

PROJET DE FIN DE FORMATION

CONCEPTION/DEVELOPPEMENT D’UNE APPLICATION WEB

Matthieu Bourgoin

Développeur Web/Web Mobile, option Cybersécurité

SESSION 19/07/2021



**TABLE DES MATIERES**

**I/** **Introduction**

**II/ Cahier des Charges**

* II – 1 *Présentation du Projet*
* II – 2 *Analyse du Projet*
* II – 3 *Objectifs du Projet*
* II – 4 *Outils couverts par le Projet*

**III/ Spécialités Fonctionnelles**

* III – 1 *Outils couverts par le Projet*
* III – 2 *Arborescence*
* III – 3 *Maquettage*
* III – 4 *Analyse Fonctionnelle*
  + *III – 4.a Diagramme de cas d’utilisation*
  + *III – 4.b Diagrammes d’activité*
  + *III – 4.c* *Diagrammes de séquence*

**IV/ Conception**

* IV – 1 *Modèle Conceptuel de Données (MCD)*
* IV– 2 *Modèle Logique de Données (MLD)*

**V/ Développement du Projet**

* V – 1 *Outils couvert par le Projet*
* V – 2 *Les contraintes de développement*
* V – 3 *Présentation d’une API*
* V – 4 *Le Code*
  + *V – 4.a Récupération d’une API et insertion en BDD*
  + *V – 4.b Développement du « front-end »*
  + *V – 4.c Création de l’API du Projet*
  + *V – 4.d Fonctionnement de l’API*

**VII/ Conclusion**

* VI – 1 *Améliorations futures*
* VII – 2 *Bilan personnel*

**VIII/ Annexes**

**I. Remerciements**

**I. Abstract**

**I. Introduction**

J’ai à un moment ou à un autre depuis ma petite enfance possédée des cartes à collectionner, le plus souvent ceci dès le primaire, là où les cartes *Pokémon* faisaient rage dans la cour de récré, on découvrait un univers au travers de ces petites cartes aux visuels passionnants.

Très rapidement, deux écoles se forment au fil des années : les personnes qui ont abandonné leurs cartes au profit d’un autre divertissement et les autres, qui ont petit à petit découvert qu’il existait de véritables règles,

relayant ainsi les cartes en véritable [jeu de cartes](https://www.actugaming.net/tag/jeu-de-societe/) avec ses propres règles.

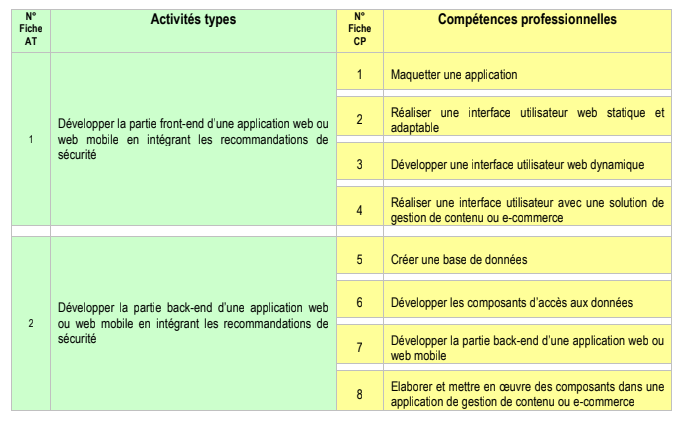
*Magic the Gathering* (ou *Magic L’Assemblée* dans la langue de Molière) fait bien entendu partie de tout ceci.

Pour présenter l’histoire de *Magic the Gathering,* le jeu est sorti en 1993, créé par Richard Garfield,à l’époque étudiant, aujourd’hui professeur de mathématiques et concepteur de divers autres jeux. *Magic the Gathering* reste le jeu de cartes à collectionner le plus joué au monde encore aujourd’hui. Avec 25 ans derrière lui, le jeu de Richard Garfield n’a jamais cessé d’évoluer au fil des années, comme par exemple la création du jeu vidéo *Magic Arena* qui propose des compétitions internationales d’eSport avec des joueurs professionnels.

\*(Ajouter parcours et projets futurs) \*

\*(Décrire le choix de passer par une API pour le projet) \*

1. **Compétences Professionnelles couvertes par le Projet**



**II. Cahier des Charges**

**II.1 Présentation du Projet**

Je vais donc vous présenter mon projet qui s’articulera autour de la création d’un paquet de cartes, communément appelé « deck » et plus précisément un système de filtre qui permettra de trier les cartes par couleurs, types, ou encore rareté par exemple. Il existe bien d’autres informations sur ces cartes, comme vous pourrez le voir ci-dessous.



**II.2 Analyse du Projet**

**Points Forts :**

* Site en Français/Anglais.
* Outil utile pour les joueurs de Magic du format Physique et Numérique.
* Actualités quotidiennes.

