午餐系統及分析

作者:吳邦寧、白翔云

[壹、研究動機 3](#_Toc531694770)

[貳、研究目的 3](#_Toc531694771)

[參、實驗器材及設備 3](#_Toc531694772)

[肆、研究過程及方式 4](#_Toc531694773)

[一、資料庫 4](#_Toc531694774)

[1. 點單查詢 5](#_Toc531694775)

[2. Procedure優化 5](#_Toc531694776)

[二、後端 5](#_Toc531694777)

[(一)、處理流程 6](#_Toc531694778)

[(二)、效能阻塞 6](#_Toc531694779)

[1. session 阻塞 6](#_Toc531694780)

[2. sql效能低下 7](#_Toc531694781)

[3. 硬碟存取阻塞 7](#_Toc531694782)

[(三)、安全性 7](#_Toc531694783)

[1. 密碼安全性 7](#_Toc531694784)

[2. SQL注入 7](#_Toc531694785)

[3. XSS 7](#_Toc531694786)

[4. 行政安全性 7](#_Toc531694787)

[5. 資料庫死結 8](#_Toc531694788)

[三、前端 8](#_Toc531694789)

[(一)、iOS前端 8](#_Toc531694790)

[1. 呼叫後端 8](#_Toc531694791)

[2. 頁面圖 8](#_Toc531694792)

[3.錯誤分析 9](#_Toc531694793)

[4.推播功能 9](#_Toc531694794)

[(二)、網頁前端 9](#_Toc531694795)

[1. ajax 9](#_Toc531694796)

[2. 分類樹 10](#_Toc531694797)

[3. 半耦合式後端 10](#_Toc531694798)

[(三)、輔助外掛 10](#_Toc531694799)

[1. 當日會計報表 10](#_Toc531694800)

[2. 新生資料匯入軟體 10](#_Toc531694801)

[(四)、前後端交換介面 10](#_Toc531694802)

[1. 輸出資料結構 10](#_Toc531694803)

[2. 命令介面 11](#_Toc531694804)

[3. JSON與編碼 11](#_Toc531694805)

[四、預測模型 11](#_Toc531694806)

[(一)、資料結構 11](#_Toc531694807)

[1. 點餐序列 12](#_Toc531694808)

[2. 點餐圖 12](#_Toc531694809)

[3. 點餐矩陣 12](#_Toc531694810)

[(一)、宏觀模型 12](#_Toc531694811)

[(二)、微觀模型 13](#_Toc531694812)

[(三)、模型應用 13](#_Toc531694813)

[五、系統使用方法 14](#_Toc531694814)

[(一)、系統流程 14](#_Toc531694815)

[(二)、學生使用方法 14](#_Toc531694816)

[(三)、代訂使用方法 14](#_Toc531694817)

[(四)、合作社使用方法 14](#_Toc531694818)

[(五)、廠商使用方法 14](#_Toc531694819)

[伍、目前研究結果 15](#_Toc531694820)

[一、使用心得 15](#_Toc531694821)

[1. 廠商、合作社 15](#_Toc531694822)

[2. 學生、代訂 15](#_Toc531694823)

[二、分析資料 15](#_Toc531694824)

[(一)、宏觀模型預測實例 15](#_Toc531694825)

[(二)、微觀模型預測實例 15](#_Toc531694826)

[(三)、預估明日應準備多少原料 15](#_Toc531694827)

[陸、參考資料及其他 15](#_Toc531694828)

# 壹、研究動機

建立一個可供廠商、消費者使用的平台

# 貳、研究目的

可追蹤特定目標族群

可預測明日之點餐數量

與產品經理討論並開發所需功能

# 參、實驗器材及設備

# 肆、研究過程及方式

　　午餐系統後端由php作為後台，mysql作為資料庫，並且輸出結果到前後端交換介面；凡是能夠操作前後端交換介面者，都可以稱呼為一個前台，包括網頁版前台、iOS前台及其他外掛插件，都可稱為一個前台。

午餐系統與大多數POS系統的不同之處在於我們擁有非常良好的可擴充性，如同電子布告欄「BBS」，任何人都能實作自己的前端，只不過我們採用JSON作為資料交換介面，而BBS採用Telnet協定。

