資料庫檢視器 - 完整項目架構

📁 項目目錄結構

```
database-viewer/
├── 📄 main.py                # FastAPI 後端主程序
├── 📄 requirements.txt       # Python 依賴項列表
├── 📄 .env                   # 環境變量配置文件（可選）
├── 📄 config.py              # 配置管理文件（可選）
├── 📄 README.md              # 項目說明文檔
├── 📄 start.py               # 啟動腳本
├── 📄 start.bat              # Windows 啟動腳本
├── 📄 start.sh               # Linux/Mac 啟動腳本
├── 📁 static/                # 靜態資源目錄
│   ├── 📄 index.html         # 主前端頁面
│   ├── 📁 css/               # CSS 樣式文件
│   │   └── 📄 style.css      # 主樣式文件
│   ├── 📁 js/                # JavaScript 文件
│   │   └── 📄 app.js         # 主應用程序 JS
│   └── 📁 assets/            # 其他資源文件
│       └── 📄 favicon.ico    # 網站圖標
├── 📁 templates/             # HTML 模板目錄（可選）
│   └── 📄 base.html          # 基礎模板
├── 📁 uploads/               # 上傳文件存儲目錄
├── 📁 temp/                  # 臨時文件目錄
├── 📁 logs/                  # 日誌文件目錄
│   └── 📄 app.log            # 應用程序日誌
├── 📁 api/                   # API 模組化目錄（可選）
│   ├── 📄 __init__.py        # Python 包初始化
│   ├── 📄 milvus_api.py      # Milvus API 端點
│   ├── 📄 duckdb_api.py      # DuckDB API 端點
│   └── 📄 models.py          # Pydantic 模型
├── 📁 utils/                 # 工具函數目錄（可選）
│   ├── 📄 __init__.py        # Python 包初始化
│   ├── 📄 database.py        # 數據庫連接工具
│   ├── 📄 file_handler.py    # 文件處理工具
```

```
|   └── 📄 logger.py          # 日誌配置
└── 📁 tests/                 # 測試文件目錄（可選）
    ├── 📄 __init__.py        # Python 包初始化
    ├── 📄 test_milvus.py     # Milvus 功能測試
    └── 📄 test_duckdb.py     # DuckDB 功能測試
```

📋 必需文件列表

1. 核心後端文件

main.py - FastAPI 主程序

```
python

# 已提供的完整後端代碼
```

requirements.txt - Python 依賴項

```
text

# 已提供的依賴項列表
```

2. 前端文件

static/index.html - 主前端頁面

```
html

# 已提供的完整前端代碼
```

3. 配置文件

.env - 環境變量配置

```
env
```

```
# Milvus 配置
MILVUS_HOST=localhost
MILVUS_PORT=19530

# 服務器配置
HOST=0.0.0.0
PORT=8000
DEBUG=True

# 文件上傳配置
MAX_FILE_SIZE=104857600  # 100MB
UPLOAD_DIR=./uploads

# 日誌配置
LOG_LEVEL=INFO
LOG_FILE=./logs/app.log

# 數據庫配置
TEMP_DIR=./temp
```

config.py - 配置管理

python

```python
import os
from pydantic import BaseSettings

class Settings(BaseSettings):
    # Milvus 設置
    milvus_host: str = "localhost"
    milvus_port: int = 19530

    # 服務器設置
    host: str = "0.0.0.0"
    port: int = 8000
    debug: bool = True

    # 文件上傳設置
    max_file_size: int = 104857600  # 100MB
    upload_dir: str = "./uploads"
    temp_dir: str = "./temp"

    # 日誌設置
    log_level: str = "INFO"
    log_file: str = "./logs/app.log"

    class Config:
        env_file = ".env"

settings = Settings()
```

4. 啟動腳本

start.py - Python 啟動腳本

```python
```

```python
#!/usr/bin/env python3
import os
import sys
import subprocess
import uvicorn
from pathlib import Path

def setup_directories():
    """創建必要的目錄"""
    directories = [
        "uploads",
        "temp",
        "logs",
        "static/css",
        "static/js",
        "static/assets"
    ]

    for directory in directories:
        Path(directory).mkdir(parents=True, exist_ok=True)

    print("✅ 目錄結構創建完成")

def check_dependencies():
    """檢查依賴項是否已安裝"""
    try:
        import fastapi
        import uvicorn
        import pymilvus
        import duckdb
        print("✅ 所有依賴項已安裝")
        return True
    except ImportError as e:
        print(f"❌ 缺少依賴項: {e}")
        print("請運行: pip install -r requirements.txt")
        return False
```

```python
def main():
    """主啟動函數"""
    print("🚀 啟動資料庫檢視器...")

    # 檢查依賴項
    if not check_dependencies():
        sys.exit(1)

    # 設置目錄
    setup_directories()

    # 啟動服務器
    print("🌐 啟動 FastAPI 服務器...")
    uvicorn.run(
        "main:app",
        host="0.0.0.0",
        port=8000,
        reload=True,
        log_level="info"
    )

if __name__ == "__main__":
    main()
```