**Points Faibles :**

* Sites similaires déjà existants.
* Difficulté à se démarquer.
* Nouveau site web.

**Charte Graphique :**

* En rapport avec les couleurs de Magic (Rouge, Bleu, Noir, Vert, Blanc).
* Intégration d’illustrations et d’artworks.
* Site épuré, lisible et cohérent.

**Cibles :**

* Joueurs actifs de Magic du format Physique et Numérique.
* Nouveaux joueurs ayant un goût pour les jeux de stratégies.
* Joueurs voulant partager ses « decklists »

\*(Rédiger sous forme de phrases) \*

**II.3 Objectifs du projet**

\*(à voir selon l’intro) \*

**II.3 Analyse Marketing**

L’analyse marketing, ou étude de marché, est la première étape dans le développement marketing d’une société. Elle est cruciale dans la mise en place d’un produit ou d’un service pour mettre en accord notre offre avec la demande.

Pour cela, il faut se renseigner, observer, analyser et comprendre le futur environnement.

Pour se faire, une analyse SWOT a été effectuée :

|  |  |  |  |
| --- | --- | --- | --- |
| Strengths  (Forces) | Weaknesses (Faiblesses) | Opportunités (Opportunités) | Threats  (Menaces) |
| * travailler en local (producteurs) | * Pas encore ouvert | * Nouveau sur le marché * peu de concurrence spécifique | * les autres restaurants déjà bien intégrés dans le paysage |

**II.2 Référencement**

**I.4 Contraintes techniques**

**III. Spécifications fonctionnelles**

**III.1 Outils couverts par le projet**

* Analyse fonctionnelle / Conception *(cf. Chapitre III et IV)*

** **

**StarUML JMerise**

* Langages d’interfaces web statiques et dynamiques

** ** 

**HTML5 CSS3 JavaScript**

* Langages et logiciels d’interactions avec la Base de Données

**PHP5 MySQL Workbench XAMPP (phpMyAdmin)**

**III.1 Arborescence**

**III.2 Maquettage**

1. **Zoning**



**Le zoning est une représentation globale d’une page qui permet de schématiser “grossièrement” une page web grâce à l’utilisation de blocs.** L’objectif est de présenter l’emplacement des zones de contenu et des grandes fonctionnalités.

C’est une première étape déterminante pour définir l’organisation générale des pages d’un projet. Elle arrive juste après la création d’une arborescence.

**Le zoning permet de présenter une** **première approche d’une page web afin d’en valider les grands axes** (zones de contenu, fonctionnalités, hiérarchie, …).

La validation des zonings constitue une étape déterminante, préalable à la définition des Wireframes. Les zonings permettent par exemple d’identifier une surcharge d’information sur la page d’accueil d’un site web.

1. **Wireframe**



Les wireframes exploitent les zonings réalisés lors de la phase précédente pour préciser chaque bloc en introduisant les contenus présents. Un wireframe est équivalent au squelette ou à la structure simple d’une page web ou de l’écran d’une application mobile. Il est beaucoup plus détaillé que le zoning.

Le wireframe est la suite logique du zoning. Chaque bloc réalisé lors de l’étape précédente se voit doté d’image(s), de texte(s) ou de vidéo(s). Ce contenu peut être fictif (ex. Lorem Ipsum) car les informations finales ne sont pas toujours connues à ce stade du projet. De plus l’étape de “wireframing” se concentre sur l’ergonomie, et utiliser de vrais textes lors de cette étape détournerait l’attention de cet objectif principal. Le wireframe n’est pas une maquette !

Un wireframe, aussi appelé “maquette fil de fer” en français, est donc une représentation filaire d’un site internet en noir et blanc, et dépourvus de couleurs, de choix de polices, de logos qui décrit la taille, l’organisation et l’emplacement précis des éléments de la page, les fonctionnalités du site, moteur de recherche, les boutons d’action (CTA) et les éléments de navigation de votre site Web sans aucune notion graphique. L’aspect visuel ne sera traité que lors de la phase suivante de maquettage.

Cette étape est d’autant plus importante lors de la [définition d’une page produite sur un site e-commerce.](https://www.imagescreations.fr/comment-optimiser-les-pages-produits-de-votre-site-e-commerce/)

[Lors d’un atelier de conception web, le Wireframe place l’ergonomie au premier plan](https://www.imagescreations.fr/atelier-de-conception-web/) (simplicité de navigation, lisibilité, …). Les wireframes permettent aux différents membres du groupe projet de se projeter et de valider définitivement les fonctionnalités attendues.

Les Wireframes sont aussi parfois utilisés dans le cadre de la rédaction de spécifications fonctionnelles pour mieux présenter chaque fonctionnalité.

Créer des wireframes permet de rendre le processus de conception itératif. Il permet de gagner du temps lors de la phase de maquettage et de design graphique. Il permet ensuite d’en gagner également lors de la phase de réalisation car les équipes de développement comprendront plus facilement le résultat final attendu. Construiriez-vous une maison sans les plans en commençant par la décoration ? Probablement, non. Et bien c’est la même chose pour un site web. C’est assez logique et l’expérience montre que cela fonctionne !

