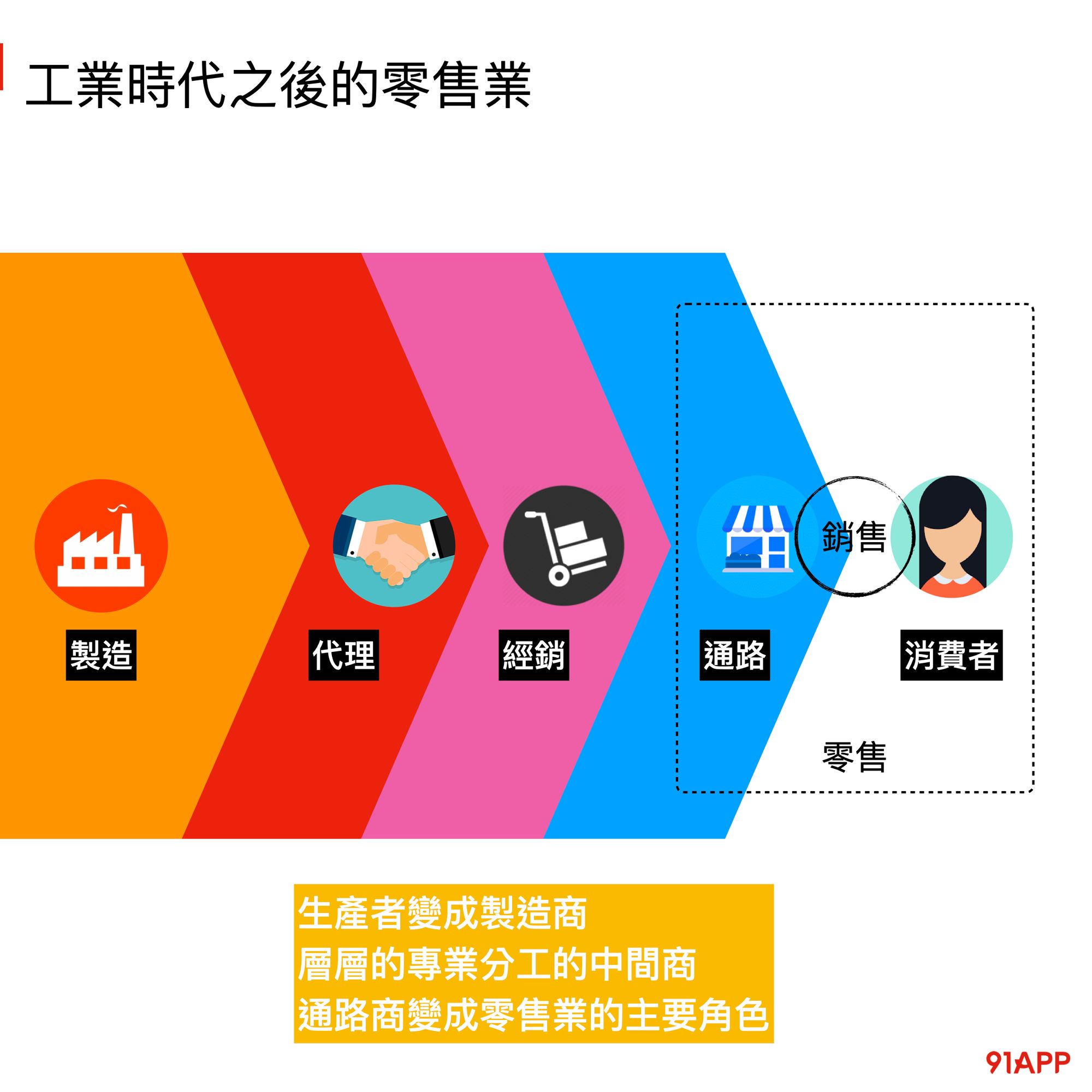
**產品生產銷售履歷鏈之區塊鏈應用**



**學校 :** 國立台北大學

**組員 :** 410785018 資工三 邱信瑋

410885021 資工三 張維哲

410885034 資工三 溫佩旻

410885005 資工三 池秉宸

**目錄**

摘要．．．．．．．．．．．．．．．．．．．．．2

程式架構及內容．．．．．．．．．．．．．．．．3

呈現結果．．．．．．．．．．．．．．．．．．． 11

**壹、摘要**

消費者愈來愈重視食安議題，對於生產者的信賴度也成為採購時的一個重要指標。以生鮮蔬果為例，習慣在一般傳統市場買菜的婆婆媽媽，多是憑藉對攤商的介紹與信任，而到超市賣場採買的消費者，通常會利用產銷履歷機制來了解產品的生產過程和安全性。加上近來受到疫情的影響，許多人改採網購訂貨、宅配到家的方式，這時產銷履歷機制也會成為消費者選擇時的一大考量依據。產銷履歷的農產品有別於傳統農業生產模式，除了得遵照「臺灣良好農業規範」（TGAP）的生產過程，包括生產場域、用藥安全、環境永續及包裝場所衛生均有嚴格的規定，而且每批產品的生產過程，都必須在政府的電子系統中建立追溯資料，並透過第三方的管理驗證，站在公平立場來查核農產品經營業者是否符合法規與TGAP的要求。因此，產銷履歷機制是一項對土地、產品及生產者的驗證制度，讓消費者可以藉由公開可追溯的生產資訊，看見生產者的用心，所以在購買產銷履歷農產品時也會更安心放心。

然而一般的產銷履歷機制還是存在著一些問題，目前產銷履歷制度所登錄的資料機制，並非為一個不可竄改的系統，生產者有可能透過更改紀錄的方式，去遮蔽一些不希望被消費者所看見的資料，這對於產銷履歷機制的公信力就可能造成負面影響。另外，生鮮蔬果和其他產品最大的不同之處，在於農作物很容易受到環境、氣候、病蟲害等多重因素的影響，造成產量與品質的不夠穩定，加上生鮮蔬果的保鮮度維持不易，導致即使臺灣的水果甜又多汁、品質非常好，但卻缺乏國際競爭力。

