蜜蜂核心計畫報告

無蜂王授粉媒合平台

進口熊蜂造成的碳排放及能源消耗

以愛禮花卉自荷蘭商 **Koppert** 進口「迷你寶 / Minipol」熊蜂蜂箱（約 **2.5 kg** ）之「油井到車輪」（Well-to-wheels）溫室氣體排放 為例：

1. Amsterdam, AMS → Taipei, TPE: **9,537.27 (km)**
2. 0.0025 (tonne) x 9,537.27 (km) x **0.550** (kgCO2e per tonne kilometer) = **13.11 kgCO2e**

	Well-to-wheels	Tank-to-wheels
GHG emissions	13.11 kgCO2e	10.28 kgCO2e
Energy consumption	149.88 MJ	117.07 MJ

1. https://www.webcargo.co/knowledge-base/tools/freight-co2-emissions-calculator/
2. List of Emissions report: https://www.co2emissiefactoren.nl/lijst-emissiefactoren/
3. CSN EN 16258: Methodology for calculation and declaration of energy consumption and GHG emissions of transport services (freight and passengers)

平台 Web-App

無蜂王授粉媒合平台

立即下單

無蜂王蜂箱以蜂王費洛蒙取代蜂巢內的蜂王，可解決瓜果農長期找不到蜜蜂授粉的困境。

臺灣大學昆蟲系教授楊恩誠研究團隊，完成智慧農業創新，利用費洛蒙讓蜂箱內僅需存有一片巢脾，以取代蜂巢內的蜂王，並將此無蜂王蜂箱應用於溫室內進行授粉

小授蜂

小型蜂箱，適合一般溫室。

$1000.00

數量

0

數量

5

大授蜂

大型蜂箱，適合大面積溫室。

$2000.00

購買者資訊

姓名*

邱先生

電子郵件*

test@example.com

地址*

台灣大學

總費用: $7000.00

送出訂單

訂單已成功提交！姓名：邱先生，電子郵件：test@example.com，地址：台灣大學，訂單內容：小授蜂 x5, 大授蜂 x1

回到首頁

入口畫面 → 選購 → 送出訂單

http://localhost:3000

無蜂王授粉媒合平台

立即下單

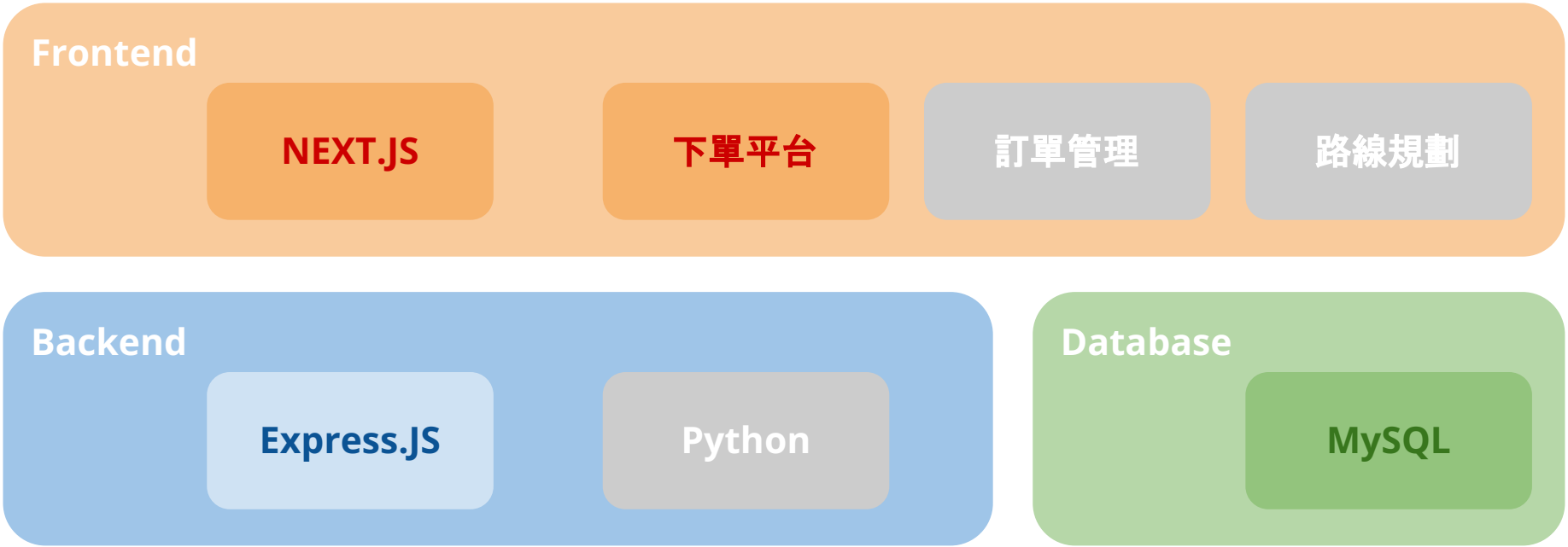
無蜂王蜂箱以蜂王費洛蒙取代蜂巢內的蜂王，可解決瓜果農長期找不到蜜蜂授粉的困境。

臺灣大學昆蟲系教授楊恩誠研究團隊，完成智慧農業創新，利用費洛蒙讓蜂箱內僅需存有一片巢脾，以取代蜂巢內的蜂王，並將此無蜂王蜂箱應用於溫室內進行授粉的任務。

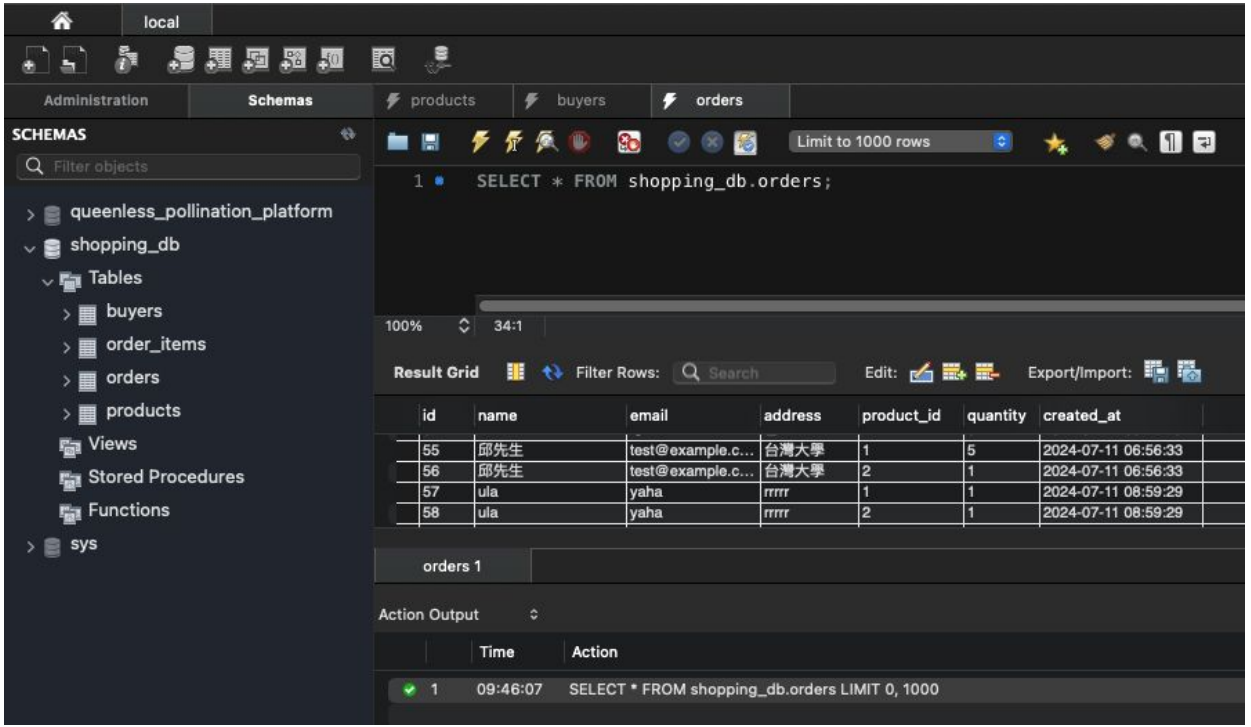
Elements Console Sources Network Performance

