國立臺北商業大學

資 訊 管 理 系

108資訊系統專案設計

**系統手冊**

****

**組 別： 第108408組**

**題 目： 省錢生活**

**指導老師： 唐日新老師**

**組 長： 10546045 陳冠都**

**組 員： 10546018 黃政源 10546026 楊秉憲**

**10546027 張詠俊**

**中華民國　108　年　11　月　20　日**

**目錄**

[**第一章 : 背景與動機** 8](#_Toc25074520)

[**1-1 個案背景** 8](#_Toc25074521)

[**1-2 問題與機會** 8](#_Toc25074522)

[**1-3 相關系統探討** 9](#_Toc25074523)

[**第二章 : 系統目標與預期成果** 10](#_Toc25074524)

[**2-1 系統目標** 10](#_Toc25074525)

[**2-2 預期成果** 10](#_Toc25074526)

[**第三章：系統規格** 11](#_Toc25074527)

[**3-1 系統架構** 11](#_Toc25074528)

[**3-2 系統軟、硬體需求與技術平台** 12](#_Toc25074529)

[**3-3 開發標準與使用工具** 12](#_Toc25074530)

[**3-4 使用標準與工具** 12](#_Toc25074531)

[**第四章：專案時程與組織分工** 13](#_Toc25074532)

[**4-1 專案時程** 13](#_Toc25074533)

[**4-2 專案組織與分工** 13](#_Toc25074534)

[**5-1 使用者需求:** 14](#_Toc25074535)

[**5-2 使用案例圖:** 14](#_Toc25074536)

[**5-3 使用個案描述:** 15](#_Toc25074537)

[**5-4 分析類別圖** 17](#_Toc25074538)

[**第六章：設計模型** 18](#_Toc25074539)

[**6-1 循序圖** 18](#_Toc25074540)

[**6-2 設計類別圖** 19](#_Toc25074541)

[**第七章：實作模型** 20](#_Toc25074542)

[**7-1 佈署圖** 20](#_Toc25074543)

[**7-2 套件圖** 20](#_Toc25074544)

[**7-3 元件圖** 21](#_Toc25074545)

[**7-4 狀態機** 21](#_Toc25074546)

[**第八章：資料庫設計** 22](#_Toc25074547)

[**8-1 資料庫關聯表** 22](#_Toc25074548)

[**8-2 表格及Meta data** 23](#_Toc25074549)

[**第九章：程式** 27](#_Toc25074550)

[**9-1 元件清單及其規格描述** 27](#_Toc25074551)

[**第十章：測試模型** 92](#_Toc25074552)

[**10-1 測試計畫** 92](#_Toc25074553)

[**10-2 測試個案與測試結果資料** 93](#_Toc25074554)

[**第十一章：操作手冊** 104](#_Toc25074555)

[**11-1 系統文件** 104](#_Toc25074556)

[**11-2 系統管理** 104](#_Toc25074557)

[**第十二章：使用手冊** 105](#_Toc25074558)

[**第十三章：感想** 107](#_Toc25074559)

[**第十四章：參考資料** 109](#_Toc25074560)

[**14-1 參考元件** 109](#_Toc25074561)

[**14-2 參考資料** 111](#_Toc25074562)

[附錄 112](#_Toc25074563)

[**附錄一 評審建議事項之修正情形** 112](#_Toc25074564)

[**附錄二 會議記錄** 113](#_Toc25074565)

**表目錄**

[表1-2-1 SWOT分析 8](#_Toc25073446)

[表1-3-1 9](#_Toc25073447)

[表3-2-1 系統軟、硬體需求與技術平台 12](#_Toc25073448)

[表3-3-1 開發標準與使用工具 12](#_Toc25073449)

[表3-4-1 使用標準與工具 12](#_Toc25073450)

[表4-2-1 專業分工 13](#_Toc25073451)

[表5-1-1 使用者需求 14](#_Toc25073452)

[表8-2-1 優惠券資料表 23](#_Toc25073453)

[表8-2-2 記帳資料表 23](#_Toc25073454)

[表8-2-3 使用者資料表 24](#_Toc25073455)

[表8-2-4 電子郵件認證資料表 24](#_Toc25073456)

[表8-2-5 token資料表 25](#_Toc25073457)

[表8-2-6 網域資料表 25](#_Toc25073458)

[表8-2-7 記帳資料表(前端) 25](#_Toc25073459)

[表8-2-8 預算資料表 26](#_Toc25073460)

[表9-1-1 Python元件清單 27](#_Toc25073461)

[表9-1-2 Swift元件清單 28](#_Toc25073462)

[表9-1-3 後端程式setting 29](#_Toc25073463)

[表9-1-4 後端程式permission 33](#_Toc25073464)

[表9-1-5 後端程式url,py 33](#_Toc25073465)

[表9-1-6 後端程式view.py 35](#_Toc25073466)

[表9-1-7 後端程式test.py 44](#_Toc25073467)

[表9-1-8 後端程式serializers.py 45](#_Toc25073468)

[表9-1-9 後端程式models.py 46](#_Toc25073469)

[表9-1-10 後端程式filter.py 47](#_Toc25073470)

[表9-1-11 後端程式insert\_coupon\_to\_mysql 48](#_Toc25073471)

[表9-1-12 後端程式KFC\_crawler 52](#_Toc25073472)

[表9-1-13 後端程式-圖片辨識 53](#_Toc25073473)

[表9-1-14 後端程式Log 54](#_Toc25073474)

[表9-1-15 後端程式Excel\_To\_SQL 55](#_Toc25073475)

[表9-1-16 後端程式insert\_mysql.py 57](#_Toc25073476)

[表9-1-17 前端程式showWalkthrough 58](#_Toc25073477)

[表9-1-18 前端程式connectDataBase 58](#_Toc25073478)

[表9-1-19 前端程式pageViewController 59](#_Toc25073479)

[表9-1-20 前端程式check 59](#_Toc25073480)

[表9-1-21 前端程式instantiateViewController 60](#_Toc25073481)

[表9-1-22 前端程式pageViewController 60](#_Toc25073482)

[表9-1-23 前端程式couponPage 60](#_Toc25073483)

[表9-1-24 前端程式createModels 61](#_Toc25073484)

[表9-1-25 前端程式createCustomViewsLayer 62](#_Toc25073485)

[表9-1-26 前端程式createPlainTextLayer 62](#_Toc25073486)

[表9-1-27 前端程式createTextWithLinesLayer 62](#_Toc25073487)

[表9-1-28 前端程式createViewGenerator 63](#_Toc25073488)

[表9-1-29 前端程式showAlertMessage 64](#_Toc25073489)

[表9-1-30 前端程式BarsChartConfig 64](#_Toc25073490)

[表9-1-31 前端程式buttonView 64](#_Toc25073491)

[表9-1-32 前端程式submit 65](#_Toc25073492)

[表9-1-33 前端程式set max lengh to text 67](#_Toc25073493)

[表9-1-34 前端程式tableViewCell 67](#_Toc25073494)

[表9-1-35 前端程式CenterLeftCell 67](#_Toc25073495)

[表9-1-36 前端程式textFieldAction 68](#_Toc25073496)

[表9-1-37 前端程式cancelPiechart 70](#_Toc25073497)

[表9-1-38 前端程式usersTable.create 70](#_Toc25073498)

[表9-1-39 前端程式http 72](#_Toc25073499)

[表9-1-40 前端程式list 73](#_Toc25073500)

[表9-1-41 前端程式download 73](#_Toc25073501)

[表9-1-42 前端程式dailyNotification 75](#_Toc25073502)

[表9-1-43 前端程式Post 76](#_Toc25073503)

[表9-1-44 前端程式logOut 77](#_Toc25073504)

[表9-1-45 前端程式screenEdgeSwiped 77](#_Toc25073505)

[表9-1-46 前端程式pickerView 78](#_Toc25073506)

[表9-1-47 前端程式getDateValue 78](#_Toc25073507)

[表9-1-48 前端程式numbers 79](#_Toc25073508)

[表9-1-49 前端程式initiate 80](#_Toc25073509)

[表9-1-50 前端程式createTable 81](#_Toc25073510)

[表9-1-51 前端程式usersTableAP 81](#_Toc25073511)

[表9-1-52 前端程式dailyDetailFormId 82](#_Toc25073512)

[表9-1-53 前端程式insert 82](#_Toc25073513)

[表9-1-54 前端程式tableViewCELL 84](#_Toc25073514)

[表9-1-55 前端程式createPlainTextLayer 84](#_Toc25073515)

[表9-1-56 前端程式DailyDetailForm 85](#_Toc25073516)

[表9-1-57 前端程式get 86](#_Toc25073517)

[表9-1-58 前端程式searchBoutton 87](#_Toc25073518)

[表9-1-59 前端程式tableViewCEllCoupon 88](#_Toc25073519)

[表9-1-60 前端程式CouponArrayFormat 89](#_Toc25073520)

[表9-1-61 前端程式CouponCellFormat 89](#_Toc25073521)

[表9-1-62 前端程式WebView 90](#_Toc25073522)

[表9-1-63 前端程式UIButton 91](#_Toc25073523)

[表10-1-1功能列表 92](#_Toc25073524)

[表10-2-1 教學 93](#_Toc25073525)

[表10-2-2 註冊 94](#_Toc25073526)

[表10-2-3 email 95](#_Toc25073527)

[表10-2-4 登入 96](#_Toc25073528)

[表10-2-5設定目標金額 97](#_Toc25073529)

[表10-2-6提醒鍵 98](#_Toc25073530)

[表10-2-7 記帳 99](#_Toc25073531)

[表10-2-8 Coupon 100](#_Toc25073532)

[表10-2-9 上傳 101](#_Toc25073533)

[表10-2-10 下載 102](#_Toc25073534)

[表10-2-11 查詢 103](#_Toc25073535)

[附錄表:評審建議及修正情況 112](#_Toc25073536)

**圖目錄**

[圖1-2-1 消費者搜尋折價券的網站類型長條圖 9](#_Toc25073583)

[圖3-1-1 系統架構 11](#_Toc25073584)

[圖3-1-2 系統主要流程 11](#_Toc25073585)

[圖4-1-1 甘特圖 13](#_Toc25073586)

[圖8-1-1關聯圖 22](#_Toc25073587)

**第一章 : 背景與動機**

**1-1 個案背景**

依據中華民國統計資訊網，近十年物價指數上升約百分之十，2018年全年平均實質月薪僅3萬8235元，低於2001年的3萬8398元，薪資倒退回17年前的水準，物價卻不斷的成長，加上近年來快節奏的趨勢，所以，而現代人較不常關注實體優惠卷及虛擬優惠卷的省錢方法，以及虛擬優惠卷並沒有有效統整，實體優惠卷則常常被使用者遺忘並過期，即使使用優惠卷省下的錢也會被使用者忽略，使這筆錢無法被適當的應用

基於以上潛在需求，我們決定製作一個APP同時結合了記帳以及優惠卷功能，透過記帳的時候選擇優惠券計算出省下來的金錢，去規劃使用者的目標，我們除了在網路上蒐集既有的優惠券，還與特定商家合作提供獨有的優惠券，在必要的花費裡找出能花到最少的金額達到最好的效果，至於具體細節以及呈現方式，會在下面幾章再多加說明。

**1-2 問題與機會**

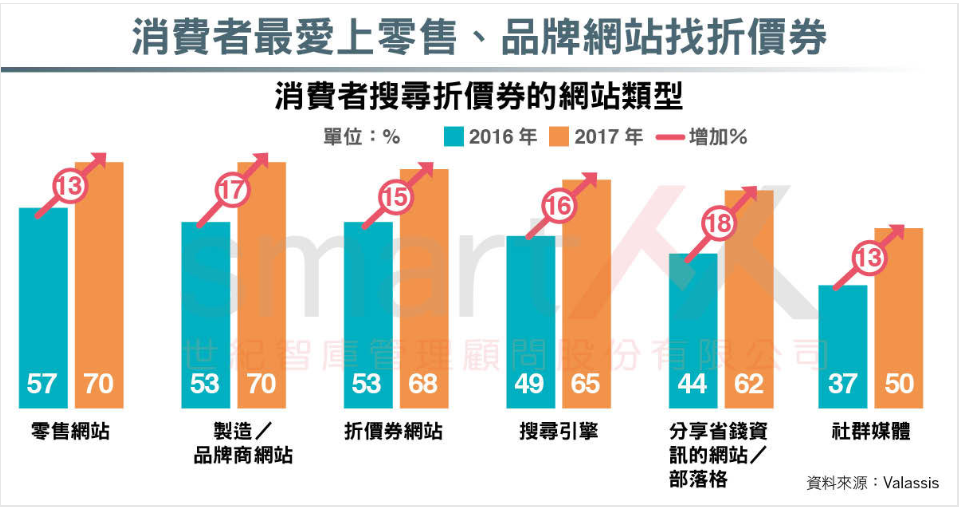
為了瞭解本系統的市場定位及發展策略，我們對本系統進行SWOT分析，如下表：

表1-2-1 SWOT分析

|  |  |
| --- | --- |
| S優勢 | W劣勢 |
| 1. 能在APP上同時達到記帳與使用優惠卷的功能 2. 獨家優惠券 | 1. 無法確認使用者是否有遺漏的消費沒有紀錄 2. 經濟資源不足無法談到很好的優惠券 |
| O機會 | T威脅 |
| 1. 在物價高漲的社會，會有許多人想要省錢 2. 電子優惠卷盛行 | 1. 無法蒐集到競爭者已與商家協議取得之獨家優惠券。 |

根據SmartM解讀，84%的消費者，會受折價券影響而選擇商店，而在智慧型手機普及下，則有35%的消費者，比前一年增加在網路上尋折價券的次數。

圖1-2-1 消費者搜尋折價券的網站類型長條圖



**1-3 相關系統探討**

表1-3-1

|  |  |  |  |
| --- | --- | --- | --- |
|  | 省錢生活 | Ahorro | 行動優惠券 |
| 記帳 | ⚫ | ⚫ |  |
| 預算規劃 | ⚫ | ⚫ |  |
| 收支控管 | ⚫ | ⚫ |  |
| 圖表分析 | ⚫ | ⚫ |  |
| 統整優惠券 | ⚫ |  | ⚫ |

**第二章 : 系統目標與預期成果**

**2-1 系統目標**

在這個優惠卷盛行的時代，我們希望可以讓更多的人群意識到這些優惠卷的好處，希望透過記帳跟優惠券的結合，讓記帳不僅只有紀錄花費而已，而是讓使用者也可以透過使用優惠券省錢，並且將省下來的這筆金額去達成預設目標。

1. 記帳:能讓使用者清楚的知道金錢收支。
2. 優惠卷:提供分析過後的優惠券讓使用者選擇適合自己的優惠卷。
3. 預設目標:使用者可以訂每個月花費的金額，如果超出的話系統會提醒使用者。

**2-2 預期成果**

記帳-

質化效益:

使用者可以有效控管可分配盈餘

訂出預算能有效審視自己是否達到預期目標

分析支出項目可以有效降低不必要的花費

量化效益:

之前主計處發布2017年數據，**全台灣家庭平均「年可支配所得101萬」平均「年消費支出是81萬」、平均「年儲蓄20萬」**，換句話說一般家庭一年最多只能拿20萬的資金出來投資。但如果能透過消費支出規劃以及把握各種省錢技巧，**一個家庭只要能從81萬中多省5%，每年就有機會多存下4萬元，相當於每年多了20%的投資資金。**

優惠卷-

質化效益:

有效滿足自己的購物慾望且CP值相對較高

量化效益:

節省一成以上花費

**第三章：系統規格**

**3-1 系統架構**

圖3-1-1 系統架構

圖3-1-2 系統主要流程

儲存

**3-2 系統軟、硬體需求與技術平台**

表3-2-1 系統軟、硬體需求與技術平台

|  |  |
| --- | --- |
| 用途 | 工具 |
| 作業系統 | IphoneX iOS 10 |
| 記憶體 | 3GB |
| 螢幕解析度 | 2436 x 1125 |
| 定位系統 | GPS |

**3-3 開發標準與使用工具**

表3-3-1 開發標準與使用工具

|  |  |  |
| --- | --- | --- |
| 用途 | | 工具 |
| 作業系統環境 | | OS、Windows 10、 |
| 資料庫 | | MySQL 8.0 |
| 撰寫工具 | | Xcode-Swift |
| 撰寫程式 | APP端 | Python、django 2.2 |
| 網頁端 | django2.2 |
| ios版本 | | iOS 10以上版本 |
| 繪圖設計軟體 | | figma |

**3-4 使用標準與工具**

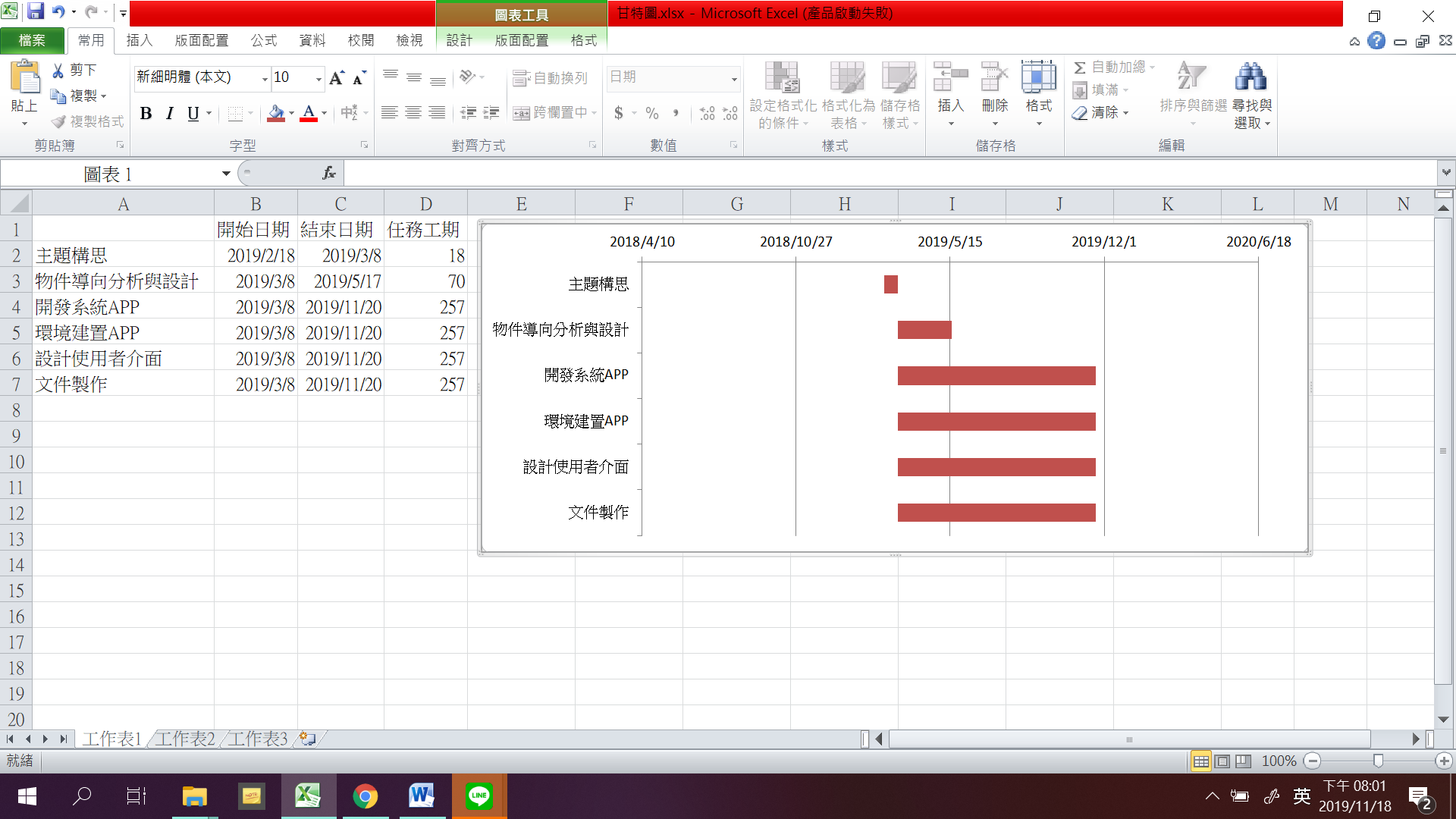
表3-4-1 使用標準與工具

|  |  |
| --- | --- |
| 用途 | 工具 |
| 系統分析與設計 | Visual Paradigm for UML 7.2 Personal Edition |
| 專案進度管理 | Microsoft Excel 2010 |
| 文件設計 | Microsoft Word 2010 |