因此透過「難以竄改和防偽機制」的區塊鏈機制，可將任何經修正的資料、數據，永久保留在記錄當中，至於防偽機制則能防止品牌商標遭人冒用或侵權，有效維護產品商譽。但如果生產者有希望不被公開的商業機密內容，例如特殊的栽種技術、原料混合的配比等，則可以透過加密記錄方式，僅分享給授權者解密調閱。另外，可以將數據轉換成有意義的務農指標，提供生產者做為日後栽種時如何調整改進的參考。此外，若是將區塊鏈和物聯網結合，更能提高供應鏈的效率和透明度，大幅縮短農作物從產地到餐桌的時間與距離。

**貳、程式架構及內容**

本次計畫，我們主要分成三部分，分別是，第一，在Remix環境下利用Solidity語言撰寫履歷的智能合約；第二，利用HTML、CSS、JavaScript語言撰寫網頁來呈現我們的履歷結果；最後，將合約、網頁以及Gananche做連結，讓使用者能很簡單的使用，像是在一般網路上購物的模樣，但另外增加了使用區塊鏈機制的履歷。

**一、履歷智能合約**

在生產鏈上我們簡單分為，生產商、製作商、代理商以及消費者，因此我們履歷的內容為下，履歷內容:

{

Id:編號:由原物料名與區塊產生時的時間戳記(timestamp)產生的32-Byte字串

Owner:擁有者:商品目前擁有者的Address

State:製程階段:商品的狀態(eg.已生產、已加工、已包裝)

由供應商填入:

{

Ingredient:原料:即供應商想生產的物品(eg.生乳、稻米)

Supply\_Name:廠商名稱:廠商申請身分已設定，自動填入

Supply\_Address:廠商地址:廠商申請身分已設定，自動填入

Supply\_Time:生產時間，根據此區塊生產時的時間戳記，自動填入

}

由加工商填入:

{

Product:產品:將原料加工後的產品(eg.鮮乳、白米)

