

國立政治大學資訊科學系研究所
碩士學位論文

去中心化數位貨幣交易記錄與查詢服務：設計
與以太坊實作

A Decentralized Digital Currency Tracing Service: Design and
Implementation on Ethereum

指導教授：郭桐惟 教授

研究生：朱奕寧 撰

中華民國一〇九年十二月

摘 要

在這個數位崛起的時代，區塊鏈是這個時代所造就的新興科技。隨著區塊鏈的發展日漸成熟，該技術也逐漸在金融、物流、穩定幣等各個層面衍生出多樣化的應用與服務。它透過密碼學與分散式系統技術，使區塊鏈中的所有節點彼此交換訊息並共同維護鏈上資料（如交易帳本）。隨著時間的推進，區塊鏈中的資料與交易量不斷成長，因此，如何於龐大的區塊鏈中取得數位貨幣交易紀錄，已成為實務應用上必須面對的問題。為此，本論文設計一去中心化數位貨幣記錄與查詢服務。此服務實作於私有鏈之智能合約中，利用跨鏈技術（見證人模式與中繼者模式）存取於公有鏈上之交易資料，並於智能合約中使用 Binary Search 演算法，提升「區塊範圍查詢」、「時間範圍查詢」功能之效率。最後，本論文將透過 Geth 建構此去中心化數位貨幣記錄與查詢服務，並架設一網頁介面以呈現服務內容。

關鍵字：區塊鏈、智能合約、跨鏈技術

ABSTRACT

Blockchain is a novel technology, and it has diversified applications and services in various fields such as finance, logistics, and stable coins. Blockchain uses cryptography and decentralized system technology to enable all nodes in the blockchain to exchange messages with each other and jointly maintain chain data such as transaction ledgers. As time progresses, blockchain stores a massive amount of data. Thus, searching data has become an important problem. Therefore, we design a decentralized digital currency tracing service, which is implemented in private blockchain smart contracts. For private blockchain smart contracts to access blockchain that stores the transaction data, we use cross-chain technology such as notary schemes and relays. In addition, we apply binary search to improve the efficiency of our searching functions, including block range query and time range query. Finally, we have implemented our system on Geth, and setup a website to demonstrate our service.

Keywords: blockchain, smart contract, cross-chain technology

目 次

摘 要	I
ABSTRACT.....	II
目 次	III
圖 次	V
第一章 緒論.....	1
第二章 技術背景.....	4
2.1 區塊鏈.....	4
2.2 智能合約	6
2.3 跨鏈技術	7
2.4 ORACLIZE (PROVABLE) 服務.....	7
第三章 相關研究.....	10
3.1 BIGCHAINDB.....	10
3.2 ETHEREUM QUERY LANGUAGE	11
3.3 ETHERSCAN.....	12
3.4 EVENT LISTENER	13
第四章 系統實作.....	14
4.1 實作系統架構	14
4.2 智能合約設計	15
4.2.1 智能合約架構	15
4.2.2 Oraclize (Provable) 服務與 Ethereum-Bridge 介紹	17
4.2.2.1 Oraclize (Provable) 服務.....	17

4.2.2.2 Ethereum-Bridge	19
4.2.3 資料處理方式	19
4.2.4 Query 功能介紹	22
4.2.4 Query 實驗結果	27
4.3 WEB 後端設計	28
第五章 系統操作介面與使用流程	29
5.1 新增數位貨幣之合約位址與相關資訊	30
5.2 修改數位貨幣之相關資訊	33
5.3 查找特定數位貨幣之交易紀錄	35
5.4 查找所有已新增之數位貨幣之交易紀錄	39
第六章 討論與結論	40
6.1 討論	40
6.2 結論	41
參考文獻	42

圖 次

圖 2.1：ORACLIZE (PROVABLE) 服務示意圖	9
圖 3.1：BIGCHAINDB 系統架構圖	11
圖 3.2：EQL 語法範例	12
圖 3.3：ETHERSCAN 使用者操作頁面	13
圖 4.1：實作系統架構圖	15
圖 4.2：智能合約設計架構圖	16
圖 4.3：ETHERSCAN API 範例	18
圖 4.4：API 之回傳結果內容	22
圖 4.5：儲存之交易內容	22
圖 4.6：比較 BINARY SEARCH 搜索效能	28
圖 5.1：網頁架構圖	29
圖 5.2：數位貨幣交易記錄與查詢服務之首頁畫面	30
圖 5.3：新增／修改數位貨幣資訊主頁	30
圖 5.4：新增數位貨幣資訊頁面	31
圖 5.5：成功新增數位貨幣資訊之畫面	32
圖 5.6：查詢數位貨幣歷史交易頁面	33
圖 5.7：修改數位貨幣資訊畫面	34
圖 5.8：查詢已儲存之數位貨幣交易內容畫面	35
圖 5.9：詳細的交易內畫面	36
圖 5.10：訂定區塊範圍篩選之畫面	36
圖 5.11：區塊範圍篩選結果之畫面	37
圖 5.12：訂定日期範圍篩選之畫面	37
圖 5.13：日期範圍篩選結果之畫面	38

