# Compte rendu projet BDD

## Objectif

Ce projet a pour but de concevoir une interface web fonctionnelle permettant de gérer une base de données de Pokémon. Sur ce site, il sera possible de lister, rechercher un Pokémon à l’aide de son prénom ou de ses caractéristiques et d’ajouter un Pokémon. De plus, nous l’avons hébergé chez OSDPROTECT, pour pouvoir y accéder vous pouvez aller sur le site [pokedex.gcoquillet.fr](https://pokedex.gcoquillet.fr).

## Outils

Pour réaliser ce projet, nous avons utilisé le langage de balisage HTML/CSS pour structurer et styliser l'interface utilisateur, le langage de programmation PHP pour la gestion des interactions entre l'utilisateur et le serveur, le langage de requête structurée SQL pour manipuler la base de données, le logiciel UwAmp comme environnement de développement local (serveur Apache/MySQL) et enfin le site GitHub pour collaborer entre nous.

## Base de données

### Création de la base de données

La base de données que nous avons créée contient 8 tables : pokemon, famille, region, link\_region, link\_type, type, link\_faiblesse et categorie.

La table Pokémon contient toutes les informations liées au Pokémon qui sont unique.

La table famille permet de mettre en lien les id des différents Pokémon formant une famille.

La table region contient toutes les régions de l’univers Pokémon.

La table type contient tous les types de Pokémon existants.

La table categorie contient toutes les catégories de Pokémon existantes.

La table link\_region permet de lier l’id d’un Pokémon avec l’id d’une région lorsque le Pokémon possède une forme régionale.

Les tables link\_faiblesse et link\_type permettent de lier l’id d’un Pokémon aux id de ses types et de ses faiblesses.

### Schéma relationnel

Pour faire ce schéma relationnel, nous avons utilisé SQL Designer, ce qui nous a permis d’avoir un code XML que vous pouvez retrouver dans le fichier ‘bdd\_schematic.xml’.

A screenshot of a graph

Description automatically generated

### Peuplement de la base

Pour pouvoir peupler la base nous avons utilisé un fichier json contenant les caractéristiques de 1025 Pokémon, que nous avons extrait du site Tyradex via son api. Au début, nous avions testé avec quelques Pokémon pour vérifier que cela fonctionnait, puis nous avons utilisé l’API de Tyradex avec un script en python qui nous a permis d’ajouter tous les autres Pokémon dans la base de données de manière automatisée. Vous pouvez retrouver ce code dans le fichier *peuplement.py.*

Pour récupérer les informations des Pokémon via l’API de Tyradex, nous avons utilisé la fonction get() du module python requests afin d’effectuer la requête à l’api. Par la suite, nous devons définir le type de données reçues par la requête avec l'instruction data = data.json() : data peut maintenant être parcourue comme un tableau dynamique de dictionnaires python.

## Développement de l’interface

### Config.php

Ce fichier permet de se connecter à la base de données, à l’aide d’un identifiant et d’un mot de passe déjà défini dans des variables. Pour ce faire, nous utilisons le module PHP PDO. De plus, l’instruction error\_reporting nous permet d’afficher les erreurs PHP si jamais il y en a afin de pouvoir y remédier. Enfin, la fonction debug nous permet d’afficher le contenu d’une variable, d’un array ou de tout élément qui lui est donné en paramètre, permettant facilement de vérifier le bon fonctionnement de notre code lorsqu’on le souhaite.

### Index.php

Cette page est la page d’accueil de notre site, c’est la page principale. Elle a pour but d’afficher le menu et d’insérer à sa suite la bonne page en fonction du paramètre a qui est communiqué via la méthode GET de php. Par exemple : si aucune action n’est réalisée, la page insèrera *lister.php*, qui permet d’afficher toutes les occurrences de notre base de données. Le menu est dynamique : l’onglet correspondant à la page insérée est mis en couleur afin de permettre à l’utilisateur de savoir facilement et rapidement où il se situe sur le site.

### Lister.php

Lister.php permet d’afficher sous forme de tableau html les Pokémon contenus dans une instance de l’objet PDOStatement $réponse. Cette variable est définie dans les autres fichiers faisant appel à lister.php pour plus de modularité. Cela nous permet de ne pas avoir à recopier son code dans chaque page où on souhaite lister des Pokémon mais juste de l’appeler avec les données déjà prédéfinies.