Process\_Name:廠商名稱:廠商申請身分已設定，自動填入

Process\_Address:廠商地址:廠商申請身分已設定，自動填入

Process\_Method:加工方法(eg.高溫殺菌、冷凍)

Expiration\_Date:有效期限

}

由代理商填入:

{

Good\_Name:商品:將產品包裝後的商品

Agent\_Name:廠商名稱:廠商申請身分已設定，自動填入

Agent\_Address:廠商地址:廠商申請身分已設定，自動填入

Proposed\_Method:使用方法(eg.直接飲用、蒸、煮)

Customer\_Service:服務專線

Price:價格

}

}

身分:

使用者加入都為 使用者

可向管理者申請成為 原物料商、加工商、代理商

申請時需要提交 欲申請身分、廠商名稱、廠商地址

日後產生履歷時，將自動填入申請時的資訊

**二、網頁撰寫**

這個部分我們採用HTML、CSS、JavaScript三種程式，搭配Bootstrap來做網頁的呈現，其中用HTML來製作網頁框架、CSS做網頁排版美化，再透過JavaScript做動態生成的部分。

我們主要做了五個頁面來呈現，分別是index(初始登入頁面)及4個不同身分的頁面 : Customer(顧客)、Supplier(供應商)、Processor(加工商)、Agent(代理商)，在初始頁面中可根據輸入的名稱而進入相對應的身分頁面，每個身分頁面中的左側會顯示帳號及目前身分，剩下的部分則是用來顯示商品，針對不同身分會有相對應的按鈕。以下會介紹每個網頁所呈現的頁面及如何排版。

1. Index

透過輸入的帳號辨別使用者身分(如圖1)。

|  |
| --- |
|  |
| 圖1 登入頁面 |

1. Customer

在頁面左側除了顯示帳號和身分以外，會多出申請身分的選項，可以選擇要申請供應商、加工商或是代理商的身分，輸入公司名稱及地址後就可以申請其他身分。剩餘的部分我們先將它的框架設為一個table，根據從Remix合約中傳回的商品數來決定顯示幾格，每一個商品都為一個box，會顯示商品名稱、履歷按鈕及購買按鈕(如圖2)，按下履歷按鈕會跳出供應商、加工商、代理商的履歷(如圖3)，按下購買可以購買此商品(如圖4)，跳出小視窗這個部分是透過Bootstrap中的組件動態生成後來完成的。

|  |
| --- |
| 一張含有 文字 的圖片  自動產生的描述 |
| 圖2 Customer頁面 |
|  |
| 圖3 履歷 |
|  |
| 圖4 購買 |

1. Supplier

顯示商品的部分我們先將它的框架設為一個table，根據從Remix合約中傳回的商品數來決定顯示幾格，每一個商品都為一個box，會顯示商品名稱及生產按鈕 (如圖5)，這邊我們會預設一些可以生產的產品，按下生產按鈕後會挑出小視窗顯示一些製造商的履歷，並根據所選的原物料填入成分欄位(如圖6)，再次按下生產後就生產成功，也可按取消按鈕取消生產，跳出小視窗這個部分是透過Bootstrap中的組件動態生成後來完成的。

|  |
| --- |
|  |
| 圖5 Supplier頁面 |
|  |
| 圖6 生產 |

1. Processor

顯示商品的部分我們先將它的框架設為一個table，根據從Remix合約中傳回的商品數來決定顯示幾格，每一個商品都為一個box，會顯示商品名稱及加工按鈕 (如圖7)，按下加工按鈕後會挑出小視窗顯示一些製造商及自己所需填寫的履歷 (如圖8)，再次按下加工按鈕後就加工成功，也可按取消按鈕取消加工，跳出小視窗這個部分是透過Bootstrap中的組件動態生成後來完成的。