**III.4 Analyse fonctionnelle**

**III.4.1 Cas d’utilisation (UseCase)**



En UML, un diagramme de cas d'utilisation permet de représenter les informations des acteurs d’un système, tel qu’une application logicielle, et leurs interactions avec ce dernier.

Un « acteur » peut donc être une personne, une organisation ou un autre système. Dans notre cas, les trois acteurs sont des personnes avec des droits différents selon leur statut.

Nous remarquons qu’une flèche d’association les relie : on parle de notion **d’héritage.**

En effet, dans notre cas « l’Administrateur » va hériter des droits du « Membre » qui va lui-même hériter des droits du « Visiteur».

Nous pouvons donc dire qu’un cas d’utilisation est un ensemble d’événements qui se produisent quand un acteur utilise une fonctionnalité pour achever un processus.

Nous pouvons noter qu’un diagramme de cas d’utilisation montre le comportement attendu du système mais qu’il n’indique pas l’ordre dans lequel les étapes sont effectuées. (*cf. Chapitre III.4.3*)

**III.4.2 Diagrammes d’activité**

1. **Diagramme d’activité de connexion**
2. **Diagramme d’activité de construction d’un deck**



Les diagrammes d'activités permettent de mettre l'accent sur les traitements. Ils sont donc particulièrement adaptés à la modélisation du cheminement de flots de contrôle et de flots de données. Ils permettent ainsi de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation.

Les diagrammes d'activités sont relativement proches des diagrammes d'états-transitions dans leur présentation, mais leur interprétation est sensiblement différente. Les diagrammes d'états-transitions sont orientés vers des systèmes réactifs, mais ils ne donnent pas une vision satisfaisante d'un traitement faisant intervenir plusieurs classeurs et doivent être complétés, par exemple, par des diagrammes de séquence. Au contraire, les diagrammes d'activités ne sont pas spécifiquement rattachés à un classeur particulier. On peut attacher un diagramme d'activités à n'importe quel élément de modélisation afin de visualiser, spécifier, construire ou documenter le comportement de cet élément.

La différence principale entre les diagrammes d'interaction et les diagrammes d'activités est que les premiers mettent l'accent sur le flot de contrôle d'un objet à l'autre, tandis que les seconds insistent sur le flot de contrôle d'une activité à l'autre.

**III.4.3 Diagrammes de séquence**

* 1. **Diagramme de séquence de connexion**



Un diagramme de séquence est un diagramme d’interaction dont le but est de décrire comment les objets collaborent au cours du temps et quelles responsabilités ils assument.

Il décrit un scénario d’un cas d’utilisation.

Un diagramme de séquence représente donc les interactions entre objets, en insistant sur la chronologie des envois de message. C’est un diagramme qui représente la structure dynamique d’un système car il utilise une représentation temporelle.

Les objets, intervenant dans l’interaction, sont matérialisés par une « ligne de vie », et les messages échangés au cours du temps sont mentionnés sous une forme textuelle.

* 1. **Diagramme de séquence de construction d’un deck**



Un diagramme de séquence est un diagramme d’interaction dont le but est de décrire comment les objets collaborent au cours du temps et quelles responsabilités ils assument.