Afin d’afficher toutes les occurrences contenues dans $réponse, nous avons créé une *while* qui, à chaque tour de boucles, définit l’array données contenant les informations du Pokémon à l’aide de la méthode fetch() de la classe PDOStatement

Par la suite, on affiche les différentes caractéristiques de chaque Pokémon à l’aide d’autres requêtes SQL nous permettant d’obtenir les types et faiblesses du Pokémon, mais aussi ses évolutions, sa variante régionale s’il en possède une et sa génération d’apparition.

### Rechercher.php

Cette page permet de rechercher un Pokémon dans la base de données. Pour cela de nombreux critères de recherche sont disponibles tels que la recherche par nom, par id, par type ou faiblesse, par génération ainsi que par catégorie.

Afin de pouvoir choisir le critère de recherche, nous avons utilisé un formulaire html avec pour méthode POST, une balise select et des balises options, nous permettant d’afficher un menu déroulant avec les différents critères de recherche ainsi qu’une balise input pour le contenu de la recherche.

Par la suite, on exécute la requête SQL pour retrouver le ou les Pokémon en fonction du critère et du contenu de la recherche, on l'exécute et l'attribue à $reponse puis on insère lister.php afin d’afficher le résultat de la recherche.

### Ajouter.php

Cette page permet à l’utilisateur d’ajouter un nouveau Pokémon via un formulaire html. Les balises input nous permettent de renseigner les informations du Pokémon mais également d’importer des images depuis l’ordinateur du client pour ce Pokémon.

Afin de renseigner les types et les faiblesses du Pokémon, nous avons dû créer un menu déroulant permettant de sélectionner plusieurs occurrences. Pour cela, nous avons créé une div contenant plusieurs balises input de type checkbox avec pour nom types[], nous permettant d'accéder aux différents types sélectionnés via un array php qui apparaît à l’aide d’un bouton html lorsque ce dernier est survolé.

Une fois les informations renseignées et le bouton ‘Ajouter’ cliqué, des requêtes SQL d’insertion sont effectuées afin d’ajouter le Pokémon dans la base de données et une fois ces requêtes terminées, le nouveau Pokémon est affiché en insérant lister.php .

### Rendu.php

Ce fichier permet d’afficher le compte rendu directement dans le site. Cela permettra à tous les utilisateurs de ce site de comprendre comment on a procédé à la création de ce site.

## Difficultés rencontrées

### Peuplement de la base de données

Pour peupler notre base de données, nous devions ajouter à la main les 1025 Pokémon existants ainsi que leurs images et leurs spécifications. Pour pallier cela, nous avons recherché des alternatives afin de faciliter la tâche. Nous avons donc trouvé l’API [Tyradex](http://tyradex.app) nous permettant en une requête http de récupérer l’entièreté des Pokémon et de leurs spécifications. Ainsi, nous avons utilisé cette api avec un script python afin de peupler notre base de données comme mentionné dans [Peuplement de la base](#_Peuplement_de_la).

### Caractères spéciaux

Lors du peuplement de la base, certaines erreurs sont apparues lors de l’insertion des informations de Pokémon sur la base de données. Ce problème était lié au nom de certains Pokémon qui contiennent des caractères non supportés par la base de données. Nous avons donc créé un fonction tri(nom:str)->str, permettant de pallier ce problème.

### Chargement des images

Lorsqu’on va sur le site, on s’est rendu compte que les images prenaient beaucoup de temps à apparaître. En effet, il y a près de 2030 images de Pokémon et ces dernières étaient stockées sur le github de l’API Tyradex. Par conséquent, le chargement des images était plutôt lent. Pour remédier à cela, nous avons téléchargé les images puis les avons importées dans le dossier images du site afin d’accélérer le temps de chargement de ces dernières.

Dorénavant, le site met environ 5 secondes à charger les 2000 images au lieu d’une quinzaine.

### Importation des images pour ajouter un Pokémon

Sur la version locale avec Uwamp, l’importation des images ne rencontrait aucun souci. Cependant, sur le site hébergé en ligne, aucune erreur n’était mentionnée mais l’image n’était en fait pas importée sur le site. Cela était en fait dû aux droits d'accès du dossier images sur le serveur web et à la taille maximale du fichier définie dans ‘ajouter.php’.