les fonctions les plus techniques, par exemple le serveur interne de sockets.

- **Clément Bisson** : plutôt intéressé par la gestion des bases de données, il a construit la base de données, a défini les requêtes, certaines fonctions du morpion (pour déterminer le gagnant) et certaines pages HTML. Il a également écrit une partie du rapport.

- **Etienne Gaucher** : il s'est occupé de l'esthétisme du site (par exemple le style du plateau de jeu), du code CSS, de certains textes et de faire la connexion à la base de données. Il a rédigé une partie du rapport.

- **Reda Aoutem** : il a travaillé sur toute la partie utilisateur et profil (connexion, déconnexion, mot de passe, création de compte, etc) et la gestion des profils utilisateurs et administrateurs.

## Organisation des fichiers / Structure du site

- **Répertoire data** : structure de la base de données du site, c'est ici que sont définies les différentes tables contenant les informations nécessaires à la bonne tenue du site. On y définit également l'utilisateur de la base de données (groupe12).

- **Répertoire public** : répertoire contenant les pages qui seront visibles par les utilisateurs. Il est divisé en plusieurs sous-répertoires selon le langage/la technologie utilisée :

    - css : dossier contenant les différents styles qui seront appliqués aux pages
    - favicon.ico : miniature présente sur l'onglet du site
    - html : ensemble des fichiers php contrôlant le visuel des pages du site. Il contient notamment l'en-tête représentant le menu, le pied de page avec le copyright, les pages correspondant aux onglets, celles gérant la connexion et l'inscription des utilisateurs et la page permettant la modification d'un profil
    - img : l'image d'un cercle et d'une croix, nécessaire au graphisme du morpion
    - js : contient le code JavaScript du morpion
    - sound : effets sonores pour le morpion
    - on retrouve évidemment le fichier index.php, dans lequel on a un routeur simple qui se charge de faire les liens entre les pages en appelant les contrôleurs concernés par l'url demandée

- **Répertoire src** : la partie back-end du site avec les fichiers PHP (connexion à la base de données, contrôleur, serveur). Il est divisé en plusieurs sous-répertoires :

    - Controllers : dans ce répertoire, on retrouve les contrôleurs qui permettent de vérifier les paramètres des requêtes http envoyées par l'utilisateur et effectuer des actions en fonction. Ils permettent aussi de vérifier si c'est un utilisateur ou un administrateur qui est connecté avant de le laisser accéder à certaines pages

    - Factory : contient le script de connexion à la base de données

    - Repositories : on y trouve les repositories, qui se chargent de faire les requêtes sur la base de données, et les classes liées à ces derniers, qui servent à créer des objets représentatifs de table de la base

    - Server : contient le serveur qui gère les sockets de connexion (permettant notamment le listage des utilisateurs connectés) ainsi que des classes utilitaires pour ce serveur

    - config : la configuration pour se connecter à la base de données

- **Makefile** : différentes commandes pour le fonctionnement du site :

    - make start : permet de lancer le serveur sur localhost : 8080

    - make socket_server.start : lance le serveur qui gère les sockets

    - db.init : initialisation de la base de données à partir de data/init.sql

    - db.drop : destruction de la base de données

    - db.reset : réinitialiser la base de données

    - user.init : création de l'utilisateur gérant la base de données

    - user.drop : suppression de l'utilisateur qui gère la base de données

    - user.reset : réinitialiser l'utilisateur qui gère la base de données

## Choix remarquables

On va revenir dans cette partie sur des choix techniques notables que l'on a faits au cours du projet.

### Le routeur

On a opté pour un routeur plutôt simple. Il s'agit d'un switch PHP qui va regarder la seconde composante de l'url (après le premier slash '/') et déterminer quel contrôleur va se charger de vérifier les paramètres de la requête http.

### Les contrôleurs

Les contrôleurs sont centraux dans un site web. Nous avons créé une classe abstraite `Controller` dont tous les contrôleurs doivent hériter. Cette classe définit une méthode `handle` qui est appelée par défaut par le routeur. Cette fonction va appeler deux méthodes (abstraite dans la classe Controller pour que les filles soient obligées d'implémenter ces fonctions) `get` et `post` en fonction de la méthode HTTP qui a été reçue dans la requête HTTP de l'utilisateur. La méthode handle vérifie également au préalable si l'utilisateur est connecté et le redirige si ce n'est pas le cas.

Ensuite les contrôleurs vérifient les potentiels paramètres dans la requête HTTP et effectuent les actions nécessaires. Par exemple, le `LoginController` regarde si le nom d'utilisateur et le mot de passe envoyés dans une requête POST correspondent à ce qui est présent dans la base de données. Il y a un contrôleur par page.