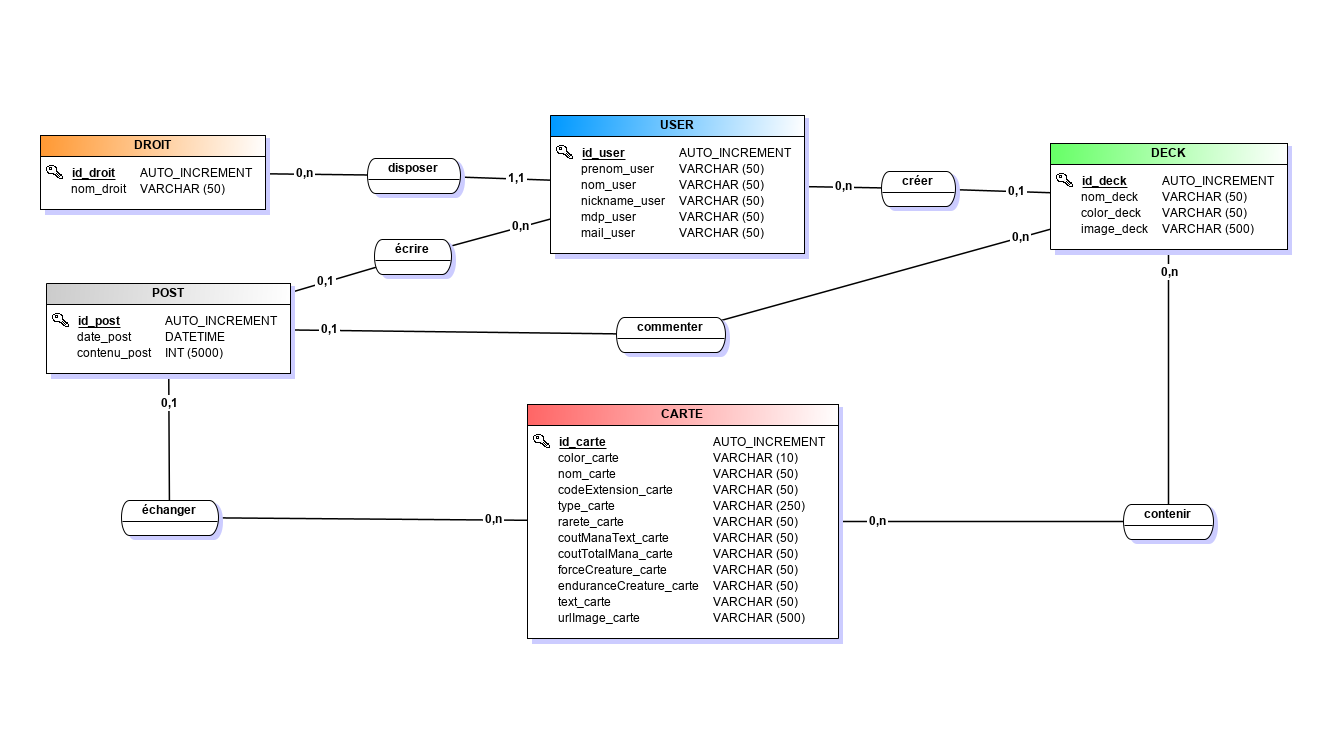
Il décrit un scénario d’un cas d’utilisation *(cf. Chapitre III.4.1*).

Un diagramme de séquence représente donc les interactions entre objets, en insistant sur la chronologie des envois de message. C’est un diagramme qui représente la structure dynamique d’un système car il utilise une représentation temporelle.

Les objets, intervenant dans l’interaction, sont matérialisés par une « ligne de vie », et les messages échangés au cours du temps sont mentionnés sous une forme textuelle.

**IV. Conception**

**IV.1 Modèle Conceptuel de Données (MCD)**

****

**MCD** est une représentation graphique de haut niveau qui permet facilement et simplement de comprendre comment les différents éléments sont liés entre eux.

Faisant partie de la boîte à outil Merise, le MCD décrit les données utilisées par le système d’information et leurs relations. Les informations sont représentées logiquement en utilisant un ensemble de règles et de diagrammes codifiés :

* Les [entités](https://www.base-de-donnees.com/entite/) (1 rectangle = 1 objet) ;
* Les [propriétés](https://www.base-de-donnees.com/propriete/) (la liste des données de l’entité) ;
* Les relations qui expliquent et précisent comment les entités sont reliées entre elles (les ovales avec leurs « pattes » qui se rattachent aux entités) ;
* Les [cardinalités](https://www.base-de-donnees.com/cardinalites/) (les petits chiffres au-dessus des « pattes »).

Utilisé assez tôt en conception de base de données, le **MCD** sert de base de travail et sera ensuite utilisé par les autres outils de Merise, à savoir le [MPD](https://www.base-de-donnees.com/mpd/) et le [MLD](https://www.base-de-donnees.com/mld/). Le MCD constitue une étape très importante de la modélisation. Si cette tâche est mal réalisée, des erreurs en cascade se produiront et rejailliront sur le MPD, le MLD et sur la base de données finale.

Cet outil **permet d’échanger entre informaticiens et non-informaticiens** sur l’outil à informatiser. On peut ainsi à partir d’un MCD **valider et préciser des règles** qui s’appliqueront à la future base de données.

Les données et les outils de traitement **font partie de la méthodologie Merise** de construction de bases de données. Un outil important concernant les données est le MCD (Modèle Conceptuel de Données).

Le MCD fournit **une représentation graphique interactive qui vous aide à comprendre facilement et intuitivement les interrelations des différents éléments** à l’aide de diagrammes codés. Les éléments du MDC sont notamment les suivants :

* Chaque rectangle représente un objet = Entités
* Les propriétés (la liste des données associées à l’entité)
* Les descriptions de la façon dont les entités sont reliées (les ovales et leurs jambes qui relient les entités)
* Ils (les chiffres au-dessus des « jambes ») sont appelés cardinalités
* Au début de la conception des bases de données, le MCD a ensuite conduit au MPD et au MLD, qui sont également des outils Merise.

**IV.2 Modèle Logique de Données (MLD)**

****

Le **MLD** ou **Modèle Logique des Données** est simplement la représentation textuelle du [MPD](https://www.base-de-donnees.com/mpd/). Il s’agit juste de la représentation en ligne du schéma représentant la structure de la base de données. Il n’y a pas de travail poussé à réaliser à cette étape, il s’agit juste d’appliquer quelques règles toutes simples. Cette étape est parfois omise.