**第四章：專案時程與組織分工**

**4-1 專案時程**

圖4-1-1 甘特圖



**4-2 專案組織與分工**

表4-2-1 專業分工

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 組員  項目 | | 10546045 | 10546018 | 10546026 | 10546027 |
| 陳冠都 | 黃政源 | 楊秉憲 | 張詠俊 |
| 規畫 | 決定主題 | ● | ● | ● | ● |
| 需求分析 | ● | ● | ● | ● |
| 資料蒐集 | ● | ● | ● | ● |
| 美工 | UI介面設計 | ○ | ○ | ● |  |
| LOGO設計 |  |  | ● |  |
| 海報設計 | ● | ○ | ● |  |
| 資料庫 | 建置伺服器 |  |  |  | ● |
| 建置資料表 |  |  |  | ● |
| 分析、儲存 |  | ○ |  | ● |
| 過濾 |  |  |  | ● |
| 資料清理 | ○ |  |  |  |
| 資料轉檔 | ● |  |  | ● |
| 程式設計 | 撰寫 | ○ | ● | ● | ● |
| 測試 | ● | ○ | ● | ● |
| 除錯 | ○ | ○ | ● | ● |
| 文件 | UML繪製 | ● | ● |  |  |
| 統整 | ● | ● | ● | ● |
| 簡報製作 | ● | ● | ● | ● |
| 文案撰寫 | ● | ● | ● | ● |

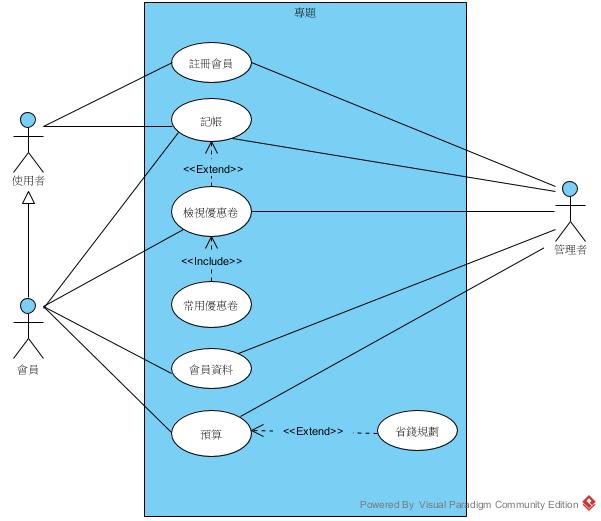
**第五章：需求模型**

**5-1 使用者需求:**

表5-1-1 使用者需求

|  |  |
| --- | --- |
| 功能需求 | 非功能需求 |
| 1. 記帳 2. 優惠券 3. 數據分析 | 1. 回應時間 ≦ 1秒 2. 手機作業系統需為ios 12.1以上版本 |

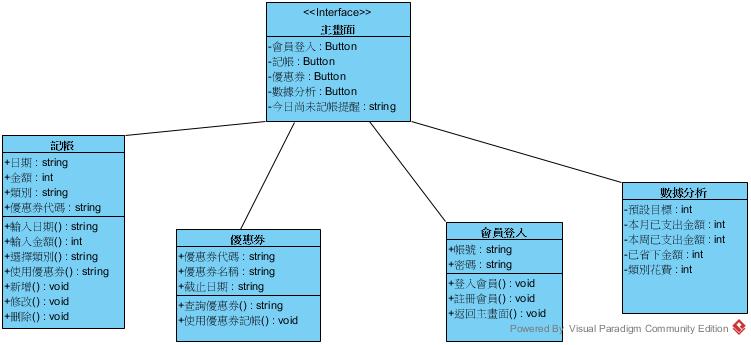
**5-2 使用案例圖:**

****

搜尋

**5-3 使用個案描述:**

|  |
| --- |
| 活動圖:記帳 |
|  |
| 活動圖:優惠券 |
|  |
| 活動圖:行為分析-使用者 |
|  |
| 活動圖:行為分析-管理員 |
|  |

**5-4 分析類別圖**

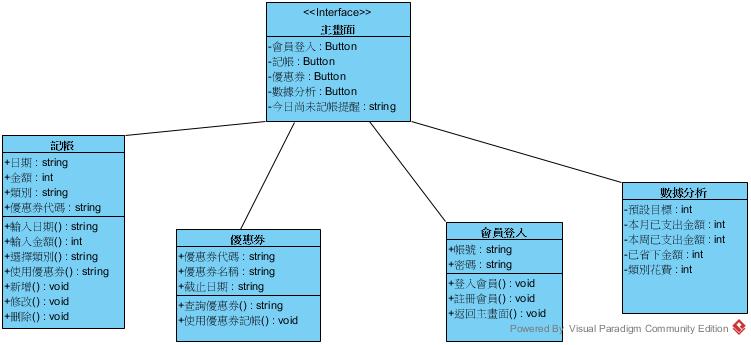
**第六章：設計模型**

**6-1 循序圖**

|  |
| --- |
| 循序圖:加入會員 |
|  |
| 循序圖:記帳 |
|  |

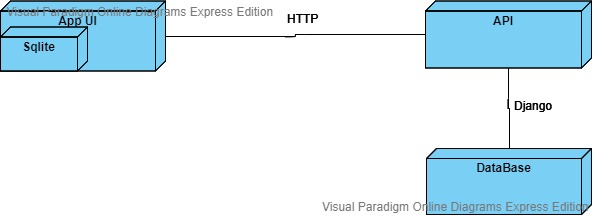
|  |
| --- |
| 循序圖:優惠券 |
|  |

**6-2 設計類別圖**

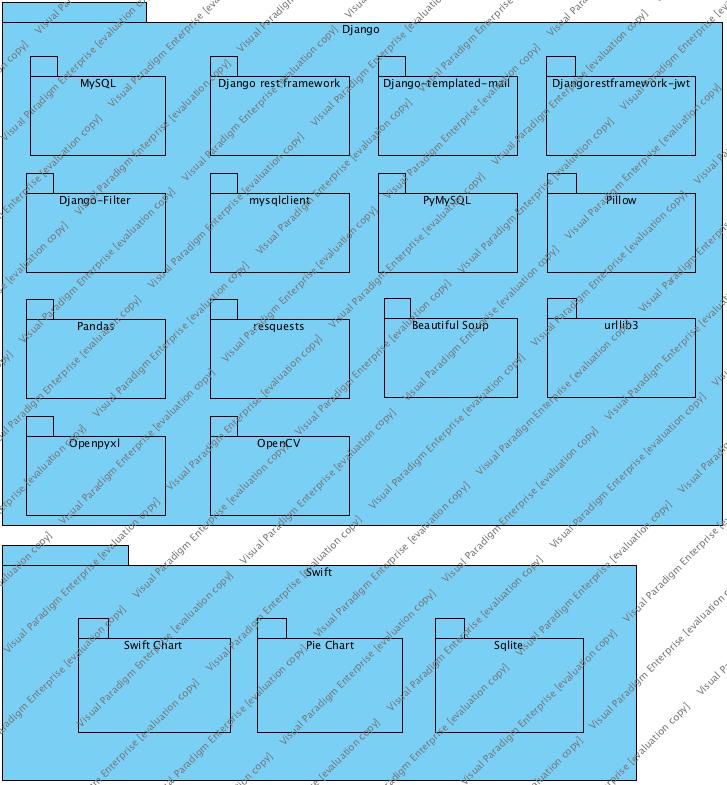


**第七章：實作模型**

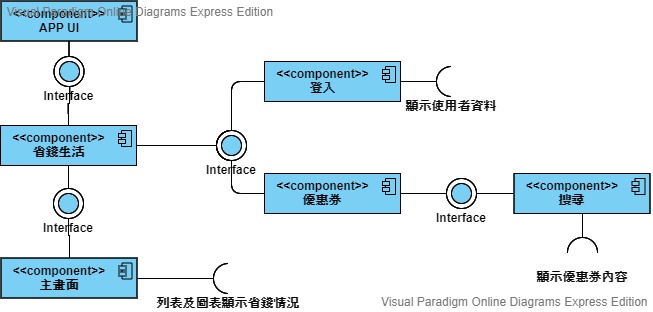
**7-1 佈署圖**

****

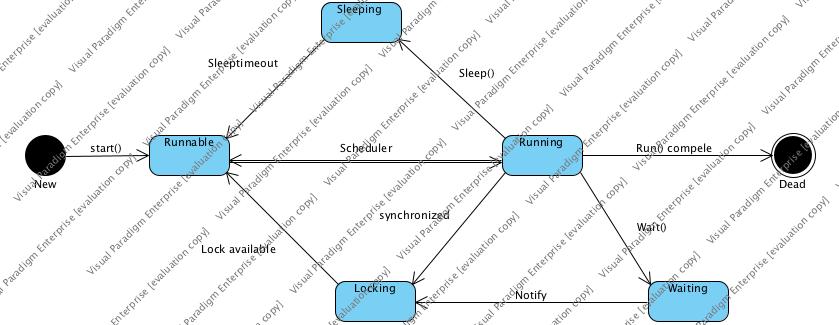
**7-2 套件圖**

****

**7-3 元件圖**

****

**7-4 狀態機**



**第八章：資料庫設計**

**8-1 資料庫關聯表**

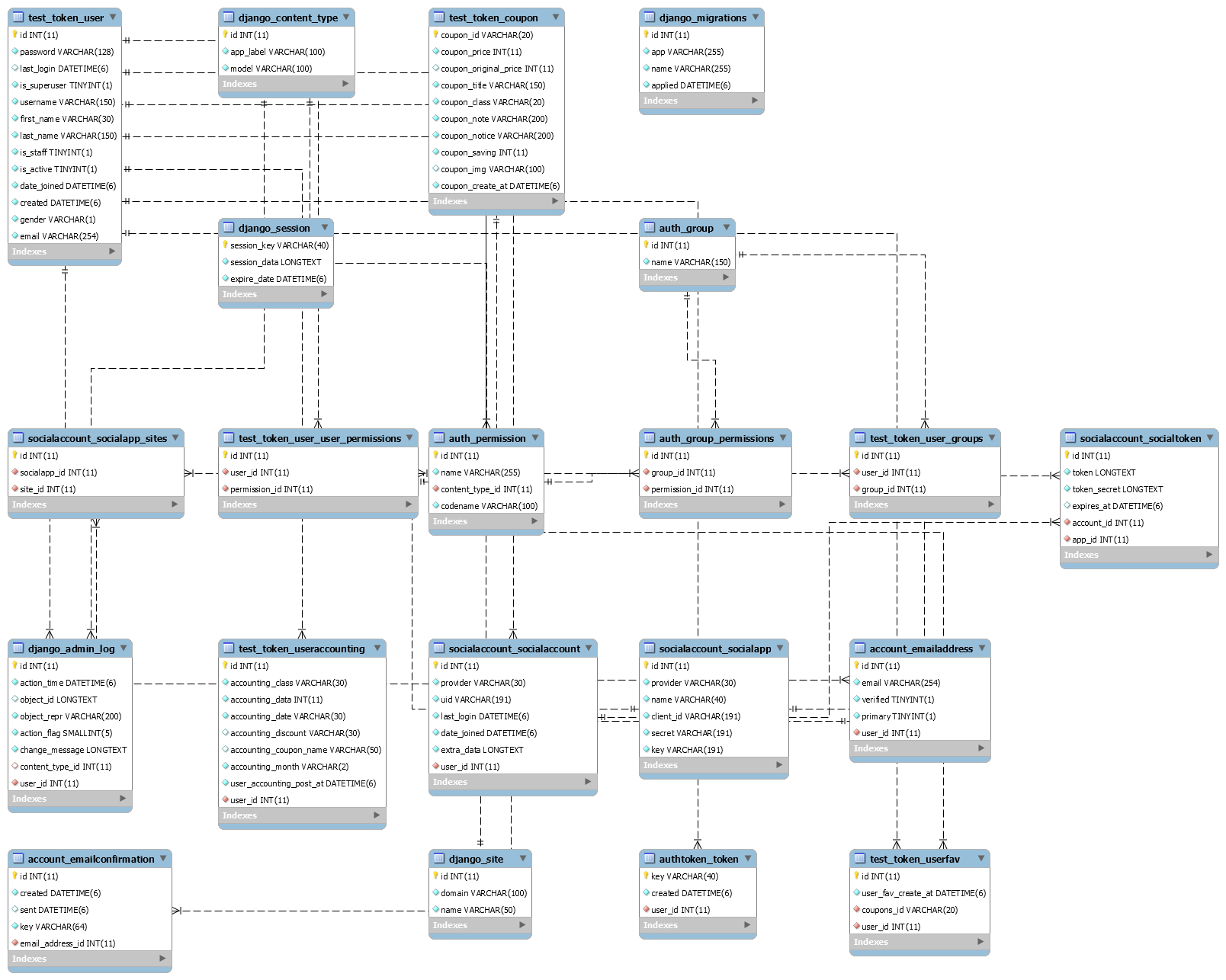
****

圖8-1-1關聯圖

**8-2 表格及Meta data**

表8-2-1 優惠券資料表

|  |  |  |  |
| --- | --- | --- | --- |
| test\_token\_coupon | | | |
| 欄位名稱 | 欄位型態 | 欄位大小 | 用途 |
| coupon\_id | varchar | 255 | PK |
| coupon\_price | int | 11 | 優惠卷優惠價格 |
| coupon\_original\_price | int | 11 | 優惠卷原本價格 |
| coupon\_title | varchar | 255 | 優惠卷顯示標題 |
| coupon\_class | varchar | 255 | 優惠卷類別 |
| coupon\_note | varchar | 255 | 優惠卷內文 |
| coupon\_notice | varchar | 255 | 優惠卷注意事項 |
| coupon\_saving | int | 11 | 優惠卷節省價錢 |
| coupon\_img | varchar | 255 | 優惠卷圖片存檔路徑 |
| coupon\_create\_at | datetime | 6 | 建立優惠卷時間 |

表8-2-2 記帳資料表

|  |  |  |  |
| --- | --- | --- | --- |
| test\_token\_accounting | | | |
| 欄位名稱 | 欄位型態 | 欄位大小 | 用途 |
| id | int | 11 | PK , 流水號 |
| user\_id | int | 11 | FK , 使用者PK |
| accounting\_class | varchar | 30 | 記帳類別 |
| accounting\_coupon\_name | varchar | 50 | 記帳優惠卷名稱 |
| accounting\_data | int | 11 | 記帳資料 |
| accounting\_date | varchar | 30 | 記帳時間 |
| accounting\_discount | varchar | 30 | 記帳折扣 |
| accounting\_month | varchar | 2 | 記帳月份 |
| accounting\_post\_at | datetime | 6 | 記帳上傳時間 |

表8-2-3 使用者資料表

|  |  |  |  |
| --- | --- | --- | --- |
| test\_token\_user | | | |
| 欄位名稱 | 欄位型態 | 欄位大小 | 用途 |
| id | int | 11 | PK , 流水號 |
| gender | varchar | 1 | 性別:M為男性, F為女性 |
| first\_name | varchar | 30 | 姓名 |
| email | varchar | 255 | 電子郵件 |
| date\_joined | datetime | 6 | 加入日期 |
| created | datetime | 6 | 創建日期 |
| is\_active | tinyint | 1 | 是否激活 |
| is\_staff | tinyint | 1 | 是否可以進入後台 |
| is\_superuser | tinyint | 1 | 是否是超級使用者 |
| last\_login | datetime | 6 | 上次登入時間 |
| last\_name | varchar | 150 | 姓名 |
| password | varchar | 128 | 密碼 |
| username | varchar | 150 | 帳號 |

表8-2-4 電子郵件認證資料表

|  |  |  |  |
| --- | --- | --- | --- |
| account\_emailaddress | | | |
| 欄位名稱 | 欄位型態 | 欄位大小 | 用途 |
| id | int | 11 | PK , 流水號 |
| email | varchar | 255 | 電子郵件地址 |
| verified | tinyint | 1 | 是否驗證 |
| primary | tinyint | 1 | 是否唯一 |
| user\_id | int | 11 | 使用者編號 |

表8-2-5 token資料表

|  |  |  |  |
| --- | --- | --- | --- |
| django\_session | | | |
| 欄位名稱 | 欄位型態 | 欄位大小 | 用途 |
| session\_key | varchar | 255 | token key |
| session\_data | longtext | 255 | token |
| expire\_date | datetime | 6 | token到期日 |

表8-2-6 網域資料表

|  |  |  |  |
| --- | --- | --- | --- |
| django\_site | | | |
| 欄位名稱 | 欄位型態 | 欄位大小 | 用途 |
| id | int | 11 | PK , 流水號 |
| domain | varchar | 255 | 網域 |
| name | varchar | 255 | 網址名稱 |

表8-2-7 記帳資料表(前端)

|  |  |  |  |
| --- | --- | --- | --- |
| usersTableAP | | | |
| 欄位名稱 | 欄位型態 | 欄位大小 | 用途 |
| id | Int | 11 | PK |
| sqDate | String | 255 | 記帳日月 |
| sqName | String | 255 | 記帳名稱 |
| sqPrice | Int | 11 | 記帳價錢 |
| sqMonth | String | 255 | 記帳月 |
| sqCoupon | String | 255 | 優惠卷 |
| sqDiscount | Int | 11 | 折扣 |
| sqSave | Int | 11 | 節省多少錢 |

表8-2-8 預算資料表

|  |  |  |  |
| --- | --- | --- | --- |
| usersTable | | | |
| 欄位名稱 | 欄位型態 | 欄位大小 | 用途 |
| id | Int | 11 | PK |
| budget | Int | 11 | 預算 |

**第九章：程式**

**9-1 元件清單及其規格描述**

表9-1-1 Python元件清單

|  |  |  |
| --- | --- | --- |
| 資料夾 | 檔案名稱 | 功能 |
| crawler | Excel\_To\_SQL.py | 將excel存入資料庫中 |
| insert\_coupon\_to\_mysql.py | 將爬蟲資料存入資料庫 |
| insert\_into\_mysql.py | 資料庫存放function |
| Convert\_Img.py | 轉換圖片和影像辨識價錢 |
| Log.py | 錯誤時撰寫Log |
| KFC\_crawler.py | 肯德基爬蟲程式 |
| KFC1104.xlsx | 爬下來後的Excel |
| media | coupons | 存放圖片的資料夾 |
| templates | registration | 存放註冊的暫時性網頁 |
| test\_token | filter.py | Filter功能 |
| model.py | 存放model，供資料庫串接使用 |
| serializers.py | 處理序列化 |
| tests.py | 單元測試 |
| views.py | 處理要回傳Json的內容 |
| migration | 紀錄資料庫更新的內容 |
| try\_token | setting.py | API config |
| url.py | API routing |
| wsgi.py | Web Server Gateway Interface |
| util | permission.py | 判斷使用者可以看到的內容 |

表9-1-2 Swift元件清單

|  |  |  |
| --- | --- | --- |
| 資料夾 | 檔案名稱 | 功能 |
| help | welcomePage.storyboard | 教學頁面設計版 |
| CustomPageViewController.swift | 客制版面設計 |
| ViewController.swift | 控制教學頁面 |
| common | LaunchScreen.storyboard | 載入頁面 |
| BridgeHeader.h | Sqlite協定 |
| AppDelegate.swift | 程式協定 |
| VarPage.swift | 設定廣域變數 |
| ViewController.swift | 呼叫help資料夾裡的東西 |
| Main.storyboard | 程式頁面設計版 |
| MotherBoard.swift | 母版 |
| pageviewcontroller.swift | 主頁面控制器 |
| CenterCenterPage.swift | 裡頁面控制器 |
| CenterPage.swift | 主頁中間頁面設計與功能 |
| CenterCenter.swift | 裡頁中間頁面設計與功能 |
| CenterLeft.swift | 裡頁左邊頁面設計與功能 |
| CenterLeftCell.swift | 定義裡頁左邊頁面的cell |
| CenterRight.swift | 裡頁右邊頁面設計與功能 |
| TargetPage.swift | 設定目標及其他設定頁面 |
| LogIn.swift | 登入頁面設計與功能 |
| SignUp.swift | 註冊頁面設計與功能 |
| ForgotPassword.swift | 忘記密碼頁面設計與功能 |
| DailyDetail.swift | 記帳頁面設計與功能 |
| DailyDetailForm.swift | 記帳cell物件化 |
| DailyDetailCell.swift | 定義記帳頁面的cell |
| RightPage.swift | Coupon頁面設計與功能 |
| CouponArrayFormat.swift | CouponCell物件化 |
| CouponCellFormat.swift | 定義Coupon頁面的Cell |
| WebView.swift | 連至Coupon圖案的頁面 |
| UibuttonExtension.swift | 按鍵反饋設計與功能 |
| RefreshContents.xib | Coupon頁面的刷新圖樣動畫設計 |

