

# FIABILITÉ DES DONNÉES

## 1. Tableau de validation

main.py

O2-fiabilite-donnees.md

docs > O2-fiabilite-donnees.md > # Fiabilité des Données > ## 1 Tableau de Validation

```
1 # Fiabilité des Données
2
3 Ce document explique comment nous vérifions et maintenons la qualité des données dans notre application maritime.
4
5
6 ## 1 Tableau de Validation
7
8 |# Modèle de Données – Backend FastAPI
9
10 Ce document décrit les entités principales utilisées dans le backend, leurs champs, contraintes et validations côté API (FastAPI + Pydantic).
11
12 | Entité | Champ | Type | Contraintes | Validation
13 |-----|-----|-----|-----|-----
14 | Capitaine | `id` | int | unique, non null | Clé primaire simulée, vérifiée à la
15 | création | | | |
16 | Capitaine | `nom` | string | non null, longueur minimale | Requis dans le modèle
17 | Capitaine | `experience` | int | >= 0, non null | Validé côté
18 | Capitaine | `specialite` | string | parmi `voilier`, `cargo`, `pêche`, `toutes` | À valider manuellement si
19 | besoin | | | |
20 | Bateau | `capitaine_id` | int/null | null ou doit exister dans la DB des capitaines | Validé manuellement dans `/
21 | affecter_capitaine` | | | |
22 | Trajet | `distance` | float | > 0, non null | Validé côté
23 API | | | |
24 --
```

Ce tableau sert de référence pour vérifier la conformité des données avant leur traitement, soulignant l'importance d'une validation systématique pour éviter les erreurs.

## 2. Vérification avec POSTMAN

Ces tests permettent de s'assurer que le système répond correctement aux données entrantes, en acceptant les données valides et en gérant les erreurs. Cela reflète une approche pratique pour confirmer la fiabilité des données dans un contexte réel, en simulant des interactions avec le système.

### a. Le cas où les données envoyés sont correctes

The screenshot displays the Postman interface for a POST request. The URL bar shows `http://127.0.0.1:8000/capitaines`. The request method is set to **POST**. The request body is a JSON object: 

```
{  "id": 1234,  "nom": "Testeur Marin",  "experience": 15,  "specialite": "pêche"}
```

. The response status is **200 OK** with a response time of 8 ms and a body size of 196 B. The response body is also a JSON object, identical to the request body.

HTTP `http://127.0.0.1:8000/capitaines` Save Share

**POST** `http://127.0.0.1:8000/capitaines` Send

Params Authorization Headers (8) **Body** Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** Beautify

```
1 {
2   "id": 1234,
3   "nom": "Testeur Marin",
4   "experience": 15,
5   "specialite": "pêche"
6 }
```

Body Cookies Headers (4) Test Results 200 OK • 8 ms • 196 B

{ } JSON Preview Visualize

```
1 {
2   "id": 1234,
3   "nom": "Testeur Marin",
4   "experience": 15,
5   "specialite": "pêche"
6 }
```

- b. Le cas où la spécialité fait pas partie du tableau de spécialités [ "voilier", "cargo", "pêche" ]

The screenshot shows a REST client interface. The top bar indicates the current request is a POST to `http://127.0.0.1:8000/capitaines`. The body tab is selected, showing a JSON payload:

```
1 {
2   "id": 1234,
3   "nom": "Testeur Marin",
4   "experience": 15,
5   "specialite": "professeur"
6 }
```

The response tab shows a `400 Bad Request` status with a response time of 4 ms and a body size of 214 B. The response body is:

```
1 {
2   "detail": "Cette spécialité ne fait pas partie de la liste des spécialités"
3 }
```

### 3. Présentation du code source

#### a. La class CAPITAINE

```
0
7  # ----- Modèle Capitaine -----
8  class Capitaine(BaseModel):
9      id: int
10     nom: str
11     experience: int
12     specialite: str
13
```

#### b. La condition permettant de vérifier si la bonne spécialité a été choisi dans le tableau

```
@app.post("/capitaines", response_model=Capitaine)
def ajouter_capitaine(capitaine: Capitaine):
    specialites_valides = ["voilier", "cargo", "pêche"]

    if capitaine.specialite not in specialites_valides:
        raise HTTPException(
            status_code=400,
            detail="Cette spécialité ne fait pas partie de la liste des spécialités"
        )

    print(f"Le capitaine {capitaine.nom} est spécialisé en {capitaine.specialite}.")
    capitaines_db.append(capitaine)
    return capitaine
```