On représente ainsi les données issues de la modélisation [Merise](https://www.base-de-donnees.com/merise/) sous la forme suivante :

* Chaque ligne représente une [table](https://www.base-de-donnees.com/table/) ;
* C’est toujours le nom de la table qui est écrit en premier ;
* Les [champs](https://www.base-de-donnees.com/champ/) sont listés entre parenthèses et séparés par des virgules ;
* Les [clés primaires](https://www.base-de-donnees.com/cle-primaire/) sont soulignées et placées au début de la liste des champs ;
* Les [clés étrangères](https://www.base-de-donnees.com/cle-etrangere/) sont préfixées par un dièse.

**V/ Développement du Projet**

**V – 1 Outils couverts par le Projet**

* Analyse fonctionnelle / Conception *(cf. Chapitre III et IV)*

** **

**StarUML JMerise**

* Langages d’interfaces web statiques et dynamiques

** ** 

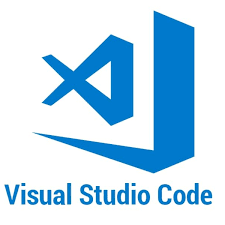
**HTML5 CSS3 JavaScript**

* Langages et logiciels d’interactions avec la Base de Données

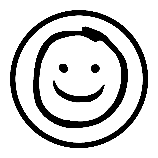
  

**PHP5 MySQL Workbench XAMPP (phpMyAdmin)**

* IDE (Integrated Development Evironment)



* Maquettage / Arborescence

**Balsamiq GlooMaps**

Star UML:

JMerise:

HTML5:

CSS3:

J’ai utilisé CSS3 pour mettre en forme et styliser les différents éléments contenus dans mes pages HTML. J’ai notamment utilisé Flexbox.

JavaScript:

PHP5:

MySQL Workbench:

XAMPP:

**Visual Studio Code:**

L’éditeur sur lequel j’ai développé mon projet. Il m’a permis grâce à plusieurs extensions d’avoir plus de facilités au niveau de l’indentation, la syntaxe ou encore un thème qui me convenait pour travailler efficacement.

**GlooMaps:**

J’ai utilisé Balsamiq pour réaliser rapidement et e fficacement les “wireframe” et “zoning” présentés dans ce mémoire.

GitHUB

**V – 2 Les contraintes de développement**

⇢ Limité dans le nombre de cartes en dû à un affichage "rapide" côté client malgré des fonctions asynchrones.

⇢

**V – 3 Présentation d’une API**

1. **Qu’est-ce qu’une API ?**

Une API (acronyme pour *Application Programming Interface*) est une interface de programmation.

Commençons par une analogie.

Nous allons prendre les anciens annuaires téléphoniques pour exemple.

⇢ Un **annuaire téléphonique** permet à un **humain**, de demander une information à un autre **humain**, par **téléphone**.

Sur le web, les API jouent le même rôle que les annuaires. Sauf que le téléphone est remplacé par internet et les humains par des ordinateurs.

⇢ Une **API**, permet à un **ordinateur** de demander une information à un autre **ordinateur**, par **internet**.

1. **A quoi sert une API ?**

Les API sont de plus en plus utilisées dans le milieu professionnel car elles répondent à plusieurs besoins. Elles permettent de **moderniser les structures des sites web et applications.**

Une API offre la possibilité aux entreprises d’ouvrir leurs données à leurs clients, fournisseurs et partenaires de manière sécurisée. Tout est interconnectable plus facilement, ce qui représente un **gain de temps et d’efficacité pour les équipes de développeurs.**

1. **Comment fonctionne une API ?**

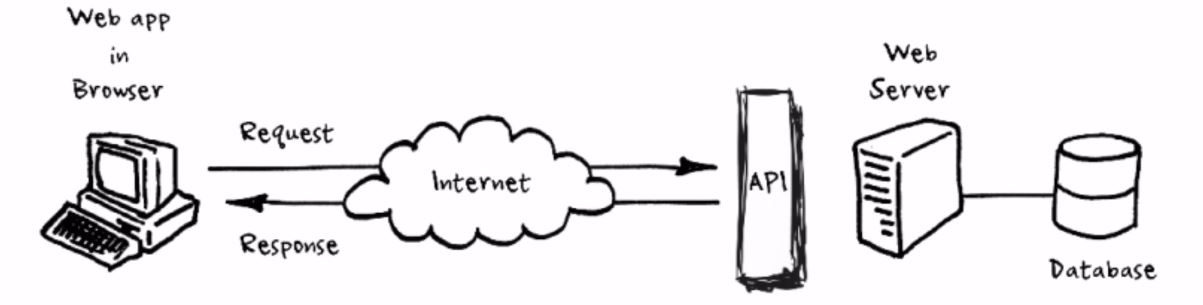
Pour développer une API, il faut un serveur et un client. **Le serveur fournit et exécute le programme de l’API**. Il attend qu’on lui envoi une requête pour lui demander des données.