|  |
| --- |
|  |
| 圖7 Proccessor 頁面 |
|  |
| 圖8 加工 |

1. Agent

顯示商品的部分我們先將它的框架設為一個table，根據從Remix合約中傳回的商品數來決定顯示幾格，每一個商品都為一個box，會顯示商品名稱及包裝按鈕 (如圖9)，按下生產按鈕後會挑出小視窗顯示一些製造商、加工商及自己所需填寫的履歷 (如圖10)，再次按下包裝後就包裝成功，也可按取消按鈕取消包裝，跳出小視窗這個部分是透過Bootstrap中的組件動態生成後來完成的。

|  |
| --- |
|  |
| 圖9 Agent頁面 |
|  |
| 圖10 包裝 |

**三、如何連接合約、網頁以及Gananche**

這次的計畫主要是利用Web3.js及jQuery來建立智能合約與網頁端的連線，我們先利用Ganache啟動我們的測試鏈，並把合約部署上去，網頁端連接到我們的HttpProvider，雖然HttpProvider的功能性有待加強，不過拿來使用我們這次的計畫是可行的，接著設定我們預設的錢包位址(web3.eth.defaultAccount)、合約位址(address)、合約的ABI，最後建立我們的合約連線(如圖 11)。

|  |
| --- |
|  |
| 圖11建立合約連線 |

我們主要使用Web3.js中methods.function()的Call&Send來呼叫合約中的函式，這部分我們Call合約中的Show\_Personal\_Data，來獲取這個錢包位址在這份合約中的註冊資料(如圖12)。

|  |
| --- |
|  |
| 圖12 獲取註冊資料 |

我們Call合約中的List\_All\_Commidity\_Id和List\_Commidity\_Data來獲得目前合約中已經被包裝好的商品，並將回傳的結果設定到我們網頁的架構中，其中我們會使用到Promise、Async 及Await，因為這些函式屬於非同步函式，因此我們必須確保先獲得回傳的資料，再進行後續的設定(如圖14)，接著開始網頁的活動，利用Await確保資料的獲取(如

圖13)。

|  |
| --- |
|  |
| 圖13 開始網頁的活動 |

|  |
| --- |
|  |
| 圖14 獲取合約中已包裝商品 |

最後，判別使用者點選了哪一個商品要進行交易，其中我們新增了Web3.js中的estimateGas函式，先去判斷合約中使用Buy()所需要的Gas，再把數值放進後續真正執行Buy()時設定的Gas，不這麼做會發現，我們很常會讓我們沒有辦法執行(Error:Out of Gas)(如圖15)。

|  |
| --- |
|  |
| 圖15判別使用者點選了哪一個商品進行交易 |

這部分是提供給身份為Customer申請其他身份使用的函式，一樣會用到EstimateGas，然後將結果Send到合約中(如圖16)。

|  |
| --- |
|  |
| 圖16 提供給身份為Customer申請其他身份使用的函式 |

**參、呈現結果**

本專題使用Solidity來撰寫智能合約，模擬生產商、加工商、代理商及顧客購買商品時的一條鏈，並透過網頁來呈現結果。當製造商選擇了欲生產的產品，並確定生產後，合約會收到這筆交易，確認過後將此項結果傳給Processor，在Processor刷新頁面時，會接收到目前有幾項商品可以加工，此時會發現新增了剛剛製造商生產的品項(如圖17)，其他頁面也是如此，當加工商加工後(圖18)，在Agent頁面刷新可看到剛剛加工完的商品(如圖19)，Agent選擇欲包裝的商品並包裝(圖20)，包裝後在Customer的頁面可以看到這項商品(圖21)，並顯示所有過去的履歷(如圖22)。

|  |
| --- |
|  |
| 圖17 左圖為Supplier介面 右圖為Processor介面 當Supplier生產產品後Processor介面就會顯示可製造的產品 |
|  |
| 圖18 Processor加工產品 |
|  |
| 圖19 Processor加工產品後，Agent的畫面 |

|  |  |
| --- | --- |
|  | |
| 圖20 Agent包裝產品 |

|  |  |
| --- | --- |
|  | |
| 圖21 Agent包裝產品後，Customer的畫面 |

|  |  |
| --- | --- |
|  | |
| 圖22 最後呈現的履歷 |