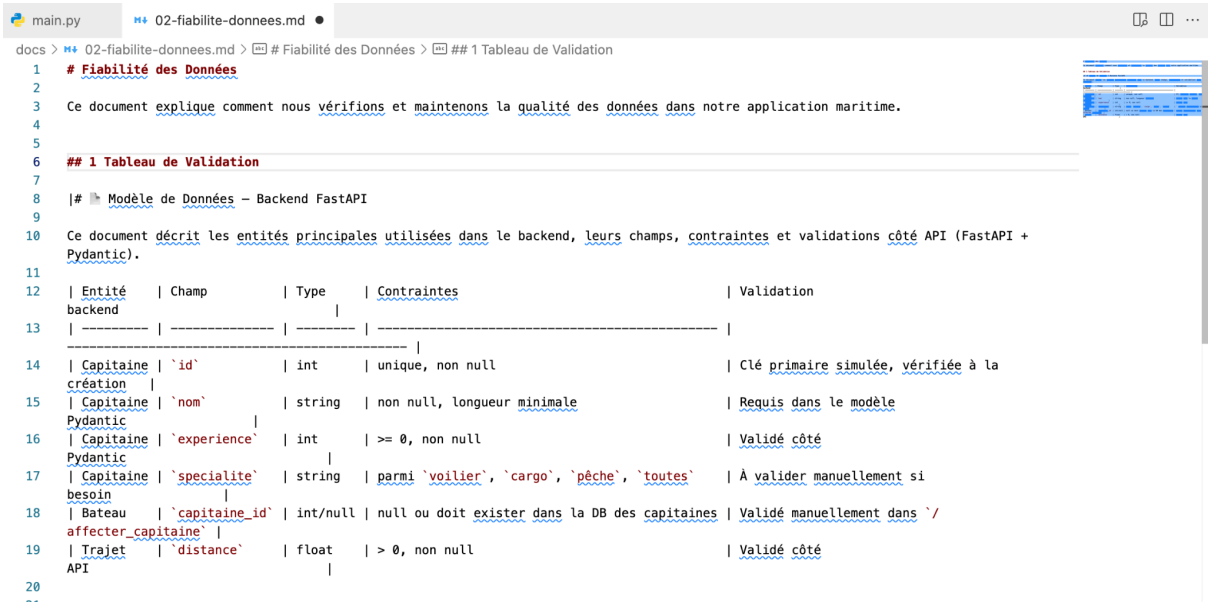
FIABILITÉ DES DONNÉES

1. Tableau de validation



main.py 02-fiabilite-donnees.md ●

docs > 02-fiabilite-donnees.md > # Fiabilité des Données > ## 1 Tableau de Validation

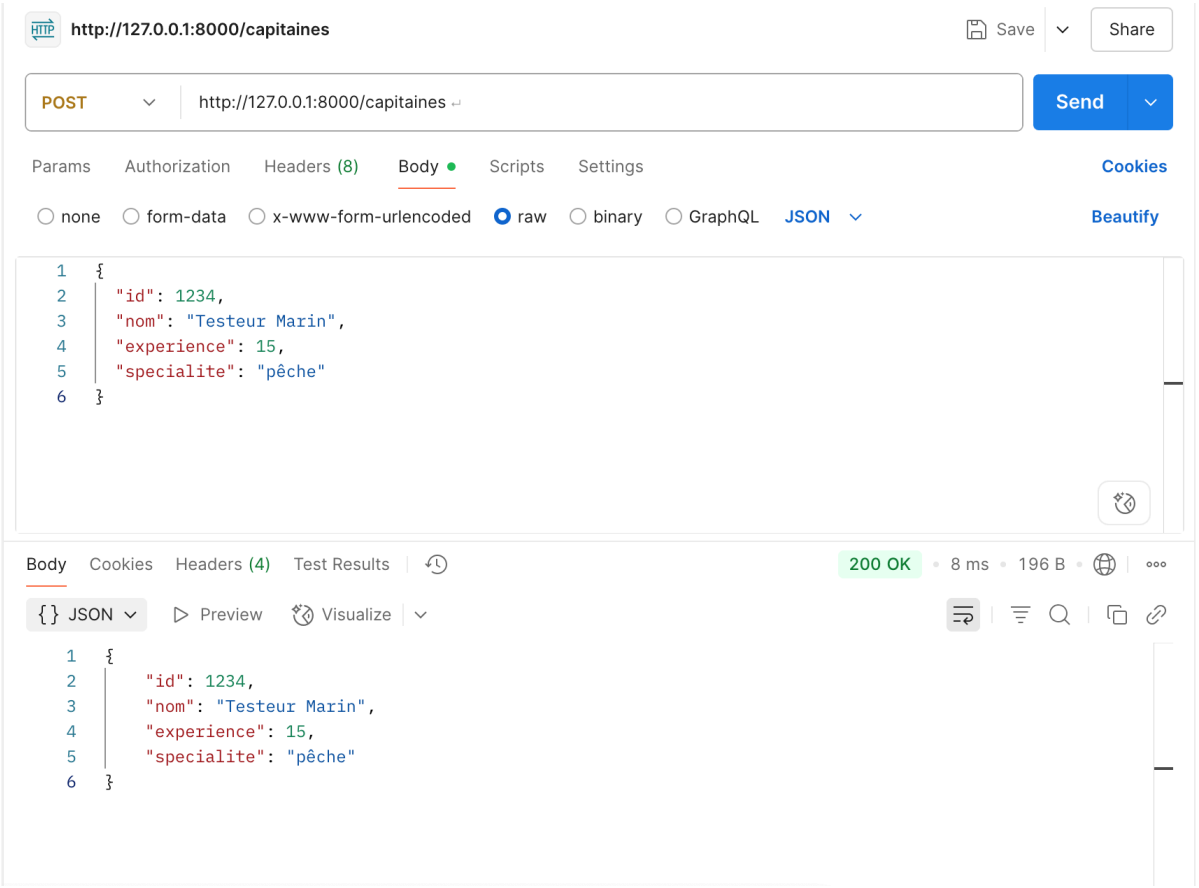
```
# Fiabilité des Données

Ce document explique comment nous vérifions et maintenons la qualité des données dans notre application maritime.


## 1 Tableau de Validation

|# Modèle de Données — Backend FastAPI

Ce document décrit les entités principales utilisées dans le backend, leurs champs, contraintes et validations côté API (FastAPI + Pydantic).

| Entité    | Champ          | Type     | Contraintes                                  | Validation
backend                          |
| --------- | -------------- | -------- | -------------------------------------------- |
---------------------------------------------- |
| Capitaine | `id`           | int      | unique, non null                             | Clé primaire simulée, vérifiée à la
création   |
| Capitaine | `nom`          | string   | non null, longueur minimale                  | Requis dans le modèle
Pydantic                  |
| Capitaine | `experience`   | int      | >= 0, non null                               | Validé côté
Pydantic                       |
| Capitaine | `specialite`   | string   | parmi `voilier`, `cargo`, `pêche`, `toutes`  | À valider manuellement si
besoin              |
| Bateau    | `capitaine_id` | int/null | null ou doit exister dans la DB des capitaines | Validé manuellement dans `/
affecter_capitaine` |
| Trajet    | `distance`     | float    | > 0, non null                                | Validé côté
API                            |