午餐系統每天都會有大量的點餐資料，如果能對這些資料進行分析，就能夠協助廠商預測明天的餐點量，於是我們設計了「宏觀模型」以及「微觀模型」。

## 一、系統使用方法

以下是使用午餐系統的流程，點單資料會跟著箭頭傳給其他人使用。

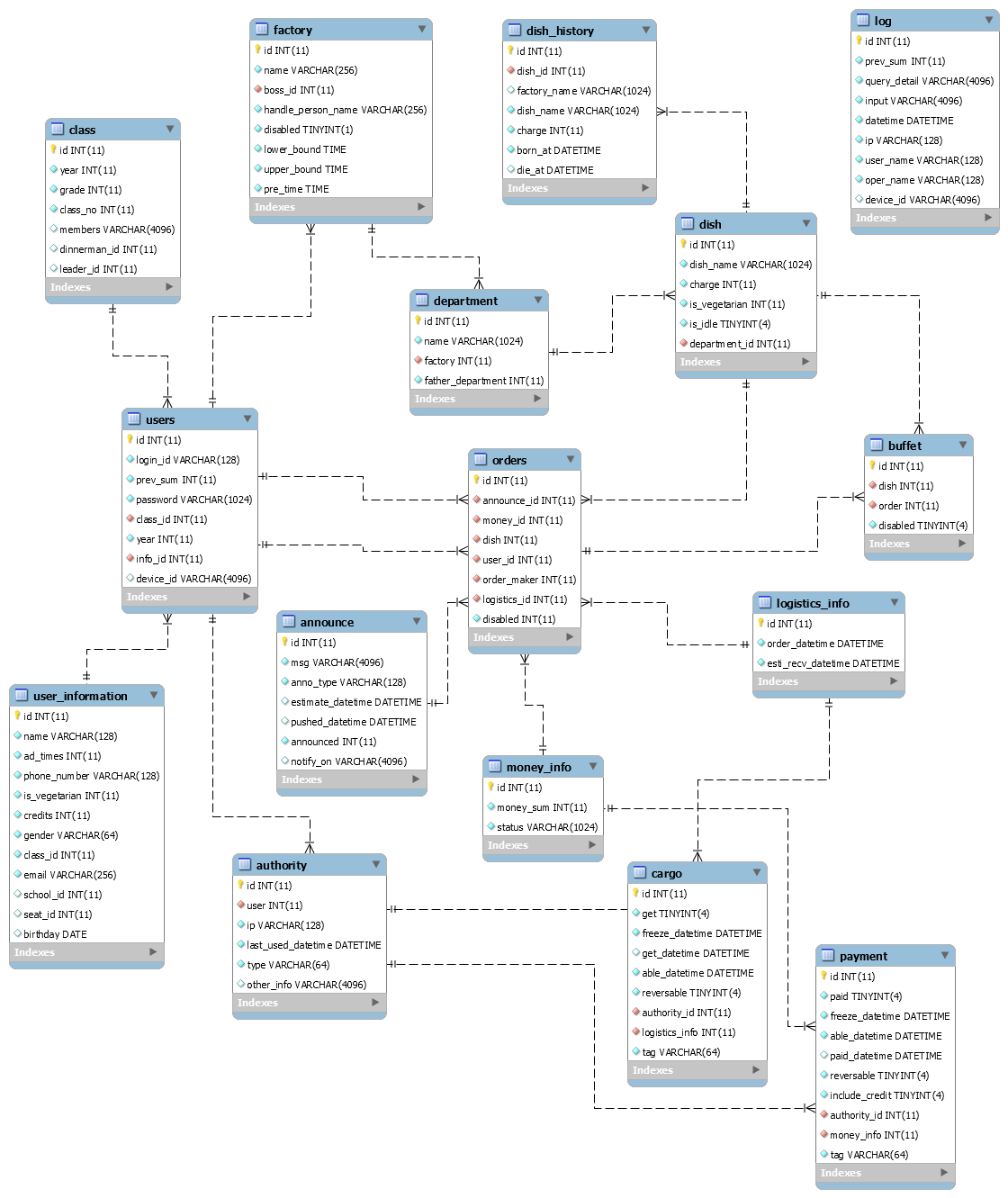
### (一)、學生、代訂使用方法

|  |  |  |  |
| --- | --- | --- | --- |
|  | 點餐 | 未付款 | 已付款 |
| 學生 |  |  |  |
| 代訂 |  |  |  |

### (二)、合作社、廠商使用方法

## 二、資料庫

　　下列為午餐系統的資料結構模型，每一條線代表一個關聯性連接。



午餐系統的資料庫為關聯性資料庫，並使用innoDB做為引擎，innoDB支援交易機制，比起myisam，innoDB更方便處理死結回溯的問題。

### (一)、動態查詢

後台會根據傳入參數，使用這六種篩選語句*(syntax)*組合成所需的篩選條件，向資料庫下達SQL指令。

|  |  |
| --- | --- |
| 時間下界 | AND (? < LO.esti\_recv\_datetime) |
| 時間上界 | AND (? > LO.esti\_recv\_datetime) |
| 針對特定使用者搜尋 | AND (? = U.id) |
| 針對某間廠商進行搜尋 | AND (? = F.id) |
| 針對特定列查詢 | AND (? = O.id) |
| 針對班級查詢 | AND ((SELECT U.class\_id FROM users AS U WHERE U.id = ?) = U.class\_id) |

在下達sql指令時，可以使用代稱*(alias)*來方便撰寫sql語句，以下是篩選語句的解釋。

|  |  |  |
| --- | --- | --- |
| 中文名稱 | 英文全名 | 英文縮寫 |
| 使用者 | User | U |
| 物流資訊 | Logistic info | LO |
| 廠商 | Factory | F |
| 點單 | Order | O |