**Le client est un programme distinct qui demande et reçoit les données** transmises par l’API. Ce client peut être un site web, une application, une machine, un logiciel métier, un smartphone, etc…

Il faut considérer l’API comme **une interface d’échange de données entre plusieurs systèmes**.

Le client envoie une requête de données au serveur. Le serveur va extraire et collecter des données qu’il va ensuite renvoyer au client.

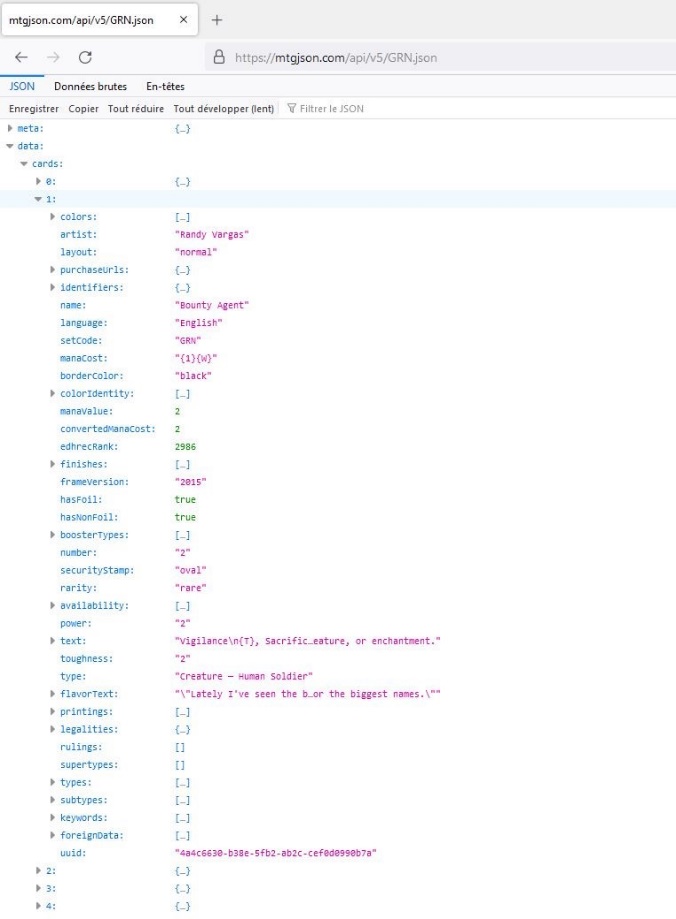
**Ce type de système permet aux différentes applications d’échanger entre elles.***(cf. schéma ci-dessous)*

****

**V – 4 Fonctionnement du code**

1. **Récupération des données via une API**

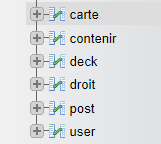
Tout d’abord, j’ai pu récupérer sur le site MTGJSON **(screenshot MTGJSON)** plusieurs données de cartes grâce à une API fournit en open source au format JSON. J’ai donc utilisé le navigateur Mozilla Firefox pour décoder le format JSON et m’afficher le résultat sous forme de tableau.



1. **Insertion des datas en BDD**

J’ai ensuite grâce à mon MCD, importer le SQL dans PHPMyAdmin afin de créer les tables dans la base de données. **(screenshot WorkBench).**

Nous pouvons donc voir que toutes les tables ont in



Comme nous pouvons le voir ci-dessous, j’ai également inséré 12 attributs, traduit sous forme de colonne dans ma table « carte » qui nous servira plus tard.



J’ai donc en premier lieu, créé un fichier PHP dans lequel la variable ***« $bdd »*** *(ligne 4)* permettra la connexion à ma base de données via l’objet ***PDO****.*



Par la suite j’ai récupéré l’url de l’API sous format JSON que j’ai inséré dans une variable *«****$pageContent****» (ligne 3)*, mise en forme grâce la fonction nativePHP *«****json\_decode****»* dans la variable « ***$jsonData****» (ligne 5*).

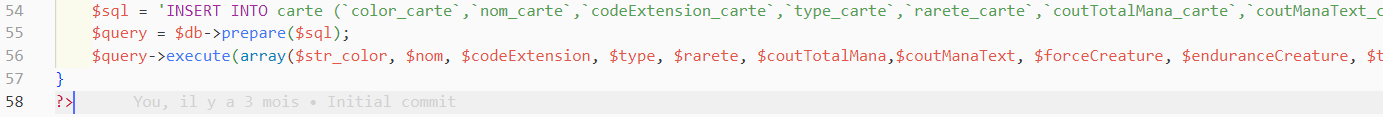
Pour éviter des erreurs d’insertion dans la table « carte », j’ai dû faire une boucle *«****foreach****» (ligne 11)* afin de pouvoir parcourir toutes les cartes*,* et les insérer.

