PyMongo

Index

Edgar López Hernandez
Belén Gamero Garcia

Tutorial

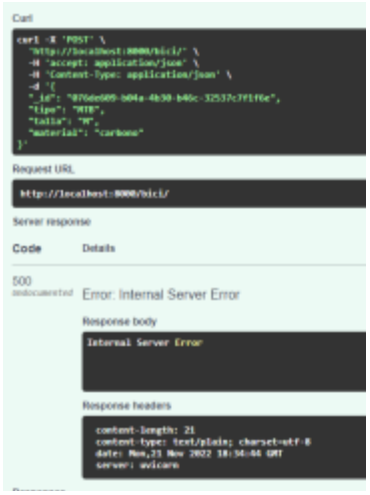
- At this point we have already done all the steps of the tutorial, obviously modifying the necessary parameters to match our Bicycles database. [PyMongo Tutorial]

Result

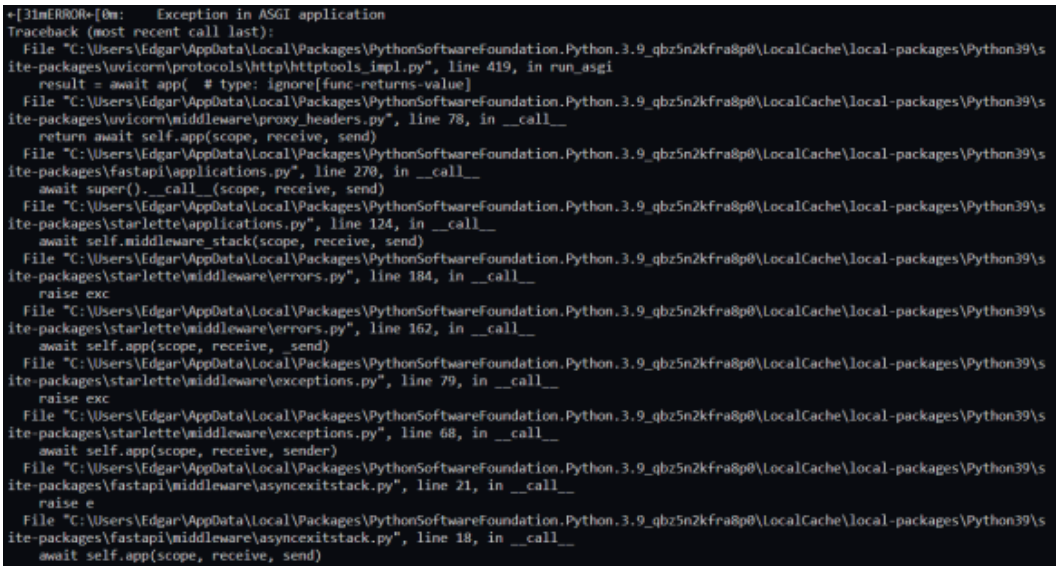
Here we create a new bicycle:


```
{
  "_id": "076de609-b04a-4b30-b46c-32537c7f1f6e",
  "tipo": "MTB",
  "talla": "M",
  "material": "carbono"
}
```


it gave us an error error



message in the terminal



Edgar López Hernandez
Belén Gamero Garcia

now we will see the content of the files

routes.py

```python
from fastapi import APIRouter, Body, Request, Response, HTTPException, status
from fastapi.encoders import jsonable_encoder
from typing import List

from models import Bicicletas, BicicletasUpdate

router = APIRouter()

@router.post("/", response_description="Create a new bici", status_code=status.HTTP_201_CREATED, response_model=Bicicletas)
def create_bicicleta(request: Request, Bicicletas: Bicicletas = Body(...)):
    bici = jsonable_encoder(Bicicletas)
    new_bici = request.app.database["Bicicletas"].insert_one(bici)
    created_bici = request.app.database["Bicicletas"].find_one(
        {"_id": new_bici.inserted_id}
    )

    return created_bici

@router.get("/", response_description="List all bicis", response_model=List[Bicicletas])
def list_bici(request: Request):
    bicis = list(request.app.database["Bicicletas"].find(limit=100))
    return bicis

@router.get("/{id}", response_description="Get a single bici by id", response_model=Bicicletas)
def find_bici(id: str, request: Request):
    if (bici := request.app.database["Bicicletas"].find_one({"_id": id})) is not None:
        return bici
    raise HTTPException(status_code=status.HTTP_404_NOT_FOUND, detail=f"Bici with ID {id} not found")

@router.put("/{id}", response_description="Update a bici", response_model=Bicicletas)
def update_bici(id: str, request: Request, Bicicletas: BicicletasUpdate = Body(...)):
    bici = {k: v for k, v in bici.dict().items() if v is not None}
    if len(bici) >= 1:
        update_result = request.app.database["Bicicletas"].update_one(
            {"_id": id}, {"$set": bici}
        )

        if update_result.modified_count == 0:
            raise HTTPException(status_code=status.HTTP_404_NOT_FOUND, detail=f"bici with ID {id} not found")
```

```python
    if (
        existing_bici := request.app.database["Bicicletas"].find_one({"_id": id})
    ) is not None:
        return existing_bici

    raise HTTPException(status_code=status.HTTP_404_NOT_FOUND, detail=f"bici with ID {id} not found")

@router.delete("/{id}", response_description="Delete a bici")
def delete_bici(id: str, request: Request, response: Response):
    delete_result = request.app.database["Bicicletas"].delete_one({"_id": id})

    if delete_result.deleted_count == 1:
        response.status_code = status.HTTP_204_NO_CONTENT
        return response

    raise HTTPException(status_code=status.HTTP_404_NOT_FOUND, detail=f"bici with ID {id} not found")
```

main.py

```python
from fastapi import FastAPI
from dotenv import dotenv_values
from pymongo import MongoClient
from routes import router as bici_router

config = dotenv_values(".env")

app = FastAPI()

@app.on_event("startup")
def startup_db_client():
    app.mongodb_client = MongoClient(config["ATLAS_URI"])
    app.database = app.mongodb_client[config["DB_NAME"]]

@app.on_event("shutdown")
def shutdown_db_client():
    app.mongodb_client.close()

app.include_router(bici_router, tags=["Bicicletas"], prefix="/bici")
```

models.py

```python
import uuid
from typing import Optional
from pydantic import BaseModel, Field

class Bicicletas(BaseModel):
    id: str = Field(default_factory=uuid.uuid4, alias="_id")
    tipo: str = Field(...)
    talla: str = Field(...)
    material: str = Field(...)

    class Config:
        allow_population_by_field_name = True
        schema_extra = {
            "example": {
                "_id": "066de609-b04a-4b30-b46c-32537c7f1f6e",
                "tipo": "MTB",
                "talla": "s",
                "material": "metal"
            }
        }

class BicicletasUpdate(BaseModel):
    tipo: Optional[str]
    talla: Optional[str]
    material: Optional[str]

    class Config:
        schema_extra = {
            "example": {
                "tipo": "MTB",
                "talla": "s",
                "material": "metal"
            }
        }
```