### (二)、Procedure優化

對於單語句*(syntax)*的sql操作，直接在php中呼叫資料庫即可；對於多語句的sql操作，則包裝成一個Procedure，方便處理死結(Deadlock)回溯(Rollback)，也不必與資料庫伺服器多次連線。

下表為該Procedure進行四種基本操作(insert/update/delete/select)的數量，包裝的操作越多，越能節省與資料庫連線的時間。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Insert | Update | Delete | Select | 總計 |
| Make order | 6 | 0 | 0 | 9 | 15 |
| Make payment | 0 | 1 | 0 | 1 | 2 |
| Login | 0 | 1 | 0 | 2 | 3 |
| Update dish | 1 | 2 | 0 | 1 | 4 |

## 三、後端

下圖為後端的架構，我們將後端分成多個模組，方便未來維護，或是開發額外的新功能。

### (一)、處理流程

　　以進行點餐操作為例，系統會先進行嚴密的輸入審查，確認是否無誤後，再將資料寫入資料庫，資料庫回傳點餐id，系統抓出該筆資料，經過編碼之後輸出到交換介面，以下為處理的流程圖。

### (二)、效能阻塞

#### 1. session 阻塞

　　php為了保證執行緒安全，同一個session同時只能給一個請求使用。在每個請求都不會提前釋放session的狀況下，同時送出大量請求，會使得效能非常低落，如下圖所示。

上圖中，請求(3)等候前面兩個請求處理完才開始執行，成為緩慢的串行命令。如果程式會先複製好session再執行，則每個請求只需要等候其他請求複製完資料，就能夠先開始執行了，如下圖所示。

　　上圖中，請求(3)僅等候前面兩個請求複製資料，因為不必等候其他請求，因此能受益於CPU的平行處理，使得效能提升。

#### 2. 資料庫存取阻塞

　　資料庫的存取速度遠遠低於記憶體的存取速度，如果每次使用常駐資料時，都向資料庫要求一次資料，則這些常駐資料請求會拖累系統效能。後端會將常駐資料先快取於session，需要使用資料時直接從session調用資料，就不必再向資料庫請求資料了。

### (三)、安全性

針對後端安全，我們進行了以下幾種保護措施。

#### 1. 密碼安全性

對於所有工作人員的密碼，皆為六個字元以上的英數混和字串，且所有的登入失敗都會寫入紀錄檔中，方便追蹤帳號是否有安全性疑慮。

#### 2. SQL注入*(Sql injection)*

不以舊版的mysql模組操作資料庫，而以新版的mysqli操作資料庫；並且將所有的statement進行prepare後bind\_param，不直接在statement中寫入值，如下列虛擬碼所示。

|  |  |
| --- | --- |
| 具有sql注入風險 | 不具有sql注入風險 |
| $sql=”select id from orders where id = $uid”  $database->execute($sql) | $sql=”select id from orders where id = ?”  $database->prepare($sql)  $database->bind\_param(“1 or True”)  $database->execute() |

#### 3. XSS

在check\_valid 模組中，嚴格限制了每個輸入參數，凡是參數不符合規定，則立刻丟出例外狀況，並寫入記錄檔中。

#### 4. 資料庫死結

　　在單線程的測試環境下，很少會遇到資料庫死結，而在系統真正運行的時候，常會遇到不可預知的死結。一個Procedure包裝了多個語句(Syntax)，若是在尚未執行完Procedure前，發生了死結，則可能會有不可預知的後果。

我們針對容易發生死結的Procedure加上start transcation、rollback、commit，若是在Procedure尚未結束前發生死結，則回溯(Rollback)整個Procedure進行的操作。