Dans cette boucle, j’ai constitué plusieurs variables pour chaque colonne de ma table et dû mettre des variables à « ***NULL*** » *(ligne 16)* car comme évoqué en introduction, les cartes ont plusieurs spécificités et ne sont pas les mêmes.

Enfin, il y a plusieurs conditions «***if***» *(ligne 25)* avec la fonction native « ***isset*** » pour une nouvelles fois éviter des erreurs d’insertion.

**

Enfin, j’exécute une requête SQL pour effectuer l’insertion finale avec une requête préparée (*ligne 54 à 56*).



Nous pouvons voir maintenant que toutes les cartes ont été insérées dans la table « cartes ».



1. **Fonctionnement de l’API du Projet pour la fonctionnalité de « création de deck »**

Commençons par expliquer mon choix de créer une API pour mon projet.

La fonctionnalité que j’ai en premier lieu décidée d’effectuer pour mon projet, fût de pouvoir « filtrer » les cartes par attributs, ici *«****color****»* et « ***type*** » pour le moment.

J’ai voulu réaliser un **affichage dynamique** pour filtrer mes cartes. J’ai donc choisi de passer par une API.

Je suis donc passé par JavaScript pour jouer le rôle de l’API et qui fera le lien entre mon fichier HTML côté client et mon fichier PHP côté serveur.

**Etape 1 :**

On crée une vue HTML pour notre page de création de decklist.

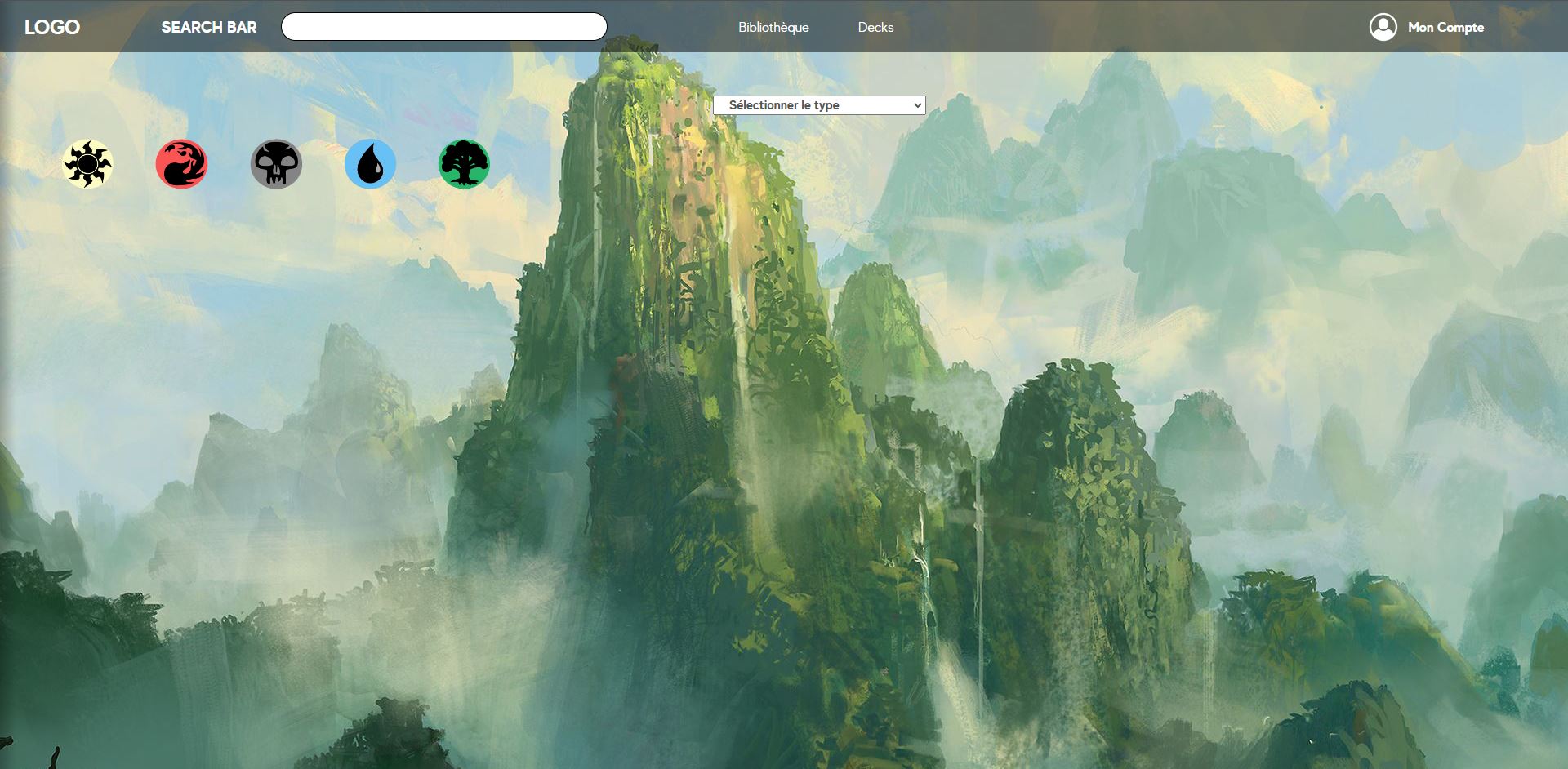


J’insère les icônes des couleurs du jeu pour m’en servir de filtre de *« couleur »*, et je crée également une liste déroulante pour filtrer les cartes par *« type »*.