top Filter Default levels No Issues 1 hidden

系統架構圖


Frontend
NEXT.JS
下單平台
訂單管理
路線規劃
Backend
Express.JS
Python
Database
MySQL

Database


local
Administration
Schemas
products
buyers
orders
SCHEMAS
Filter objects
queenless_pollination_platform
shopping_db
Tables
buyers
order_items
orders
products
Views
Stored Procedures
Functions
sys
Limit to 1000 rows
1 SELECT * FROM shopping_db.orders;
100%
34:1
Result Grid
Filter Rows:
Search
Edit:
Export/Import:
id name email address product_id quantity created_at
55 邱先生 test@example.c... 台灣大學 1 5 2024-07-11 06:56:33
56 邱先生 test@example.c... 台灣大學 2 1 2024-07-11 06:56:33
57 ula yaha rrrrr 1 1 2024-07-11 08:59:29
58 ula yaha rrrrr 2 1 2024-07-11 08:59:29
orders 1
Action Output
Time Action
1 09:46:07 SELECT * FROM shopping_db.orders LIMIT 0, 1000

開發中功能

媒合平台:最佳化單一起點對多個目的地的配送路徑。

1. 建立距離矩陣
2. 透過演算法求最佳解
3. 將模擬結果用 Google Maps 呈現

Step 1: 建立距離矩陣

使用 Google Maps API 窮舉出所有地點兩兩距離的排列組合。

地點	台北101	台中市政府	高雄六合夜市	花蓮火車站	台南孔廟
台北101	0	167	357	158	316
台中市政府	167	0	190	210	143
高雄六合夜市	357	190	0	318	49
花蓮火車站	158	210	318	0	282
台南孔廟	316	143	49	282	0

Step 2: 透過演算法求最佳解 (1/8)

旅行商問題（Traveling Salesman Problem, TSP），**即尋找一條經過所有指定城市，且只經過一次的最短閉合路徑的問題。**

此問題透過不同的演算法求解，如：

a. 匈牙利算法（Hungarian algorithm）
b. 蟻群算法（Ant Colony Optimization）
c. 遺傳算法（Genetic Algorithm）

以下以**匈牙利算法** 舉例。

Step 2: 透過演算法求最佳解 (2/8)

匈牙利算法(Hungarian algorithm):

1. **行減法:在每一行找出最小 值,然後減去這個最小 值,讓每一行至少有一個零。**

2. 列減法:在每一列找出最小值,然後減去這個最小值,讓每一列至少有一個零。
3. 覆蓋零:使用最少的水平線和垂直線覆蓋所有的零。
4. 檢查覆蓋線數:如果線的數量等於任務的數量,則找到最佳分配方案。如果不是,則進行調整。
5. 調整矩陣:找到未被覆蓋的元素中的最小值,進行調整,創建新的零。

	任務1	任務2	任務3
工人1	**2**	3	**1**
工人2	3	**2**	**1**
工人3	4	3	5

Step 2: 透過演算法求最佳解 (3/8)

匈牙利算法(Hungarian algorithm):

1. **行減法:在每一行找出最小 值,然後減去這個最小 值,讓每一行至少有一個零。**

2. 列減法:在每一列找出最小值,然後減去這個最小值,讓每一列至少有一個零。
3. 覆蓋零:使用最少的水平線和垂直線覆蓋所有的零。
4. 檢查覆蓋線數:如果線的數量等於任務的數量,則找到最佳分配方案。如果不是,則進行調整。
5. 調整矩陣:找到未被覆蓋的元素中的最小值,進行調整,創建新的零。

	任務1	任務2	任務3
工人1	2-2	3-2	1-1
工人2	3-2	2-2	1-1
工人3	4-2	3-2	5-1

Step 2: 透過演算法求最佳解 (4/8)

