

重韻開發指南

第二版

# 目錄

[目錄 2](#_Toc170242137)

[原則 3](#_Toc170242138)

[常規流程 4](#_Toc170242139)

[項目架構 5](#_Toc170242140)

[模組化設計 5](#_Toc170242141)

[前後端分離 6](#_Toc170242142)

[API風格 8](#_Toc170242143)

[代碼分層 10](#_Toc170242144)

[代碼目錄 11](#_Toc170242145)

[父系目錄 – 項目根目錄 11](#_Toc170242146)

[子系目錄 - 模塊 12](#_Toc170242147)

[Docker Compose 13](#_Toc170242148)

[代碼版本 14](#_Toc170242149)

[編程語言 15](#_Toc170242150)

[後端框架/工具推薦 16](#_Toc170242151)

[數據庫 16](#_Toc170242152)

[框架 17](#_Toc170242153)

[API規範 20](#_Toc170242154)

[身份驗證 22](#_Toc170242155)

[前端框架/工具推薦 23](#_Toc170242156)

[Web Framework 23](#_Toc170242157)

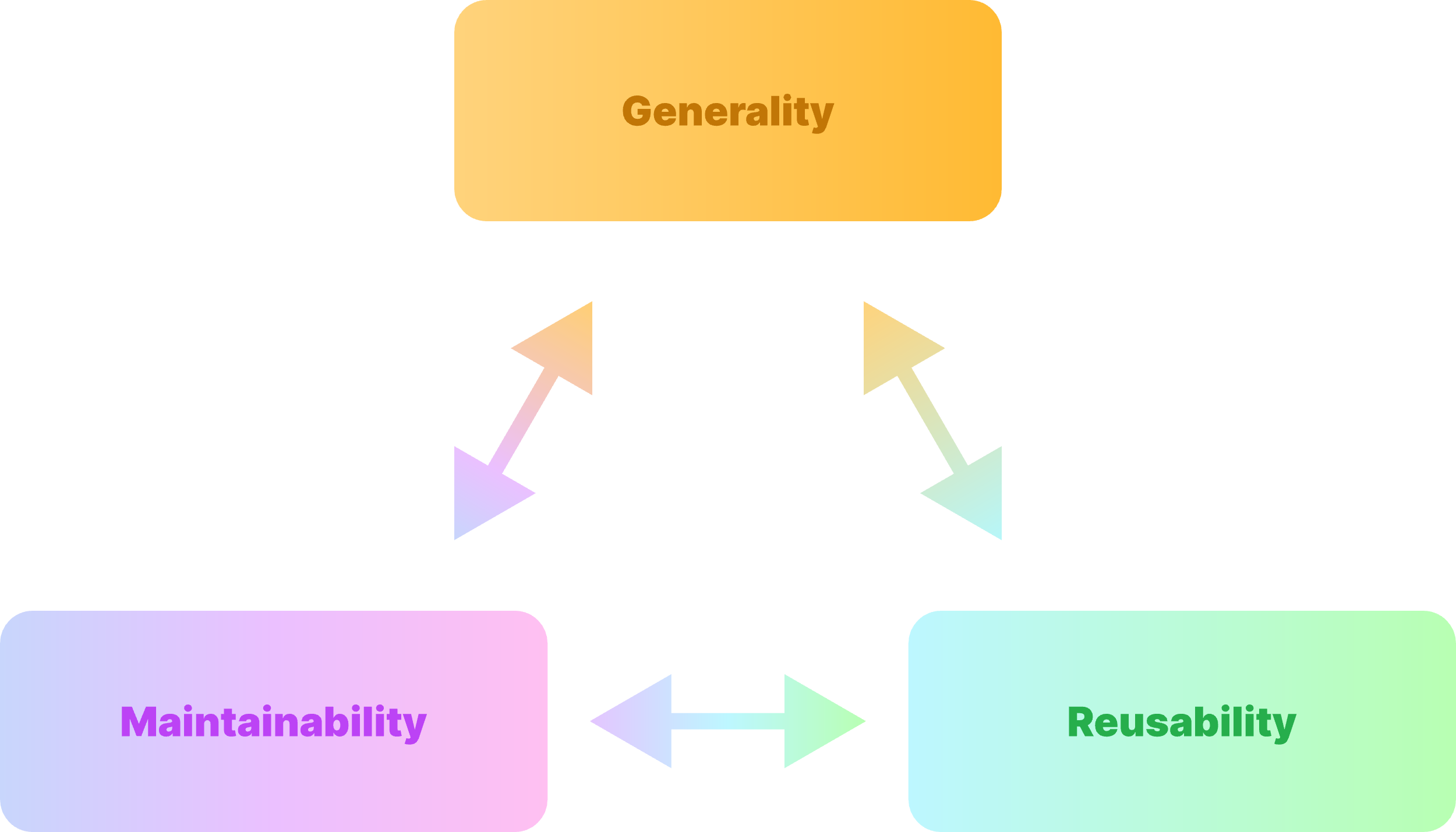
[Theme 23](#_Toc170242158)