*(image 1)*

Nous pouvons donc voir le résultat de la vue HTML, où les icônes de couleurs sont bien présentes ainsi que la liste déroulante.



**Etape 2 :**

J’ai créé un fichier Javascript qui va me permettre de déclencher les différents évènements d’affichage que je souhaite pour mon application, c’est-à-dire filtrer les cartes de manière dynamique sans rechargement de la page, d’où l’API et les fonctions asynchrones que nous verrons plus en détails plus tard.

En premier lieu, j’ai créé des constantes où je vais stocker des url qui vont récupérer toutes les cartes dans mon fichier backend que nous verrons plus tard.



*(image 2)*

Par la suite, j’ai utilisé les fonctions natives de Javascript pour créer les évènements. Ici ce sont seulement des clics.

*Event « Color »* *(image 3)* *Event « Type » (image 4)*

**Etape 3 :**

Comme nous pouvons le constater il y a plusieurs couleurs *(image 3)* et nous allons choisir la couleur *« blue »* comme exemple.

Nous créons donc une variable dans notre Javascript dans laquelle nous insérons la fonction ***« document.querySelector »*** qui va nous permettre de récupérer le contenu HTML ciblé, ici *« l’id #blue » (image 1, ligne 52).*

*Par la suite, la fonction* ***« add.EventListener »*** *(image3, ligne 154) va réagir au clic, et déclencher la fonction* ***« showBlueCards ».*** *(image 5)*



*(image 5)*

**Etape 4 :**

Nous remarquons que cette fonction est une fonction *« asynchrone ».* L’idée principale de l’asynchrone est que notre script Javascript puisse continuer à s’exécuter (ici nos cartes) pendant qu’une certaine opération plus longue ou demandant une réponse est en cours (la réponse de notre API). Elle permet donc un affichage plus rapide des pages et une meilleure expérience utilisateur.

Dans cette fonction, je créé une variable ***« $data »*** *(ligne 101, image 5)*dans laquelle je stock l’url de la constante ***« urlU »*** *(image 2, ligne 8) précédé de la fonction* ***« fetch »*** qui va attendre de récupérer les ressources au travers de notre API, dans notre backend, grâce à la fonction ***« await »*** (fonctionrelative à une fonction asynchrone)

**Etape 5 :**

Nous nous retrouvons donc dans notre fichier backend, qui va permettre de vérifier si les url transmises sont « présentes » grâce à la condition ***« isset »***, elle-même dans une condition ***« if ».*** *(image 6, ligne 2)*

******

*(image 6)*

**Etape 6 :**

Une fois, l’url vérifiée, la fonction ***« reqBlueCards »*** s’exécute.



*(image 7)*

Nous remarquons, que la fonction ***« include »*** *(ligne 158),* nous permet de nous mettre en relation avec notre base de données avant n’importe quelle requête SQL.

Dans la variable ***« $requête »,*** j’utilise la requête SQL ***« SELECT »*** *(ligne 160, image 7)*que l’on va stocker.

Je crée par la suite une variable ***« $output »*** qui va aller parcourir grâce à la fonction ***« fetchAll »*** toute ma table ***« carte »*** en base de données et me permettre de stocker toutes les cartes contenant la couleur bleue sous forme de tableau.

Enfin, une fois cette fonction exécutée et le tableau retourné, on va l’encoder en format de données JSON grâce à la fonction ***« json\_encode »*** *(image 6, ligne 26) et le renvoyer à l’API.*

**Etape 7 :**

Nous retournons dans le script Javascript, et plus précisément dans la fonction ***« showBlueCards ». (image 5)***

La réponse attendue par la fonction « fetch » étant revenu, je vais la stocker dans la constante ***« json » (ligne 103)*** dans laquelle je vais instancier un tableau du résultat de la constante ***« data ».***

Je crée une variable ***« url\_image »*** (ligne 106) qui va stocker tous les id des images présentes dans la base de données contenant la couleur bleue, récupérer grâce à la boucle ***« for »*** (ligne 109).

Enfin, j’affiche dans la div ***« zone »*** *(image1, ligne 57)* de manière dynamique, toutes les cartes contenant la couleur bleue auxquelles je donne une taille en hauteur et en largeur. *(image 5, ligne 248)*

Voici le résultat :

