CSI – Architecture d'application – mini projet d'évaluation

L'objectif de cette évaluation est de créer une API REST pour une base de données.

Vous pouvez reprendre la base de données que vous avez créé pour l'évaluation de BDD Requêtes.

Vous pouvez réaliser ce mini-projet en binôme.

Critères:

- La base de données doit faire entre deux tables reliées par une liaison 1-n ou n-n. La base de données que vous avez réalisé pour l'évaluation de BDD Requêtes remplit normalement ces critères. Si celle-ci est plus complexe que la base demandée dans le présent devoir (ce qui est normalement le cas), vous ne réaliserez les fonctions d'API que pour deux des tables de la base – mais celles-ci doivent être reliées par une liaison 1-n ou n-n.
- Votre API doit implémenter l'ensemble des requêtes GET, GET (id), POST, PUT, DELETE pour chacune des deux tables.
- Votre API doit être auto-découvrable. La liaison 1-n doit être encodée sous forme d'une url (dans un sens) et d'une liste d'url dans l'autres sens (voir ci-dessous)
- Votre API doit intégrer le filtrage selon certains critères à définir.

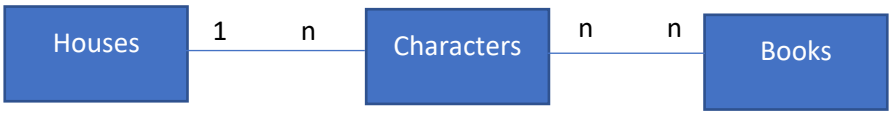
- Sur un deuxième projet Web, écrivez les vues correspondant au CRUD de l'une des tables.

- [BONUS] Votre API **peut** intégrer des éléments avancés: pagination, recherche.

Travail à réaliser

- Documenter: Définir les EndPoints de votre API, leurs adresses et leurs utilisations. Vous donnerez également les filtres implémentés. Une fois l'API créée, vous rajouterez également un exemple de requête et de réponse.
    - Exemple: https://anapioficeandfire.com/Documentation#characters
- Développer : développer l'API correspondant à vos EndPoints définis. Votre API doit être interrogeable avec Postman.
- Créer un site front end (découplé de votre API) qui permettra de faire un CRUD sur l'une de vos entités. Il n'est pas nécessaire ici de travailler sur l'aspect esthétique.

Réalisation de l'auto-découvrabilité




```
{
        "url": "https://anapioficeandfire.com/api/characters/583",
        "name": "Jon Snow",
        "gender": "Male",
        "culture": "Northmen",
        "born": "In 283 AC",
        "died": "",
        "titles": [
                "Lord Commander of the Night's Watch"
        ],
        "aliases": [
                "Lord Snow",
                "Ned Stark's Bastard",
                "The Snow of Winterfell",
                "The Crow-Come-Over",
                "The 998th Lord Commander of the Night's Watch",
                "The Bastard of Winterfell",
                "The Black Bastard of the Wall",
                "Lord Crow"
        ],
        "father": "",
        "mother": "",
        "spouse": "",
        "allegiances": [
                "https://anapioficeandfire.com/api/houses/362"
        ],
        "books": [
                "https://anapioficeandfire.com/api/books/5"
        ],
        "povBooks": [
                "https://anapioficeandfire.com/api/books/1",
                "https://anapioficeandfire.com/api/books/2",
                "https://anapioficeandfire.com/api/books/3",
                "https://anapioficeandfire.com/api/books/8"
        ],
        "tvSeries": [
                "Season 1",
                "Season 2",
                "Season 3",
                "Season 4",
                "Season 5",
                "Season 6"
        ],
        "playedBy": [
                "Kit Harington"
        ]
}
```


NOTA : pour les requêtes PORT et PUT (par exemple ici d'un `characters`), on considèrera que l'on connait les id des autres ressources liées (ici `houses` et `books`).

Langages

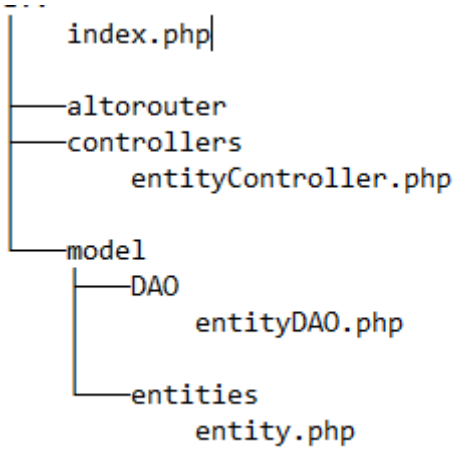
Vous pouvez réaliser votre API en PHP ou en javascript, selon votre préférence.

Pour PHP

Le TP que nous avons réalisé offre une bonne trame de base à la réalisation de votre API. Deux points sont cependant à approfondir :

- Proposer un algorithme de routage plus rigoureux que celui utilisé en TP. Pour cela, vous pouvez utiliser une classe toute faite, comme la classe altorouter (https://altorouter.com/, https://github.com/dannyvankooten/AltoRouter)
- L'inclusion des adresses pour l'auto-discovery, qui ne sont pas automatiquement renvoyée par la base de données. Pour cela, la bonne façon de coder implique de mettre en œuvre un pattern DAO, où les entités (l'équivalent des beans en java) implémentent l'interface JsonSerializable (exemple ici : https://www.codebyamir.com/blog/object-to-json-in-php).

Exemple d'architecture :



Pour javascript

Vous avez à disposition un projet d'API à l'adresse suivante :

https://github.com/Floriansp40/api-cocktails

Une démonstration sera nécessaire pour ceux souhaitant réaliser l'API en javascript