表9-1-3 後端程式setting

|  |  |
| --- | --- |
| 程式名稱 | setting |
| 功能 | 配置和設定元件 |
| **import** os  **from** crawler.password **import** My\_password  *# Build paths inside the project like this: os.path.join(BASE\_DIR, ...)*  BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))  *# Quick-start development settings - unsuitable for production*  *# See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/*  *# SECURITY WARNING: keep the secret key used in production secret!*  SECRET\_KEY = **'72#jc)ljx@)@qvx\_av9fc=i%jxtwe0g=t)bplo0j)@\_as7t5!9'**  *# SECURITY WARNING: don't run with debug turned on in production!*  DEBUG = **True**  *#利於使用ngrok測試*  ALLOWED\_HOSTS = [**"\*"**]  INSTALLED\_APPS = [  **'django.contrib.admin'**,  **'django.contrib.auth'**,  **'django.contrib.contenttypes'**,  **'django.contrib.sessions'**,  **'django.contrib.messages'**,  **'django.contrib.staticfiles'**,  *#rest auth setting*  **'rest\_framework'**,  **'rest\_framework.authtoken'**,  **'rest\_auth'**,    *#rest auth Registration*  **'django.contrib.sites'**,  **'allauth'**,  **'allauth.account'**,  **'rest\_auth.registration'**,  *#第三方登入 facebook*  **'allauth.socialaccount'**,  *# 'allauth.socialaccount.providers.facebook',*  **'allauth.socialaccount.providers.google'**,  **'test\_token'**,  **'django\_filters'**, *#自定義過濾器*  ]  *# rest-auth registration setting (設定要用哪一個domain)*  SITE\_ID = 2  *#=================django-restframework-jwt=======================*  *# #設定token時效*  **import** datetime  JWT\_AUTH = {  *#允不允許重新刷新*  **'JWT\_ALLOW\_REFRESH'**: **True**,  *#失效時間(有動還是會失效)*  **'JWT\_REFRESH\_EXPIRATION\_DELTA'**: datetime.timedelta(days=30),  *#失效時間(都不動的話)*  **'JWT\_EXPIRATION\_DELTA'**: datetime.timedelta(days=1),  }  *#=================django-restframework-jwt=======================*  MIDDLEWARE = [  **'django.middleware.security.SecurityMiddleware'**,  **'django.contrib.sessions.middleware.SessionMiddleware'**,  **'django.middleware.common.CommonMiddleware'**,  **'django.middleware.csrf.CsrfViewMiddleware'**,  **'django.contrib.auth.middleware.AuthenticationMiddleware'**,  **'django.contrib.messages.middleware.MessageMiddleware'**,  **'django.middleware.clickjacking.XFrameOptionsMiddleware'**,  ]  ROOT\_URLCONF = **'try\_token.urls'**  TEMPLATES = [  {  **'BACKEND'**: **'django.template.backends.django.DjangoTemplates'**,  **'DIRS'**: [os.path.join(BASE\_DIR, **'templates'**)],  **'APP\_DIRS'**: **True**,  **'OPTIONS'**: {  **'context\_processors'**: [  **'django.template.context\_processors.debug'**,  **'django.template.context\_processors.request'**,  **'django.contrib.auth.context\_processors.auth'**,  **'django.contrib.messages.context\_processors.messages'**,  ],  },  },  ]  WSGI\_APPLICATION = **'try\_token.wsgi.application'**  *# Database*  *# https://docs.djangoproject.com/en/2.2/ref/settings/#databases*  DATABASES = {  **'default'**: {  **'ENGINE'**: **'django.db.backends.mysql'**,  **'NAME'**: **'ntub\_coupon\_backend'**,  **'USER'**: **'root'**,  **'PASSWORD'**: My\_password,  **'HOST'**: **'127.0.0.1'**,  **'PORT'**: **'3306'**,  *#第三方登入。。。。不設定migration會出錯*  **'OPTIONS'**: {**'init\_command'**: **'SET default\_storage\_engine=INNODB;'**}  },  }  *# Password validation*  *# https://docs.djangoproject.com/en/2.2/ref/settings/#auth-password-validators*  AUTH\_PASSWORD\_VALIDATORS = [  {  **'NAME'**: **'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator'**,  },  {  **'NAME'**: **'django.contrib.auth.password\_validation.MinimumLengthValidator'**,  },  {  **'NAME'**: **'django.contrib.auth.password\_validation.CommonPasswordValidator'**,  },  {  **'NAME'**: **'django.contrib.auth.password\_validation.NumericPasswordValidator'**,  },  ]  AUTH\_USER\_MODEL = **'test\_token.User'**  PASSWORD\_HASHERS = [  **'django.contrib.auth.hashers.PBKDF2SHA1PasswordHasher'**,  ]  *# Internationalization*  *# https://docs.djangoproject.com/en/2.2/topics/i18n/*  LANGUAGE\_CODE = **'en-us'**  TIME\_ZONE = **'UTC'**  USE\_I18N = **True**  USE\_L10N = **True**  USE\_TZ = **True**  *# Static files (CSS, JavaScript, Images)*  *# https://docs.djangoproject.com/en/2.2/howto/static-files/*  STATIC\_URL = **'/static/'**  MEDIA\_URL= **'/media/'**  MEDIA\_ROOT = os.path.join(BASE\_DIR, **'media'**)  *# 如果會發生Timeout的事情，要先檢查網路是否正常*  EMAIL\_BACKEND = **'django.core.mail.backends.smtp.EmailBackend'**  EMAIL\_HOST = **'smtp.gmail.com'**  EMAIL\_HOST\_USER = **'jmm.chang@gmail.com'**  EMAIL\_HOST\_PASSWORD = **My\_password**  EMAIL\_PORT = 587  EMAIL\_USE\_TLS = **True**  *#讓第三方登入重新導向login頁面*  LOGIN\_REDIRECT\_URL = **'/login'**  *#讓user一定要填email*  ACCOUNT\_EMAIL\_REQUIRED = **True**  **#設定認證方式**  ACCOUNT\_EMAIL\_VERIFICATION = **"mandatory"**  REST\_USE\_JWT = **True**  REST\_FRAMEWORK = {  **'DEFAULT\_AUTHENTICATION\_CLASSES'**: [  **'rest\_framework\_jwt.authentication.JSONWebTokenAuthentication'**,  **'rest\_framework.authentication.BasicAuthentication'**,  ],  **'DEFAULT\_FILTER\_BACKENDS'**: [**'django\_filters.rest\_framework.DjangoFilterBackend'**]  }  *# after confirming email , redirect url*  ACCOUNT\_LOGIN\_ON\_EMAIL\_CONFIRMATION = **False**  ACCOUNT\_EMAIL\_CONFIRMATION\_ANONYMOUS\_REDIRECT\_URL = **'/login'**  STATIC\_URL = **'/static/'**  STATICFILES\_DIRS = [  os.path.join(BASE\_DIR,**'static'**)  ] | |

表9-1-4 後端程式permission

|  |  |
| --- | --- |
| 程式名稱 | permission |
| 功能 | 管理使用者可看見的內容 |
| **from** rest\_framework **import** permissions  **class** IsOwnerOrReadOnly(permissions.BasePermission):  *"""*  *Object-level permission to only allow owners of an object to edit it.*  *Assumes the model instance has an `owner` attribute.*  *"""*  **def** has\_object\_permission(self, request, view, obj):  *# Read permissions are allowed to any request,*  *# so we'll always allow GET, HEAD or OPTIONS requests.*  **if** request.method **in** permissions.SAFE\_METHODS:  **return True**  *# Instance must have an attribute named `owner`.*  *#obj可以看成資料庫裡model，這裡把owner改成我們資料库中的user*  **return** obj.user == request.user | |

表9-1-5 後端程式url,py

|  |  |
| --- | --- |
| 程式名稱 | url.py |
| 功能 | API routing |
| **from** django.contrib **import** admin  **from** django.urls **import** path, include  **from** rest\_framework.routers **import** DefaultRouter  **from** django.conf.urls **import** url  **from** django.conf **import** settings  **from** django.conf.urls.static **import** static  **from** test\_token.views **import** (CouponViewSet,  UserViewSet,  UserFavViewset,  ActivateUserByGet,  ResetPasswordUserByGet,  UserAccountingViewset, )  *#rest\_auth*  **from** rest\_auth.views **import** (  LoginView, LogoutView, UserDetailsView, PasswordChangeView,  PasswordResetView, PasswordResetConfirmView  )  **from** django.views.generic **import** TemplateView  *# JWT*  **from** test\_token.views **import** ObtainJSONWebToken  **from** rest\_framework\_jwt.views **import** refresh\_jwt\_token  **from** rest\_framework\_jwt.views **import** verify\_jwt\_token  *# rest auth google*  **from** test\_token.views **import** GoogleLogin  router = DefaultRouter()  router.register(**r'coupon'**,CouponViewSet,base\_name=**'coupon'**)  router.register(**r'users'**,UserViewSet,base\_name=**'users'**)  router.register(**r'userfavs'**, UserFavViewset, base\_name=**"userfavs"**)  router.register(**r'useraccouting'**, UserAccountingViewset, base\_name=**"useraccouting"**)  urlpatterns = [  *# rest-auth setting / Registration*  *##-------------------------rest-auth登入----------------------------------------------------------*  *# URLs that do not require a session or valid token*  url(**r'^password/reset/$'**, PasswordResetView.as\_view(),  name=**'rest\_password\_reset'**),  url(**r'^password/reset/confirm/$'**, PasswordResetConfirmView.as\_view(),  name=**'rest\_password\_reset\_confirm'**),  url(**r'^users/login/$'**, LoginView.as\_view(), name=**'rest\_login'**),  *# URLs that require a user to be logged in with a valid session / token.*  url(**r'^users/logout/$'**, LogoutView.as\_view(), name=**'rest\_logout'**),  url(**r'^user/$'**, UserDetailsView.as\_view(), name=**'rest\_user\_details'**),  url(**r'^password/change/$'**, PasswordChangeView.as\_view(),  name=**'rest\_password\_change'**),  *##-------------------------rest-auth登入----------------------------------------------------------*  *#因為rest-auth預設使用allauth，所以要匯入allauth的url，才不會造成NoReverse的錯誤*  url(**r'^accounts/'**, include(**'allauth.urls'**)),  *#reset password 預設會用到django的admin管理網站，所以我們先匯入他測試看看*  url(**r'^'**, include(**'django.contrib.auth.urls'**)),  url(**r'^rest-auth/registration/'**, include(**'rest\_auth.registration.urls'**)),  path(**""**,include(router.urls)),  url(**r'^rest-auth/google/$'**, GoogleLogin.as\_view(), name=**'google\_login'**),  *#JWT 套用*  path(**'api-token-auth/'**, ObtainJSONWebToken.as\_view()),  path(**'api-token-refresh/'**, refresh\_jwt\_token),  path(**'api-token-verify/'**, verify\_jwt\_token),  path(**'admin/'**, admin.site.urls),  path(**'api-auth/'**, include(**'rest\_framework.urls'**, namespace=**'rest\_framework'**)),  ] + static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT) | |

表9-1-6 後端程式view.py

|  |  |
| --- | --- |
| 程式名稱 | view.py |
| 功能 | 設定要回傳的JSON |
| *# Create your views here.*  **from** rest\_framework **import** viewsets  **from** rest\_framework.parsers **import** JSONParser  **from** rest\_framework.permissions **import** IsAuthenticated,AllowAny  **from** rest\_framework **import** filters*# Create your views here.*  **from** rest\_framework **import** viewsets  **from** rest\_framework.parsers **import** JSONParser  **from** rest\_framework.permissions **import** IsAuthenticated,AllowAny  **from** rest\_framework **import** filters  **from** rest\_framework\_jwt.authentication **import** JSONWebTokenAuthentication  *#permission(用於我的最愛)*  **from** util.permission **import** IsOwnerOrReadOnly  **from** rest\_framework.response **import** Response  **from** rest\_framework **import** status,generics,mixins,reverse  **from** .models **import** Coupon,User,UserFav,UserAccounting  **from** .serializers **import** CouponSerializer,UserSerializer ,UserFavSerializer,UserAccountingSerializer  *#設定分頁*  **from** rest\_framework.pagination **import** PageNumberPagination  *#設定過濾器*  **from** .filter **import** CouponFilter  **from** django\_filters.rest\_framework **import** DjangoFilterBackend  *#設定搜尋，排序*  **from** rest\_framework.filters **import** SearchFilter , OrderingFilter  *# to post activate*  **from** rest\_framework.views **import** APIView  **import** requests  *#隨機密碼*  **from** .random\_passwd **import** RandomPassword  *#其他*  **from** django.contrib.auth.tokens **import** default\_token\_generator  **from** templated\_mail.mail **import** BaseEmailMessage  **from** djoser **import** utils  **from** djoser.conf **import** settings  **class** PasswordResetEmail(BaseEmailMessage):  template\_name = **'email/password\_reset.html'**  **def** get\_context\_data(self):  context = super(PasswordResetEmail, self).get\_context\_data()  user = context.get(**'user'**)  context[**'uid'**] = utils.encode\_uid(user.pk)  context[**'token'**] = default\_token\_generator.make\_token(user)  context[**'url'**] = settings.PASSWORD\_RESET\_CONFIRM\_URL.format(\*\*context)  *# context['new\_password'] = RandomPassword() ##修改的地方*  **return** context  *##激活帳號*  **class** ActivateUserByGet(APIView):  **def** get (self, request, uid, token):  *#獲取網址中的uid和token，透過requests中的post方法自行post進去做激活*  protocol = **'https://' if** request.is\_secure() **else 'http://'**  web\_url = protocol + request.get\_host()  post\_url = web\_url + **"/auth/users/activate/"**  post\_data = {**'uid'**: uid, **'token'**: token}  result = requests.post(post\_url, data = post\_data)  *#判斷成功激活或失敗*  **if** result.status\_code == 204:  **return** Response({**'detail'**: **'all good sir'**})  **else**:  **return** Response(result.json())  *##重設密碼*  **class** ResetPasswordUserByGet(APIView):  **def** get (self, request, uid, token):  *#獲取網址中的uid和token，透過requests中的post方法自行post進去做激活*  new\_password = RandomPassword()  protocol = **'https://' if** request.is\_secure() **else 'http://'**  web\_url = protocol + request.get\_host()  post\_url = web\_url + **"/auth/password/reset/confirm/"**  post\_data = {**'uid'**: uid, **'token'**: token,**'new\_password'**:new\_password}  result = requests.post(post\_url, data = post\_data)  *#判斷成功激活或失敗*  **if** result.status\_code == 204:  **return** Response({**'new\_password'**:new\_password})  **else**:  **return** Response(result.json())  *##Coupont自訂分頁*  **class** CouponPagination(PageNumberPagination):  *#每頁顯示個數*  page\_size = 10  *#允許動態改變每頁顯示個數*  page\_size\_query\_param = **'page\_size'**  *#設定頁碼參數*  page\_query\_param = **'page'**  *#最多多少頁*  max\_page\_size = 100  **class** UserViewSet(viewsets.GenericViewSet,  mixins.RetrieveModelMixin,  mixins.ListModelMixin):  queryset = User.objects.all()  serializer\_class = UserSerializer  *# 透過自定permission，來判斷使用者是誰*  permission\_classes = (IsAuthenticated, IsOwnerOrReadOnly)  lookup\_field = **'user'**  **def** get\_queryset(self):  *# 只能看到自己的收藏，不能看到別人的*  **return** User.objects.filter(user=self.request.user)  *#viewsets.ModelViewSet 測試用*  *# Create your views here.*  **class** CouponViewSet(viewsets.GenericViewSet  ,mixins.RetrieveModelMixin  , mixins.ListModelMixin):  queryset = Coupon.objects.all()  serializer\_class = CouponSerializer  *#載入自定義Pagination*  pagination\_class = CouponPagination  *#自定義filter*  filter\_backends = (DjangoFilterBackend,SearchFilter,OrderingFilter)  filter\_class = CouponFilter  *#search設定*  search\_fields = (**"coupon\_title"**,**"coupon\_class"**)  *#ordering\_fields --> 設定可排序的參數 ordering -> 默認排序*  ordering\_fields = (**'coupon\_price'**,**"coupon\_saving"**)  ordering=(**'coupon\_title'**,)  *# permission設置(JWT)*  permission\_classes = (IsAuthenticated, IsOwnerOrReadOnly)  *#因為在serializers 是 coupons ->所以這裡不是coupon\_id*  lookup\_field = **'coupon\_id'**  *#我的最愛*  **class** UserFavViewset(viewsets.GenericViewSet  ,mixins.RetrieveModelMixin  , mixins.ListModelMixin  , mixins.CreateModelMixin  , mixins.DestroyModelMixin):  queryset = UserFav.objects.all()  serializer\_class = UserFavSerializer  *#透過自定permission，來判斷使用者是誰*  permission\_classes = (IsAuthenticated, IsOwnerOrReadOnly)  *#因為在serializers 是 coupons ->所以這裡不是coupon\_id*  lookup\_field = **'coupons\_id'**  **def** get\_queryset(self):  *# 只能看到自己的收藏，不能看到別人的*  **return** UserFav.objects.filter(user=self.request.user)  *#記帳功能*  **class** UserAccountingViewset(viewsets.GenericViewSet  ,mixins.RetrieveModelMixin  , mixins.ListModelMixin  , mixins.CreateModelMixin  , mixins.DestroyModelMixin):  queryset = UserAccounting.objects.all()  serializer\_class = UserAccountingSerializer  *# #透過自定permission，來判斷使用者是誰*  permission\_classes = (IsAuthenticated, IsOwnerOrReadOnly)  lookup\_field = **'id'**  *#search功能設定*  filter\_backends = [filters.SearchFilter]  search\_fields = [**'accounting\_date'**]  **def** get\_queryset(self):  *# 只能看到自己的記帳，不能看到別人的*  **return** UserAccounting.objects.filter(user=self.request.user)  **from** rest\_framework\_jwt.serializers **import** JSONWebTokenSerializer  **from** rest\_framework\_jwt.views **import** JSONWebTokenAPIView  **from** rest\_framework\_jwt.views **import** jwt\_response\_payload\_handler  **from** try\_token **import** settings **as** Mysetting  **class** ObtainJSONWebToken(JSONWebTokenAPIView):  serializer\_class = JSONWebTokenSerializer  **def** post(self, request):  serializer = self.get\_serializer(data=request.data)  **if** serializer.is\_valid():  user = serializer.object.get(**'user'**) **or** request.user  *# check if settings swith is on / then check validity*  **if** Mysetting.ACCOUNT\_EMAIL\_VERIFICATION == **"mandatory"**:  email\_address = user.emailaddress\_set.get(email=user.email)  **if not** email\_address.verified:  **return** Response(status=403, data=**'E-mail is not verified.'**)  token = serializer.object.get(**'token'**)  response\_data = jwt\_response\_payload\_handler(token, user, request)  **return** Response(response\_data)  **return** Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  *# trying to use rest auth ( google )*  **from** allauth.socialaccount.providers.google.views **import** GoogleOAuth2Adapter  **from** rest\_auth.registration.views **import** SocialLoginView  **class** GoogleLogin(SocialLoginView):  adapter\_class = GoogleOAuth2Adapter  **from** rest\_framework\_jwt.authentication **import** JSONWebTokenAuthentication  *#permission(用於我的最愛)*  **from** util.permission **import** IsOwnerOrReadOnly  **from** rest\_framework.response **import** Response  **from** rest\_framework **import** status,generics,mixins,reverse  **from** .models **import** Coupon,User,UserFav,UserAccounting  **from** .serializers **import** CouponSerializer,UserSerializer ,UserFavSerializer,UserAccountingSerializer  *#設定分頁*  **from** rest\_framework.pagination **import** PageNumberPagination  *#設定過濾器*  **from** .filter **import** CouponFilter  **from** django\_filters.rest\_framework **import** DjangoFilterBackend  *#設定搜尋，排序*  **from** rest\_framework.filters **import** SearchFilter , OrderingFilter  *# to post activate*  **from** rest\_framework.views **import** APIView  **import** requests  *#隨機密碼*  **from** .random\_passwd **import** RandomPassword  *#其他*  **from** django.contrib.auth.tokens **import** default\_token\_generator  **from** templated\_mail.mail **import** BaseEmailMessage  **from** djoser **import** utils  **from** djoser.conf **import** settings  **class** PasswordResetEmail(BaseEmailMessage):  template\_name = **'email/password\_reset.html'**  **def** get\_context\_data(self):  context = super(PasswordResetEmail, self).get\_context\_data()  user = context.get(**'user'**)  context[**'uid'**] = utils.encode\_uid(user.pk)  context[**'token'**] = default\_token\_generator.make\_token(user)  context[**'url'**] = settings.PASSWORD\_RESET\_CONFIRM\_URL.format(\*\*context)  *# context['new\_password'] = RandomPassword() ##修改的地方*  **return** context  *##激活帳號*  **class** ActivateUserByGet(APIView):  **def** get (self, request, uid, token):  *#獲取網址中的uid和token，透過requests中的post方法自行post進去做激活*  protocol = **'https://' if** request.is\_secure() **else 'http://'**  web\_url = protocol + request.get\_host()  post\_url = web\_url + **"/auth/users/activate/"**  post\_data = {**'uid'**: uid, **'token'**: token}  result = requests.post(post\_url, data = post\_data)  *#判斷成功激活或失敗*  **if** result.status\_code == 204:  **return** Response({**'detail'**: **'all good sir'**})  **else**:  **return** Response(result.json())  *##重設密碼*  **class** ResetPasswordUserByGet(APIView):  **def** get (self, request, uid, token):  *#獲取網址中的uid和token，透過requests中的post方法自行post進去做激活*  new\_password = RandomPassword()  protocol = **'https://' if** request.is\_secure() **else 'http://'**  web\_url = protocol + request.get\_host()  post\_url = web\_url + **"/auth/password/reset/confirm/"**  post\_data = {**'uid'**: uid, **'token'**: token,**'new\_password'**:new\_password}  result = requests.post(post\_url, data = post\_data)  *#判斷成功激活或失敗*  **if** result.status\_code == 204:  **return** Response({**'new\_password'**:new\_password})  **else**:  **return** Response(result.json())  *##Coupont自訂分頁*  **class** CouponPagination(PageNumberPagination):  *#每頁顯示個數*  page\_size = 10  *#允許動態改變每頁顯示個數*  page\_size\_query\_param = **'page\_size'**  *#設定頁碼參數*  page\_query\_param = **'page'**  *#最多多少頁*  max\_page\_size = 100  **class** UserViewSet(viewsets.GenericViewSet,  mixins.RetrieveModelMixin,  mixins.ListModelMixin):  queryset = User.objects.all()  serializer\_class = UserSerializer  *# 透過自定permission，來判斷使用者是誰*  permission\_classes = (IsAuthenticated, IsOwnerOrReadOnly)  lookup\_field = **'user'**  **def** get\_queryset(self):  *# 只能看到自己的收藏，不能看到別人的*  **return** User.objects.filter(user=self.request.user)  *#viewsets.ModelViewSet 測試用*  *# Create your views here.*  **class** CouponViewSet(viewsets.GenericViewSet  ,mixins.RetrieveModelMixin  , mixins.ListModelMixin):  queryset = Coupon.objects.all()  serializer\_class = CouponSerializer  *#載入自定義Pagination*  pagination\_class = CouponPagination  *#自定義filter*  filter\_backends = (DjangoFilterBackend,SearchFilter,OrderingFilter)  filter\_class = CouponFilter  *#search設定*  search\_fields = (**"coupon\_title"**,**"coupon\_class"**)  *#ordering\_fields --> 設定可排序的參數 ordering -> 默認排序*  ordering\_fields = (**'coupon\_price'**,**"coupon\_saving"**)  ordering=(**'coupon\_title'**,)  *# permission設置(JWT)*  permission\_classes = (IsAuthenticated, IsOwnerOrReadOnly)  *#因為在serializers 是 coupons ->所以這裡不是coupon\_id*  lookup\_field = **'coupon\_id'**  *#我的最愛*  **class** UserFavViewset(viewsets.GenericViewSet  ,mixins.RetrieveModelMixin  , mixins.ListModelMixin  , mixins.CreateModelMixin  , mixins.DestroyModelMixin):  queryset = UserFav.objects.all()  serializer\_class = UserFavSerializer  *#透過自定permission，來判斷使用者是誰*  permission\_classes = (IsAuthenticated, IsOwnerOrReadOnly)  *#因為在serializers 是 coupons ->所以這裡不是coupon\_id*  lookup\_field = **'coupons\_id'**  **def** get\_queryset(self):  *# 只能看到自己的收藏，不能看到別人的*  **return** UserFav.objects.filter(user=self.request.user)  *#記帳功能*  **class** UserAccountingViewset(viewsets.GenericViewSet  ,mixins.RetrieveModelMixin  , mixins.ListModelMixin  , mixins.CreateModelMixin  , mixins.DestroyModelMixin):  queryset = UserAccounting.objects.all()  serializer\_class = UserAccountingSerializer  *# #透過自定permission，來判斷使用者是誰*  permission\_classes = (IsAuthenticated, IsOwnerOrReadOnly)  lookup\_field = **'id'**  *#search功能設定*  filter\_backends = [filters.SearchFilter]  search\_fields = [**'accounting\_date'**]  **def** get\_queryset(self):  *# 只能看到自己的記帳，不能看到別人的*  **return** UserAccounting.objects.filter(user=self.request.user)  **from** rest\_framework\_jwt.serializers **import** JSONWebTokenSerializer  **from** rest\_framework\_jwt.views **import** JSONWebTokenAPIView  **from** rest\_framework\_jwt.views **import** jwt\_response\_payload\_handler  **from** try\_token **import** settings **as** Mysetting  **class** ObtainJSONWebToken(JSONWebTokenAPIView):  serializer\_class = JSONWebTokenSerializer  **def** post(self, request):  serializer = self.get\_serializer(data=request.data)  **if** serializer.is\_valid():  user = serializer.object.get(**'user'**) **or** request.user  *# check if settings swith is on / then check validity*  **if** Mysetting.ACCOUNT\_EMAIL\_VERIFICATION == **"mandatory"**:  email\_address = user.emailaddress\_set.get(email=user.email)  **if not** email\_address.verified:  **return** Response(status=403, data=**'E-mail is not verified.'**)  token = serializer.object.get(**'token'**)  response\_data = jwt\_response\_payload\_handler(token, user, request)  **return** Response(response\_data)  **return** Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  *# trying to use rest auth ( google )*  **from** allauth.socialaccount.providers.google.views **import** GoogleOAuth2Adapter  **from** rest\_auth.registration.views **import** SocialLoginView  **class** GoogleLogin(SocialLoginView):  adapter\_class = GoogleOAuth2Adapter | |