## 原則

在開發過程中，確保明確定義項目的目標和期望結果。這有助於確保整個團隊的共識，並使開發過程更具方向性和效率。

為了項目的長期發展，需要考慮以下三點：

1. 通用性（Generality）：軟件應具有廣泛的應用價值，而不僅僅針對特定需求或情境。
2. 可維護性（Maintanability）：軟件應易於維護和修改，包括代碼可讀性、詳細的文檔等。
3. 複用性（ Reusability ）：軟件應具有可重用的模塊，以便在不同的項目中重複使用。



## 常規流程

1. 提出概念（idea）
2. ⁠列出需求
3. ⁠設計架構
4. ⁠定義功能
5. ⁠UI設計
6. ⁠開發
7. ⁠QA測試
8. ⁠上架

## 項目架構

### 模組化設計

模組化設計可以為開發過程帶來永續性和效率。

模組化設計將系統分解為許多小單元，也稱為模組（modules）或區塊（blocks），它們可以在不同的系統中獨立建構和使用。就像一個拼圖，每個拼圖代表一個模組（modules），整個拼圖就代表一個完整的系統。

一張含有 文字, 螢幕擷取畫面, 圖形, 字型 的圖片

自動產生的描述

透過模組化，可以將系統的不同功能和組件分離，封裝成獨立的模組，促進代碼和元件的可重用性。這使得我們可以靈活地選擇和組合這些模組來滿足不同專案的需求，避免每個開發週期都需要使用不同的技術，節省了大量的時間和精力。

## 前後端分離

為了在設計上實現更好的模組化，我們採用前後端分離的方式，透過API來實現不同模組之間的通訊。此設計解決方案具有多個優點。

一張含有 文字, 螢幕擷取畫面, 字型, 圖形 的圖片

自動產生的描述

首先，是可以最小化不同前後端之間的依賴關係。只需要專注於自己的功能，而不需要考慮其他對方的具體實作細節。這種鬆散耦合的設計允許獨立開發、測試和維護，同時提高代碼的可維護性。

其次，實現可重用性和可擴充性。這種設計允許前端和後端職責輕鬆分離開發團隊，前後端之間只透過API提供一組明確定義的接口，通過接口就可以允許一個模組使用其他模組的功能，而無需了解其內部實作細節。只要保持API接口一樣，就能夠輕鬆更換、升級或擴展單個模組/前後端，而不會影響整個系統。