## 四、前端

### (一)、iOS前端

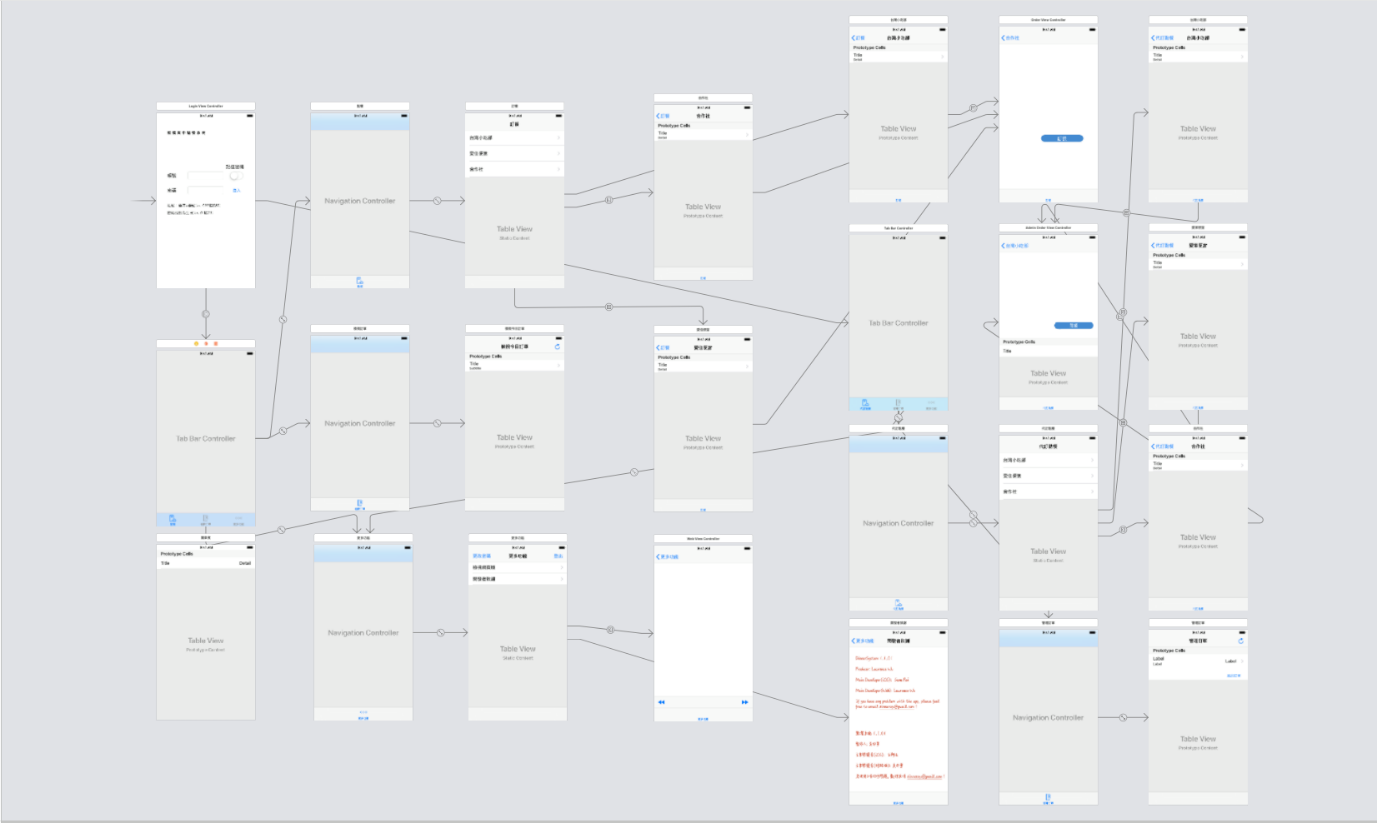
iOS前端為專屬開發給蘋果使用者的操作介面，由於網頁版前端較不簡潔，我們開發了專屬於蘋果使用者的前端，該前端符合iOS的人機互動指南。

