



Documentation automatique avec FastAPI

Sommaire

- **Documentation automatique avec FastAPI**
 - **Sommaire**
 - **1. Documentation générée grâce aux schémas Pydantic**
 - **2. Interfaces graphiques disponibles**
 - **3. Exemple d'annotation d'un endpoint**
 - **4. Autres options de documentation**
 - **5. Résumé**

1. Documentation générée grâce aux schémas Pydantic

FastAPI utilise les modèles Pydantic (comme `UserCreate`, `UserRead`, etc.) pour :

- Générer automatiquement la structure des données attendues en entrée (exemple : corps JSON)
- Afficher les formats des données retournées (réponses)
- Valider les données envoyées par le client selon ces schémas

Cela permet d’avoir une documentation précise, toujours à jour et cohérente avec le code.

2. Interfaces graphiques disponibles

FastAPI génère deux interfaces interactives pour tester et documenter l’API :

Interface	URL par défaut	Description
Swagger UI	<code>/docs</code>	Documentation interactive et visuelle très complète, avec possibilité de tester les requêtes
ReDoc	<code>/redoc</code>	Documentation plus détaillée, orientée lecture et compréhension

Représentation visuelle :

FastAPI

0.10.0

0.10.0

OpenAPI JSON

Utilisateurs

GET

/users/

List Users

default

POST

/users/

Créer un utilisateur

GET

/users/{user_id}/read

Read User

GET

/users/{user_id}/get

Get User

Schemas

HTTPValidationError

Expand all

object

UserCreate

Expand all

object

UserRead

Expand all

object

ValidationError

Expand all

object

Search...

Utilisateurs

POST

Créer un utilisateur

GET

Read User

GET

Get User

FastAPI (0.1.0)

Download OpenAPI specification: Download

Utilisateurs

List Users

Responses

>

200 Successful Response

Créer un utilisateur

Crée un nouvel utilisateur dans la base.

name

nom complet de l'utilisateur

email

adresse email valide

GET

/users/

Response samples

200

Content type

application/json

Copy

null

POST

/users/

Request samples

Payload

docs

redoc

3. Exemple d'annotation d'un endpoint

```
from fastapi import FastAPI, Path, Query, Body

app = FastAPI()

@app.post(
    "/users/",
    response_model=UserRead,
    summary="Créer un utilisateur"
)
def create_user(
    user: UserCreate=Body(..., example={"name": "Alice", "email": "alice@mail.com"})
):
    """
    Crée un nouvel utilisateur dans la base.

    - **name**: nom complet de l'utilisateur
    - **email**: adresse email valide
    """
    # logique de création ici...
    return user # Exemple simplifié
```

- `response_model=UserRead` : indique le schéma attendu en réponse (sert à générer la doc et la validation)
- `summary` : résumé affiché dans Swagger UI
- `Body(..., example=...)` : exemple concret affiché dans la doc, aide à comprendre le format attendu
- La docstring de la fonction est utilisée comme description détaillée visible dans Swagger UI

POST **/users/** Créer un utilisateur

Crée un nouvel utilisateur dans la base.

- **name**: nom complet de l'utilisateur
- **email**: adresse email valide

Parameters

No parameters

Request body **required**

Example Value	Schema
<pre>{ "name": "Alice", "email": "alice@mail.com" }</pre>	

4. Autres options de documentation

- **Descriptions sur les paramètres** : on peut ajouter des descriptions dans les paramètres **Path**, **Query**, ou **Body** :

```
@app.get("/users/{user_id}")
def read_user(
    user_id: int = Path(..., description="L'ID unique de l'utilisateur"),
    active: bool | None = Query(None, description="Filtrer selon l'état actif")
):
    ...
```

GET
/users/{user_id}/read
Read User

Parameters

Name	Description
user_id * required integer (path)	L'ID unique de l'utilisateur <input type="text" value="user_id"/>
active boolean (boolean null) (query)	Filtrer selon l'état actif <input type="text" value="--"/>

- **Tags** : regrouper les endpoints par catégories dans la doc

```
app = FastAPI()

@app.get("/users/", tags=["Utilisateurs"])
def list_users():
    ...
```

Utilisateurs

GET
/users/
List Users

- **Réponses personnalisées** : on peut documenter les codes HTTP attendus

```
from fastapi.responses import JSONResponse
from fastapi import status

@app.get(
    "/users/{user_id}",
    responses={
        404: {"description": "Utilisateur non trouvé"},
        200: {"description": "Utilisateur trouvé"}
    }
)
```

```
def get_user(user_id: int):  
    ...
```

Responses

Code

Description

200

Utilisateur trouvé

Media type

application/json



Controls Accept header.

Example Value | Schema

"string"

404

Utilisateur non trouvé

422

Validation Error

5. Résumé

Grâce aux schémas Pydantic et aux annotations sur les endpoints, FastAPI produit une documentation :

- **automatique, précise et interactive**
- **facile à maintenir** (car directement liée au code)
- **accessible aux développeurs et consommateurs d'API**