首先是前端主機，例如我們中心的ETF訊息中心網站、市場訊息中心網站還有新版官網，另外集保ETF觀測站也會來我們的後端拿資料。

接下來是後端主機，後端主機會去跟資料庫、NAS或是排程主機拿資料，來去回應前端主機發送過來的請求。

後端的程式有分成系統組開發的以及英科公司開發的，系統組開發的Servlet程式這四個網站都有服務到，英科公司開發的Servlet就是服務新版官網，然後英科公司開發的Servlet與債券系統之間有TCP的溝通來達成債券殖利率百元價試算以及債券附條件試算。

----------------------------------------------------------------------------

DMZ: 非軍事區，用來保護內部網路組織抵禦外部威脅，位於公司內部網絡和 Internet 之間的小型子網，充當兩者之間的緩衝區。

隱藏後端服務：通過將 Tomcat 置於內部網絡，外部用戶無法直接訪問，減少了直接攻擊的可能性。

----------------------------------------------------------------------------

Nginx:

靜態HTTP伺服器，前端的的靜態資源(HTML/CSS/圖片)會放到Nginx裡面，當request path有對到對應的設定，就會到指定的位置取得靜態資源。

------------------------------------------------------------------------------

Tomcat: 一個開源的Web容器，他允許開發者部署和運行基於Java的Web應用程式。

Java Servlet: 用Java編寫的伺服器端程式，主要功能是接收client發送來的請求，生成需要的數據，並將其返回給客戶端。

API server: 提供應用程式介面(API)服務的伺服器，提供官網、ETF訊息中心網站、集保ETF觀測站API服務。

-------------------------------------------------------------------------------

NAS: 儲存資料供Tomcat伺服器讀取，例如: 盤後報表CSV、  
興櫃即時行情、債券即時行情

-------------------------------------------------------------------------------

排程主機:

-------------------------------------------------------------------------------

網站後台: 、 SQL server:

-------------------------------------------------------------------------------

一站式架構(舊架構):

**前後端不分離**：前端（如 HTML、CSS、JavaScript）和後端（如數據庫、業務邏輯）都由同一個項目或同一套服務器進行處理。前端的頁面通常由後端服務器渲染和生成。

技術實現: 舊架構使用 PHP後端語言來渲染頁面。

前端和後端程式碼共存於一個應用中，後端直接生成完整的 HTML 頁面並返回給客戶端。

-------------------------------------------------------------------------------

前後端分離(新架構):

**前後端分離**：前端和後端被分開成獨立的模塊或應用程序，前端通常是單頁應用（SPA）或多頁應用，通過 API（例如 REST API 或 GraphQL）與後端進行通信

-------------------------------------------------------------------------------

前後端分離優點:

-------------------------------------------------------------------------------

Nginx反向代理:

反向代理好處:

後續辦理事項:

工作職掌:

前後端不分離和前後端分離是兩種不同的網頁應用架構設計方式，它們在實現方式、技術選擇、維護性、性能和開發流程上有顯著差異。以下是它們的主要區別：

**1. 定義**

* **前後端不分離**：前端（如 HTML、CSS、JavaScript）和後端（如數據庫、業務邏輯）都由同一個項目或同一套服務器進行處理。前端的頁面通常由後端服務器渲染和生成。
* **前後端分離**：前端和後端被分開成獨立的模塊或應用程序，前端通常是單頁應用（SPA）或多頁應用，通過 API（例如 REST API 或 GraphQL）與後端進行通信。

**2. 技術實現**

* **前後端不分離**：
  + 典型的架構會使用像 PHP、JSP、ASP.NET 等後端技術來渲染頁面。
  + 前端和後端代碼共存於一個應用中，後端直接生成完整的 HTML 頁面並返回給客戶端。
* **前後端分離**：
  + 前端和後端通常使用不同的技術棧。前端可以使用框架如 React、Vue、Angular，後端可以使用 Node.js、Django、Spring 等。
  + 前端通過 HTTP 請求與後端進行數據交互，後端僅負責提供數據和業務邏輯，前端自己渲染 UI。

**3. 數據交換方式**

* **前後端不分離**：
  + 主要通過後端模板引擎將數據嵌入到 HTML 中，再返回給瀏覽器。
  + 數據和視圖緊密耦合。
* **前後端分離**：
  + 通常通過 API 以 JSON 或 XML 格式交換數據。
  + 視圖渲染和數據請求分開進行。

**4. 開發模式**

* **前後端不分離**：
  + 開發者往往需要同時處理前端和後端邏輯，開發者需要具備前後端的綜合技能。
  + 版本迭代時，修改界面設計或業務邏輯可能需要同時修改後端的代碼。
* **前後端分離**：
  + 前端和後端可以分別由不同的團隊負責，專業化程度更高。
  + 開發者可以專注於各自領域，前端開發者專注於用戶界面和體驗設計，後端開發者專注於數據和業務邏輯。

**5. 性能**

* **前後端不分離**：
  + 對於簡單的網站來說，不分離架構可以更容易實現且性能足夠，因為不需要頻繁的 API 調用。
  + 不過，當應用規模增大時，後端需要負責渲染和業務邏輯，可能會導致性能瓶頸。
* **前後端分離**：
  + 在大型應用中，前後端分離可以提高性能，前端可以實現更快的響應速度（因為 API 請求比完整頁面刷新更輕量）。
  + 前端可以利用瀏覽器的緩存和異步加載來提高用戶體驗。

**6. 維護性與可擴展性**

* **前後端不分離**：
  + 當前後端代碼高度耦合時，維護起來比較困難，任何改動都可能影響到整個應用的其他部分。
  + 隨著項目的複雜性增加，代碼會變得難以擴展和維護。
* **前後端分離**：
  + 前端和後端是獨立的模塊，維護起來更靈活，改動其中一端通常不會影響到另一端。
  + 更容易擴展，因為前後端可以獨立進行性能優化或技術更新。

**7. SEO（搜尋引擎優化）**

* **前後端不分離**：
  + 後端生成的靜態 HTML 更有利於 SEO，因為搜尋引擎可以輕鬆抓取和索引內容。
* **前後端分離**：
  + 單頁應用可能會遇到 SEO 的挑戰，因為內容是通過 JavaScript 動態渲染的，部分搜尋引擎可能無法有效索引這些內容。
  + 解決方案包括使用伺服器端渲染（SSR）或靜態網站生成器（如 Next.js）。

**總結**

* **前後端不分離**更適合小型或中型項目，特別是對開發人員較少、快速迭代的情況，實現起來更簡單。
* **前後端分離**則更適合大型或複雜的項目，開發過程更靈活且專業化，前後端團隊可以獨立發展，提高可擴展性和維護性。