以下為iOS前端的登入畫面，我們可以見到前端十分簡潔，與使用者的互動簡單明瞭。

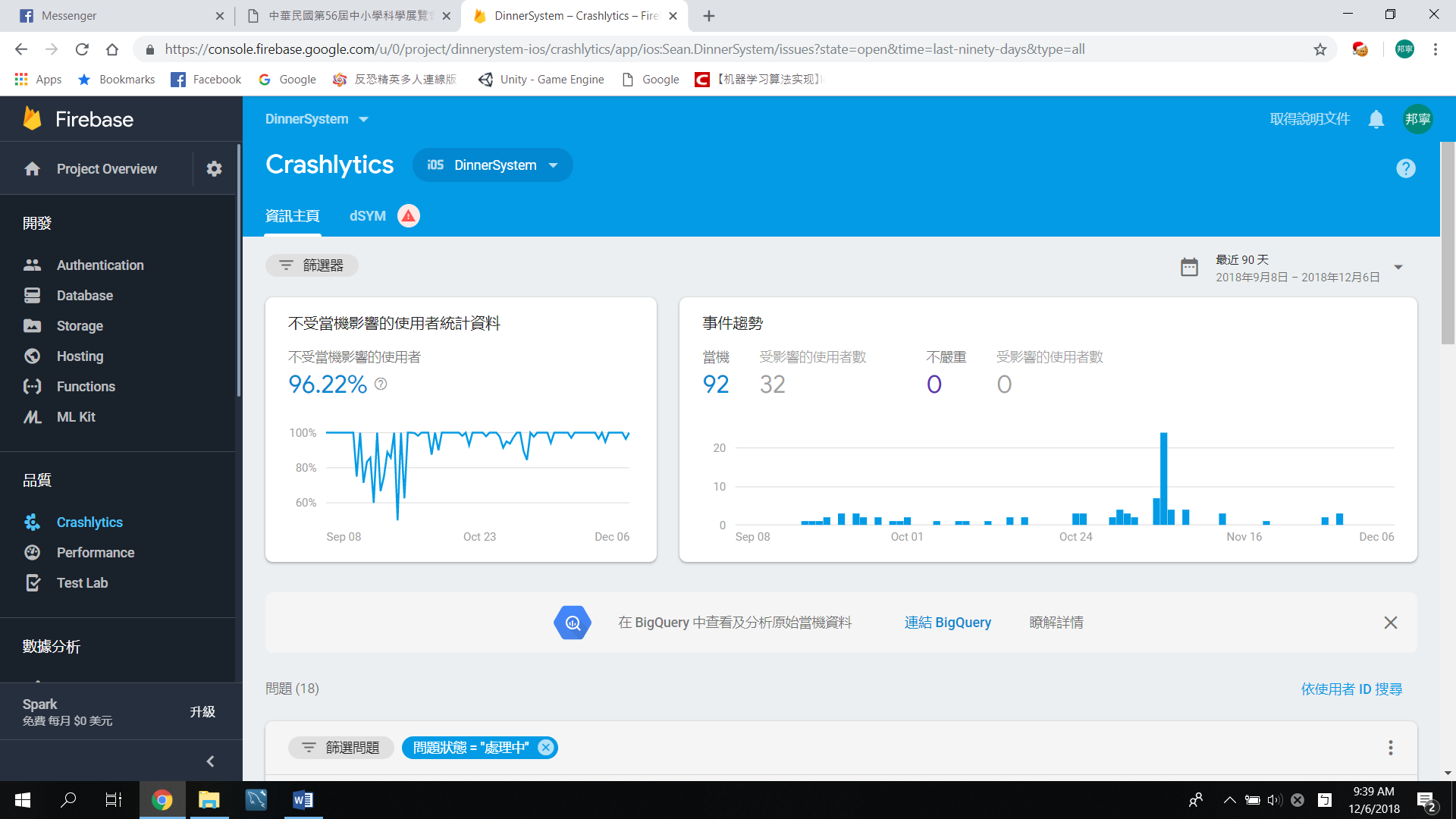
#### 1. 呼叫後端

採用Alamofire第三方API(Application Interface) 進行HTTP請求(request)及回應(response)，上傳時以get方法訪問(Request)伺服器，並使用Swift原生(Native)API中的JSONDecoder來解析伺服器回傳的Json字串。

#### 2. 頁面圖

以上為iOS前端的頁面(Layout)關係圖，我們可以見到十分複雜的設計關係圖，每一條線代表觸發任何事件後可以從第一個頁面(Activity)轉跳至下一個頁面(Activity)。

#### 3.錯誤分析

 發生任何錯誤狀況時，App會傳送資料給Crashlytics，在開發者頁面中可以立即看到發生的例外狀況，也會發送電子郵件給開發者，讓開發者能夠在最快的時間內修正錯誤，以下是crashlytics的統計分析，在九月時系統剛剛上線，較不穩定，進入十月之後系統就幾乎沒有任何問題了。

