

Mise en place d'un web service sous FastAPI SPGoO – Août 2026 – YS

Objectif : mettre en place un service web pour accès de type CRUD à une base de données MongoDB.

source : <https://github.com/ystroppa/anfNoSQL2026>

Annuaire des chercheurs — FastAPI + MongoDB

Web service exposant un CRUD sur une collection MongoDB de chercheurs (nom, prénom, statut, affiliations, discipline, sujets de recherche).

Structure

```
chercheurs-api/
├── app/
│   ├── main.py          # routes FastAPI
│   ├── crud.py          # opérations MongoDB (create/read/update/delete)
│   ├── models.py        # modèles Pydantic
│   └── database.py       # connexion Motor (driver MongoDB async)
├── requirements.txt
├── docker-compose.yml   # MongoDB local en un clic
└── .env.example
```

Lancer le démonstrateur

1. Démarrer MongoDB (si vous n'avez pas déjà une instance) :

```
docker compose up -d
```

2. Installer les dépendances et lancer l'API :

```
python -m venv venv && source venv/bin/activate # Windows :
venv\Scripts\activate
pip install -r requirements.txt
cp .env.example .env
uvicorn app.main:app --reload
```

3. Ouvrir la documentation interactive :

<http://localhost:8000/docs> — Swagger UI générée automatiquement par FastAPI, permet de tester chaque route directement dans le navigateur.

Modèle de données

```
{
  "nom": "Curie",
  "prenom": "Marie",
  "statut": "Directrice de recherche",
  "affiliations": ["Institut du Radium", "CNRS"],
  "discipline": "Physique",
  "sujets_recherche": ["Radioactivité", "Isotopes"]
}
```

Routes disponibles

Méthode	Route	Description
POST	/chercheurs	Ajouter un chercheur
GET	/chercheurs	Lister les chercheurs (filtres optionnels : discipline, statut, affiliation)
GET	/chercheurs/{id}	Récupérer un chercheur
PATCH	/chercheurs/{id}	Modifier un chercheur (mise à jour partielle)
DELETE	/chercheurs/{id}	Supprimer un chercheur
POST	/chercheurs/{id}/affiliations	Ajouter une affiliation
DELETE	/chercheurs/{id}/affiliations/{valeur}	Enlever une affiliation
POST	/chercheurs/{id}/sujets	Ajouter un sujet de recherche
DELETE	/chercheurs/{id}/sujets/{valeur}	Enlever un sujet de recherche

Les routes `affiliations` / `sujets` permettent d'ajouter ou d'enlever un seul élément de la liste (opérateurs MongoDB `$addToSet` / `$pull`) sans avoir à renvoyer la liste complète — pratique pour un client léger.

Exemples avec curl

```
# Ajouter un chercheur
curl -X POST http://localhost:8000/chercheurs \
  -H "Content-Type: application/json" \
  -d '{
    "nom": "Curie", "prenom": "Marie",
    "statut": "Directrice de recherche",
    "affiliations": ["Institut du Radium"],
    "discipline": "Physique",
    "sujets_recherche": ["Radioactivité"]
  }'

# Lister les chercheurs en physique
curl "http://localhost:8000/chercheurs?discipline=Physique"

# Modifier le statut
curl -X PATCH http://localhost:8000/chercheurs/<id> \
  -H "Content-Type: application/json" \
  -d '{"statut": "Professeure"}'

# Ajouter une affiliation
curl -X POST http://localhost:8000/chercheurs/<id>/affiliations \
  -H "Content-Type: application/json" -d '{"valeur": "CNRS"}'

# Enlever une affiliation
curl -X DELETE "http://localhost:8000/chercheurs/<id>/affiliations/CNRS"
```

```
# Supprimer un chercheur  
curl -X DELETE http://localhost:8000/chercheurs/<id>
```

Notes de conception

- **Motor** (driver MongoDB async officiel) est utilisé plutôt que PyMongo synchrone, pour rester cohérent avec le modèle async de FastAPI.
- Les identifiants MongoDB (**ObjectId**) sont convertis en chaînes de caractères aux frontières de l'API : le client ne manipule jamais de type BSON.
- **PATCH** (mise à jour partielle) est utilisé plutôt que **PUT** : seuls les champs fournis sont modifiés, les autres restent inchangés.
- La logique MongoDB est isolée dans `crud.py`, séparée des routes HTTP — elle reste testable indépendamment de FastAPI.

Ecran de visualisation des entrées de ce web service à partir de l'url suivante :

<http://localhost:8000/docs>

Lancement du service à partir d'un terminal

```
(formmongodb) (base) ystroppa@isto.cnrs-orleans.fr@5CG6092ZML:~/formation_NOSQL/chercheurs-api-mongodb$  
uvicorn app.main:app --reload  
INFO: Will watch for changes in these directories: ['/home/ystroppa@isto.cnrs-orleans.fr/formation_NOSQL/chercheurs-  
api-mongodb']  
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)  
INFO: Started reloader process [1065901] using WatchFiles  
INFO: Started server process [1065903]  
INFO: Waiting for application startup.  
INFO: Application startup complete.
```

Annuaire des chercheurs 1.0.0 OAS 3.1

/openapi.json

Démonstrateur FastAPI exposant un CRUD sur une collection MongoDB de chercheurs.

default

POST	/chercheurs	Ajouter un chercheur	⌵
GET	/chercheurs	Lister les chercheurs	⌵
GET	/chercheurs/{chercheur_id}	Récupérer un chercheur	⌵
PATCH	/chercheurs/{chercheur_id}	Modifier un chercheur (mise à jour partielle)	⌵
DELETE	/chercheurs/{chercheur_id}	Supprimer un chercheur	⌵
POST	/chercheurs/{chercheur_id}/affiliations	Ajouter une affiliation	⌵
DELETE	/chercheurs/{chercheur_id}/affiliations/{valeur}	Enlever une affiliation	⌵
POST	/chercheurs/{chercheur_id}/sujets	Ajouter un sujet de recherche	⌵
DELETE	/chercheurs/{chercheur_id}/sujets/{valeur}	Enlever un sujet de recherche	⌵

Schemas

Chercheur > Expand all object

ChercheurCreate > Expand all object

ChercheurUpdate > Expand all object

HTTPValidationError > Expand all object

ItemTexte > Expand all object

ValidationError > Expand all object