## API風格

在一般開發中，我們採用了RESTful API設計風格來實作我們的API介面。

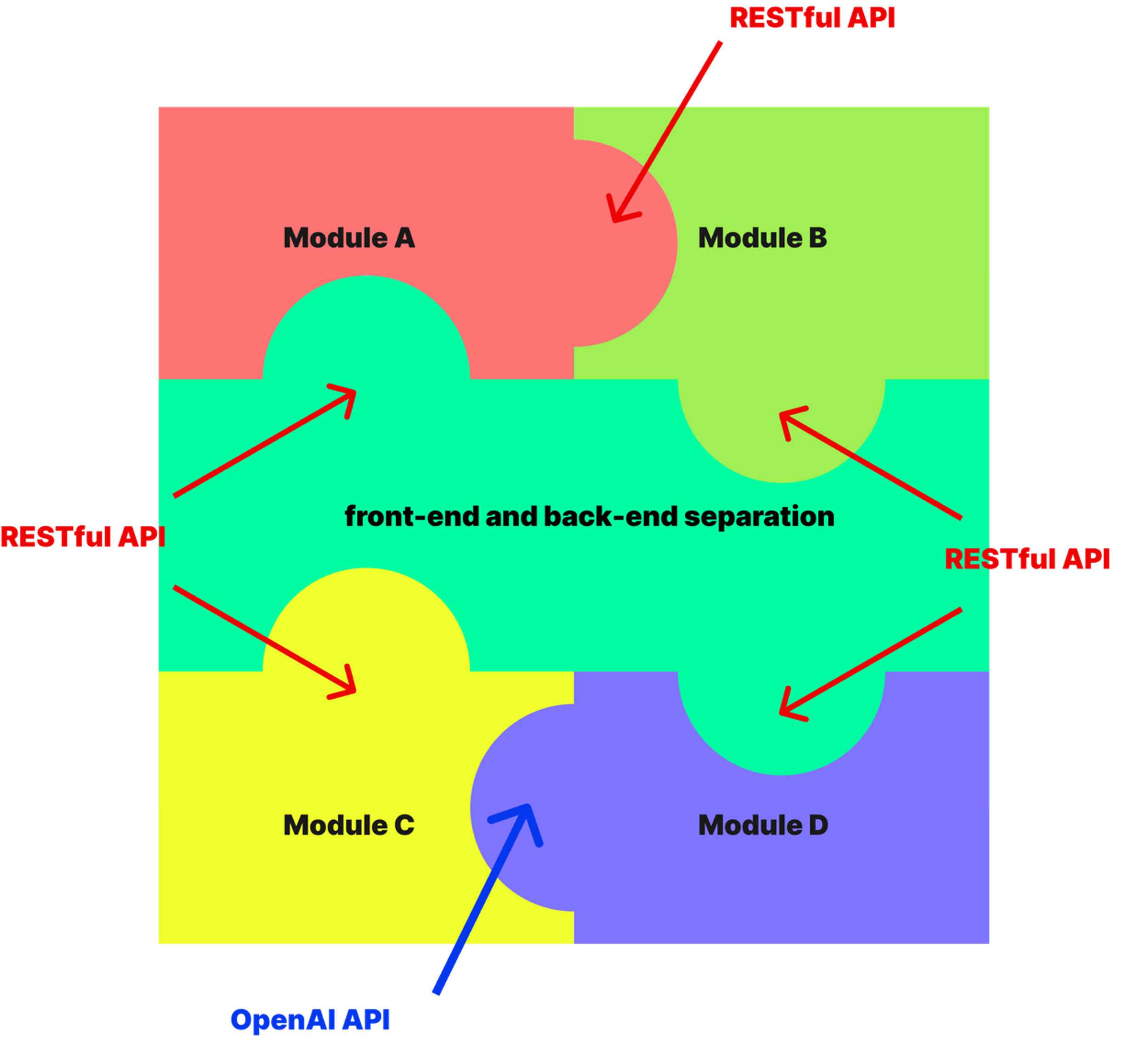
RESTful API是一種基於HTTP協定的設計風格，利用統一的介面和狀態碼提供輕量級、可擴充、無狀態的資源操作。

