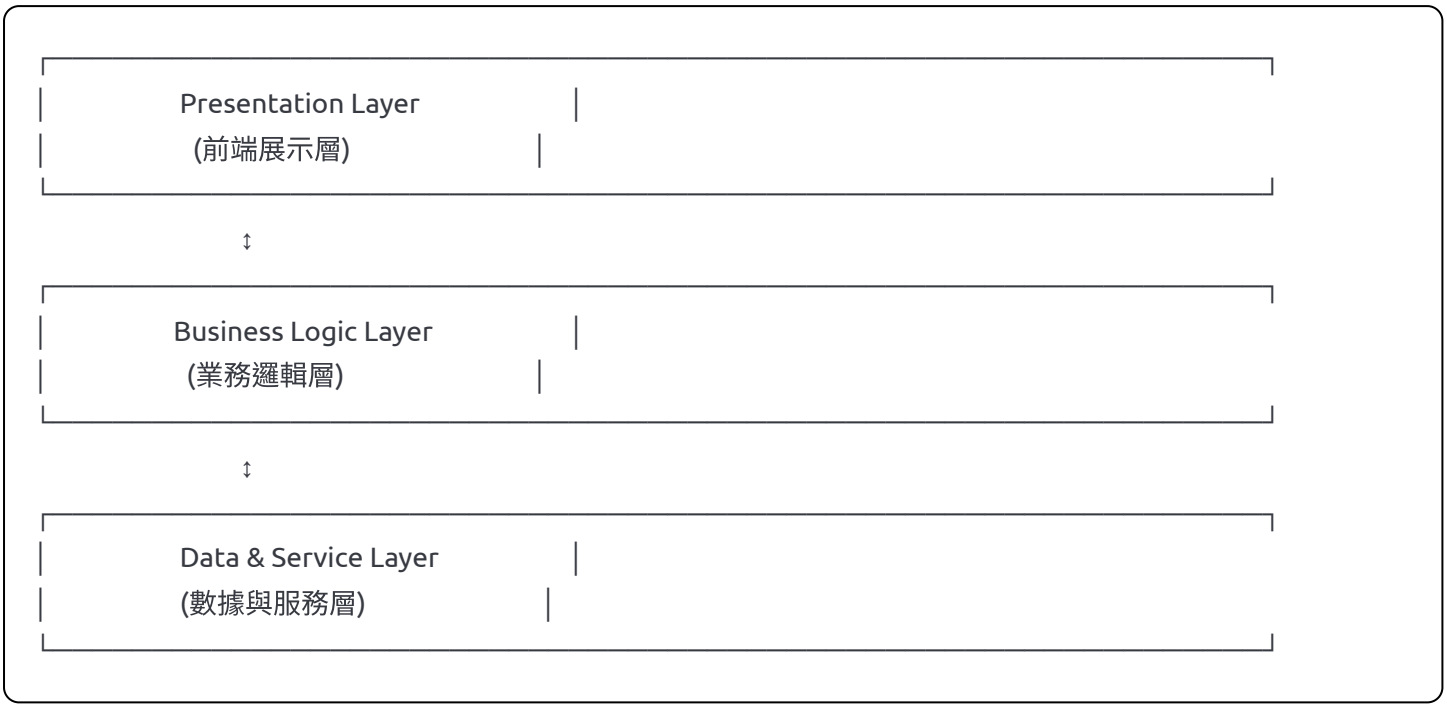
TeachBrief.AI 三層架構設計

🎯 架構總覽



🖥️ 第一層：Presentation Layer (前端展示層)

核心技術棧

- **框架**: Next.js 14 + TypeScript
- **樣式**: Tailwind CSS + Headless UI
- **狀態管理**: Zustand / Redux Toolkit
- **文件上傳**: React Dropzone
- **圖表組件**: Recharts / Chart.js

