from fastapi import FastAPI, HTTPException  
from pydantic import BaseModel  
from typing import List, Optional  
  
app = FastAPI()  
  
# In-memory data store  
books\_db = {}  
authors\_db = {}  
  
# Models  
class Author(BaseModel):  
 id: int  
 name: str  
 nationality: Optional[str] = None  
  
class Book(BaseModel):  
 id: int  
 title: str  
 author\_id: int  
 year: Optional[int] = None  
  
# ------------------ AUTHORS ENDPOINTS ------------------ #  
  
@app.post("/authors/", response\_model=Author)  
def create\_author(author: Author):  
 if author.id in authors\_db:  
 raise HTTPException(status\_code=400, detail="Author already exists")  
 authors\_db[author.id] = author  
 return author  
  
@app.get("/authors/", response\_model=List[Author])  
def get\_authors():  
 return list(authors\_db.values())  
  
@app.get("/authors/{author\_id}", response\_model=Author)  
def get\_author(author\_id: int):  
 if author\_id not in authors\_db:  
 raise HTTPException(status\_code=404, detail="Author not found")  
 return authors\_db[author\_id]  
  
@app.put("/authors/{author\_id}", response\_model=Author)  
def update\_author(author\_id: int, author: Author):  
 if author\_id != author.id or author\_id not in authors\_db:  
 raise HTTPException(status\_code=404, detail="Author not found")  
 authors\_db[author\_id] = author  
 return author  
  
@app.delete("/authors/{author\_id}")  
def delete\_author(author\_id: int):  
 if author\_id not in authors\_db:  
 raise HTTPException(status\_code=404, detail="Author not found")  
 del authors\_db[author\_id]  
 return {"detail": "Author deleted"}  
  
# ------------------ BOOKS ENDPOINTS ------------------ #  
  
@app.post("/books/", response\_model=Book)  
def create\_book(book: Book):  
 if book.id in books\_db:  
 raise HTTPException(status\_code=400, detail="Book already exists")  
 if book.author\_id not in authors\_db:  
 raise HTTPException(status\_code=404, detail="Author does not exist")  
 books\_db[book.id] = book  
 return book  
  
@app.get("/books/", response\_model=List[Book])  
def get\_books():  
 return list(books\_db.values())  
  
@app.get("/books/{book\_id}", response\_model=Book)  
def get\_book(book\_id: int):  
 if book\_id not in books\_db:  
 raise HTTPException(status\_code=404, detail="Book not found")  
 return books\_db[book\_id]  
  
@app.put("/books/{book\_id}", response\_model=Book)  
def update\_book(book\_id: int, book: Book):  
 if book\_id != book.id or book\_id not in books\_db:  
 raise HTTPException(status\_code=404, detail="Book not found")  
 books\_db[book\_id] = book  
 return book  
  
@app.delete("/books/{book\_id}")  
def delete\_book(book\_id: int):  
 if book\_id not in books\_db:  
 raise HTTPException(status\_code=404, detail="Book not found")  
 del books\_db[book\_id]  
 return {"detail": "Book deleted"}