RESTful API的設計風格包括以下特點：

統一方法：RESTful API 採用一套統一的 HTTP 方法（GET、POST、PUT、DELETE 等）和 URL 來進行資源操作。這使得不同的客戶端（例如Web應用、行動應用）可以使用相同的接口進行操作，降低學習成本並提高開發效率。

輕量級且可擴充性：RESTful API 使用輕量級資料格式（例如 JSON 或 XML）進行資源傳輸，導致資料負載較小且易於解析。此外，由於 RESTful API 結構簡單明了，因此可以直接擴充新的資源類型或操作。

無狀態：RESTful API 是無狀態的，這表示每個請求都是獨立的，不依賴先前的請求狀態。這使得 API 服務的可擴展性和部署更加容易，而無需管理每個請求的狀態。



在LLM/RAG開發中，我們採用了OpenAI Chat Completions API風格來實現API接口。透過將所有LLM視為遵循這種風格的服務，我們可以將LLM部署為獨立伺服器，從而能夠與其他應用程式整合並在不同場景中使用。

## 代碼分層

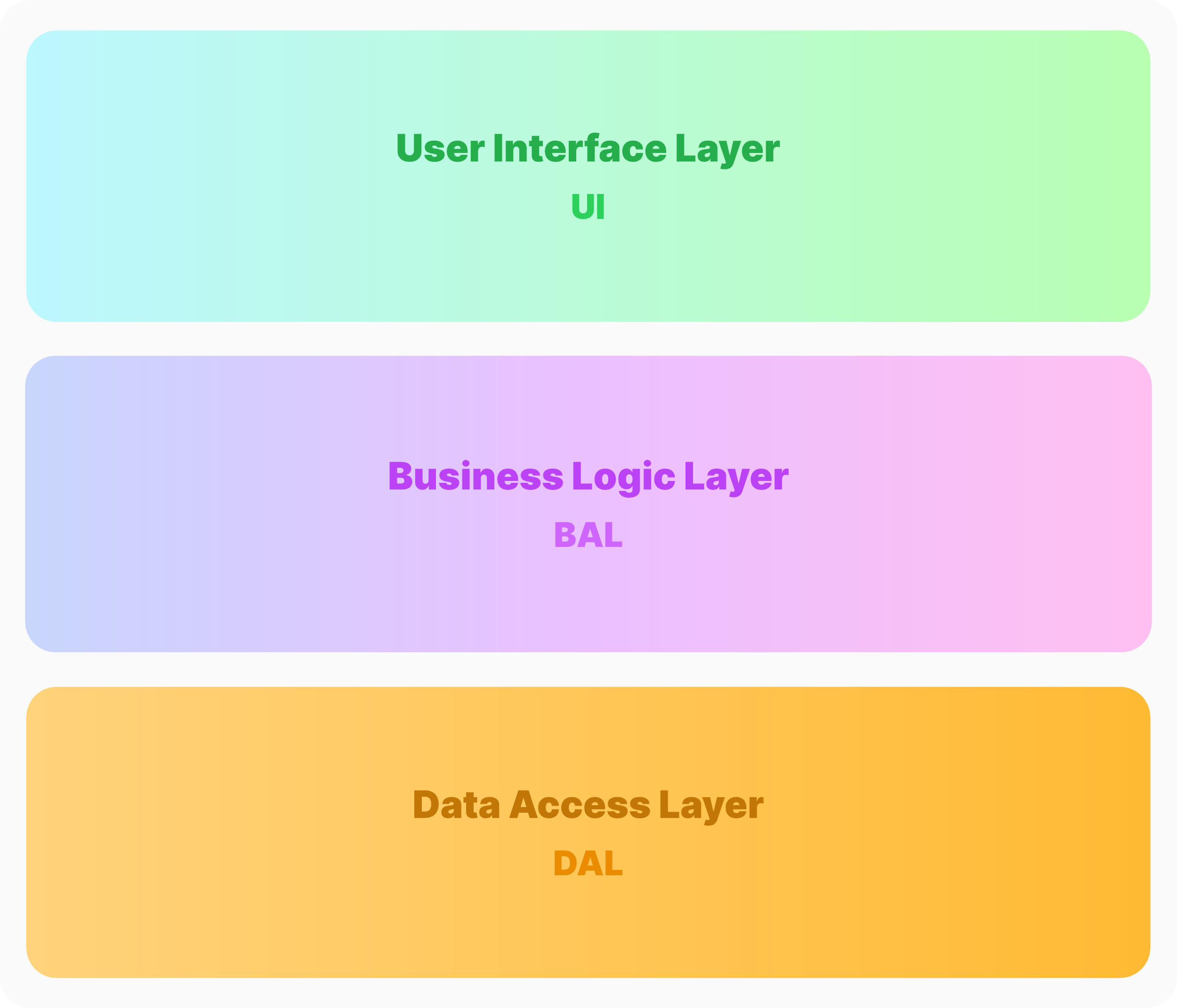
參考《軟體工程理論與實務應用》的系統模型章節 和 GoFrame框架的代碼分層設計，我們將會採用三層架構設計（Three-tier Architecture）。