主要組件模組

```
src/
├── components/
│   ├── FileUpload/       # 多文件上傳組件
│   ├── PPTEditor/        # 簡報編輯器
│   ├── PreviewPanel/     # 預覽面板
│   ├── TemplateSelector/  # 模板選擇器
│   └── ProgressTracker/   # 生成進度追蹤
├── pages/
│   ├── dashboard/        # 主控台
│   ├── create/           # 簡報創建頁面
│   ├── edit/             # 編輯頁面
```

```
│   └── library/          # 簡報庫
└── hooks/
    ├── useFileProcessor.ts  # 文件處理 hook
    ├── usePPTGenerator.ts   # 簡報生成 hook
    └── useAIIntegration.ts  # AI 整合 hook
```

頁面流程

1. **文件上傳頁** → 支援多格式文件上傳 (PDF, DOC, TXT, Images)
2. **內容配置頁** → 設定簡報參數 (主題、風格、頁數)
3. **生成處理頁** → 即時顯示 AI 處理進度
4. **編輯預覽頁** → 簡報內容編輯與預覽
5. **匯出下載頁** → 多格式匯出 (PPTX, PDF, Google Slides)

⚙️ 第二層：Business Logic Layer (業務邏輯層)

核心技術棧

- **主框架**: FastAPI + Python 3.11
- **非同步處理**: Celery + Redis
- **API 閘道**: Nginx (反向代理)
- **認證授權**: JWT + OAuth 2.0

服務模組架構

```
app/
├── api/
│   ├── v1/
│   │   ├── presentations/   # 簡報相關 API
│   │   ├── files/           # 文件處理 API
│   │   ├── ai/              # AI 服務 API
│   │   └── templates/       # 模板管理 API
├── services/
│   ├── content_processor/   # 內容處理服務
│   ├── ai_orchestrator/     # AI 模型協調器
│   ├── presenton_client/    # Presenton API 客戶端
│   └── image_generator/     # 圖像生成服務
├── models/
│   ├── presentation.py      # 簡報數據模型
│   ├── user.py              # 用戶模型
│   └── file.py              # 文件模型
└── utils/
    ├── file_parser.py       # 文件解析工具
```

```
├── content_merger.py  # 內容整合工具
└── export_handler.py  # 匯出處理工具
```

核心業務邏輯

1. AI 模型協調器 (AI Orchestrator)

```python
class AIOrchestrator:
    def __init__(self):
        self.llm_models = {
            'mistral': MistralClient(),
            'phi2': Phi2Client(),
            'llama3': LlamaClient(),
            'zephyr': ZephyrClient()
        }
        self.image_models = {
            'sdxl': StableDiffusionXL(),
            'blip2': BLIP2Client()
        }

    async def generate_content(self, files, instructions):
        # 多文件整合與摘要
        # 自動選擇最適合的 LLM 模型
        # 生成簡報大綱與內容
        pass

    async def generate_images(self, content_context):
        # 根據內容生成配圖
        # 圖片延伸與優化
        pass
```

2. Presenton API 整合服務

```python
```

```python
class PresentonIntegration:
    def __init__(self):
        self.base_url = "http://presenton-api:5000"

    async def create_presentation(self, enhanced_content):
        # 將 AI 處理後的內容轉換為 Presenton 格式
        # 調用 Presenton API 生成基礎簡報
        # 後處理和優化簡報內容
        pass

    async def enhance_with_ai(self, presentation_data):
        # 用我們的 AI 模型增強 Presenton 生成的內容
        # 添加自定義配圖和樣式
        pass
```

3. 內容處理管道

```python
class ContentProcessingPipeline:
    async def process_files(self, uploaded_files):
        # 步驟 1: 文件解析和內容提取
        # 步驟 2: 多文件內容整合
        # 步驟 3: 關鍵信息摘要
        # 步驟 4: 簡報結構化
        # 步驟 5: 生成講稿和配圖建議
        pass
```

🗄️ 第三層：Data & Service Layer (數據與服務層)

