# Gestion d'une collection de cartes

**Vous avez probablement tous entendu parler des jeux de cartes à collectionner tels que Magic, Pokemon, Yu-Gi oh ! et consorts. Dans cet exercice nous allons travailler sur une telle page qui sert à afficher une collection de cartes Magic avec le calcul d'un prix de chaque carte de la collection ainsi que le calcul du prix de la collection entière.**

Ce qu'on fait ici pour des cartes Magic pourrait bien sûr se décliner pour tout autre jeu de cartes, ou toute autre collection de manière générale. Ne vous gênez surtout pas pour détourner le code de la page afin de l'adapter à d'autres besoins ☺

# Objectifs pédagogiques

Grâce à ce projet vous aborderez les points suivants :

- La **programmation orientée objets** en PHP
- L'exploitation d'une **base de données**
- L'utilisation d'un **framework** PHP ultra léger dédié à l'accès à la base de données (**Medoo**)
- La navigation dans le **DOM** en PHP avec la récupération de données
- L'utilisation de **jQuery** et de librairies permettant de faire du **lazyLoading** et de **l'autocomplétion** lors d'une recherche

# Présentation du résultat

Vous allez travailler sur une unique page web (tout se passera en chargeant l'url index.php) qui affichera les cartes Magic de votre collection. Cette collection sera déjà fournie dans une base de données que vous devrez importer localement. Vous pourrez modifier cette base de données, mais vous ne serez pas obligés de le faire. La version fournie est suffisante.

Voici quelques screenshot montrant ce qu'on veut :

Magic: The Gathering | Valeur totale de la collection : 508.94€ | Trier par | MAJ toutes les cartes | Rechercher