表9-1-7 後端程式test.py

|  |  |
| --- | --- |
| 程式名稱 | test.py |
| 功能 | 單元測試 |
| **from django.test import TestCase**  **from test\_token.models import User , Coupon , UserAccounting**  **from rest\_framework import status**  **from rest\_framework.test import APITestCase**  ***#Django Model Test***  **class UserModelTestCase(TestCase):**  **def test\_create\_user(self):**  **user = User(username="jack")**  **user.save()**  **db\_user = User.objects.get(username=user.username)**  **self.assertEqual(user == db\_user , True)**  **def test\_create\_coupon(self):**  **coupon = Coupon(coupon\_id = "655" , coupon\_price = 87 , coupon\_original\_price = 107 , coupon\_title = "測試"**  **,coupon\_class = "食物" , coupon\_note = "測試" , coupon\_notice ="測試" , coupon\_saving = 20)**  **coupon.save()**  **db\_coupon = Coupon.objects.get(coupon\_id = coupon.coupon\_id)**  **self.assertEqual(coupon == db\_coupon , True)**  **def test\_create\_useraccounting(self):**  **user = User(username="jack2")**  **user.save()**  **useraccounting = UserAccounting(user\_id=user.id,accounting\_class='食物',accounting\_data=111,accounting\_date='1110',accounting\_discount='八折'**  **,accounting\_coupon\_name='優惠卷測試',accounting\_month='11')**  **useraccounting.save()**  **db\_useraccounting = UserAccounting.objects.get(user\_id = user.id)**  **self.assertEqual(useraccounting == db\_useraccounting, True)**  ***#API Test***  **class AccountTests(APITestCase):**  **def test\_create\_account(self):**  **url = '/rest-auth/registration/'**  **data = {'username': 'benson','email':'aa@bb.cc', 'password1':'as12345678' , 'password2':'as12345678'}**  **response = self.client.post(url, data, format='json')**  **self.assertEqual(response.status\_code, status.HTTP\_201\_CREATED)**  **self.assertEqual(User.objects.count(), 1)**  **self.assertEqual(User.objects.get().username, 'benson')**  **def test\_email\_confirm(self):**  **url = "/api-token-auth/"**  **data = {"username":"benson" , "password":"as12345678"}**  **response = self.client.post(url, data, format='json')**  **self.assertEqual(response.status\_code, status.HTTP\_400\_BAD\_REQUEST)** | |

表9-1-8 後端程式serializers.py

|  |  |
| --- | --- |
| 程式名稱 | serializers.py |
| 功能 | 將資料序列化 |
| **from rest\_framework import serializers**  ***#使用者收藏***  **from rest\_framework.validators import UniqueTogetherValidator**  **from .models import Coupon,User,UserFav,UserAccounting**  **class UserSerializer(serializers.HyperlinkedModelSerializer):**  **class Meta:**  **model = User**  **fields = ('url', 'id','username', 'email', 'last\_name', 'password')**  **class CouponSerializer(serializers.ModelSerializer):**  ***# image\_url = serializers.SerializerMethodField('get\_url')***  **coupon\_img = serializers.ImageField(required=False,max\_length=None,**  **allow\_empty\_file=True, use\_url=True)**  **class Meta:**  **model = Coupon**  **fields = ('coupon\_id','coupon\_title','coupon\_class','coupon\_note','coupon\_notice','coupon\_original\_price',**  **'coupon\_price','coupon\_saving','coupon\_img')**  **class UserFavSerializer(serializers.ModelSerializer):**  ***#獲取當前使用者是誰***  **user = serializers.HiddenField(**  **default=serializers.CurrentUserDefault()**  **)**  **class Meta:**  ***# 設定validate使一個coupons只能收藏一次***  **validators = [**  ***#只能一個***  **UniqueTogetherValidator(**  **queryset=UserFav.objects.all(),**  **fields=('user', 'coupons'),**  ***# message的信息可以自定义***  **message="已經收藏"**  **)**  **]**  **model = UserFav**  ***# 必填，因為取消收藏時要知道ID，所以ID也是必填***  **fields = ("user", "coupons", 'id')**  **class UserAccountingSerializer(serializers.ModelSerializer):**  ***#獲取當前使用者是誰***  **user = serializers.HiddenField(**  **default=serializers.CurrentUserDefault()**  **)**  **class Meta:**  **model = UserAccounting**  ***#必填***  **fields = ("user", 'id' , "accounting\_data" , "accounting\_date" , "accounting\_class","accounting\_month",**  **"accounting\_discount","accounting\_coupon\_name")** | |

表9-1-9 後端程式models.py

|  |  |
| --- | --- |
| 程式名稱 | models.py |
| 功能 | 設定資料模型 |
| **from django.db import models**  **import datetime**  ***# Create your models here.***  **from django.contrib.auth.models import AbstractUser**  **class User(AbstractUser):**  **created = models.DateTimeField(auto\_now\_add=True,verbose\_name="建立時間")**  **gender=models.CharField(max\_length=1,verbose\_name="性別")**  **email = models.EmailField(unique=True)**  **REQUIRED\_FIELDS = ["gender","email"]**  **class Meta:**  **ordering = ('created',)**  **class Coupon(models.Model):**  ***# user = models.ForeignKey(User,on\_delete=models.CASCADE)***  **coupon\_id = models.CharField(max\_length=20, primary\_key=True,verbose\_name="優惠卷編號")**  **coupon\_price = models.IntegerField(verbose\_name="優惠卷特價價格")**  **coupon\_original\_price = models.IntegerField(blank=True,null=True,verbose\_name="優惠卷原始價格")**  **coupon\_title = models.CharField(max\_length=150,verbose\_name="優惠卷標題")**  **coupon\_class = models.CharField(max\_length=20,verbose\_name="優惠卷類別")**  **coupon\_note = models.CharField(max\_length=200,verbose\_name="優惠卷內容")**  **coupon\_notice = models.CharField(max\_length=200,verbose\_name="優惠卷注意事項")**  **coupon\_saving = models.IntegerField(verbose\_name="優惠卷折扣")**  **coupon\_img = models.ImageField(upload\_to='coupons', blank=True,null=True,verbose\_name="優惠卷圖片")**  **coupon\_create\_at = models.DateTimeField(auto\_now\_add=True,verbose\_name="優惠卷創建時間")**  **class Meta:**  **verbose\_name = '優惠卷'**  **verbose\_name\_plural = verbose\_name**  **ordering = ('coupon\_create\_at',)**  **def \_\_str\_\_(self):**  **return self.coupon\_title**  ***#使用者收藏***  **class UserFav(models.Model):**  ***#兩個外鍵行成關聯表***  **user = models.ForeignKey(User, on\_delete=models.CASCADE, verbose\_name="使用者")**  **coupons = models.ForeignKey(Coupon, on\_delete=models.CASCADE, verbose\_name="優惠卷", help\_text="優惠卷id")**  **user\_fav\_create\_at = models.DateTimeField("添加時間",auto\_now\_add=True)**  **class Meta:**  **verbose\_name = '使用者收藏'**  **verbose\_name\_plural = verbose\_name**  **unique\_together = ("user", "coupons")**  **def \_\_str\_\_(self):**  **return self.user.username**  ***#使用者記帳***  **class UserAccounting(models.Model):**  **user = models.ForeignKey(User, on\_delete=models.CASCADE, verbose\_name="使用者")**  **accounting\_class = models.CharField(max\_length=30,verbose\_name="記帳類別")**  **accounting\_data = models.IntegerField(default=0,verbose\_name="記帳資料")**  **accounting\_date = models.CharField(max\_length=30 , default=str(datetime.datetime.now().strftime("%m%d")),**  **verbose\_name="記帳日期")**  **accounting\_discount = models.CharField(max\_length=30,blank=True,null=True,verbose\_name="記帳折扣")**  **accounting\_coupon\_name = models.CharField(max\_length=50,blank=True,null=True,verbose\_name="優惠卷名稱")**  **accounting\_month = models.CharField(max\_length=2,verbose\_name="記帳月份")**  **user\_accounting\_post\_at = models.DateTimeField(auto\_now\_add=True,verbose\_name="記帳創建時間")**  **class Meta:**  **verbose\_name = '使用者記帳'**  **verbose\_name\_plural = verbose\_name**  **def \_\_str\_\_(self):**  **return self.user.username** | |

表9-1-10 後端程式filter.py

|  |  |
| --- | --- |
| 程式名稱 | filter.py |
| 功能 | 過濾資料功能 |
| **import django\_filters**  **from .models import Coupon**  **class CouponFilter(django\_filters.rest\_framework.FilterSet):**  ***#field\_name -> 要被過濾的東西 lookup -> 要如何過濾(這裡是小於等於)***  **price\_min = django\_filters.NumberFilter(field\_name="coupon\_price", lookup\_expr='gte')**  **price\_max = django\_filters.NumberFilter(field\_name="coupon\_price", lookup\_expr='lte')**  **class Meta:**  ***#和serializers的寫法一樣***  **model = Coupon**  **fields = ['price\_min', 'price\_max']** | |

表9-1-11 後端程式insert\_coupon\_to\_mysql

|  |  |
| --- | --- |
| 程式名稱 | insert\_coupon\_to\_mysql |
| 功能 | 將優惠卷輸入進資料庫 |
| **import requests**  **from bs4 import BeautifulSoup**  **import os**  **from urllib.request import urlretrieve**  **import pandas as pd**  ***#資料庫存檔***  **import mysql.connector**  **from insert\_mysql import prevent\_duplicate**  **from password import My\_password**  ***#建立資料庫連線***  **mydb = mysql.connector.connect(**  **user='root',**  **passwd=My\_password,**  **host='localhost',**  **database='mangerdb',**  **)**  **mycursor = mydb.cursor()**  **mycursor.execute('use mangerdb')**  **url = "https://www.4freeapp.com/"**  **respond = requests.get(url, headers={'User-agent': 'Mozilla/5.0'})**  **html = BeautifulSoup(respond.text)**  **main = html.find\_all("div", class\_="caption")**  ***# 準備空的 dataFrame***  **df = pd.DataFrame(columns=['id', 'title', 'class', 'content'])**  **df1 = pd.DataFrame(columns=['title', 'adress'])**  ***# 儲存網頁內所有內容***  **add\_String = ''**  **add\_Tag = ''**  **today\_judge = ''**  ***# 判斷檔名***  **def judge(x):**  **if "jpg" in x or "png" in x:**  **return True**  **else:**  **return False**  **for \_ in main:**  **index\_title = \_.find("a").string *# 這個會有\n***  **index\_url = \_.find("a").get("href")**  ***# 直接使用網站的title會有\n的問題，所以要先排除\n或\造成的錯誤("\"會造成註解)***  **Index\_Title = index\_title.split('\n')[1]**  ***# judge the url***  **today\_judge = index\_url**  **index\_respond = requests.get(index\_url, headers={'User-agent': 'Mozilla/5.0'})**  **index\_html = BeautifulSoup(index\_respond.text)**  **index\_main = index\_html.find\_all("a", attrs={'style': 'margin-left: 1em; margin-right: 1em;'})**  ***# 網頁內容***  **index\_content = index\_html.find\_all("span", attrs={'style': 'font-size: large;'})**  ***# 網頁 tag***  **index\_tag = index\_html.find\_all("div", class\_="widget-tags")**  ***# MySQL Myclass***  **for text in index\_tag:**  **print(Index\_Title)**  **text\_a = text.find\_all("a")**  **for str in text\_a:**  ***# 物件類別***  **class\_tab = str.string**  **web\_tag = class\_tab.split("\n")[1]**  **add\_Tag = add\_Tag + web\_tag + "/"**  ***# MySQL content***  **for text in index\_content:**  **if text.string is not None:**  ***# add\_text.append(text.string.splitlines())***  ***# splitlines將\n分開***  **add\_String = add\_String + text.string.splitlines()[0] + "\ "**  ***# MySQL ID***  **for i in index\_main:**  **downloadURL = i.get("href")**  **ds = downloadURL.split("/")**  ***##設定ID***  **coupon\_id = ds[5]**  ***#將資料輸入進資料庫***  **prevent\_duplicate(coupon\_id, Index\_Title.splitlines()[-1], add\_Tag, add\_String)**  ***#將資料存入DataFrame***  **s = pd.Series([coupon\_id, Index\_Title.splitlines()[-1], add\_Tag, add\_String],**  **index=['id', 'title', 'class', 'content'])**  **df = df.append(s, ignore\_index=True)**  ***# 重製儲存字串***  **add\_String = ''**  **add\_Tag = ''**  ***# 儲存圖片***  **for i in index\_main:**  ***# 設定資料夾路徑，怕標題有/***  **dname = "C:/Users/Jun/Desktop/coupon/" + Index\_Title.split("/")[0] + "/"**  **if not os.path.exists(dname):**  **os.mkdir(dname)**  **downloadURL = i.get("href")**  **print(downloadURL)**  **ds = downloadURL.split("/")**  **filetype = ds[-1].split(".")[-1]**  **judgeURL = filetype**  **if judgeURL:**  **fpath = dname + ds[5] + "." + filetype**  **urlretrieve(downloadURL, fpath)**  **s1 = pd.Series([Index\_Title.splitlines()[-1], fpath], index=['title', 'adress'])**  **df1 = df1.append(s1, ignore\_index=True)**  **print("判斷標準", today\_judge)**  **import datetime**  **now = datetime.datetime.now().strftime("%Y-%m-%d")**  **now = now.split("-")**  **begin = datetime.date(int(now[0]), int(now[1]), int(now[2]))**  **end = datetime.date(2019, 5, 11)**  **d = begin**  **delta = datetime.timedelta(days=1)**  **while d >= end:**  **print(d.strftime("%Y-%m-%d"))**  **url = "https://www.4freeapp.com/search?updated-max=" + d.strftime(**  **"%Y-%m-%d") + "T10%3A53%3A00%2B08%3A00&max-results=7#PageNo=2"**  **respond = requests.get(url, headers={'User-agent': 'Mozilla/5.0'})**  **html = BeautifulSoup(respond.text)**  **main = html.find\_all("div", class\_="caption")**  ***# 網頁內容***  **index\_content = index\_html.find\_all("span", attrs={'style': 'font-size: large;'})**  ***# 網頁 tag***  **index\_tag = index\_html.find\_all("div", class\_="widget-tags")**  **for times in reversed(main):**  **index\_title = times.find("a").string *# 這個會有\n***  **index\_url = times.find("a").get("href")**  **Index\_Title = index\_title.split('\n')[1] *##把indextitle的\n排除***  **if today\_judge == index\_url:**  **today\_judge = main[-1].find("a").get("href")**  **break**  **print("頁尾", main[-1].find("a").get("href"))**  **print("判斷標準", today\_judge)**  **print(Index\_Title)**  **index\_respond = requests.get(index\_url, headers={'User-agent': 'Mozilla/5.0'})**  **index\_html = BeautifulSoup(index\_respond.text)**  **index\_main = index\_html.find\_all("a", attrs={'style': 'margin-left: 1em; margin-right: 1em;'})**  ***# 網頁內容***  **index\_content = index\_html.find\_all("span", attrs={'style': 'font-size: large;'})**  ***# 將標題和內容存成CSV***  **for text in index\_content:**  **if text.string is not None:**  ***# add\_text.append(text.string.splitlines())***  ***# splitlines將\n分開***  **add\_String = add\_String + text.string.splitlines()[0] + "\ "**  ***# MySQL Myclass***  **for text in index\_tag:**  **print(Index\_Title)**  **text\_a = text.find\_all("a")**  **for str in text\_a:**  ***# 物件類別***  **class\_tab = str.string**  **web\_tag = class\_tab.split("\n")[1]**  **add\_Tag = add\_Tag + web\_tag + "/"**  ***# MySQL ID***  **for i in index\_main:**  **downloadURL = i.get("href")**  **ds = downloadURL.split("/")**  ***##設定ID***  **coupon\_id = ds[5]**  **print(coupon\_id)**  ***# 將資料輸入進資料庫***  **prevent\_duplicate(coupon\_id, Index\_Title.splitlines()[-1], add\_Tag, add\_String)**  ***#將資料存進DataFrame***  **s = pd.Series([coupon\_id, Index\_Title.splitlines()[-1], add\_Tag, add\_String],**  **index=['id', 'title', 'class', 'content'])**  **df = df.append(s, ignore\_index=True)**  ***# 重製儲存字串***  **add\_String = ''**  **add\_Tag = ''**  ***##儲存***  **for i in index\_main:**  **print(Index\_Title)**  ***# 設定資料夾路徑，怕標題有/***  **dname = "C:/Users/Jun/Desktop/coupon/" + Index\_Title.split("/")[0] + "/"**  **if not os.path.exists(dname):**  **os.mkdir(dname)**  **downloadURL = i.get("href")**  **print(downloadURL)**  **ds = downloadURL.split("/")**  **filetype = ds[-1].split(".")[-1]**  **if judge(filetype):**  **fpath = dname + ds[5] + "." + filetype**  **urlretrieve(downloadURL, fpath)**  **s1 = pd.Series([Index\_Title.splitlines()[-1], fpath], index=['title', 'adress'])**  **df1 = df1.append(s1, ignore\_index=True)**  **d -= delta**  **print(df)**  **print(df1)**  **df.to\_csv("title\_content.csv", encoding='utf-8', index=False)**  **df1.to\_csv("title\_address.csv", encoding='utf-8', index=False)** | |

