資工系實務專題研究計畫成果報告 線上整合教學平台

專題編號：111-CSIE-S020
執行期限：110 年第 1 學期至 111 年第 1 學期
指導教授：郭忠義 教授
專題參與人員： 108360201 劉承翰
108590025 劉立傑
108590032 林奕杰

一、摘要

以分離式的網頁架構，開發出一個線上教學的平台，稱為 eLearning。

整個專案分為前端網頁、後端 API 伺服器、檔案伺服器與 Jitsi[1]線上會議伺服器，所有的系統皆以 Docker[2]容器化建置部屬，以 API 作為服務之間的橋樑。

關鍵詞：React、Golang、MariaDB、Docker、Restful、HLS

二、緣由與目的

自從新冠肺炎疫情爆發至今已超過兩年的時間，到目前為止，我們進行了多次的遠距教學。然而線上教學的平台五花八門，除了本校自有的北科 i 學園 Plus 外，常用的還有 Zuvio，Moodle 等學習平台，老師可能會需要透過 Teams 或 Google Meet 進行線上教學，同時可能以 Zuvio 等學習平台進行點名、隨堂問題，再回到北科 i 學園中分派作業、上傳教材，無疑會花費許多的時間，在使用及整合上產生了許多的不便，因此，我們的目標是希望可以打造一個整合所有教學需求的線上平台，藉由多元化的功能整合滿足所有教學的需求。

三、研究方法

(一) 研究範圍

主要功能參考範圍即為目前常見的線上學習平台及北科 i 學園 Plus。以整合教學的需求功能為主要方向，其次是前後端分離的現代 Web Dev。

(二) 使用技術方法

1 前端－React

使用 React 進行 SPA 開發，為提升程式可讀性、避免例外問題發生，使用 Typescript 語言編寫，同時利用 Prettier、eslint 等工具來使每位成員的 Coding Style 能維持一致，請求部分則使用 Axios 以 JWT 做為身分驗證方式進行處理，再引入 Redux-persist[3]讓資料可以持久化儲存。

因為本系統之教師端有許多文章編輯功能，如發布公告、編輯題目，為了提升其使用的方便度，我們採用 Draft.js[4]框架來實現富文本(RichText)編輯功能，並加入插入圖片、超連結等選項讓使用者可以編寫出更完整的文本。

最後再透過 Eletron[5]、React Native[6]的方式進行 PWA 及跨平台的實現，提升不同平台下的使用者體驗。

2 後端－Golang

Golang 是一款由 Google 開發的程式語言，其設計特性使它可以方便、快速的用於多核處理、網路應用的開發，因此選用此語言進行開發。

系統架構方面依 MVC 架構進行設計，網路通訊使用 GoFiber[7]套件實現，資料庫使用 MariaDB，並選用 Gorm[8]套件自動建置、更新 Schema、實現物件關聯對映(ORM)。

3 開發流程

DevOps 部分使用 GitLab CI/CD 與 Docker 做結合的解決方案，達到容器化的

自動部署的目的，通過 GitLab CI 設置推送觸發建置、測試到部屬的 Pipeline 的流程，實現了持續集成與把關程式碼品質的目的。

因後端部分採用 RestfulAPI 做為接口，在大量 API 與前端交互的情況下，我們使用了 Swagger[9]製作規格文件，避免前後端溝通上出現問題，同時也可以直接對單一 API 進行測試。

(三) 架構流程

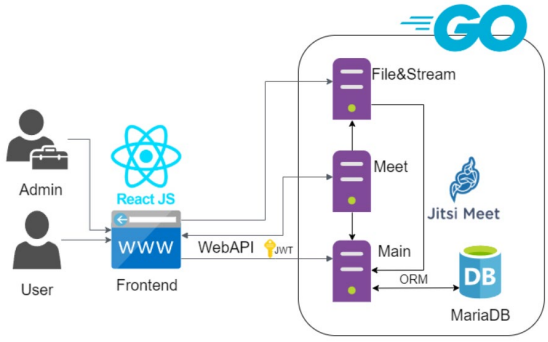

圖 1 系統架構圖

本專題分為前端使用者介面、後端伺服器，檔案伺服器與會議伺服器。

前端 Web 框架作為操作系統的使用者介面，對網頁元件操作，使用 API 與後端伺服器溝通並將資料整理成使用者方便觀看之方式顯示。

後端伺服器作為整個 elearning 的核心，負責所有課程相關資料的處理，並通過 API 的形式會傳所需要的資料。

檔案伺服器主要作為課程教材與錄影的儲存以及影片串流使用若是有影片資料上傳，檔案伺服器會自動產生 HLS 媒體流，達到影音線上串流的功能。

elearning 平台線上會議採用 Jitsi 作為解決方案，線上會議具備語音、視訊、畫面分享、文字聊天與錄影等功能，並在 Jitsi 設定了會議相關事件的觸發器，包括了創建、刪除、加入、離開等，以及自動開始錄影等功能。

四、研究結果

本專題通過整合線上學習所需要的功能開發出一個可跨平台的網路 APP，教師端可建立題庫、發布點名、公告、管理學生、發布隨堂問答、發布測驗並進行成績考核，同時可以上傳教材、進行線上教學；學生端則可以進行與教師端功能相對應之動作。

五、結論

本專題已建構出基礎的功能及介面，並提供容易擴充的架構與部屬方式，將這些各個服務拆分獨立運行的容器，並且彼此透過 API 連接資料，不僅增加開發及除錯的靈活度，也讓各服務的隔離性增加，達到提升內聚力減少耦合度的效果，對於可用性以及延伸度大幅增加，更加優化後續維護及延續開發的模式。

四、參考文獻

[1] Jitsi meet
https://jitsi.github.io/handbook/docs/intro
[2] Docker
https://docs.docker.com/
[3] React redux
https://react-redux.js.org/
[4] Draft.js
https://draftjs.org/
[5] Electron
https://www.electronjs.org/docs/latest
[6] React Native
https://reactnative.dev/docs/components-and-apis
[6] React Native
https://reactnative.dev/docs/components-and-apis
[8] Gorm
https://gorm.io/docs/index.html
[9] go-swagger
https://goswagger.io