### Mana Drain


#22 | Legends | Played

122€

Dernière mise à jour : 08/09/2020

Mettre à jour | Page détaillée

### Académie Tolariane


#18 | Epopée d'Urza | Excellent

70€

Dernière mise à jour : 08/09/2020

Mettre à jour | Page détaillée

### Lac de montagne bouillant


#12 | Zendikar | Near Mint

46€

Dernière mise à jour : 08/09/2020

Mettre à jour | Page détaillée

### Forêt pluviale embrumée


#11 | Zendikar | Excellent

45€

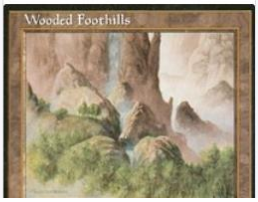
Dernière mise à jour : 08/09/2020

Mettre à jour | Page détaillée

### Maze of Ith


### Fondrière Sanguinolente


### Contreforts Boisés




### Faille cyclonique

Magic: The Gathering

Valeur totale de la collection : 508.94€

Trier par

MAJ toutes les cartes

Rechercher


#22 Legends Played

122€

Dernière mise à jour : 08/09/2020

Mettre à jour Page détaillée

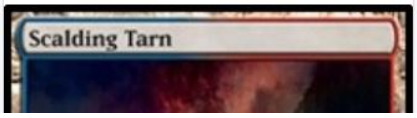

#18 Epopée d'Urza Excellent

70€

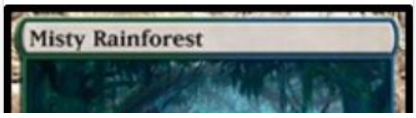
Dernière mise à jour : 08/09/2020

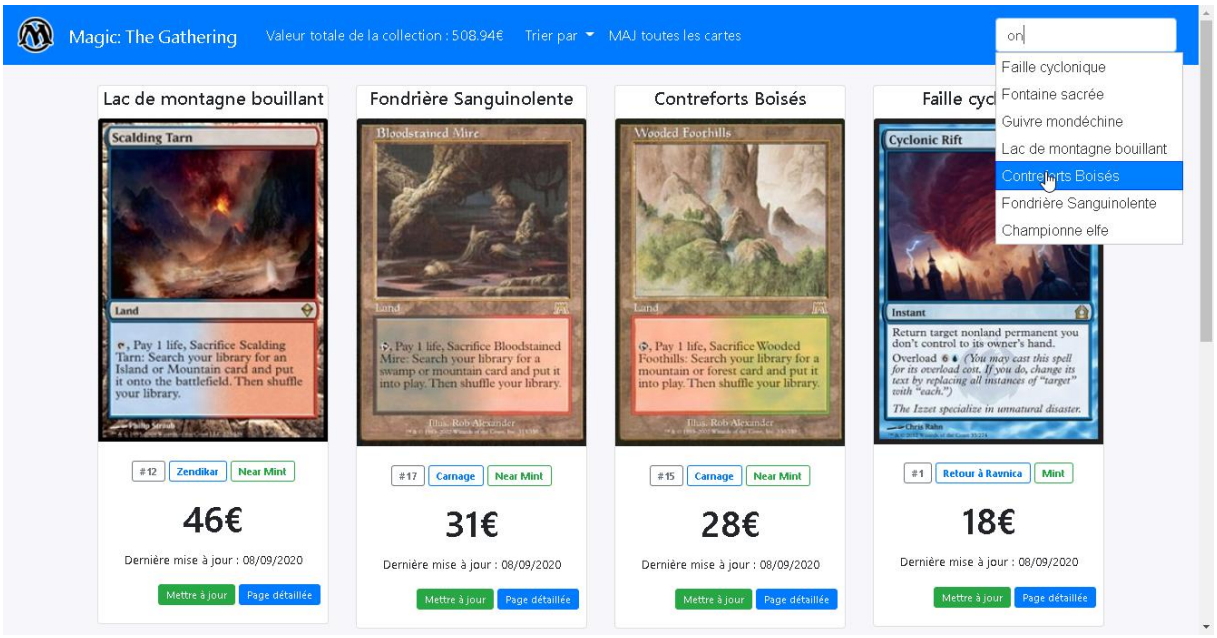
Mettre à jour Page détaillée

Lac de montagne bouillant




Forêt pluviale embrumée






Pour atteindre ce résultat là, vous utiliserez les fichiers déjà fournis que vous devrez compléter pour répondre à chacun des besoins énoncés ci-dessous :

1) En chargeant la page, vous afficherez sous la forme de *cards* de Bootstrap votre collection de cartes Magic. Cette collection provient de la base de données.
2) Chaque carte affiche son prix et ses informations (vous êtes libres du format) ainsi que deux boutons permettant d'aller vers la page détaillée de la carte (c'est l'url associée à la carte dans la base de données) et un lien de mise à jour de la carte (on reviendra dessus plus tard)
3) On veut pouvoir trier les cartes *a minima* selon les critères suivants : prix (croissant et décroissant), nom de la carte (croissant), id (croissant) et date de mise à jour (croissant). D'autres critères de tri peuvent être rajoutés si vous le souhaitez.
4) Un bouton permet aussi, d'un seul clic, de lancer une mise à jour globale de toutes les cartes de la collection
5) Enfin, on veut aussi pouvoir utiliser la recherche pour afficher les cartes contenant le texte saisi dans leur titre. Les cartes ne correspondant pas sont masquées, et quand le champ de rechercher redevient vide on réaffiche tout. De plus, pendant que l'utilisateur saisit du texte, une fenêtre d'autocomplétion apparaît proposant les choix possibles à l'utilisateur

# Architecture du programme

Outre la page **index.php**, qui est la seule page affichable du site, le programme est composé des fichiers et dossiers suivants :

- Dossier JS : contenant jquery-ui (composants supplémentaires pour jquery dont l'autocomplétion) et le fichier **script.js** destiné à recevoir votre code javascript
- Dossier img : contenant la favicon du site
- Dossier CSS : contenant la feuille de style de jquery-ui ainsi que votre feuille de style à vous **style.css**

- La fichier **Medoo.php** qui contient le framework Medoo, un framework ultra léger dédié aux accès à la base de données (apprenez à l'utiliser pour faire vos requêtes https://medoo.in/doc )
- Un fichier **Card.php** contenant la classe Card, permettant de créer des objets « Card », les cartes que nous allons manipuler ainsi que toutes les opérations appliquées sur ces cartes.
- Un fichier MKMParser qui va utiliser le fichier **simple_html_dom.php** (manipulation du DOM en PHP). Ce fichier MKMParser contient une classe qui va se charger d'analyser la page web d'une carte en question sur le site MKM dans le but d'y extraire les informations telles que image et prix, qu'on va ensuite stocker en base de données

# Fonctionnement de l'application

En lisant le contenu du fichier **index.php** vous retrouverez le fonctionnement décrit ci-dessous :

1. Lors du chargement de la page **index.php**, sans aucun paramètre, la page se met automatiquement en mode de tri « *prix décroissant* » par défaut.
2. On envoie une requête à la base de données qui permet alors de récupérer un tableau contenant toutes les cartes dans l'ordre de tri retenu (la requête se passe dans la classe Card)
3. On calcule aussi, via une requête dans la classe Stats, la valeur totale de la collection
4. Ensuite on itère sur toutes les cartes récupérées, et on génère une *card* bootstrap pour afficher chaque carte dans notre page, avec pour chaque carte, un lien vers le site de référence, et un lien permettant de lancer une mise à jour de la carte, en rappelant **index.php** mais avec le paramètre update ayant comme valeur l'id de la carte.
5. Si **index.php** est appelée avec le paramètre update, deux cas sont possible. Soit on lance une mise à jours de toutes les cartes (de manière séquentielle), soit on ne met à jour que la carte dont l'id est passée en paramètre.
6. La procédure de mise à jour d'une carte débute par le chargement de la page web des infos de la carte sur le site www.cardmarket.com. On construit le **DOM** de cette page (comme en JS, mais en PHP cette fois) et on navigue dans ce **DOM** pour récupérer des infos sur la carte telles que le prix (qu'on calcule) et l'url de l'image. Une fois les infos de la cartes mises à jour, on envoie ces informations en base de données et on raffiche toute la collection.
7. Des filtres de tri permette de modifier l'ordre d'affichage de la liste de cartes. A chaque changement d'ordre de tri, la page est rechargée avec un paramètre sort. C'est la valeur de ce paramètre qui définira le tri à effectuer. Le tri sera pris en charge par la requête envoyée à la base de données, c'est elle qui fournira la liste de cartes dans l'ordre souhaité.
8. Enfin, une barre de recherche permet de filtrer les cartes affichées. A chaque caractère tapé, la sélection s'affine et une boite d'autocomplétion s'affiche automatiquement proposant les choix valides parmi toutes les cartes disponibles.