表9-1-12 後端程式KFC\_crawler

|  |  |
| --- | --- |
| 程式名稱 | KFC\_crawler |
| 功能 | 爬蟲 |
| **import requests**  **from bs4 import BeautifulSoup**  **import shutil**  **import pandas as pd**  **from Log import ErrorWrite**  **import datetime**  **from PIL import Image**  **from Convert\_Img import convert\_Image, change\_Image\_to\_text**  ***# connect to web***  **url = "https://www.kfcclub.com.tw/tw/Coupon/"**  **respond = requests.get(url, headers={'User-agent': 'Mozilla/5.0'})**  **html = BeautifulSoup(respond.text)**  **main = html.find\_all('div', attrs={'style': 'cursor:pointer;text-align:center'})**  ***# df -> 'coupon\_id' , '標題' , '內文' , '注意事項'***  **df = pd.DataFrame(columns=['id', 'title', 'note', 'notice', 'price'])**  ***# log 檔案名稱***  **log\_file\_name = datetime.datetime.now().strftime('%Y-%m-%d-%H%M%S') + '.txt'**  ***# 計算現在為第幾筆***  **cnt = 1**  ***# get data by atr***  **for item in main:**  **img\_src = item.find("img")['src']**  **img\_name = item.find("img")['code']**  ***# 網頁內容***  **title = item.find("img")['t']**  **note = item.find\_all("p")**  **expire\_date = ""**  **for expire in note:**  **expire\_date = expire\_date + expire.text**  **notice = item.find("img")['onclick'].split(',')[3]**  ***# ------------寫入pd和檢查是否有空值--------------***  **if not img\_name or not title or not expire\_date or not notice:**  **ErrorWrite(log\_file\_name, "第{0}筆資料，欄位出現空值\n".format(cnt))**  ***# ------------儲存圖片--------------***  **r = requests.get(img\_src, stream=True, headers={'User-agent': 'Mozilla/5.0'}) *# stream=True 则会推迟响应内容的下载***  **img\_path = "C:/Users/case-pc/Desktop/KFC/" + img\_name + ".jpg"**  **if r.status\_code == 200:**  **with open(img\_path, 'wb') as f:**  **r.raw.decode\_content = True**  **shutil.copyfileobj(r.raw, f)**  ***# -------------辨識價格-----------------***  **img = Image.open(img\_path)**  ***# 剪裁圖片***  **img = img.crop((35, 70, 100, 105))**  **img = convert\_Image(img)**  **price = change\_Image\_to\_text(img)**  ***# ---------------------------儲存資料-----------------------***  **s = pd.Series([img\_name, title, expire\_date, notice, price],**  **index=['id', 'title', 'note', 'notice', 'price'])**  **df = df.append(s, ignore\_index=True)**  **cnt = cnt + 1**  **df.to\_excel("KFC.xlsx", encoding='utf-8', index=False)** | |

表9-1-13 後端程式-圖片辨識

|  |  |
| --- | --- |
| 程式名稱 | 圖片辨識 |
| 功能 | Convert\_Img |
| **import pytesseract**  **import re**  **from PIL import Image**  ***#去雜訊***  **def convert\_Image(img,standard=127.5):**  ***"""***  ***轉換成灰階***  ***"""***  **image = img.convert('L')**  **"""**  **二值化**  **-> 根據閥值standard , 將所有像素都置為 0(黑色) 或 255(白色), 便於接下來的分割**  **"""**  **pixels = image.load()**  **for x in range(image.width):**  **for y in range(image.height):**  **if pixels[x, y] > standard:**  **pixels[x, y] = 255**  **else:**  **pixels[x, y] = 0**  **return image**  ***#使用 pytesseract 庫來識別圖片中的字符***  **def change\_Image\_to\_text(img):**  ***""""***  ***如果出現 使用pytesseract出现错误：“[WinError 2] 系统找不到指定的文件***  ***則需要手動設定路徑，將 testdata\_dir\_config 和 tesseract\_cmd 設定成自己下載的tesseract.exe所在路徑***  ***tesseract.exe download -> https://github.com/UB-Mannheim/tesseract/wiki***  ***ref - > https://blog.csdn.net/jacke121/article/details/75443785***  ***"""***  **pytesseract.pytesseract.tesseract\_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'**  **testdata\_dir\_config = "C:\Program Files\Tesseract-OCR\tesseract.exe"**  **textCode = pytesseract.image\_to\_string(img, lang='eng', config=testdata\_dir\_config)**  ***# 去掉非法字符，只保留字母數字***  **textCode = re.sub("\W", "", textCode)**  **return textCode** | |

表9-1-14 後端程式Log

|  |  |
| --- | --- |
| 程式名稱 | Log |
| 功能 | 撰寫錯誤訊息 |
| **import os**  **def ErrorWrite(filename,content):**  **if os.path.exists(filename):**  **f = open(filename, 'a' , encoding='utf-8')**  **f.write(content)**  **f.close()**  **else:**  **f = open(filename, 'w' , encoding='utf-8')**  **f.write(content)**  **f.close()** | |

表9-1-15 後端程式Excel\_To\_SQL

|  |  |
| --- | --- |
| 程式名稱 | Excel\_To\_SQL |
| 功能 | 將Excel轉成SQL |
| **import pandas as pd**  **import mysql.connector**  **from password import My\_password**  **import re**  ***# 切割特殊符號，使內容只剩下中文，英文，數字***  ***# ref ----> https://www.jianshu.com/p/a5aa570d8173***  **def limit\_unicode(text):**  ***# 抓出不是中文 ， 英文 ，數字的符號***  **cop = re.compile("[^\u4e00-\u9fa5^a-z^A-Z^0-9^)^(^\uff0c^;^\^/^.^:^\u0020^-]")**  ***# 将抓到的符號替换成空白***  **result = cop.sub('', text)**  **return result**  **class Convert\_Excel\_To\_MySQL():**  **def \_\_init\_\_(self,file\_path):**  **self.file\_path = file\_path**  **self.df = pd.read\_excel(file\_path).dropna(thresh=3)**  **self.col = list(self.df.columns)**  **def Set\_DataSet(self):**  ***"""***  ***將 Excel的資料轉成Tuple***  ***"""***  **arr = []**  **for i in self.df.index:**  **row\_val = []**  **for col in self.col:**  **row\_val.append(limit\_unicode(str(self.df[col][i])))**  **arr.append(tuple(row\_val))**  **self.dataset = arr**  **def Connect\_To\_Sql(self, conn , db):**  ***"""***  ***連線到SQL***  **:param *conn: 連線進資料庫***  **:param *db: 資料庫名稱***  ***"""***  **self.\_conn = conn**  **self.\_db = db**  **self.\_mycursor = self.\_conn.cursor()**  **self.\_mycursor.execute('use ' + str(self.\_db))**  **def Get\_InsertSQL(self , tb ,colSet):**  ***"""***  ***取得SQL字串***  **:param *tb: 資料表名稱***  **:param *colSet: 欄位***  ***"""***  **sql = "insert into "+str(tb) + "("**  **sql\_value = ""**  **for db\_col in colSet:**  **sql = sql + " " + str(db\_col)**  **sql\_value += "%s "**  **if(db\_col != colSet[-1]):**  **sql += ","**  **sql\_value += ","**  **sql += ") value "**  **sql = sql + "( " + sql\_value + ")"**  **self.insert\_sql = sql**  **def Bulk\_Insert(self):**  **try:**  **self.\_mycursor.executemany(self.insert\_sql, self.dataset)**  **self.\_conn.commit()**  **except mysql.connector.Error as error:**  **print(error)**  **self.\_conn.rollback()**  **def Close\_Conn(self):**  **self.\_mycursor.close()**  **self.\_conn.close()**  ***#建立資料庫連線***  **mydb = mysql.connector.connect(**  **user='root',**  **passwd=My\_password,**  **host='127.0.0.1',**  **database='ntub\_coupon\_backend',**  **auth\_plugin='mysql\_native\_password'**  **)**  **My\_Excel = Convert\_Excel\_To\_MySQL("KFC1104.xlsx")**  ***#取得Excel資料***  **My\_Excel.Set\_DataSet()**  ***#連線進資料庫***  **My\_Excel.Connect\_To\_Sql(mydb,"ntub\_coupon\_backend")**  ***# 取得SQl字串***  **My\_Excel.Get\_InsertSQL("test\_token\_coupon" , ["coupon\_id","coupon\_title","coupon\_note",**  **"coupon\_notice","coupon\_price","coupon\_original\_price",**  **"coupon\_img" , "coupon\_class" ,"coupon\_create\_at","coupon\_saving"])**  ***# print(My\_Excel.df["original price"])***  **print(My\_Excel.dataset)**  **print(My\_Excel.insert\_sql)**  **My\_Excel.Bulk\_Insert()** | |

表9-1-16 後端程式insert\_mysql.py

|  |  |
| --- | --- |
| 程式名稱 | insert\_mysql.py |
| 功能 | 將資料輸入進資料庫裡 |
| **import mysql.connector**  **import datetime**  **class Set\_Mysql():**  **def \_\_init\_\_(self,\_id,\_title,\_class,\_content , \_conn , \_db):**  **self.\_id = \_id**  **self.\_title = \_title**  **self.\_class = \_class**  **self.\_content = \_content**  **self.\_conn = \_conn**  **self.\_db = \_db**  **def Connect\_To\_Sql(self):**  ***"""***  ***連線到資料庫***  ***"""***  **self.\_mycursor = self.\_conn.cursor()**  **self.\_mycursor.execute('use ' + str(self.\_db))**  **def prevent\_duplicate(self):**  ***"""***  ***輸入進資料庫***  ***"""***  **title\_data = (self.\_title,)**  **sql = "select \* from mangerdb\_item where title = %s"**  **self.\_mycursor.execute(sql, title\_data)**  **myresult = self.\_mycursor.fetchall()**  **if myresult:**  **print('重複的資料', 'id', id, '標題', title\_data[0])**  **else:**  **try:**  **insert\_sql = "insert ignore into mangerdb\_item (id , title ,Myclass , content) values (%s,%s,%s,%s)"**  **insert\_data = (self.\_id, self.\_title, self.\_class, self.\_content)**  **self.\_mycursor.execute(insert\_sql, insert\_data)**  **self.\_conn.commit()**  **except mysql.connector.Error as error:**  **self.\_conn.rollback()**  **finally:**  **if self.\_mycursor.rowcount:**  **print("資料成功輸入")**  **else:**  **time\_id = datetime.datetime.now().strftime("%Y%m%d%H%M%S")**  **insert\_data = (self.\_id, self.\_title, self.\_class, self.\_content)**  **self.\_mycursor.execute(insert\_sql, insert\_data)**  **self.\_conn.commit()**  **print(self.\_mycursor.rowcount, "record inserted.", "id改為時間參數")** | |

表9-1-17 前端程式showWalkthrough

|  |  |
| --- | --- |
| 程式名稱 | showWalkthrough |
| 功能 | 呼叫教學頁面 |
|  | |

表9-1-18 前端程式connectDataBase

|  |  |
| --- | --- |
| 程式名稱 | connectDataBase |
| 功能 | 連至sqlite |
|  | |

表9-1-19 前端程式pageViewController

|  |  |
| --- | --- |
| 程式名稱 | pageViewController |
| 功能 | 控制頁面轉換 |
| **func** pageViewController(**\_** pageViewController: UIPageViewController, viewControllerAfter viewController: UIViewController) -> UIViewController? {    **if** **let** index = list.index(of: viewController) , index < list.count - 1{  **return** list[index + 1 ]    }  **return** **nil**  }    **func** pageViewController(**\_** pageViewController: UIPageViewController, viewControllerBefore viewController: UIViewController) -> UIViewController? {    **if** **let** index = list.index(of: viewController) , index > 0{  **return** list[index - 1 ]    }  **return** **nil**  } | |

表9-1-20 前端程式check

|  |  |
| --- | --- |
| 程式名稱 | check |
| 功能 | 確認使用者有無登入 |
| **if** !userToken.bool(forKey: "fisrtCheck"){  **if** firstCheck == "0" && logoutCheck == **false**{  firstCheck = firstCheckStore  }**else** **if** firstCheck == "0"{  **self**.showAlertMessage(title: "發大財", message: "登入以開啟其餘功能")  }  }    // if !userToken.bool(forKey: "sqliteCheck"){  **if** secondCheck == **false**{  secondCheck = sqliteCheck  } | |

表9-1-21 前端程式instantiateViewController

|  |  |
| --- | --- |
| 程式名稱 | instantiateViewController |
| 功能 | 印出頁面 |
| **let** m0 = storyboard?.instantiateViewController(withIdentifier: "m0")  **let** m1 = storyboard?.instantiateViewController(withIdentifier: "m1")  **let** m2 = storyboard?.instantiateViewController(withIdentifier: "m2")    list.append(m0!)  list.append(m1!)  list.append(m2!)    setViewControllers([list[1]], direction: .forward, animated: **true**, completion: **nil**)  dataSource = **self** | |

表9-1-22 前端程式pageViewController

|  |  |
| --- | --- |
| 程式名稱 | pageViewController |
| 功能 | 裡頁控制頁面轉換 |
| **func** pageViewController(**\_** pageViewController: UIPageViewController, viewControllerAfter viewController: UIViewController) -> UIViewController? {    **if** **let** index = list.index(of: viewController) , index < list.count - 1{  **return** list[index + 1 ]    }  **return** **nil**  }  **func** pageViewController(**\_** pageViewController: UIPageViewController, viewControllerBefore viewController: UIViewController) -> UIViewController? {    **if** **let** index = list.index(of: viewController) , index > 0{  **return** list[index - 1 ]  }  **return** **nil**  } | |

表9-1-23 前端程式couponPage

|  |  |
| --- | --- |
| 程式名稱 | couponPage |
| 功能 | 連至CouponPage |
| **@IBAction** **func** couponPage(**\_** sender: **Any**) {  **self**.performSegue(withIdentifier: "goToPC", sender: **nil**)  }  **override** **func** prepare(for segue: UIStoryboardSegue, sender: **Any**?) {    **if** segue.identifier == "goToPC" {    **let** secondVC = segue.destination **as**! pageviewcontroller    secondVC.received = 2    }  } | |

表9-1-24 前端程式createModels

|  |  |
| --- | --- |
| 程式名稱 | createModels |
| 功能 | 製作piechart模型 |
| **func** createModels() -> [PieSliceModel] {    **var** models: [PieSliceModel] = []  connectDataBase()  **do** {  //MARK: **- get month data -**  **let** todaysDate = Calendar.current.date(byAdding: .month, value: 0, to: Date())  **let** monthFormatter = DateFormatter()  monthFormatter.dateFormat = "MM"  **let** monthString = monthFormatter.string(from: todaysDate!)  **let** monthdata = monthString  **if** secondCheck == **true**{  **let** lastMonthKind = usersTableAP  .filter(sqMonth == monthdata)  .select(**self**.sqPrice.sum,sqName).group(sqName)  **for** user **in** (**try**! databaseCon.prepare(lastMonthKind)) {    **let** randomColor = Int.random(in: 0...10)  **let** model =  PieSliceModel(  value:Double(user[sqPrice.sum]!)  ,color: colors[randomColor]  ,obj:String("\(user[sqName])")  )  models.append(model)  }  }  } **catch** {  print(error)  }    print(models)  **return** models  } | |

表9-1-25 前端程式createCustomViewsLayer

|  |  |
| --- | --- |
| 程式名稱 | createCustomViewsLayer |
| 功能 | 定義piechart模型數值 |
| **func** createCustomViewsLayer() -> PieCustomViewsLayer {  **let** viewLayer = PieCustomViewsLayer()  **let** settings = PieCustomViewsLayerSettings()  settings.viewRadius = 135  settings.hideOnOverflow = **true**  viewLayer.settings = settings    viewLayer.viewGenerator = createViewGenerator()  **return** viewLayer  } | |

表9-1-26 前端程式createPlainTextLayer

|  |  |
| --- | --- |
| 程式名稱 | createPlainTextLayer |
| 功能 | 定義piechart模型之文字 |
| func createPlainTextLayer() -> PiePlainTextLayer {  let textLayerSettings = PiePlainTextLayerSettings()  textLayerSettings.viewRadius = 90  textLayerSettings.hideOnOverflow = true  textLayerSettings.label.font = UIFont.systemFont(ofSize: 10)  let formatter = NumberFormatter()  formatter.maximumFractionDigits = 1  textLayerSettings.label.textGenerator = {slice in  return formatter.string(from: slice.data.percentage \* 100 as NSNumber).map{"\($0)%"} ?? ""  }  let textLayer = PiePlainTextLayer()  textLayer.settings = textLayerSettings  return textLayer  } | |

表9-1-27 前端程式createTextWithLinesLayer

|  |  |
| --- | --- |
| 程式名稱 | createTextWithLinesLayer |
| 功能 | 定義piechart模型之數字及連接線 |
| func createTextWithLinesLayer() -> PieLineTextLayer {  let lineTextLayer = PieLineTextLayer()  var lineTextLayerSettings = PieLineTextLayerSettings()  lineTextLayerSettings.lineColor = UIColor.lightGray  let formatter = NumberFormatter()  formatter.maximumFractionDigits = 1  lineTextLayerSettings.label.font = UIFont.systemFont(ofSize: 14)  lineTextLayerSettings.label.textGenerator = {slice in  return formatter.string(from: slice.data.model.value as NSNumber).map{"\($0)"} ?? ""  }  lineTextLayer.settings = lineTextLayerSettings  return lineTextLayer  } | |

表9-1-28 前端程式createViewGenerator

|  |  |
| --- | --- |
| 程式名稱 | createViewGenerator |
| 功能 | 創造piechart製造功能 |
| **func** createViewGenerator() -> (PieSlice, CGPoint) -> UIView {  **return** {slice, center **in**    **let** container = UIView()  container.frame.size = CGSize(width: 100, height: 40)  container.center = center  **let** view = UIImageView()  view.frame = CGRect(x: 30, y: 0, width: 40, height: 40)  container.addSubview(view)    **let** tranObj = slice.data.model.obj **as**! String    // if tran == "Food" || slice.data.id == 1 {  **let** specialTextLabel = UILabel()  specialTextLabel.textAlignment = .center    **if** tranObj == "Food"{  specialTextLabel.text = "Food"  }**else** **if** tranObj == "Other" {  specialTextLabel.text = "Other"  }**else** **if** tranObj == "Place" {  specialTextLabel.text = "Place"  }**else** **if** tranObj == "Sneak" {  specialTextLabel.text = "Sneak"  }**else** **if** tranObj == "Tools" {  specialTextLabel.text = "Tools"  }**else** **if** tranObj == "Traffic" {  specialTextLabel.text = "Traffic"  }**else** **if** tranObj == "Beverage" {  specialTextLabel.text = "Beverage"  }  specialTextLabel.sizeToFit()  specialTextLabel.font = UIFont.boldSystemFont(ofSize: 10)  specialTextLabel.frame = CGRect(x: 0, y: 40, width: 100, height: 20)  container.addSubview(specialTextLabel)  container.frame.size = CGSize(width: 100, height: 60)  // }  **return** container  }    } | |