### (二)、網頁前端

網頁前端擁有最完整的功能，而且當其他前端無法運作時，網頁前端會確保大多數使用者還能夠使用系統，不會使整套系統停止運作。

#### 1. ajax

對於批次點餐、繳款、上傳，我們使用ajax技術向伺服器後端請求資料。使用ajax技術不只可以避免前後端耦合，還可以受惠於平行處理使得效能增加。

#### 2. 分類樹

當我們想要確認繳款101點的所有餐時，我們不必一一查看所有點單，我們只需要查看101的分類就可以了，上圖為分類樹的抽象概念，下圖為實際使用分類樹。



由右圖可見，下列該分類樹的階層。

1. 班級
2. 是否付款
3. 廠商
4. 餐點
5. 點單

分類樹還會順便把金額數量加總，方便點帳。在金額無誤時，就直接按下已繳款按鈕；金額有誤時，從階層式的資料中找出是哪裡點錢點錯了即可。

#### 3. 後端運算

為了避免讓前端處理大量資料，我們選擇先在伺服器使用分類樹整理資料。

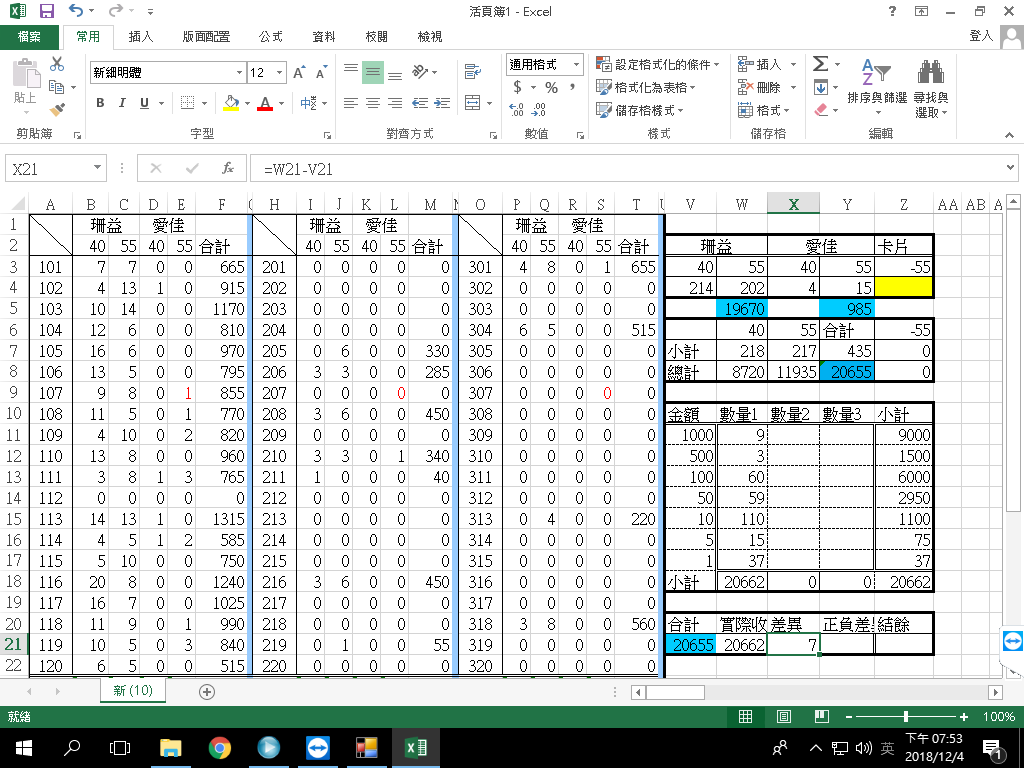
在前端的collapsable模組中，先呼叫後端主模組backend\_main，抓取點餐資料，再將資料傳給collapsable進行整理並轉為HTML字串，再將已經轉為HTML碼的字串傳給前端，雖然浪費了多一點的網路，但是能夠節省前端運算的大量時間。

### (三)、輔助外掛

#### 1. 當日會計報表

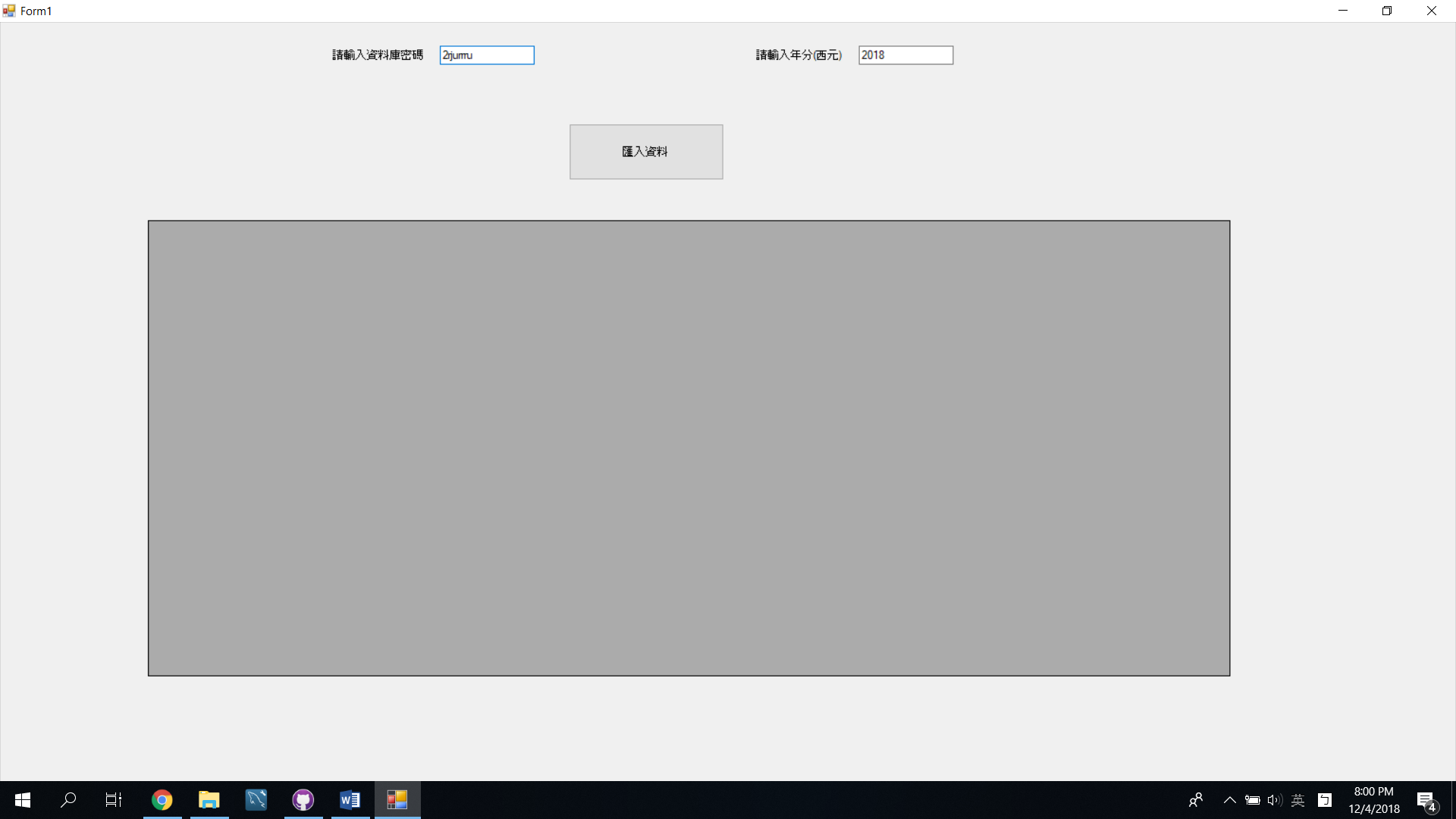
會計報表可以視為一個前端，會將點餐資料統整之後輸出給Excel，以下為十月二十三號的會計報表。