```

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.

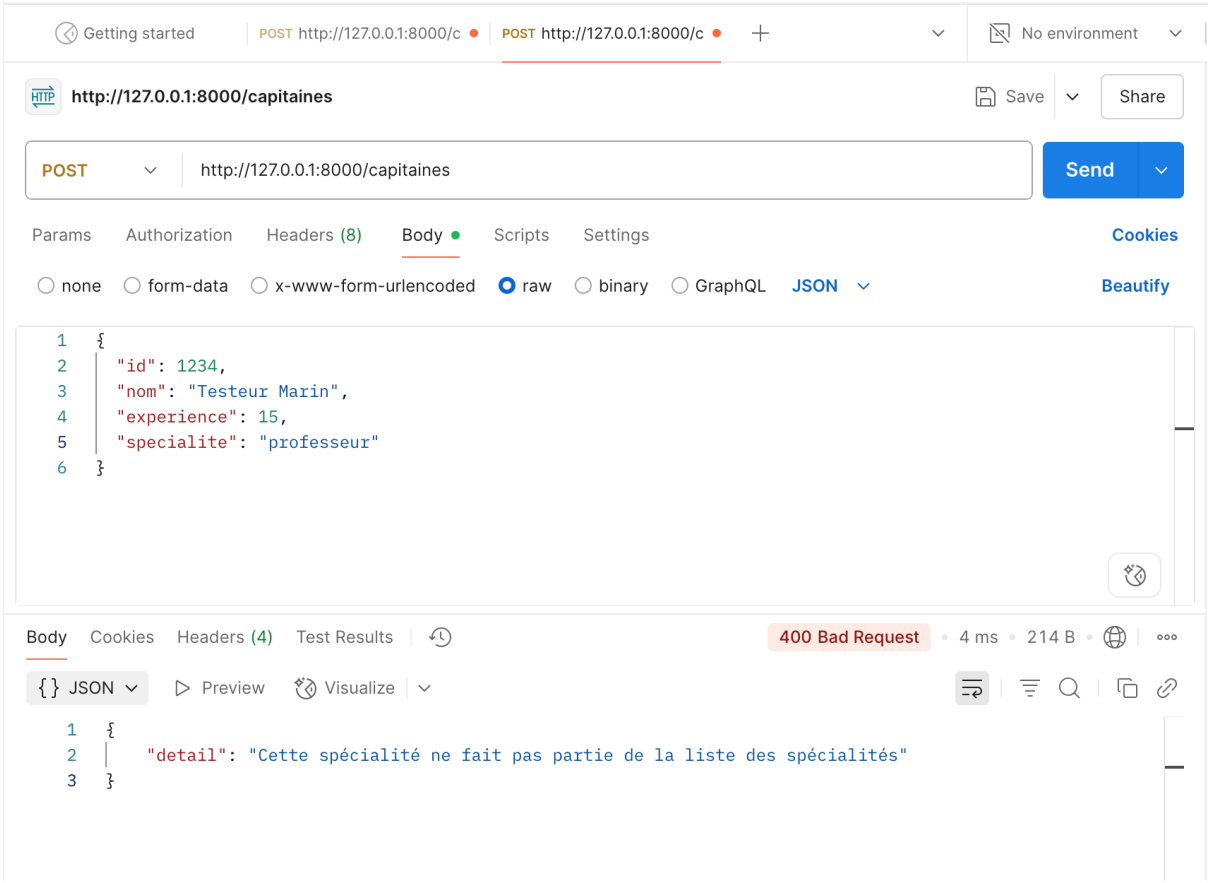
2. Vérification avec POSTMAN

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



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



3. Présentation du code source
    a. La class CAPITAINE

```python
# --------- Modèle Capitaine ----------
class Capitaine(BaseModel):
    id: int
    nom: str
    experience: int
    specialite: str
```

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

```python
@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
```