表9-1-29 前端程式showAlertMessage

|  |  |
| --- | --- |
| 程式名稱 | showAlertMessage |
| 功能 | 呼喚警訊功能 |
| **func** showAlertMessage(title: String, message: String) {  **let** inputErrorAlert = UIAlertController(title: title, message: message, preferredStyle: .alert) //產生AlertController  **let** okAction = UIAlertAction(title: "確認", style: .default, handler: **nil**) // 產生確認按鍵  inputErrorAlert.addAction(okAction) // 將確認按鍵加入AlertController  **self**.present(inputErrorAlert, animated: **true**, completion: **nil**) // 顯示Alert  }  } | |

表9-1-30 前端程式BarsChartConfig

|  |  |
| --- | --- |
| 程式名稱 | BarsChartConfig |
| 功能 | 設定barchart數值 |
| **let** chartConfig = BarsChartConfig(  valsAxisConfig : ChartAxisConfig(from:0 , to:100 ,by:10)  )  **let** frame = CGRect(x : 50 , y : 50 ,width:**self**.view.frame.width/1.5,height:80)  **let** show = Double(showingNumber ?? 0)  **let** chart = BarsChart(  frame : frame,  chartConfig : chartConfig,  xTitle : "",  yTitle : "",  bars :[  ("",show)  ],  color:UIColor.init(displayP3Red: 1, green: 0.81, blue: 0.62, alpha: 1),  barWidth: 30  )  **self**.view.addSubview(chart.view)  **self**.chartView = chart | |

表9-1-31 前端程式buttonView

|  |  |
| --- | --- |
| 程式名稱 | buttonView |
| 功能 | 設定按鍵反饋 |
| **@IBOutlet** **weak** **var** buttonView: UIButton!  **@IBAction** **func** beginEdit(**\_** sender: **Any**) {    buttonView.flash()  buttonView.setImage(UIImage(named: "searchReload")?.withRenderingMode(.alwaysOriginal), for:[])  buttonCheck = **false**  }  **@IBAction** **func** endEdit(**\_** sender: **Any**) {  buttonView.flash()  buttonView.setImage(UIImage(named: "search")?.withRenderingMode(.alwaysOriginal), for:[])  buttonCheck = **true**  } | |

表9-1-32 前端程式submit

|  |  |
| --- | --- |
| 程式名稱 | submit |
| 功能 | 查詢記帳 |
| **@IBAction** **func** submit(**\_** sender: **Any**) {    **if** buttonCheck == **true** {    **if** searchBar.text != ""{    centerLeftList.removeAll()  tableView.reloadData()  **var** searchControll = usersTableAP.filter(sqMonth == "\(searchBar.text ?? "")")    **if** (searchBar.text?.count ?? 0 ) == 4{  searchControll = usersTableAP.filter(sqDate == "\(searchBar.text ?? "")")  }**else**{  searchControll = usersTableAP.filter(sqMonth == "\(searchBar.text ?? "")")  }  **do** {    **let** users = **try** **self**.database.prepare(searchControll)  **for** user **in** users {  print("userId: \(user[**self**.id]), date: \(user[**self**.sqDate]),name: \(user[**self**.sqName]),price: \(user[**self**.sqPrice]), month: \(user[sqMonth])")  **let** daily = DailyDetailForm(Date: "\(user[**self**.sqDate])", Name: " \(user[**self**.sqName])", Price: "\(user[**self**.sqPrice])", Coupon: " \(user[sqCoupon])",id: "\(user[**self**.id])",save:"-\(user[sqSave])")    centerLeftList.append(daily)  **let** indexPath = IndexPath(row: centerLeftList.count - 1,section :0)  tableView.insertRows(at:[indexPath], with: .automatic)  tableView.reloadData()  }  }**catch** {  print(error)  }  }**else**{    centerLeftList.removeAll()  tableView.reloadData()  **do** {  print("2")  **let** users = **try** **self**.database.prepare(usersTableAP)  **for** user **in** users {  **var** getSqCoupon = "\(user[sqCoupon])"  **var** getsqSave = "-\(user[sqSave])"  **if** "\(user[sqCoupon])" == ""{  getSqCoupon = "none"  getsqSave = ""  }  print("userId: \(user[**self**.id]), date: \(user[**self**.sqDate]),name: \(user[**self**.sqName]),price: \(user[**self**.sqPrice])")    **let** daily = DailyDetailForm(Date: "\(user[**self**.sqDate])", Name: " \(user[**self**.sqName])", Price: "\(user[**self**.sqPrice])", Coupon: getSqCoupon,id:"\(user[**self**.id])",save: getsqSave)    centerLeftList.append(daily)      **let** indexPath = IndexPath(row: centerLeftList.count - 1,section :0)  tableView.insertRows(at:[indexPath], with: .automatic)  tableView.reloadData()    }  }**catch** {  print(error)  }  }  }  //MARK: **--search cancel--**  **else**{    centerLeftList.removeAll()  tableView.reloadData()  searchBar.text = ""  **do** {  **let** users = **try** **self**.database.prepare(usersTableAP)  **for** user **in** users {  **var** getSqCoupon = "\(user[sqCoupon])"  **var** getsqSave = "-\(user[sqSave])"  **if** "\(user[sqCoupon])" == ""{  getSqCoupon = "none"  getsqSave = ""  }  print("userId: \(user[**self**.id]), date: \(user[**self**.sqDate]),name: \(user[**self**.sqName]),price: \(user[**self**.sqPrice])")    **let** daily = DailyDetailForm(Date: "\(user[**self**.sqDate])", Name: " \(user[**self**.sqName])", Price: "\(user[**self**.sqPrice])", Coupon: getSqCoupon,id: "\(user[**self**.id])",save: getsqSave)    centerLeftList.append(daily)      **let** indexPath = IndexPath(row: centerLeftList.count - 1,section :0)  tableView.beginUpdates()  tableView.insertRows(at:[indexPath], with: .automatic)  tableView.endUpdates()  }  }**catch** {  print(error)  }  searchBar.endEditing(**true**)  }  } | |

表9-1-33 前端程式set max lengh to text

|  |  |
| --- | --- |
| 程式名稱 | **set max lengh to text** |
| 功能 | 設定輸入長度 |
| //MARK: **- set max lengh to text -**  **func** textField(**\_** textField: UITextField, shouldChangeCharactersIn range: NSRange, replacementString string: String) -> Bool {  **guard** **let** textFieldText = searchBar.text,  **let** rangeOfTextToReplace = Range(range, in: textFieldText) **else** {  **return** **false**  }  **let** substringToReplace = textFieldText[rangeOfTextToReplace]  **let** count = textFieldText.count - substringToReplace.count + string.count  **return** count <= 4  } | |

表9-1-34 前端程式tableViewCell

|  |  |
| --- | --- |
| 程式名稱 | tableViewCell |
| 功能 | 設定cell數值 |
| **func** tableView(**\_** tableView: UITableView, heightForFootererInSection section: Int) -> CGFloat {  **return** cellSpacingHeight  }    **func** tableView(**\_** tableView: UITableView, viewForFootererInSection section: Int) -> UIView? {  **let** footerView = UIView()  footerView.backgroundColor = UIColor.clear  **return** footerView  }    **func** tableView(**\_** tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {  **let** headerView = UIView()  headerView.backgroundColor = UIColor.clear  **return** headerView  } | |

表9-1-35 前端程式CenterLeftCell

|  |  |
| --- | --- |
| 程式名稱 | CenterLeftCell |
| 功能 | 定義cell |
| **import** UIKit  **class** CenterLeftCell: UITableViewCell {  **@IBOutlet** **weak** **var** clDate: UILabel!  **@IBOutlet** **weak** **var** clName: UILabel!  **@IBOutlet** **weak** **var** clPrice: UILabel!  **@IBOutlet** **weak** **var** clCoupon: UILabel!  **@IBOutlet** **weak** **var** clSave: UILabel!    **func** setCenterLeftList(centerLeftList: DailyDetailForm){  clName.text = centerLeftList.Name  clPrice.text = centerLeftList.Price  clDate.text = centerLeftList.Date  clCoupon.text = centerLeftList.Coupon  clSave.text = centerLeftList.save  }  **override** **var** frame: CGRect {  **get** {  **return** **super**.frame  }  **set** {  **var** frame = newValue  //frame.origin.x += 15  //frame.size.width -= 2 \* 15  frame.origin.x = 0;//这里间距为10，可以根据自己的情况调整  frame.size.width -= 2 \* frame.origin.x;  frame.size.height = 55;  **super**.frame = frame  }  }  } | |

表9-1-36 前端程式textFieldAction

|  |  |
| --- | --- |
| 程式名稱 | textFieldAction |
| 功能 | searchBar功能 |
| **@IBAction** **func** textFieldAction(**\_** sender: **Any**) {  preview.textColor = UIColor.clear  **if** searchBar.text != ""{  models.removeAll()  pie.clear()  **do** {  **let** documentDirectory = **try** FileManager.default.url(for: .documentDirectory, in: .userDomainMask, appropriateFor: **nil**, create: **true**)  **let** fileUrl = documentDirectory.appendingPathComponent("users").appendingPathExtension("sqlite3")  **let** database = **try** Connection(fileUrl.path)  **self**.database = database  print("connected")      **let** thisMonthKind = usersTableAP  .filter(sqMonth == searchBar.text!)  .select(**self**.sqPrice.sum,sqName).group(sqName)  **let** users = **try** **self**.database.prepare(thisMonthKind)    **let** thisMonthSave = usersTableAP  .filter(sqMonth == searchBar.text!)  .select(sqSave.sum,sqMonth).group(sqMonth)  **let** usersSave = **try** **self**.database.prepare(thisMonthSave)    **for** user **in** users {    **let** randomColor = Int.random(in: 0...4)  **let** model = PieSliceModel(value:Double(user[sqPrice.sum]!), color: colors[randomColor],obj:String("\(user[sqName])"))    models.append(model)    }    **for** user **in** usersSave {    save.text! = "已省下 \(user[sqSave.sum]!)"    }    } **catch** {  print(error)  }  save.textColor = UIColor(red: 0.06, green: 0.23, blue: 0.31, alpha: 1)  monthNumber.textColor = UIColor(red: 0.06, green: 0.23, blue: 0.31, alpha: 1)    **if** searchBar.text! == "01"{  monthNumber.text! = "1月"  }**else** **if** searchBar.text! == "02"{  monthNumber.text! = "2月"  }**else** **if** searchBar.text! == "03"{  monthNumber.text! = "3月"  }**else** **if** searchBar.text! == "04"{  monthNumber.text! = "4月"  }**else** **if** searchBar.text! == "05"{  monthNumber.text! = "5月"  }**else** **if** searchBar.text! == "06"{  monthNumber.text! = "6月"  }**else** **if** searchBar.text! == "07"{  monthNumber.text! = "7月"  }**else** **if** searchBar.text! == "08"{  monthNumber.text! = "8月"  }**else** **if** searchBar.text! == "09"{  monthNumber.text! = "9月"  }**else** **if** searchBar.text! == "10"{  monthNumber.text! = "10月"  }**else** **if** searchBar.text! == "11"{  monthNumber.text! = "11月"  }**else** **if** searchBar.text! == "12"{  monthNumber.text! = "12月"  }  searchBar.text! = ""    }  viewDidLoad()  searchBar.endEditing(**true**)  } | |

表9-1-37 前端程式cancelPiechart

|  |  |
| --- | --- |
| 程式名稱 | cancelPiechart |
| 功能 | 取消searchBar |
| **func** cancelPiechart() -> [PieSliceModel]{  **do** {  **let** documentDirectory = **try** FileManager.default.url(for: .documentDirectory, in: .userDomainMask, appropriateFor: **nil**, create: **true**)  **let** fileUrl = documentDirectory.appendingPathComponent("users").appendingPathExtension("sqlite3")  **let** database = **try** Connection(fileUrl.path)  **self**.database = database  print("connected")    **let** todaysDate = Calendar.current.date(byAdding: .month, value: 0, to: Date())  **let** monthFormatter = DateFormatter()  monthFormatter.dateFormat = "MM"  **let** monthString = monthFormatter.string(from: todaysDate!)  **let** monthdata = monthString    **let** thisMonthKind = usersTableAP  .filter(sqMonth == monthdata)  .select(**self**.sqPrice.sum,sqName).group(sqName)  **let** users = **try** **self**.database.prepare(thisMonthKind)  **for** user **in** users {    **let** randomColor = Int.random(in: 0...4)  **let** model = PieSliceModel(value:Double(user[sqPrice.sum]!), color: colors[randomColor],obj:String("\(user[sqName])"))    models.append(model)    }    } **catch** {  print(error)  }  **return** models  } | |

表9-1-38 前端程式usersTable.create

|  |  |
| --- | --- |
| 程式名稱 | usersTable.create |
| 功能 | 創造table |
| **do** {  **let** documentDirectory = **try** FileManager.default.url(for: .documentDirectory, in: .userDomainMask, appropriateFor: **nil**, create: **true**)  **let** fileUrl = documentDirectory.appendingPathComponent("users").appendingPathExtension("sqlite3")  **let** database = **try** Connection(fileUrl.path)  **self**.database = database  } **catch** {  print(error)  }    print("CREATE TAPPED")    **let** createTable = **self**.usersTable.create { (table) **in**  table.column(**self**.id, primaryKey: **true**)  table.column(**self**.budget, unique: **true**)  }    **do** {  **try** **self**.database.run(createTable)  print("Created Table")  } **catch** {  print(error)  }    **let** insert = usersTable.insert(budget <- 0)  **if** **let** rowId = **try**? database.run(insert) {  print("插入成功：\(rowId)")  } **else** {  print("插入失败")  }    // let del = usersTable  **let** firstDel = usersTable.filter(id == 2)  **if** **let** count = **try**? **self**.database.run(firstDel.delete()) {  print("删除的条数为：\(count)")  } **else** {  print("删除失败")  }    print("LIST TAPPED")    **do** {  **let** users = **try** **self**.database.prepare(**self**.usersTable)  **for** user **in** users {  showTarget.text = "\((user[(**self**.budget)]))"  print("userId: \(user[**self**.id]), budget: \((user[**self**.budget]))")  }  } **catch** {  print(error)  }  } | |

表9-1-39 前端程式http

|  |  |
| --- | --- |
| 程式名稱 | http |
| 功能 | 連至資料庫 |
| // MARK: **- upload Post -**  request.setValue("JWT "+(Token **as**! String), forHTTPHeaderField: "Authorization")  **let** postData = [  "accounting\_class":"\(user[**self**.sqName])"  ,"accounting\_data":"\(user[**self**.sqPrice])"  ,"accounting\_date":"\(user[**self**.sqDate])"  ,"accounting\_month":"\(user[**self**.sqMonth])"  ,"accounting\_coupon\_name":"\(user[sqCoupon])"  ,"accounting\_discount":"\(user[sqSave])"  ]  **let** postString = postData.compactMap({ (key, value) -> String **in**  **return** "\(key)=\(value)"  }).joined(separator: "&")  request.httpBody = postString.data(using: .utf8)  // 后面不解释了，和GET的注释一样    **let** task = session.dataTask(with: request) {(data, response, error) **in**  **do** {  **let** r = **try** JSONSerialization.jsonObject(with: data!, options: JSONSerialization.ReadingOptions.mutableContainers) **as**! NSDictionary  print(r)  print("2")  } **catch** {  print("无法连接到服务器")  print("張詠峻喔")  **return**  }  }  task.resume()    arrCheck = postData  }  **let** del = usersTableAP    **if** **let** count = **try**? **self**.database.run(del.delete()) {  print("删除的条数为：\(count)")  } **else** {  print("删除失败")  }  } **catch** {  print(error)  }    showAlertMessage(title:"已成功上傳至雲端",message:"")  } | |

表9-1-40 前端程式list

|  |  |
| --- | --- |
| 程式名稱 | list |
| 功能 | 更新目標金額 |
| **@IBAction** **func** list(**\_** sender: **Any**) {  //button feedback  **let** generator = UINotificationFeedbackGenerator()  generator.notificationOccurred(.success)  print("UPDATE TAPPED")  **guard** **let** userIdString = update.text,  **let** budget = Int(update.text ?? "0")  **else** { **return** }  print(userIdString)  print(budget)  **let** user = **self**.usersTable.filter(**self**.id == 1)  **let** updateUser = user.update(**self**.budget <- budget)  **do** {  **try** **self**.database.run(updateUser)  } **catch** {  print(error)  }  **do** {  **let** users = **try** **self**.database.prepare(**self**.usersTable)  **for** user **in** users {  showTarget.text = "\((user[**self**.budget]))"  print("userId: \(user[**self**.id]), budget: \(user[**self**.budget])")  }  } **catch** {  print(error)  }  } | |

表9-1-41 前端程式download

|  |  |
| --- | --- |
| 程式名稱 | download |
| 功能 | 存雲端下載資料 |
| **@IBAction** **func** download(**\_** sender: **Any**) {  // 创建一个会话，这个会话可以复用  **let** session = URLSession(configuration: .default)  // 设置URL  **var** request = URLRequest(url:URL(string:"https://"+UrlId+".ngrok.io/useraccouting/")!)  // 创建一个网络任务  // if userToken.bool(forKey: "Token") == true{  request.setValue("JWT "+(Token **as**! String), forHTTPHeaderField: "Authorization")  // }  request.httpMethod = "GET"  **let** task = session.dataTask(with: request) {(data, response, error) **in**  **do** {  connectDataBase()  **let** del = usersTableAP    **if** **let** count = **try**? **self**.database.run(del.delete()) {  print("删除的条数为：\(count)")  } **else** {  print("删除失败")  }    **let** r = **try** JSONSerialization.jsonObject(with: data!, options:JSONSerialization.ReadingOptions.mutableContainers)    print(r)  **let** rArray = r **as**! NSArray    **for** i **in** rArray{    **let** download = i **as**! NSDictionary  **let** sqDateInsert = "\(download["accounting\_date"] ?? "")"  **let** sqNameInsert = "\(download["accounting\_class"] ?? "")"  **let** sqPriceInsert = "\(download["accounting\_data"] ?? "")"  **let** sqPriceFinal = Int(sqPriceInsert) ?? 0  **let** sqMonthInsert = "\(download["accounting\_month"] ?? "")"  **let** sqCouponInsert = "\(download["accounting\_coupon\_name"] ?? "")"  **let** sqSaveInsert = "\(download["accounting\_discount"] ?? "")"  **let** sqSaveFinal = Int(sqSaveInsert) ?? 0    **let** insertUser = usersTableAP.insert(**self**.sqDate <- sqDateInsert ,**self**.sqName <- sqNameInsert, **self**.sqPrice <- sqPriceFinal,**self**.sqMonth <- sqMonthInsert,sqCoupon <- sqCouponInsert, sqSave <- sqSaveFinal,sqDiscount <- sqPriceFinal)  **do** {  **try** **self**.database.run(insertUser)  } **catch** {  print(error)  }  print(sqDateInsert,sqNameInsert,sqPriceInsert,sqMonthInsert,sqSaveFinal)  print(type(of: sqDateInsert),type(of: sqNameInsert),type(of: sqPriceInsert),type(of: sqMonthInsert),type(of: sqSaveFinal))  }  }**catch** {  // 如果连接失败就...  print("无法连接到服务器download")  **let** content = UNMutableNotificationContent()  content.title = "發大財"  // content.subtitle = "subtitle："  content.body = "請先登入"  content.badge = 0  content.sound = UNNotificationSound.default  **let** trigger = UNTimeIntervalNotificationTrigger(timeInterval: 1, repeats: **false**)  **let** request = UNNotificationRequest(identifier: "notification", content: content, trigger: trigger)  UNUserNotificationCenter.current().add(request, withCompletionHandler: {error **in**  print("建立通知on")  })  **return**  }  }  task.resume()  showAlertMessage(title:"已成功下傳至手機",message:"")  } | |

表9-1-42 前端程式dailyNotification

|  |  |
| --- | --- |
| 程式名稱 | dailyNotification |
| 功能 | 定義通知 |
| **@IBAction** **func** dailyNotification(**\_** sender: **Any**) {    **let** content = UNMutableNotificationContent()  content.title = "發大財"  // content.subtitle = "subtitle："  content.body = "記一筆？ 台灣發大財關心您"  content.badge = 0  content.sound = UNNotificationSound.default    **let** content2 = UNMutableNotificationContent()  content2.title = "發大財"  // content.subtitle = "subtitle："  content2.body = "關提醒了 別忘記記帳！ 省錢生活關心您"  content2.badge = 0  content2.sound = UNNotificationSound.default    // let date = Date(timeIntervalSinceNow: 36000)  // let triggerDaily = Calendar.current.dateComponents([.hour, .minute, .second], from: date)  // let trigger = UNCalendarNotificationTrigger(dateMatching: triggerDaily, repeats: true)  **let** triggerOn = UNTimeIntervalNotificationTrigger(timeInterval: 60, repeats: **true**)  **let** triggerOff = UNTimeIntervalNotificationTrigger(timeInterval: 1, repeats: **false**)    **if** (sender **as** AnyObject).isOn {  controll = triggerOn  controllContent = content  **let** request = UNNotificationRequest(identifier: "notification", content: controllContent, trigger: controll)  UNUserNotificationCenter.current().add(request, withCompletionHandler: {error **in**  print("建立通知on")  })  userNotification.set(alertCheck, forKey: "userNotificationValue")  alertCheck = **false**  }**else**{  controll = triggerOff  controllContent = content2  **let** request = UNNotificationRequest(identifier: "notification", content: controllContent, trigger: controll)  UNUserNotificationCenter.current().add(request, withCompletionHandler: {error **in**  print("建立通知off")  })  userNotification.set(alertCheck, forKey: "userNotificationValue")  alertCheck = **true**  }  alertImageControll()    }  // MARK:**- Notification View image -**    **func** alertImageControll(){    **let** noticifationValue = userNotification.value(forKey: "userNotificationValue")    **if** noticifationValue **as**! Bool? == **true**{  alertView.image = UIImage(named: "alertOn")  }**else**{  alertView.image = UIImage(named: "alertOff")    }  } | |