它將系統分為三層：表示層、業務邏輯層 和 資料訪問層。

在表示層，重點是設計和實現使用者介面，旨在提供使用者友善的體驗。

在業務邏輯層，處理系統的邏輯功能和服務。此層負責處理和執行業務邏輯。

在資料訪問層，與資料庫進行直接通訊，管理進出系統的資料流。



這種架構有效地分離了系統的不同功能和職責，從而實現了可擴展性和穩定性。

MVC

## 代碼目錄

### 父系目錄 – 項目根目錄

將不同功能模塊分為frontend、backed和 \_document 等目錄，使代碼模塊化，統一每個項目的代碼目錄，可以方便長期維護。

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

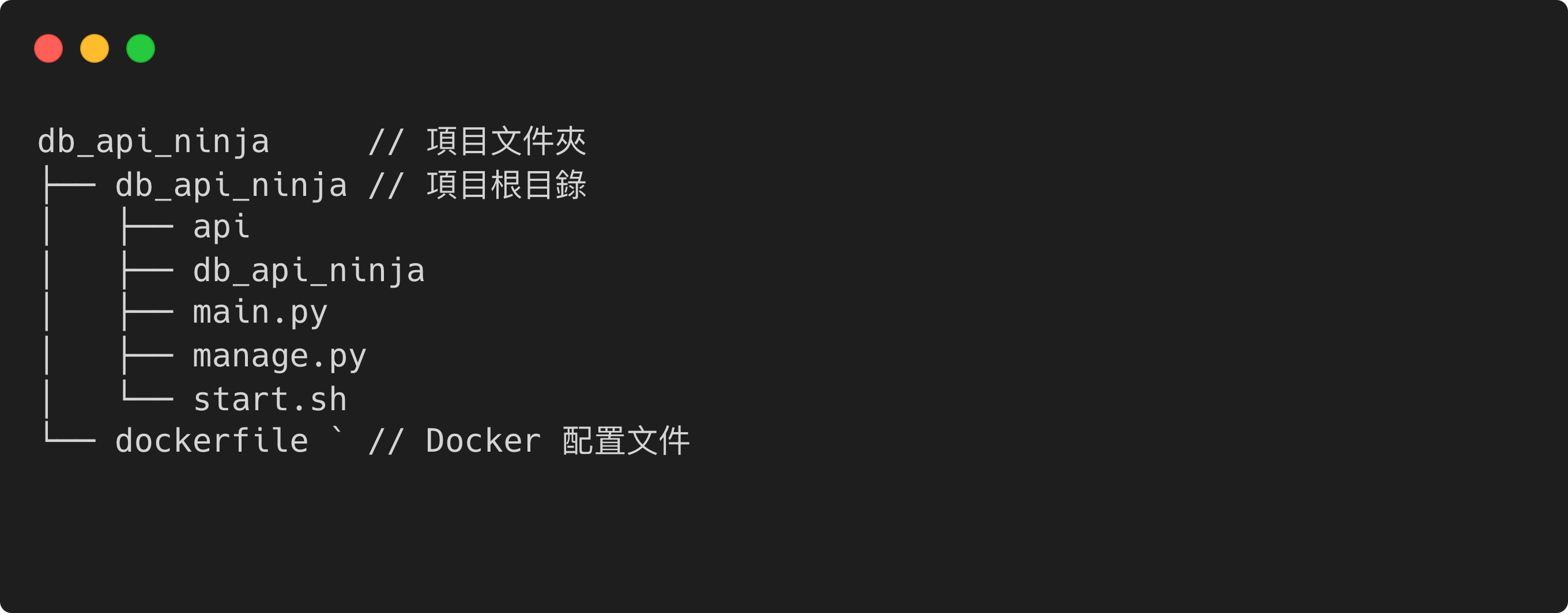
README.md 文件提供項目的概述。.gitignore 文件列出了需要被忽略的常見文件和目錄。frontend 目錄包含與用戶界面相關的代碼文件。backend 目錄包含實現業務邏輯的代碼文件，同時包含數據訪問層的代碼文件。