我們可以看到當天的差異只有7，而這大多是人為計算錯誤，不必太在意。



#### 2. 新生資料匯入軟體

匯入軟體必須跟資料庫伺服器放在一起，才能存取資料庫，新生資料匯入軟體會先從Excel資料表裡面抓取新生資料，再將資料轉成SQL語句，交給資料庫處理。



### (四)、前後端交換介面

前後端交換介面確保後端與前端的獨立，任何能夠操作命令介面，與使用者互動，並且能夠擷取回傳資料，不論語言、平台撰寫成的程式，皆可被視為一個前端；任何能夠根據命令回傳出相對應的程式，不論語言、不論平台，也皆可被視為一個後端。

#### 輸出資料結構

原始輸出為JSON字串，我們將JSON轉換成表格，以利理解，以下表格為精簡後的點單資料結構。

|  |  |
| --- | --- |
| **id** | 22410 |
| **user** | |  |  | | --- | --- | | **id** | 11184 | | **name** | 白翔云 | | **seat\_no** | 20904 | |
| **dish** | |  |  | | --- | --- | |  | | | **dish\_name** | 卡拉雞 雞塊 三副菜 | | **dish\_id** | 1 | | **dish\_cost** | 55 | |

#### 命令介面

以下表格為精簡後的命令界面，「-」代表沒有參數。

|  |  |  |
| --- | --- | --- |
| login | id=[使用者帳號] | password=[密碼] |
| select\_self | esti\_start=[時間上界] | esti\_start=[時間上界] |
| set\_payment | oid=[點餐單號] | target=[是否繳款] |
| make\_order | did=[餐點號碼] | esti\_recv=[預計送達時間] |

如第一列所示，其意義代表：「使用get/post方法造訪後台，以cmd為索引，傳入字串login；以id為索引，傳入使用者帳號；以password為索引，傳入密碼」。

#### 3. JSON與編碼

採用UTF-8編碼，包含BOM Header，對於JSON中包含的中文字串不予加以編碼，對於「”」字元更改為「\”」，對於「\」字元更改為「\\」。

## 五、預測模型

我們先跟廠商、合作社打聽了一下平常點餐的趨勢，廠商表示「韓式拌飯」、「牛肉燴飯」特別多人點購，而且訂購便當的意願隨著在學校的時間漸漸降低。

我們經過初步分析，得知高一最多人點餐，高二的點餐人數較少，高三幾乎沒有人點餐，每天的點餐人數大約以87份餐/天緩慢下滑，

### (一)、資料結構

原始的點餐資料可以視為下列表格，「-」代表沒有點餐。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 甲生 | 乙生 | 丙生 | 丁生 | 戊生 |
| 08/07 | 韓式拌飯 | 烤肉飯 | - | 烤肉飯 | 素便當 |
| 08/08 | - | 烤肉飯 | 韓式拌飯 | - | 素便當 |
| 08/09 | 素便當 | 韓式拌飯 | 素便當 | 烤肉飯 | 素便當 |
| 08/10 | 烤肉飯 | 烤肉飯 | - | 韓式拌飯 | 素便當 |
| 08/11 | 韓式拌飯 | 烤肉飯 | 韓式拌飯 | 韓式拌飯 | 素便當 |

以下根據所需的原料進行分類，由上往下進行篩選，例如「韓式拌飯」符合第一行的Regex篩選，故屬於調味飯類；「烤肉飯」不符合第一行的Regex篩選，而符合第三行的Regex篩選，故屬於便當類。

|  |  |
| --- | --- |
| 調味飯類 | Regex: "(焗)|(拌飯)" |
| 麵類 | Regex: "(烏龍)|(麵)|(湯)" |
| 便當類 | Regex: "(副菜)|(飯)" |
| 小吃類 | Regex: "((餃)|(蔥抓餅)|(鍋貼)|(板條))" |
| 鍋類 | Regex: "(鍋)|(粥)" |
| 其他 | 不滿足以上篩選者 |

#### 1. 點餐序列

將甲生的點餐資料提取出來，並且加以分類，再填上是否有點餐，即為點餐序列。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 甲生 | 08/07 | 08/08 | 08/09 | 08/10 | 08/11 |
| 餐點類別 | 調味飯類 | - | 其他 | 便當類 | 調味飯類 |
| 是否點餐 | True | False | True | True | True |

