

# Mise en pratique API JPA / Mongo

Maintenant que vous savez ce qu'est une appli Spring Boot et interagir avec, votre objectif est de mettre en place une API de gestion de Pals dont les attributs sont :

```
id
key
name
wiki
types []
suitability []
  type
  level
drops []
skills []
  level
  name
  type
  cooldown
  power
  description
stats
  hp
  attack
    melee
    ranged
  defense
  speed
    ride
    run
    walk
  stamina
  support
  food
rarity
price
size
```

Si vous vous demandez ce qu'est un Pal : <https://store.steampowered.com/app/1623730/Palworld/>.

Votre première étape sera d'importer toutes les data contenues dans le pals.json dans votre base de données, de la technologie choisie, à chaque lancement de l'application. Le tout en utilisant Spring Boot et Spring JPA/Mongodb bien sûr.

Vous devrez respecter les principes REST et les 4 types d'opérations : Get, Post, Put, Delete.

Il faudra respecter la séparation des couches grâce à des packages bien nommés. Le pattern Controller / Service / Repository (ou DAO) sera attendu au minimum.

Une architecture en Clean Architecture avec les couches Application / Domain / Infrastructure serait un plus.

Votre code devra être testé unitairement (au moins 50% de couverture) et avec au moins un test d'intégration.

**@WebMvcTest** & **@MockMvc** (<https://spring.io/guides/gs/testing-web>) est un plus pour le test d'intégration, mais **@SpringBootTest** en appelant directement les méthodes des points d'entrées des contrôleurs est suffisant.

## L'API doit pouvoir gérer

GET par id  
GET par name  
GET par type

SAVE nouveau Pal  
GET skills d'un Pal  
ADD un skill d'un Pal  
MODIFY un skill d'un Pal

GET types d'un Pal  
ADD type d'un Pal  
REMOVE type d'un Pal

**BONUS** : avec des routes comme par exemple `/pal/sortedByRarity/{number of item per page}/{page numero}` donc avec utilisation de l'objet Pageable

GET x Pal SORTED BY rarity  
GET x Pal SORTED BY price

**BONUS 2** :  
Ajouter des test Gerkin & Cucumber.