最重要的是，docker-compose.yml 文件則於配置Docker Compose相關的設置。有了它所有項目都可以一鍵運行。

### 子系目錄 - 模塊

子系目錄也就是我們的功能模塊，可以是一個API接口，一個用戶介面，也可以是一個完整的項目。模塊的結構與父系目錄大致相同，這可以方便遞歸項目或模塊。

例子： API接口

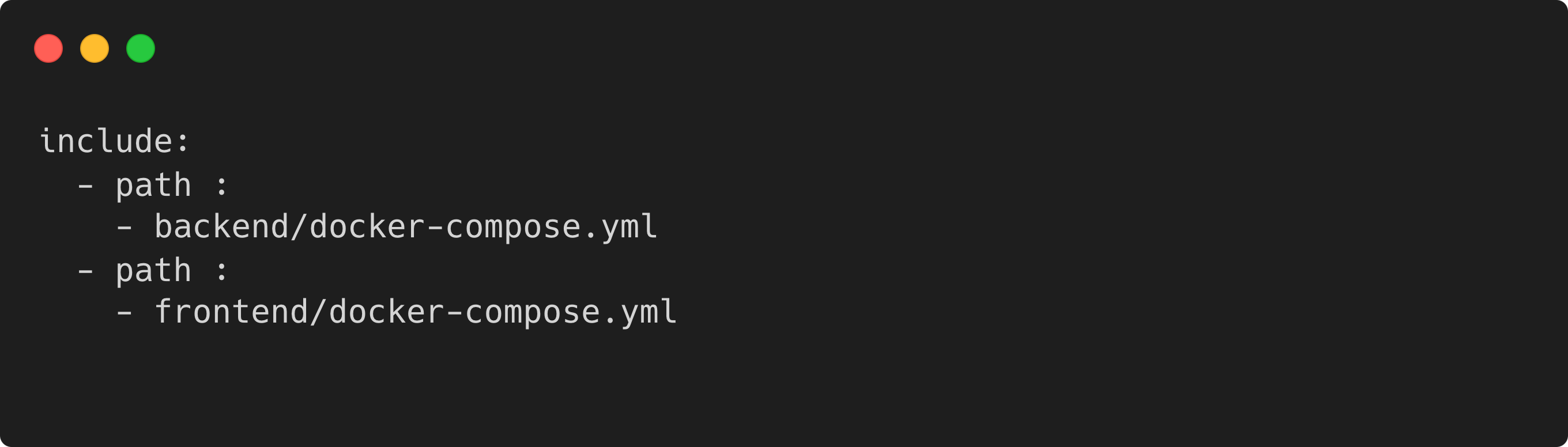


當然，子系目錄可以根據情況自行改變，但必須提供dockerfile 或 docker-compsoe.yml。

## Docker Compose

frontend 和 backend 目錄各自有一個docker-compose.yml文件，使兩者可以分開開發和部署。

當開發完成時，需要一鍵開啟整個項目，則使用項目根目綠的doker-compose.yml。



這個配置文件使用了 include 關鍵字，允許在一個主要的 Docker Compose 文件中引入其他的Docker Compose 文件。 過 include 指令，它們的內容將被合併到主要的 Docker Compose 檔案中，以便一起運行和管理容器。