start.bat - Windows 批處理啟動腳本

```batch
```

```
@echo off
echo 🚀 啟動資料庫檢視器...

REM 檢查 Python 是否安裝
python --version >nul 2>&1
if errorlevel 1 (
    echo ❌ Python 未安裝或未添加到 PATH
    pause
    exit /b 1
)

REM 檢查虛擬環境
if exist "venv\Scripts\activate.bat" (
    echo 🔧 激活虛擬環境...
    call venv\Scripts\activate.bat
)

REM 安裝依賴項
echo 📦 檢查依賴項...
pip install -r requirements.txt

REM 啟動應用程序
echo 🌐 啟動服務器...
python start.py

pause
```

start.sh - Linux/Mac Shell 啟動腳本

```
bash
```

```
#!/bin/bash
echo "🚀 啟動資料庫檢視器..."

# 檢查 Python 是否安裝
if ! command -v python3 &> /dev/null; then
    echo "❌ Python3 未安裝"
    exit 1
fi

# 檢查並激活虛擬環境
if [ -d "venv" ]; then
    echo "🔧 激活虛擬環境..."
    source venv/bin/activate
fi

# 安裝依賴項
echo "📦 檢查依賴項..."
pip install -r requirements.txt

# 創建必要目錄
mkdir -p uploads temp logs static/{css,js,assets}

# 啟動應用程序
echo "🌐 啟動服務器..."
python3 start.py
```

5. 文檔文件

README.md - 項目說明文檔

```
markdown
```

```
# 資料庫檢視器

## 快速開始

1. 安裝 Python 3.8+
2. 克隆或下載項目文件
3. 運行啟動腳本：
   - Windows: `start.bat`
   - Linux/Mac: `./start.sh`
   - 或直接運行: `python start.py`

## 訪問地址

- 主界面: http://localhost:8000
- API 文檔: http://localhost:8000/docs

## 功能說明

- Milvus 集合檢視器
- DuckDB 數據檢視器

詳細使用說明請參考使用指南。
```

🚀 部署步驟

第一步：創建項目目錄

```bash
mkdir database-viewer
cd database-viewer
```

第二步：創建文件

按照上述架構創建所有必需的文件和目錄。

第三步：設置 Python 環境

```bash
# 創建虛擬環境
python -m venv venv

# 激活虛擬環境
# Windows:
venv\Scripts\activate
# Linux/Mac:
source venv/bin/activate

# 安裝依賴項
pip install -r requirements.txt
```

第四步：啟動應用程序

```bash
# 方法 1：使用啟動腳本
python start.py

# 方法 2：直接運行主程序
python main.py

# 方法 3：使用 uvicorn
uvicorn main:app --host 0.0.0.0 --port 8000 --reload
```

📂 可選的模組化結構

如果項目變得更複雜，可以考慮以下模組化結構：

```
database-viewer/
├── app/                    # 應用程序主目錄
│   ├── __init__.py
│   ├── main.py             # FastAPI 應用程序
│   ├── config.py           # 配置管理
│   ├── dependencies.py     # 依賴注入
│   ├── api/                # API 路由
│   │   ├── __init__.py
│   │   ├── milvus.py
│   │   └── duckdb.py
│   ├── models/             # 數據模型
│   │   ├── __init__.py
│   │   ├── milvus.py
│   │   └── duckdb.py
│   ├── services/           # 業務邏輯服務
│   │   ├── __init__.py
│   │   ├── milvus_service.py
│   │   └── duckdb_service.py
│   └── utils/              # 工具函數
│       ├── __init__.py
│       ├── logger.py
│       └── file_handler.py
├── static/                 # 靜態文件
├── uploads/                # 上傳文件
├── temp/                   # 臨時文件
├── logs/                   # 日誌文件
├── tests/                  # 測試文件
├── requirements.txt
├── .env
└── run.py                  # 啟動入口點
```

🔧 開發建議

1. **版本控制**：使用 Git 管理代碼
2. **環境隔離**：使用虛擬環境

3. **配置管理**：使用 .env 文件管理配置
4. **日誌記錄**：配置適當的日誌級別
5. **錯誤處理**：實現全面的錯誤處理機制
6. **測試**：編寫單元測試和集成測試

這個架構提供了一個清晰、可維護的項目結構，適合在 localhost 上開發和測試。