數據存儲

```
├── PostgreSQL          # 主要關聯數據
│   ├── users           # 用戶資料
│   ├── presentations    # 簡報元數據
│   ├── templates       # 模板數據
│   └── usage_logs      # 使用記錄
├── MongoDB             # 非結構化數據
│   ├── file_contents    # 文件原始內容
│   ├── ai_generations   # AI 生成記錄
│   └── presentation_cache # 簡報緩存
├── MinIO/S3            # 文件存儲
│   ├── uploaded_files/   # 用戶上傳文件
│   ├── generated_ppts/   # 生成的簡報
```

```
│   ├── images/          # 生成的圖片
│   └── exports/         # 匯出文件
└── Redis                # 緩存和任務隊列
    ├── session_cache    # 會話緩存
    ├── ai_model_cache   # 模型推理緩存
    └── celery_queue     # 任務隊列
```

AI 模型服務部署

```
├── Ollama 服務集群
│   ├── mistral-7b-instruct
│   ├── phi-2
│   ├── llama3-8b
│   └── zephyr-7b
├── ComfyUI 圖像服務
│   ├── stable-diffusion-xl
│   └── blip-2
├── Hugging Face Transformers
│   ├── 文本摘要模型
│   └── 文本分類模型
└── Presenton API 服務
    └── 簡報生成核心引擎
```

外部服務整合

```
├── 雲端服務
│   ├── Google Drive API   # Google Slides 匯出
│   ├── OneDrive API       # PowerPoint Online 整合
│   └── Dropbox API        # 文件同步
├── 通知服務
│   ├── Email Service      # 完成通知
│   └── Webhook            # 狀態回調
└── 監控服務
    ├── Prometheus         # 指標監控
    ├── Grafana            # 監控儀表板
    └── Sentry             # 錯誤追蹤
```

🔄 完整工作流程

用戶操作流程

1. **文件上傳** → 前端上傳多份教學文件
2. **參數設定** → 選擇模板、風格、語言等參數

3. **AI 處理** → 中間層協調多個 AI 模型處理內容
4. **Presenton 生成** → 調用 Presenton API 生成基礎簡報
5. **AI 增強** → 用自有 AI 模型增強簡報內容和配圖
6. **用戶編輯** → 前端提供編輯界面供用戶調整
7. **多格式匯出** → 支援 PPTX、PDF、Google Slides 匯出

技術處理流程

```mermaid
graph TD
    A[用戶上傳文件] --> B[文件解析與內容提取]
    B --> C[多文件內容整合]
    C --> D[AI 摘要與結構化]
    D --> E[調用 Presenton API]
    E --> F[AI 配圖生成]
    F --> G[內容優化與後處理]
    G --> H[用戶編輯界面]
    H --> I[多格式匯出]
```

🚀 部署建議

容器化部署 (Docker Compose)

```yaml
```

```yaml
version: '3.8'
services:
 frontend:
  build: ./frontend
  ports: ["3000:3000"]

 api:
  build: ./backend
  ports: ["8000:8000"]

 ollama:
  image: ollama/ollama
  volumes: ["./models:/root/.ollama"]

 presenton:
  image: presenton/api
  ports: ["5000:5000"]

 postgres:
  image: postgres:15

 redis:
  image: redis:alpine

 minio:
  image: minio/minio
```

擴展性考量

- **水平擴展**: API 服務可通過 Load Balancer 進行多實例部署
- **模型分離**: AI 模型服務可獨立擴展，根據使用量調整資源
- **緩存策略**: 多層緩存提升響應速度
- **異步處理**: 使用 Celery 處理耗時任務，提升用戶體驗

📊 預期效益

- ✅ 教師備課時間減少 50% 以上
- ✅ 提升教材品質與視覺表現
- ✅ 降低教學資料重工與重複編輯
- ✅ 支援多語言和多種教學模板
- ✅ 完全開源，適合教育機構部署