# Votre travail

Votre travail consistera à compléter les méthodes dont le corps a été supprimé, dans les fichiers **Card.php**, **MKMParser.php** et **Stats.php**

Des commentaires devraient vous permettre de comprendre ce qu'on attend de vous à ces différents endroits. Par ordre chronologique voici ce qu'il est conseillé de faire :

1. Commencez par importer la base de données dans votre SGBD. Vous trouverez une table **cartes** qui contiendra la liste des cartes de votre collection ainsi qu'une table **conditions** détaillant les différents états possibles pour les cartes allant de *Mint* (neuf) jusqu'à *Poor* pour les cartes très abimées. Il existe donc une clé étrangère liant le champ « **condition** » de la table **cartes** avec la table **conditions**
2. Copiez le dossier « **mycollection** » dans le répertoire « **www** » de Laragon et ouvrez dans votre navigateur l'URL : http://localhost/mycollection/index.php
   La page affichera une erreur, c'est normal, il va falloir écrire le code manquant pour que tout fonctionne.
3. Configurez les constantes dans le fichier **Constants.php** pour que ça corresponde bien aux accès de votre base de données
4. Lisez le fichier **index.php** pour comprendre ce qu'on veut réaliser, vous n'avez normalement pas besoin de le modifier
5. Votre travail se situera dans les fichiers **Card.php**, **MKMParser.php** et **Stats.php**. A chaque fois, vous trouverez un commentaire **//TODO** qui vous explique ce qui est attendu. Complétez ces bouts de code et vous devriez obtenir le résultat présenté dans les screenshots ! Débrouillez-vous pour faire disparaître les erreurs d'exécutions que vous allez obtenir du fait que le code n'est plus syntaxiquement correct et saisissez ce qu'il faut pour que le code fonctionne.
6. Vous n'aurez pas à toucher aux autres fichiers, utilisez l'interface qui vous est fournie, **l'autocomplétion** et le **lazyloading** sont déjà mis en place. A vous de comprendre comment ça marche et d'améliorer tout ça si vous le souhaitez. Pour bien comprendre l'intérêt du lazyloading, lorsque votre page fonctionnera, passez en mode mobile et scrollez rapidement sur la page. Vous verrez les images se charger au fur et à mesure, et non pas toutes d'un coup au chargement de la page.

# Améliorations

Sous la forme de bonus vous pouvez apporter les améliorations suivantes :

- Le calcul de la moyenne se fait sur la base de toutes les valeurs relevées dans la page, avec potentiellement des valeurs extrêmes pouvant donner un résultat éloigné du cas médian. Une façon d'avoir un résultat encore plus pertinent pourrait être obtenu en retirant les extrema (les x plus hautes et plus basses valeurs) de chaque carte avant de calculer sa moyenne. On peut adapter la valeur de x en fonction du nombre de valeurs dont nous disposons. Par exemple si on a au moins 7 valeurs pour la moyenne, on retire la plus et la plus basse et on calcule la moyenne sur 5 valeurs. Au-dessus de 10 on retire les 2 plus hautes et 2 plus basses et au-dessus de 20, on retire les 3 plus hautes et 3 plus basses.
- Le fait d'utiliser une table pour les conditions permet de facilement faire des requêtes sur celles-ci. Vous pouvez dès lors rajouter un filtre de tri permettant de classer les cartes par ordre de conditions croissantes (de Poor à Mint) ou décroissantes (de Mint à Poor).
- Au contraire des conditions, les extensions des cartes sont écrites en « dur » dans la base de données, rendant plus difficiles (et moins pertinentes) les requêtes sur celles-ci. Restructurez la base de données en ajoutant une table « extensions » qui contiendra, comme la table condition l'ensemble des extensions de votre collection, puis faîtes en sorte de lier la table cartes à cette table extension.
- Enfin, vous pourrez ajouter une nouvelle fonctionnalité faisant qu'au clic sur le bouton/tag de l'extension de la carte (sous le visuel) seules les cartes de cette extension s'affichent – les

autres sont masquées. Dans la version de base vous pouvez cliquer sur le tag, mais rien ne se passe.