這樣可以幫助將整個應用程式的不同組件（例如後端和前端）進行分離並管理，使得 Docker Compose 配置更加模組化和易於維護。

## 代碼版本

我們使用Git 進行版本控制去進行一個完整的項目工作流程：

在開始專案之前，應先閱讀專案文檔，以了解專案的目標、架構和現有功能。

然後，就是最重要的版本管理，我們的分支策略主要包括三個主要分支：

1. main：主要開發分支，包含項目的最新穩定版本。
2. stage：stage分支用作預生產環境，用於測試和質量保證（QA）活動。從main分支部署到此處以進行測試。
3. production：production分支代表生產環境。在stage 分支上進行充分測試後，將更改合併到 production 分支進行最終部署。

在開發新功能或修補問題時，建議從main分支創建新的分支。

這樣可以在不影響main分支的情況下獨立進行工作：

一張含有 文字, 螢幕擷取畫面, 字型, 多媒體軟體 的圖片

自動產生的描述

通過使用這種分支命名可以更容易地識別分支的目的和相關的貢獻者。

## 編程語言

我們選擇Python和 JavaScript （Nodejs）作為主要程式語言，並使用pyenv和nvm進行版本管理。 同時，如果需要性能優先的場景，可以使用golang作為副助程式語言。

Python需要Python 3.10.12或更高版本。

Node.js需要 Node.js 20.7.0 或更高版本。

Golang 沒有指定特定版本。

值得注意的是，除了使用 JavaScript，還會使用 TypeScript。

TypeScript和python都有 type hit 的功能，為了長期發展，增加可維護性，需要經常使此功能。

## 後端框架/工具推薦

後端推薦的編程語言是python，雖然python的性能不如其他語言，但是python對開發來說有很多好處。

1. 簡潔性：相對其他語言，python 代碼易於閱讀和理解，為長期維護提供便利。
2. 多樣性：python 擁有龐大而活躍的第三方庫和工具，節省了時間和精力，同時有更多的 選擇和靈活性。
3. 可靠性：Python 的第三方庫通常是經過廣泛測試和使用的。這些庫中的許多都是成熟的、穩定的解決方案，能夠滿足各種需求。

### 數據庫

數據庫主要使用PostgreSQL。PostgreSQL 是一個完全開源的關聯式數據庫管理系統 (RDBMS)，可以自由地下載、使用、修改和分發它，而且不需要支付任何費用，有效減少成本。同時，是目前廣泛使用的數據庫之一，意味著可以輕鬆地獲取支援、解決問題和分享資源。

還有，我們推薦使用 pgAdmin 作為管理和檢查數據的工具。pgAdmin 是一個開源的圖形用戶界面工具，專為 PostgreSQL 設計，提供了許多方便的功能和工具，可以輕鬆地管理和操作數據庫。

### 框架

有請GPT介紹幫手介紹：

這些都是不同的網絡開發框架結合不同的數據庫 ORM 工具。

#### FastAPI + SqlModel

FastAPI 是一個基於 Python 的現代、快速（高性能）的 Web 應用框架，並且具有自動化的 API 文檔生成功能。SqlModel 是一個基於 SQLAlchemy 的工具，它提供了一種定義數據模型的方式，並能夠自動生成對應的數據庫表結構。這個組合可以讓你使用 FastAPI 快速構建 Web API，同時使用 SqlModel 管理數據模型和數據庫。