表9-1-43 前端程式Post

|  |  |
| --- | --- |
| 程式名稱 | **Post** |
| 功能 | 連至資料庫 |
| //MARK: **- Post -**  // 这个session可以使用刚才创建的。  **let** session = URLSession(configuration: .default)  // 设置URL  **var** request = URLRequest(url:URL(string:"https://"+UrlId+".ngrok.io/api-token-auth/")!)  request.httpMethod = "POST"  request.setValue("application/x-www-form-urlencoded", forHTTPHeaderField: "Content-Type")    // 设置要post的内容，字典格式    **let** postData = ["username":"\(String(username.text!))"  ,"password":"\(String(password.text!))"  ]  **let** postString = postData.compactMap({ (key, value) -> String **in**  **return** "\(key)=\(value)"  }).joined(separator: "&")  request.httpBody = postString.data(using: .utf8)  print("3")  **let** task = session.dataTask(with: request) {(data, response, error) **in**  **do** {  **let** jsonString = NSString(data: data!, encoding: String.Encoding.utf8.rawValue)! **as** String  print(jsonString)  **let** r = **try** JSONSerialization.jsonObject(with: data!, options: JSONSerialization.ReadingOptions.allowFragments) **as**! NSDictionary  print("2")    **let** rToken = r["token"]**as**! String  //MARK: **- store the Token -**  **self**.showAlertMessage(title:"登入成功",message:"")  userToken.set(rToken , forKey: "Token")  userToken.synchronize()  // if r["non\_field\_errors"] != nil{  // let rerror = r["non\_field\_errors"]as! String  // self.showAlertMessage(title:"登入失敗",message:rerror)  // }  } **catch** {  print("无法连接到服务器")  print("張詠峻喔")  print(error)  **self**.showAlertMessage(title:"登入失敗",message:"\(error)")  **return**  }  }  task.resume()  print("done") | |

表9-1-44 前端程式logOut

|  |  |
| --- | --- |
| 程式名稱 | logOut |
| 功能 | 登出 |
| **@IBAction** **func** logOut(**\_** sender: **Any**) {  userToken.set("" , forKey: "Token")  userToken.set("" , forKey: "userId")  userToken.set("", forKey: "userPassword")  showAlertMessage(title:"登出成功",message:"")  firstCheck = "0"  userToken.set(0, forKey: "firstCheck")  userToken.synchronize()  username.text = ""  password.text = ""  logoutCheck = **true**  } | |

表9-1-45 前端程式screenEdgeSwiped

|  |  |
| --- | --- |
| 程式名稱 | screenEdgeSwiped |
| 功能 | 右滑返回 |
| **@objc** **func** screenEdgeSwiped(**\_** recognizer: UIScreenEdgePanGestureRecognizer) {  **if** recognizer.state == .recognized {  **let** vc = (storyboard?.instantiateViewController(withIdentifier: "pageController"))  vc?.modalPresentationStyle = .fullScreen  vc?.modalTransitionStyle = .crossDissolve  present(vc!, animated: **true** )  }  }  } | |

表9-1-46 前端程式pickerView

|  |  |
| --- | --- |
| 程式名稱 | pickerView |
| 功能 | 設定選取器 |
| **@IBOutlet** **weak** **var** pickerView: UIPickerView!    **@IBOutlet** **weak** **var** tableView: UITableView!  **let** types = ["Tools","Traffic","Beverage","Place","Food","Sneak","Other"]  **func** numberOfComponents(in pickerView: UIPickerView) -> Int {  **return** 1  }    **func** pickerView(**\_** pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int {  **return** types.count  }    **func** pickerView(**\_** pickerView: UIPickerView, titleForRow row: Int, forComponent component: Int) -> String? {  **return** types[row]  }      **func** pickerView(**\_** pickerView: UIPickerView, didSelectRow row: Int, inComponent component: Int) {  typeShow = types[row]  }  **func** pickerView(**\_** pickerView: UIPickerView, attributedTitleForRow row: Int, forComponent component: Int) -> NSAttributedString? {  **return** NSAttributedString(string:types[row] , attributes: [NSAttributedString.Key.foregroundColor: UIColor.black])  } | |

表9-1-47 前端程式getDateValue

|  |  |
| --- | --- |
| 程式名稱 | getDateValue |
| 功能 | 定義選取器 |
| **var** datedata : String!  **var** monthdata: String!    **@IBOutlet** **weak** **var** getDateValue: UIDatePicker!  **@IBAction** **func** dateAction(**\_** sender: UIDatePicker) {  **let** dateValue = DateFormatter()  dateValue.dateFormat = "MMdd" // 設定要顯示在Text Field的日期時間格式  datedata = dateValue.string(from: getDateValue.date) // 更新Text Field的內容  // getDateValue.setValue(UIColor.black, forKeyPath: "textColor") 轉動時變顏色    **let** monthValue = DateFormatter()  monthValue.dateFormat = "MM"  monthdata = monthValue.string(from: getDateValue.date)  } | |

表9-1-48 前端程式numbers

|  |  |
| --- | --- |
| 程式名稱 | numbers |
| 功能 | 記帳的計算器 |
| **@IBAction** **func** numbers(**\_** sender: UIButton) {  **let** inputNumber=sender.tag-1  **if** total.text != **nil** {  **if** total.text == "0"||total.text == "+"||total.text == "-"{  total.text = "\(inputNumber)"  }**else**{  total.text = total.text!+"\(inputNumber)"  }  }  numberOnScreen = Double(total.text!) ?? 0  }      **@IBAction** **func** clean(**\_** sender: UIButton) {  total.text = "0"  numberOnScreen = 0  previousNumber = 0  performingMath = **false**  operation = .none  startNew = **true**  }      **@IBAction** **func** add(**\_** sender: UIButton) {  total.text = "+"  previousNumber = numberOnScreen  performingMath = **true**  operation = .add  }    **@IBAction** **func** substract(**\_** sender: UIButton) {  total.text = "-"  previousNumber = numberOnScreen  performingMath = **true**  operation = .substract  }      **@IBAction** **func** answer(**\_** sender: **Any**) {  **if** performingMath == **true**{  **switch** operation{  **case** .add:  numberOnScreen = previousNumber+numberOnScreen  makeOkNumberString(from: numberOnScreen)  **case** .substract:  numberOnScreen = previousNumber-numberOnScreen  makeOkNumberString(from: numberOnScreen)  **case** .none:  total.text = "0"  }  performingMath = **false**  startNew = **true**  }  }    **@IBAction** **func** deleteNum(**\_** sender: **Any**) {  **if** total.text?.count == 1{  total.text = "0"  }**else**{  numberOnScreen = Double(total.text!.dropLast())!  makeOkNumberString(from: numberOnScreen)  }  performingMath = **true**  startNew = **false**  }      **func** makeOkNumberString(from number:Double) {  **if** floor(number) == number{  total.text = "\(Int(number))"  }**else**{  total.text = "\(number)"  }  } | |

表9-1-49 前端程式initiate

|  |  |
| --- | --- |
| 程式名稱 | **initiate** |
| 功能 | 初始化選取器 |
| // MARK: **- ----initiate the date to today**  **let** todaysDate = Date()  **let** dateFormatter = DateFormatter()  dateFormatter.dateFormat = "MMdd"  **let** dateString = dateFormatter.string(from: todaysDate)  datedata = dateString    **let** monthFormatter = DateFormatter()  monthFormatter.dateFormat = "MM"  **let** monthString = monthFormatter.string(from: todaysDate)  monthdata = monthString    // MARK: **- ----initiate the picker value**  pickerView.selectRow(0, inComponent: 0, animated: **true**) //initiate the value of picker    typeShow = "Tools"  selectedBackgroundView.backgroundColor = UIColor.clear    dailyDetailForm = createArray() | |

表9-1-50 前端程式createTable

|  |  |
| --- | --- |
| 程式名稱 | createTable |
| 功能 | 創造sqlite的記帳table |
| **let** createTable = usersTableAP.create { (table) **in**  table.column(**self**.id, primaryKey: **true**)  table.column(**self**.sqDate, unique: **false**)  table.column(**self**.sqName, unique: **false**)  table.column(**self**.sqPrice, unique: **false**)  table.column(**self**.sqCoupon, unique: **false**)  table.column(**self**.sqDiscount, unique: **false**)  table.column(**self**.sqMonth, unique: **false**)  table.column(**self**.sqSave, unique: **false**)    }  userToken.set(**true**, forKey: "sqliteCheck")  secondCheck = **true** | |

表9-1-51 前端程式usersTableAP

|  |  |
| --- | --- |
| 程式名稱 | usersTableAP |
| 功能 | 讀取sqlite並印出 |
| **do** {  **let** listQuery = usersTableAP.order(sqDate,sqName)  **let** users = **try** **self**.database.prepare(listQuery)  **for** user **in** users {  **var** getSqCoupon = "\(user[**self**.sqCoupon])"  **var** getSqSave = "-\(user[sqSave])"  **if** "\(user[**self**.sqCoupon])" == ""{  getSqCoupon = "none"  getSqSave = ""  }    print("userId: \(user[**self**.id]), date: \(user[**self**.sqDate]),name: \(user[**self**.sqName]),price: \(user[**self**.sqPrice]),coupon:\(user[**self**.sqCoupon]),month: \(user[**self**.sqMonth])")    **let** daily = DailyDetailForm(Date: "\(user[**self**.sqDate])", Name: " \(user[**self**.sqName])", Price: "\(user[**self**.sqPrice])", Coupon: getSqCoupon,id:"\(user[**self**.id])",save:getSqSave)    dailyDetailForm.append(daily)    **let** indexPath = IndexPath(row: dailyDetailForm.count - 1,section :0)  tableView.beginUpdates()  tableView.insertRows(at:[indexPath], with: .automatic)  tableView.endUpdates()    }    **let** query = usersTableAP.select(**self**.sqPrice.sum,sqName)  **for** user **in** (**try**? database?.prepare(query))!! {  **if** user[**self**.sqPrice.sum] ?? 0 > 0 {  print("\(user[**self**.sqPrice.sum]! )")  }**else**{  print("0")  }  }    } **catch** {  print(error)  }  // MARK: **-------swipe------**  **let** edgePan = UIScreenEdgePanGestureRecognizer(target: **self**, action: **#selector**(screenEdgeSwiped))  edgePan.edges = .right    view.addGestureRecognizer(edgePan)  } | |

表9-1-52 前端程式dailyDetailFormId

|  |  |
| --- | --- |
| 程式名稱 | dailyDetailFormId |
| 功能 | Sqlite刪除功能 |
| **let** dailyDetailFormId = Int(dailyDetailForm[indexPath.row-1].id) ?? 0  print(dailyDetailFormId)  **let** firstDel = usersTableAP.filter(id == dailyDetailFormId)  **if** **let** count = **try**? **self**.database.run(firstDel.delete()) {  print("删除的条数为：\(count)")  } **else** {  print("删除失败")  } | |

表9-1-53 前端程式insert

|  |  |
| --- | --- |
| 程式名稱 | insert |
| 功能 | Sqlite insert功能 |
| **@IBAction** **func** insert(**\_** sender: UIButton) {  print(cellData)  insertNewRow()  tableView.scrollToBottomRow()  total.text = "0"  total.textColor = UIColor(red: 0.048, green: 0.18, blue: 0.235, alpha: 1)  resetTheCouponValue()  }  **func** insertNewRow() {    print("INSERT TAPPED")  **let** getCouponTitle:String? = couponTitle  **var** getCouponDiscount:Int? = 0  **var** getCouponSave:Int? = 0  **if** couponDiscount != "0"{  getCouponDiscount = Int("\(couponDiscount)")  getCouponSave = Int("\(couponSave)")  }  **guard** **let** sqDate = datedata,  **let** sqName = typeShow,  **let** sqPrice = Int(total.text ?? "0"),  **let** sqMonth = monthdata,  **let** sqCoupon = getCouponTitle,  **let** sqDiscount = getCouponDiscount,  **let** sqSave = getCouponSave  **else** { **return** }    **let** insertUser = usersTableAP.insert(**self**.sqDate <- sqDate  , **self**.sqName <- sqName  , **self**.sqPrice <- sqPrice  , **self**.sqCoupon <- sqCoupon  , **self**.sqDiscount <- sqDiscount  , **self**.sqMonth <- sqMonth  , **self**.sqSave <- sqSave)    **do** {  **try** **self**.database.run(insertUser)  **let** dailyInsert = DailyDetailForm(Date: "\(sqDate)", Name: " \(sqName)", Price: "\(sqPrice)", Coupon: "none",id:"0",save:"")  **let** dailyInsertWithCoupon = DailyDetailForm(Date: "\(sqDate)", Name: " \(sqName)", Price: "\(sqPrice)", Coupon: "\(sqCoupon)",id:"0",save: "-\(sqSave)")    **if** couponDiscount == "0"{  dailyDetailForm.append(dailyInsert)  }**else**{  dailyDetailForm.append(dailyInsertWithCoupon)  }    **let** indexPath = IndexPath(row: dailyDetailForm.count - 1,section :0)  tableView.beginUpdates()  tableView.insertRows(at:[indexPath], with: .automatic)  tableView.endUpdates()  } **catch** {  print(error)  }  **let** generator = UINotificationFeedbackGenerator()  generator.notificationOccurred(.success)  } | |

表9-1-54 前端程式tableViewCELL

|  |  |
| --- | --- |
| 程式名稱 | tableViewCELL |
| 功能 | 設定CELL數值 |
| **func** tableView(**\_** tableView: UITableView, numberOfRowsInSection section: Int) -> Int {  **return** dailyDetailForm.count  }    **func** tableView(**\_** tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  **let** dailyDetailForms = dailyDetailForm[indexPath.row]    **let** cell = tableView.dequeueReusableCell(withIdentifier: "DailyDetailCell") **as**! DailyDetailCell      cell.setDailyDetailForm(dailyDetailForm: dailyDetailForms)  cell.selectedBackgroundView = selectedBackgroundView    // note that indexPath.section is used rather than indexPath.row  //cell.textLabel?.text = self.animals[indexPath.section]    // add border and color  // cell.contentView.transform = CGAffineTransform(rotationAngle: .pi)  cell.backgroundColor = UIColor.init(white: 1, alpha: 0.75)  cell.layer.shadowColor = UIColor.lightGray.cgColor  cell.layer.masksToBounds = **false**  cell.layer.shadowOpacity = 0.75  cell.layer.shadowOffset = CGSize(width: 5, height: 5)  cell.layer.shadowRadius = 5  cell.layer.cornerRadius = 12  cell.clipsToBounds = **false**  cell.id.text = dailyDetailForms.id    **return** cell    }  } | |

表9-1-55 前端程式createPlainTextLayer

|  |  |
| --- | --- |
| 程式名稱 | createPlainTextLayer |
| 功能 | 連至sqlite |
| **func** scrollToBottomRow() {  DispatchQueue.main.async {  **guard** **self**.numberOfSections > 0 **else** { **return** }  // Make an attempt to use the bottom-most section with at least one row  **var** section = max(**self**.numberOfSections - 1, 0)  **var** row = max(**self**.numberOfRows(inSection: section) - 1, 0)  **var** indexPath = IndexPath(row: row, section: section) // Ensure the index path is valid, otherwise use the section above (sections can  // contain 0 rows which leads to an invalid index path)  **while** !**self**.indexPathIsValid(indexPath) {  section = max(section - 1, 0)  row = max(**self**.numberOfRows(inSection: section) - 1, 0)  indexPath = IndexPath(row: row, section: section)  // If we're down to the last section, attempt to use the first row  **if** indexPath.section == 0 {  indexPath = IndexPath(row: 0, section: 0)  **break**  }  }  // In the case that [0, 0] is valid (perhaps no data source?), ensure we don't encounter an  // exception here  **guard** **self**.indexPathIsValid(indexPath) **else** { **return** }  **self**.scrollToRow(at: indexPath, at: .bottom, animated: **true**)  }  }  **func** indexPathIsValid(**\_** indexPath: IndexPath) -> Bool {  **let** section = indexPath.section  **let** row = indexPath.row  **return** section < **self**.numberOfSections && row < **self**.numberOfRows(inSection: section)  }  } | |

表9-1-56 前端程式DailyDetailForm

|  |  |
| --- | --- |
| 程式名稱 | DailyDetailForm |
| 功能 | 物件化 |
| **class** DailyDetailForm {    **var** Date : String  **var** Name : String  **var** Price : String  **var** Coupon : String  **var** id: String  **var** save :String    **init**(Date: String, Name: String, Price: String, Coupon: String,id:String,save:String){  **self**.Date = Date  **self**.Name = Name  **self**.Price = Price  **self**.Coupon = Coupon  **self**.id = id  **self**.save = save  }  } | |

表9-1-57 前端程式get

|  |  |
| --- | --- |
| 程式名稱 | get |
| 功能 | 連至資料庫 |
| **func** get(){  DispatchQueue.main.async {  // 创建一个会话，这个会话可以复用  **let** session = URLSession(configuration: .default)  // 设置URL  **var** request = URLRequest(url:URL(string:"https://"+UrlId+".ngrok.io/coupon/?page=1")!)  // 创建一个网络任务  // if userToken.bool(forKey: "Token") == true{  request.setValue("JWT "+String(Token **as**! String) ?? "", forHTTPHeaderField: "Authorization")  // }  request.httpMethod = "GET"  **let** task = session.dataTask(with: request) {(data, response, error) **in**  **do** {  // 返回的是一个json，将返回的json转成字典r  **var** r = **try** JSONSerialization.jsonObject(with: data!, options:JSONSerialization.ReadingOptions.mutableContainers) **as**! NSDictionary    **let** jsonString = NSString(data: data!, encoding: String.Encoding.utf8.rawValue)! **as** String  **let** rArray = (r["results"]**as**! NSArray)    **for** i **in** rArray{    **let** couponBook = i **as**! NSDictionary  // print(couponBook["coupon\_class"] ?? "")  **let** coupon1 = CouponArrayFormat(Store: "\(couponBook["coupon\_title"]!)", Content:"\(couponBook["coupon\_note"]!)",Kind:"\(couponBook["coupon\_class"]!)", url:"\(couponBook["coupon\_img"]!)",discount:"\(couponBook["coupon\_price"]!)",save:"\(couponBook["coupon\_saving"]!)")  **self**.couponList.append(coupon1)  **self**.tableView.reloadData()  print("\(couponBook["coupon\_class"]!)","\(couponBook["coupon\_class"]!)","\(couponBook["coupon\_class"]!)","\(couponBook["coupon\_img"]!)")  }  }**catch** {  // 如果连接失败就...  print("无法连接到服务器coupon")    DispatchQueue.main.async {  **self**.showAlertMessage(title: "發大財", message: "請重新登入")  }  **return**  }  }  // 运行此任务  task.resume()  }  } | |

表9-1-58 前端程式searchBoutton

|  |  |
| --- | --- |
| 程式名稱 | searchBoutton |
| 功能 | searchBar按鍵設定 |
| **@IBAction** **func** searchBoutton(**\_** sender: **Any**) {    DispatchQueue.main.async {    **self**.search = **self**.searchBar.text!    // 创建一个会话，这个会话可以复用  **let** session = URLSession(configuration: .default)  // 设置URL  **if** **self**.searchBar.text != ""{    **var** request = URLRequest(url:URL(string:"https://"+UrlId+".ngrok.io/coupon/?search=\(**self**.search)")!)  // 创建一个网络任务  // if userToken.bool(forKey: "Token") == true{  request.setValue("JWT "+String(Token **as**! String) ?? "", forHTTPHeaderField: "Authorization")  // }  request.httpMethod = "GET"  **let** task = session.dataTask(with: request) {(data, response, error) **in**  **do** {  // 返回的是一个json，将返回的json转成字典r  **var** r = **try** JSONSerialization.jsonObject(with: data!, options:JSONSerialization.ReadingOptions.mutableContainers) **as**! NSDictionary    **let** jsonString = NSString(data: data!, encoding: String.Encoding.utf8.rawValue)! **as** String    **let** rArray = (r["results"]**as**! NSArray)    **for** i **in** rArray{    **let** couponBook = i **as**! NSDictionary  // print(couponBook["coupon\_class"] ?? "")  **let** coupon1 = CouponArrayFormat(Store: "\(couponBook["coupon\_title"]!)", Content:"\(couponBook["coupon\_note"]!)",Kind:"\(couponBook["coupon\_class"]!)", url:"\(couponBook["coupon\_img"]!)",discount:"\(couponBook["coupon\_price"]!)",save:"\(couponBook["coupon\_saving"]!)")    **self**.couponList.append(coupon1)    **self**.tableView.reloadData()  print("\(couponBook["coupon\_class"]!)","\(couponBook["coupon\_class"]!)","\(couponBook["coupon\_class"]!)","\(couponBook["coupon\_img"]!)")  }  }**catch** {  // 如果连接失败就...  print("无法连接到服务器coupon")  **return**  }    }    // 运行此任务  task.resume()    }**else**{  **self**.searchBar.endEditing(**true**)  }  }  } | |