### La connexion

Lorsqu'un utilisateur se connecte, plusieurs de ses coordonnées sont stockées dans les sessions (`$_SESSION`) comme le pseudo et son statut d'administrateur.

### Le serveur de sockets

Pour lister les utilisateurs connectés, nous avons utilisé des sockets PHP. Parallèlement au site, on lance un serveur qui gère des connexions de sockets. A chaque

fois que l'utilisateur charge une page, on va tenter de créer une connexion persistante vers ce serveur. La connexion persistante va permettre de conserver la socket à chaque rechargement de page.

Ensuite quand un utilisateur se connecte, on envoie l'information au serveur qui va garder dans une liste le pseudonyme de l'utilisateur. Après, quand un utilisateur va sur la page d'accueil, on va demander au serveur la liste des utilisateurs connectés. Cela nous permet alors de lister les utilisateurs connectés.

Le serveur gère également le changement de pseudonyme pour éviter des situations étranges et il gère surtout les parties de Morpion. Dans l'idéal, il aurait aussi permis de gérer des interactions immédiates entre les joueurs, par exemple dans les parties joueur contre joueur de Morpion.

### Les parties de Morpion

Comme dit précédemment, les parties de Morpion sont gérées par le serveur de sockets. Quand on arrive sur la page de jeu, le contrôleur va exiger au serveur de sockets de créer une nouvelle partie pour l'utilisateur. Ensuite, un script javascript va se charger de récupérer ce que l'utilisateur joue et, lorsqu'il récupère l'action de l'utilisateur, il envoie une requête HTTP à la page de jeu. La requête est récupérée par le `GameController`. Celui-ci envoie l'action au serveur de sockets qui va vérifier que le coup est bien valide, et le jouer si c'est le cas. Il vérifie ensuite si la partie est terminée. Si ce n'est pas le cas, il fait jouer l'IA, vérifie si l'IA a gagné et retourne le résultat.

### Un exemple de requête SQL

La page d'historique liste l'historique des parties. Pour cela, on va récupérer un objet appelé `mainRepository` qui va permettre de récupérer n'importe quel repository. Cet objet est initialisé lors de la création du contrôleur. Ensuite, le repository concerné va faire une requête à la base de données avec une méthode toute prête `fetchAllByUser`. Ainsi on évite de créer des requêtes n'importe où et n'importe comment.

Finalement, on va récupérer la liste renvoyée par la requête dans une variable. Elle va être utilisée dans la vue pour afficher une liste de chaque partie jouée par l'utilisateur depuis qu'il a été créé.

## Difficultés/solutions

- Le manque de temps a été un obstacle dans la mise en place du site. Par exemple, l'implémentation du serveur de sockets a nécessité de nombreuses heures. En effet, comme personne ne connaissait le sujet, il a fallu faire des recherches, et cela requiert des compétences approfondies.

- Nous avons connu quelques difficultés techniques, notamment lors de l'installation de PHP et surtout de pgsql. Cela a créé des incompatibilités entre nos ordinateurs, selon le système d'exploitation que nous employons (Dual Boot ou machine virtuelle).

- Pour que des utilisateurs puissent jouer ensemble, il a fallu mettre en place des sockets. Nous avons eu beaucoup de problèmes avec les interactions immédiates entre joueurs, nous avons donc choisi de faire des parties en joueur contre IA plutôt que joueur contre joueur, et laisser cette dernière fonctionnalité pour un terrain d'approfondissement et d'amélioration du projet.

## Amélioration

Ce que l'on pourrait faire si l'on avait plus de temps :

- héberger notre site sur un serveur afin qu'il ne soit plus seulement accessible en local
- implémenter les rubriques que l'on n'a pas eu le temps de faire
- ajouter le fait de pouvoir jouer contre un autre utilisateur
- ajouter des effets visuels sur le morpion
- ajouter une fonctionnalité de chat pour discuter avec les autres utilisateurs

## Mode d’emploi

A la première utilisation, afin de créer la base de données, il faut entrer les commandes suivantes :

```
make user.init
make db.init
```

Ensuite, pour lancer le site, il faut utiliser sur deux terminaux différents :

```
make socket_server.start
make start
```

Il faut ensuite entrer l’adresse suivante dans un navigateur web :

```
http://127.0.0.1:8080
```

## Conclusion

Même si nous avons dévié quelque peu de notre projet initial, nous avons su conserver le coeur de celui-ci, qui est le jeu de Morpion. Le site est fonctionnel, protégé par un système d’authentification et les différents menus sont présentés de façon claire. On peut alors aller s’affronter dans une partie de morpion.