#### Django + rest\_framework

Django 是一個強大的 Python Web 應用框架，它提供了完整的開發工具和功能，包括數據庫 ORM、用戶身份驗證、URL 路由、模板引擎等。Django REST framework 是 Django 的一個擴展，它提供了用於構建 Web API 的工具和功能，包括序列化、視圖、路由、身份驗證等。這個組合非常適合用於構建功能豐富的 Web API。

#### Gin + Gorm

Gin 是一個用於 Golang 的輕量級 Web 框架，它具有高性能和低內存占用的特點。Gorm 是 Golang 中常用的數據庫 ORM 工具，支持多種數據庫。這個組合可以讓你使用 Gin 快速構建 Web 應用，同時使用 Gorm 管理數據庫。

#### Goframe

Goframe 是一個基於 Golang 的 Web 應用框架，它具有高性能和豐富的功能。Goframe 內置了一個輕量級的 ORM 工具，可以方便地進行數據庫操作。這個組合讓你能夠使用 Goframe 快速構建 Web 應用，同時內置的 ORM 工具可以簡化數據庫操作。

#### Django + Django-ninja

這個組合同樣是使用 Django 框架，但使用的是 Django-ninja 擴展。Django-ninja 是一個用於構建快速、簡單和靈活的 Web API 的工具。它提供了一個簡潔的 API 語法，讓你能夠快速定義和處理 API 請求。這個組合適合那些希望使用 Django 并且更偏向於使用函數式 API 語法的開發者。

我們建議使用Django + Django-ninja 的框架組合作為後端的開發方案，因為 Django 自帶ORM功能，非常方便開發。 加上Django-ninja 大量參考FastAPI 的 設計，語法非常簡潔，能一句話完成就絕對不超過兩句。

兩個框架都擁有完善的身份認證功能，減少我不必要的壓力。真的好爽。

可能你會問，為何不使用更成熟的 Django + rest\_framework 呢？因為 Django-ninja 同 fastapi 一樣，內置swegger-ui，一會兒會詳細說明。

測試框架

單元測試是軟體開發中的一個重要實踐，可以確保代碼品質，提高代碼可維護性。同時可以促進團隊合作，代碼有一套完整的單元測試時，其他團隊成員可以更容易地理解和使用代碼。

我們建議使用Pytest用於編寫和運行單元測試。它具有簡單易用的語法和豐富的功能，讓你能夠輕鬆地組織、執行和報告測試結果。

### API規範

在之前的章節中，曾提到會使用RESTful API風格，但這還不夠細節。 為了擁有更高通用性的 API，我們會遵守 OpenAPI 規範。

OpenAPI（前身為Swagger）是一個用於描述和設計 RESTful API 的開放式規範。使用 JSON 或 YAML 格式的文件來定義 API 的結構、端點、參數、請求和響應等信息。

Swagger 同樣是一種用於描述和設計 RESTful API 的規範工具。最重要是， Swagger可以自動生成互動式的 API 文件和客戶端程式碼。 Swagger UI 是 Swagger 工具集的一個組件，它提供了一個互動式的、可視化的界面，用於渲染和展示 Swagger 描述文件生成的 API 文檔。

一張含有 文字, 螢幕擷取畫面, 數字, 字型 的圖片

自動產生的描述

一個openapi.json的樣子。

一張含有 文字, 螢幕擷取畫面, 軟體, 電腦 的圖片

自動產生的描述

有了他，前後端交流更暢通。在巿面上，有一些工具可以利用這個規範生成前端的接口，例如OpenAPI Typescript Codegen，減低前端寫請求文件的煩惱。（還在實驗中）

### 身份驗證

推薦兩種，一個是利用Django的身份認證，一個是使用Django-ninja- jwt 的身份認證。

## 前端框架/工具推薦

### Web Framework

vue svelte

nuxt

sveltekit

### Theme

tailwind css

daisy ui