#### 2. 點餐圖

根據原點餐序列，刪去沒有點餐的那一天，即如下表。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 甲生 | 08/07 | 08/09 | 08/10 | 08/11 |
| 餐點類別 | 調味飯類 | 其他 | 便當類 | 調味飯類 |
| 是否點餐 | True | True | True | True |

將每一天的轉移視為圖上的一個邊，每一個分類視為圖上的一個節點，則可得到一個有向圖，我們稱呼為點餐圖。

若將每一個人的點餐序列寫入同一張點餐圖，則可得到類似下面的有向圖。

### (一)、宏觀模型

宏觀模型適合去預測全校明天的點餐數量，能夠在短時間內預測明天的點餐數量，不必進行大量運算，不過模型的精確度相較於微觀模型較低。

模型背後的預測原理可以譬喻為「使每個人隨機在城鎮間移動，移動無限次之後每個城鎮的人數多寡將趨近平衡，城鎮人數多寡即為預測值」，城鎮為「餐點」，移動為「由甲餐轉而訂購乙餐」，城鎮人數多寡為「訂購的數量」。

我們可以將現實世界中的操作抽象化，成為圖論中的操作，方便程式預測，下表即為現實世界中的操作與圖論中的操作對照表。

|  |  |
| --- | --- |
| 圖論上的操作 | 現實世界的意義 |
| 從圖上的任意一個點出發 | 第一天先隨便點一道菜 |
| 經由任意一條邊 | 經過了一天 |
| 到達圖上的另外一個點 | 第二天點了另外一道餐 |
| 經過一個自環 | 經過了一天，決定要吃同一道菜 |
| 到了同一個點 | 第三天點了同一道菜 |

#### 1. 鄰接機率矩陣

對於任意有向圖，將(節點到節點所有邊的數量)/(節點的出度)，寫入矩陣中的，則之值為從節點移動步之後到達節點的機率。

下圖為點餐圖，表一為該圖的鄰接矩陣，鄰接矩陣之值為節點到節點所有邊的數量，表二為該圖的鄰接機率矩陣，表三為鄰接機率矩陣，其中，意思為從節點i移動到節點j的機率為多少。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 韓式拌飯 | 烤肉飯 | 燒肉飯 |
| 韓式拌飯 | 3 | 3 | 1 |
| 烤肉飯 | 5 | 6 | 5 |
| 燒肉飯 | 1 | 0 | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | 韓式拌飯 | 烤肉飯 | 燒肉飯 |
| 韓式拌飯 | 1/3 | 1/3 | 1/8 |
| 烤肉飯 | 5/9 | 2/3 | 5/8 |
| 燒肉飯 | 1/9 | 0 | 1/4 |

對於矩陣，我們可以得知，即為從節點i經過任意節點k再到達節點j之機率和，同時也為節點i到移動兩步到達節點j的機率。

#### 2. 馬可夫矩陣

對於列向量，x代表原本有多少人訂購i餐，將K若，則，代表經過n天後，x’人會訂購第i號餐。經過多次轉移之後，我們知道，

#### 求穩定矩陣

我們有兩種方法可以求穩定矩陣，

下表為兩種演算法的比較，模型會根據傳入的參數決定最佳演算法。

|  |  |  |
| --- | --- | --- |
|  | 矩陣快速冪 | 反矩陣解聯立 |
| 迭代次數 |  | - |
| 誤差值 |  | 0 |
| 優勢 | 能夠根據昨天的資料，預測未來某一天的資料。 | 能夠求得實際上的穩定狀態。 |
| 劣勢 | 會有極小的誤差值。 | 僅能夠獲得長期下來的穩定狀態。 |

#### 4. 線性回歸

### (二)、微觀模型

方便細部追蹤一個小群體

#### 1. 決策模型

#### 2. 預測模型

### (三)、模型應用

#### 1. 模型的比較

|  |  |  |  |
| --- | --- | --- | --- |
|  | 宏觀模型 | 微觀模型A | 微觀模型B |
| alpha/cycles | - |  |  |
| 樣本數 | - | 30個 | 30個 |
| 訓練時間 |  |  |  |
| 信心區間/水準 |  |  |  |
| 優勢 |  |  |  |
| 劣勢 |  |  |  |

下圖為宏觀模型的預測圖

左圖為微觀模型A的預測圖，右圖為微觀模型B的預測圖。

根據便當所需原料，能夠給予廠商一個預估值，避免廠商浪費過多食材。

# 伍、目前研究結果

## 一、使用心得

### (一)、廠商、合作社

### (二)、學生、代訂

## 二、分析資料

### (一)、宏觀模型預測實例

### (二)、微觀模型預測實例

### (三)、預估明日應準備多少原料

# 陸、參考資料及其他

板橋高中資訊培訓講義

資訊之芽

StackOverflow

## 一、發展期望