表9-1-59 前端程式tableViewCEllCoupon

|  |  |
| --- | --- |
| 程式名稱 | tableViewCEllCoupon |
| 功能 | 設定cell數值 |
| **func** tableView(**\_** tableView: UITableView, numberOfRowsInSection section: Int) -> Int {  **return** couponList.count  }    **func** tableView(**\_** tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  **let** couponLists = couponList[indexPath.row]  **let** cell = tableView.dequeueReusableCell(withIdentifier: "CouponCellFormat") **as**! CouponCellFormat  cell.setCoupon(coupon: couponLists)  cell.selectedBackgroundView = selectedBackgroundView  // note that indexPath.section is used rather than indexPath.row  //cell.textLabel?.text = self.animals[indexPath.section]  // add border and color  cell.backgroundColor = UIColor.init(white: 1, alpha: 0.75)  cell.layer.shadowColor = UIColor.lightGray.cgColor  cell.layer.masksToBounds = **false**  cell.layer.shadowOpacity = 0.75  cell.layer.shadowOffset = CGSize(width: 5, height: 5)  cell.layer.shadowRadius = 5  cell.layer.cornerRadius = 12  cell.clipsToBounds = **false**  cell.url.text = couponLists.url  cell.discount.text = couponLists.discount  cell.Store.text = couponLists.Store  cell.save.text = couponLists.save    **return** cell  } | |

表9-1-60 前端程式CouponArrayFormat

|  |  |
| --- | --- |
| 程式名稱 | CouponArrayFormat |
| 功能 | 物件化 |
| **import** Foundation  **import** UIKit  **class** CouponArrayFormat {    **var** Store : String  **var** Content : String  **var** Kind : String  **var** url: String  **var** discount : String  **var** save:String    **init**(Store: String, Content: String, Kind: String,url:String,discount:String,save:String){  **self**.Store = Store  **self**.Content = Content  **self**.Kind = Kind  **self**.url = url  **self**.discount = discount  **self**.save = save  }  } | |

表9-1-61 前端程式CouponCellFormat

|  |  |
| --- | --- |
| 程式名稱 | CouponCellFormat |
| 功能 | 協定 |
| **class** CouponCellFormat: UITableViewCell {  **@IBOutlet** **weak** **var** Store: UILabel!  **@IBOutlet** **weak** **var** Content: UILabel!  **@IBOutlet** **weak** **var** Kind: UILabel!  **@IBOutlet** **weak** **var** url: UILabel!  **@IBOutlet** **weak** **var** discount: UILabel!  **@IBOutlet** **weak** **var** save: UILabel!    **func** setCoupon(coupon: CouponArrayFormat){  Store.text = coupon.Store  Content.text = coupon.Content  Kind.text = coupon.Kind  url.text = coupon.url  discount.text = coupon.discount  save.text = coupon.save  }      **override** **var** frame: CGRect {  **get** {  **return** **super**.frame  }  **set** {  **var** frame = newValue  //frame.origin.x += 15  //frame.size.width -= 2 \* 15  frame.origin.x = 10;//这里间距为10，可以根据自己的情况调整  frame.size.width -= 2 \* frame.origin.x;  frame.size.height -= 1 \* frame.origin.x;  **super**.frame = frame  }  }    } | |

表9-1-62 前端程式WebView

|  |  |
| --- | --- |
| 程式名稱 | WebView |
| 功能 | Coupon內頁 |
| **class** WebView: UIViewController {    **var** receivedUrl:String = ""    **@IBAction** **func** nextPage(**\_** sender: **Any**) {  **self**.performSegue(withIdentifier: "goToPC", sender: **nil**)  }    **override** **func** prepare(for segue: UIStoryboardSegue, sender: **Any**?) {  **if** segue.identifier == "goToPC" {  **let** secondVC = segue.destination **as**! pageviewcontroller  secondVC.received = 2  }  }    **@IBAction** **func** back(**\_** sender: **Any**) {  resetTheCouponValue()  }    **@IBOutlet** **weak** **var** WebView: WKWebView!    **override** **func** viewDidLoad() {  **super**.viewDidLoad()  **let** url = URL(string: couponListUrl)  **let** request = URLRequest(url :url!)    WebView.load(request)  } | |

表9-1-63 前端程式UIButton

|  |  |
| --- | --- |
| 程式名稱 | UIButton |
| 功能 | 定義按鍵反饋 |
| **extension** UIButton {    **func** pulsate() {  **let** pulse = CASpringAnimation(keyPath: "transform.scale")  pulse.duration = 0.6  pulse.fromValue = 0.95  pulse.toValue = 1  pulse.autoreverses = **true**  pulse.repeatCount = 2  pulse.initialVelocity = 0.5  pulse.damping = 1.0    layer.add(pulse, forKey: **nil**)  }  **func** flash() {  **let** flash = CABasicAnimation(keyPath: "opacity")  flash.duration = 0.75  flash.fromValue = 0  flash.toValue = 1  flash.timingFunction = CAMediaTimingFunction(name: CAMediaTimingFunctionName.easeInEaseOut  layer.add(flash, forKey: **nil**)  }  **func** shake() {  **let** shake = CABasicAnimation(keyPath: "position")  shake.duration = 0.1  shake.repeatCount = 2  shake.autoreverses = **true**    **let** fromPoint = CGPoint(x: center.x - 5, y: center.y)  **let** fromValue = NSValue(cgPoint: fromPoint)    **let** toPoint = CGPoint(x: center.x + 5, y: center.y)  **let** toValue = NSValue(cgPoint: toPoint)    shake.fromValue = fromValue  shake.toValue = toValue    layer.add(shake, forKey: **nil**)    }  } | |

**第十章：測試模型**

**10-1 測試計畫**

測試方式分別以使用者的角度實際操作系統之各個功能是否正常運作，下表為各項測試功能以及對應的測試項目。

表10-1-1功能列表

|  |  |
| --- | --- |
| 功能 | 測試項目 |
| 教學 | 測試UI是否有正常出現。 |
| 註冊 | 測試註冊是否正常。 |
| email | 測試是否出現驗證信。 |
| 登入 | 測試登入是否是否正常。 |
| 設定目標金額 | 測試輸入與目標金額是否一致。 |
| 提醒鍵 | 測試是否在訊息框出現提醒訊息。 |
| 記帳 | 測試記帳功能。 |
| Coupon | 測試優惠券顯示是否正常以及與記帳功能連結是否正常。 |
| 上傳 | 測試是否成功上傳使用者資料至雲端 |
| 下載 | 測試雲端資料是否正常下載至手機並執行。 |
| 查詢 | 測試是否可以查詢。 |

**10-2 測試個案與測試結果資料**

表10-2-1 教學

|  |  |  |  |
| --- | --- | --- | --- |
| 測試項目 | 教學 | 測試結果是否成功 | 是 |
| 測試畫面 | | | |
|  | |  | |

表10-2-2 註冊

|  |  |  |  |
| --- | --- | --- | --- |
| 測試項目 | 註冊 | 測試結果是否成功 | 是 |
| 測試畫面 | | | |
|  | |  | |

表10-2-3 email

|  |  |  |  |
| --- | --- | --- | --- |
| 測試項目 | email | 測試結果是否成功 | 是 |
| 測試畫面 | | | |
|  | | | |

表10-2-4 登入

|  |  |  |  |
| --- | --- | --- | --- |
| 測試項目 | 登入 | 測試結果是否成功 | 是 |
| 測試畫面 | | | |
|  | |  | |

表10-2-5設定目標金額

|  |  |  |  |
| --- | --- | --- | --- |
| 測試項目 | 設定目標金額 | 測試結果是否成功 | 是 |
| 測試畫面 | | | |
|  | |  | |

表10-2-6提醒鍵

|  |  |  |  |
| --- | --- | --- | --- |
| 測試項目 | 提醒鍵 | 測試結果是否成功 | 是 |
| 測試畫面 | | | |
|  | |  | |

表10-2-7 記帳

|  |  |  |  |
| --- | --- | --- | --- |
| 測試項目 | 記帳 | 測試結果是否成功 | 是 |
| 測試畫面 | | | |
|  | |  | |

表10-2-8 Coupon

|  |  |  |  |
| --- | --- | --- | --- |
| 測試項目 | Coupon | 測試結果是否成功 | 是 |
| 測試畫面 | | | |
|  | | | |

表10-2-9 上傳

|  |  |  |  |
| --- | --- | --- | --- |
| 測試項目 | 上傳 | 測試結果是否成功 | 是 |
| 測試畫面 | | | |
|  | |  | |

表10-2-10 下載

|  |  |  |  |
| --- | --- | --- | --- |
| 測試項目 | 下載 | 測試結果是否成功 | 是 |
| 測試畫面 | | | |
|  | |  | |

表10-2-11 查詢

|  |  |  |  |
| --- | --- | --- | --- |
| 測試項目 | 查詢 | 測試結果是否成功 | 是 |
| 測試畫面 | | | |
|  | | | |

**第十一章：操作手冊**

**11-1 系統文件**

|  |  |
| --- | --- |
| 系統元件表 | |
| 系統元件 | 省錢生活 |
| 軟體類別 | APP |
| 支援語言 | 中文 |
| 價格 | 免費 |
| 最低版本需求 | iOS 10以上版本 |
| 權限 | 網路設定 |
| 接受網路資料 |

**11-2 系統管理**

|  |  |
| --- | --- |
| 環境資料 | 依照優惠券官方資料 |
| 畫面設計 | 根據使用者回饋評論優化使用者介面 |
| 可選資訊 | 定期增加近期流行之優惠券 |

**第十二章：使用手冊**

|  |  |
| --- | --- |
| Step1 進入APP | 說明 |
|  | 登入APP即出現教學訊息，往左滑即可了解如何使用這個APP。 |
| Step2 註冊與登入與驗證email | 說明 |
|  | 註冊並填寫使用者資料，提供使用者GOOGLE信箱做驗證，登入後即可使用其他功能。 |
| Step3 Coupon | 說明 |
|  | 顯示優惠券店家名稱及折扣金額，點擊後即可使用優惠券並到記帳畫面直接紀錄於使用者資料。 |
| Step4 其他功能 | 說明 |
|  | 搜尋月份，顯示記帳資料和各月份圖表，並可上傳至雲端提供換機下載功能，並提供使用者提醒功能。 |

**第十三章：感想**

楊秉憲:

這次的專題讓我從零到有得去學習前端APP開發，從看不懂swift語言到能夠自己做出一個專題，經過一次又一次的查閱網路上的歷史資料，使我有辦法慢慢摸清楚程式開發的過程，也因為這次專題的經歷，使我第一次感覺有做專案的感覺，前端的建構元設計，API對接以各種layout的刻畫設計，以及套件的使用讓我對專案又了解了一點，也使我對即將進入的資訊界有一定的了解。

黃政源:

一開始發想很多主題，但可行性太低或市面上已經有這些功能而打消念頭，中間也考慮過要不要換題目，後來再權宜選擇之下選擇了現在這個主題。

開始著手這個主題，本來由詠俊負責的爬蟲部分改由我負責，還有幫忙秉憲定義物件，這時才知道自己的能力不足，對於程式能力的不足在這時候完全放大，需要大量的搜尋資料及同學的幫忙，有了這次的專題，提升了程式能力也發現自己的渺小，需要一直的進修才能跟上科技進步。

這一次專題最困難的應該就是時程的規劃，組員一起討論的時間甚至個人接觸專題的時間，一開始沒有規劃好，想著未來還有很長的時間可以完成，因為我想要跨考研究所，幾乎很多時間都在補習班，還有組員們都要去實習上班，結果進度落後不少，只能利用瑣碎時間完成專題，透過這次的專題大大瞭解時間規劃的重要，能盡早完成的事要盡早完成、盡早決定，決定跨考以及專題時間嚴重重疊，真的心力交瘁，這個教訓在未來的人生一定有很大的幫助。

值得慶幸的是，大家合作的過程中都沒有發生任何間隙，很感謝組員的幫忙跟老師的指導，這次專題才能順利完成。

張詠俊:

經過這次專題，讓我從頭學起該如何爬蟲，以及如何建置API，對於API接口有更進一步的了解，而也因為這次的經驗，了解到了語言的重要性，許多文件和資料都是英文，所以對於寫程式而言最重要的還是英文，而經過這次專題，慢慢地學會讓自己習慣查找外國文件，以及當遇到bug的時候該怎麼解決，而這也是第一次在學校以來，有完整製作一個專案的感覺，也慢慢了解有關於後端的架構，以及資料流等等，算是對於未來的工作有初步的認識，也期望自己將來可以擁有更多經驗，去慢慢補足自己不足的地方。

陳冠都:

經過這次的專題之後，才體會到開發一個APP需要多少的創意，還要兼具很好的執行力，才可以成功完成一個APP，也體認到了自己能力的不足，很多地方都需要老師和組員的教導才能順利完成，也很感謝老師和組員很有耐心的指導我，才能順利完成自己的任務，組員們也給了我很多信心，把他們認為我擅長的工作放心的教給我，我也必須負起自己的責任將大家交給我的任務給做到好!

**第十四章：參考資料**

**14-1 參考元件**

前端\_swift

Swift -- <https://github.com/apple/swift>

主要撰寫語言，用於製作iOS系統

Pie chart --<https://github.com/i-schuetz/PieCharts/commits/master>

Easy to use and highly customizable pie charts library for iOS

Swift 4.2, iOS 8+

Swift chart --<https://github.com/i-schuetz/SwiftCharts>

Easy to use and highly customizable charts library for iOS

SQLite -- <https://github.com/stephencelis/SQLite.swift#notification-settings>

A type-safe, [Swift](https://swift.org/)-language layer over [SQLite3](http://www.sqlite.org/).

[SQLite.swift](https://github.com/stephencelis/SQLite.swift) provides compile-time confidence in SQL statement syntax *and* intent.

後端\_django

Django -- <https://github.com/django/django>

後端撰寫語言，用來製作API串接於前端

MySQL -- <https://dev.mysql.com/doc/refman/8.0/en/charset-metadata.html>

主要資料庫，用於存放資料

Django rest framework -- <https://www.django-rest-framework.org/>

主要API框架，用來撰寫API

Django-Filter -- <https://github.com/carltongibson/django-filter>

Django-filter is a reusable Django application allowing users to declaratively add dynamic QuerySet filtering from URL parameters.

Django-templated-mail -- <https://github.com/sunscrapers/django-templated-mail>

A simple wrapper for django.core.mail.EmailMultiAlternatives which makes use of Django template system to store email content in separate file.

Djangorestframework-jwt -- <https://github.com/jpadilla/django-rest-framework-jwt>

使用JWT進行拿取資料

mysqlclient -- <https://pypi.org/project/mysqlclient/>

連接Django和MySQL

PyMySQL -- <https://github.com/PyMySQL/PyMySQL>

連接Django和MySQL

Pillow -- <https://pillow.readthedocs.io/en/stable/>

Pillow is the friendly PIL fork by [Alex Clark and Contributors](https://github.com/python-pillow/Pillow/graphs/contributors). PIL is the Python Imaging Library by Fredrik Lundh and Contributors.

Pandas -- <https://pandas.pydata.org/>

處理和過濾爬下了的資料

requests -- <https://pypi.org/project/requests/2.7.0/>

爬蟲所需套件

Beautiful Soup -- <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

爬蟲所需套件

urllib3 -- <https://urllib3.readthedocs.io/en/latest/>

爬蟲所需套件

Openpyxl -- <https://openpyxl.readthedocs.io/en/stable/>

Excel轉檔套件

OpenCV -- <https://opencv.org/>

影像辨識用套件

**14-2 參考資料**

<https://hackmd.io/dWVWPt_aSjWbq_9OHFYMKQ>

<https://hackmd.io/UXzC0EWYSo2fAfqEq37LXQ>

<https://hackmd.io/jyMDtbDdQ6Spjgtl6Wv9Hw>

<https://www.smartm.com.tw/article/33353932cea3>

附錄

**附錄一 評審建議事項之修正情形**

附錄表:評審建議及修正情況

|  |  |
| --- | --- |
| 第一位評審   1. 主題不夠明確，優惠卷與記帳功能相關性不夠 2. 分工宜再強化 3. UI design需根據使用慣例調整 4. 資料庫設計不完整   第二位評審   1. 應參考已有的市場優惠卷，增加未被開發的機制 2. 參考Line、大賀票卷、玉山PI卡 3. 如何完整收集all優惠卷，市面已有 4. 介面應美化 5. 功能應以圖式菜單來供user使用   第三位評審   1. 分工有問題 2. 沒有人做美工 3. 進度延遲 4. 兩位同學讓我感覺跟這個專題無關 5. Github沒有每週上傳 6. 建議專題名稱要聚焦 7. 動機太弱，不夠強 8. 聚焦支出與coupon的Mapping 9. 女性：31-45歲的TA不夠行 10. Coupon的資料要能拿得到   第四位評審   1. 主題應修改為省錢… 2. 優惠券與記帳的差異為何 3. Github缺會議記錄 4. 表3-1沒資料 5. 每章的節次若只有一節，就不需要特地方一個節次出來，如：6-1。 6. 有些表格裡的資料不完整 | 已修正   1. 分工重新調整，每個人都接觸程式。 2. UI design重新配置，色調及介面盡量達到初次使用之使用者需求。 3. 資料庫已修正。 4. 蒐集市面上熱門優惠券。 5. 與特定店家協議優惠券。 6. 專題名稱更正為省錢生活。 7. 已增加會議紀錄 8. 已修正專題文件內容。 |

**附錄二 會議記錄**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **會議名稱** | **第一次專題會議** | | | |
| **時間** | **2019/9/9** | | **地點** | **圖書館-研討室二** |
| **主席** | **唐日新老師** | | **紀錄** | **陳冠都** |
| **參與人員** | **108408全體組員** | | | |
| **會議內容** | | | | |
| **此次會議之討論內容：**   * **討論專題時程**   發現原本的主題不夠有特別性，所以修改了原本的目標。  決定找尋商家合作擁有獨特的優惠券。 | | | | |
| **指導老師** | | **組長** | | **紀錄者** |
| **唐日新老師** | | **陳冠都** | | **陳冠都** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **會議名稱** | **第二次專題會議** | | | |
| **時間** | **2019/09/23** | | **地點** | **圖書館-研討室二** |
| **主席** | **唐日新老師** | | **紀錄** | **陳冠都** |
| **參與人員** | **108408全體組員** | | | |
| **會議內容** | | | | |
| **第一次會議之討論內容：**   * 討論專題時程   發現原本的主題不夠有特別性，所以修改了原本的目標。  **此次會議之討論內容：**   * 討論專題時程   老師建議我們修改logo的顏色、紀錄coupon省下的金額、找尋商家合作。  老師希望未來規畫要有辦法搜尋折價的排序、結合外送跟網拍。  10/7前完成爬蟲、分析、logo修改、找尋商家、APP圖表。 | | | | |
| **指導老師** | | **組長** | | **紀錄者** |
| **唐日新老師** | | **陳冠都** | | **陳冠都** |
| **會議名稱** | **第三次專題會議** | | | |
| **時間** | **2019/10/29** | | **地點** | **研討室三** |
| **主席** | **唐日新老師** | | **紀錄** | **陳冠都** |
| **參與人員** | **108408全體組員** | | | |
| **會議內容** | | | | |
| **第二次會議之討論內容：**   * 討論專題時程   老師建議我們修改logo的顏色、紀錄coupon省下的金額、找尋商家合作。  老師希望未來規畫要有辦法搜尋折價的排序、結合外送跟網拍。  **此次會議之討論內容：**   * 討論專題時程   目前進度只缺了紀錄coupon省下的金額、海報、文件、連資料庫。  專題名稱最終修改為省錢生活。  要在11/15之前完成交給老師檢查。 | | | | |
| **指導老師** | | **組長** | | **紀錄者** |
| **唐日新老師** | | **陳冠都** | | **陳冠都** |