匈牙利算法(Hungarian algorithm):

1. **行減法:在每一行找出最小 值,然後減去這個最小 值,讓每一行至少有一個零。**

2. 列減法:在每一列找出最小值,然後減去這個最小值,讓每一列至少有一個零。
3. 覆蓋零:使用最少的水平線和垂直線覆蓋所有的零。
4. 檢查覆蓋線數:如果線的數量等於任務的數量,則找到最佳分配方案。如果不是,則進行調整。
5. 調整矩陣:找到未被覆蓋的元素中的最小值,進行調整,創建新的零。

	任務1	任務2	任務3
工人1	0	1	0
工人2	1	0	0
工人3	2	1	4

Step 2: 透過演算法求最佳解 (5/8)

匈牙利算法(Hungarian algorithm):

1. 行減法:在每一行找出最小值，然後減去這個最小值，讓每一行至少有一個零。
2. **列減法:在每一列找出最小 值，然後減去這個最小 值，讓每一列至少有一個零。**
3. 覆蓋零:使用最少的水平線和垂直線覆蓋所有的零。
4. 檢查覆蓋線數:如果線的數量等於任務的數量，則找到最佳分配方案。如果不是，則進行調整。
5. 調整矩陣:找到未被覆蓋的元素中的最小值，進行調整，創建新的零。

	任務1	任務2	任務3
工人1	0	1	0
工人2	1	0	0
工人3	2-1	1-1	4-1

Step 2: 透過演算法求最佳解 (6/8)

匈牙利算法(Hungarian algorithm):

1. 行減法:在每一行找出最小值，然後減去這個最小值，讓每一行至少有一個零。
2. **列減法:在每一列找出最小 值，然後減去這個最小 值，讓每一列至少有一個零。**
3. 覆蓋零:使用最少的水平線和垂直線覆蓋所有的零。
4. 檢查覆蓋線數:如果線的數量等於任務的數量，則找到最佳分配方案。如果不是，則進行調整。
5. 調整矩陣:找到未被覆蓋的元素中的最小值，進行調整，創建新的零。

	任務1	任務2	任務3
工人1	0	1	0
工人2	1	0	0
工人3	1	0	3

Step 2: 透過演算法求最佳解 (7/8)

匈牙利算法(Hungarian algorithm):

1. 行減法:在每一行找出最小值，然後減去這個最小值，讓每一行至少有一個零。
2. 列減法:在每一列找出最小值，然後減去這個最小值，讓每一列至少有一個零。
3. **覆蓋零:使用最少的水平線和垂直線覆蓋所有的零。**
4. 檢查覆蓋線數:如果線的數量等於任務的數量，則找到最佳分配方案。如果不是，則進行調整。
5. 調整矩陣:找到未被覆蓋的元素中的最小值，進行調整，創建新的零。

在此可以最少用三條線

	任務1	任務2	任務3
工人1	0	1	0
工人2	1	0	0
工人3	1	0	3

Step 2: 透過演算法求最佳解 (8/8)

匈牙利算法(Hungarian algorithm):

1. 行減法:在每一行找出最小值，然後減去這個最小值，讓每一行至少有一個零。
2. 列減法:在每一列找出最小值，然後減去這個最小值，讓每一列至少有一個零。
3. 覆蓋零:使用最少的水平線和垂直線覆蓋所有的零。
4. **檢查覆蓋線數:如果線的數量等於任務的數量，則找到最佳分配方案。如果不是，則進行調整。**
5. 調整矩陣:找到未被覆蓋的元素中的最小值，進行調整，創建新的零。

最佳解: 2+1+3=6

工人1 => 任務1

工人2 => 任務3

工人3 => 任務2

	任務1	任務2	任務3
工人1	0 2	1 3	0 1
工人2	1 3	0 2	0 1
工人3	1 4	0 3	3 5

Step 3: 將模擬結果用 Google Maps 呈現

1. 使用 Google Maps API 輸入地點，並產生路徑
2. 將回傳結果繪製於客製化地圖
3. 將地圖嵌入使用者介面