圖 5.14：交易發生時間篩選之畫面	38
圖 5.15：特定位址篩選之畫面	38
圖 5.16：特定位址篩選結果之畫面	39
圖 5.17：數位貨幣篩選結果之畫面	39

第一章 緒論

隨著網際網路的普及，已有各式各樣的數位化商品與服務陸續出現，隨之改變大眾的生活習慣。然而，因近年來區塊鏈技術日漸成熟，貨幣的數位化才得以被實現。除了數位貨幣之外，也衍生出金融性商品、物流追蹤、產品保固卡等各個層面的應用與服務。重要的是，區塊鏈具備下列三大特性：（一）交易紀錄不得竄改、（二）可追溯性、（三）去中心化，下列分別介紹此三項特性。

（一）交易紀錄難以被竄改：採用密碼學原理將交易紀錄儲存於區塊鏈中，並透過時間順序排序，一旦數據寫入至區塊鏈中，便可以保證其交易紀錄無法遭人輕易竄改。

（二）可追溯性：區塊鏈系統為塊鏈式結構，鏈上數據依時間順序環環相扣，造就了區塊鏈之可追溯性。

（三）去中心化：因這個特性讓區塊鏈被稱之為「分散式帳本」，由系統中所有節點共同維護與更新同一本帳本，且沒有中央伺服器來管理秩序。

區塊鏈便是能同時滿足交易紀錄不得竄改、可追溯性且去中心化的安全系統。隨著區塊鏈技術的演進，現今也發展出了智能合約 [1]，其合約內容與代碼皆由程式所編寫而成，是區塊鏈中一種制定合約時所使用的特殊協議，主要用於提供驗證及執行智能合約內所訂定的條件。目前大多應用於發行數位資產（如 ERC-20 Token [2]）、儲存資料或是開發金融類型協議（如 DeFi [3]）。智能合約也具有高交易效率，合約的流程幾乎為自動化，使得交易效率提高；也能透過客製化合約，依照用戶的需求進行修改。

因應不同使用者需求及應用場景的不同，區塊鏈主要可分為公有鏈和私有鏈。公有鏈為任何人都能參與的區塊鏈，不需要通過授權，即可自由加入或退出，如比特幣 [4]或以太幣 [5]等；私有鏈則是需要受到授權才能參與的區塊鏈，

並非任何人都能加入，像是 Facebook Libra [6]、R3 Corda [7]等。因參與私有鏈的節點有權限限制和可控制的區塊鏈系統設置，因此，私有鏈往往可以有較快的交易處理速度、每個區塊可容納更多的交易量且能夠節省需花費的交易手續費。本論文將聚焦於私有鏈上。

隨著時間的演進，區塊鏈中的資料與交易量不斷成長，但區塊鏈並沒有提供搜尋鏈上資料、查詢交易等功能可供使用者進行查找。因此，如何於龐大的區塊鏈中取得數位貨幣（於本論文中為 ERC-20 Token 代幣）之相關交易紀錄、獲取特定資訊、亦或是政府機關和監察單位等須進行審核與查找鏈上數據等，現已成為實務應用上必須面對的問題。如今有以下現有技術與相關文獻來解決該問題，像是 BigchainDB [8]、EQL [9]與 Etherscan [10]。BigchainDB 為分散式資料庫與區塊鏈的結合而成之技術平台，但無法查詢智能合約內部所記錄之交易資訊（如 ERC-20 Token 之交易數量）；EQL 則是提供使用者可透過類似 SQL [11]的查詢語言檢索區塊鏈中的數據（像是交易內容、區塊內容等），但實際上 EQL 與前者相同皆無法查詢智能合約內部所記錄之交易資訊；Etherscan 雖為現今最為成熟的 Ethereum Block Explorer，其檢索區塊鏈中的數據功能齊全，也提供智能合約的內部交易資訊，但沒有提供區塊範圍查詢、時間範圍查詢等功能。為此，本論文欲設計一去中心化數位貨幣交易記錄與查詢服務。更明確地說我們希望以「去中心化」的方式、「不遺漏任何交易紀錄」、「無誤的交易內容」且「交易來源可信任」的情況下，設計一查詢數位貨幣之歷史交易服務。重要的是，該服務也提供了區塊範圍查詢、時間範圍查詢、交易發送方與交易接收方等查詢功能。以下是本論文的研究目標：

- 設計一去中心化數位貨幣交易記錄與查詢服務，並保證不遺漏任何交易紀錄、無誤的交易內容且交易來源可信任
- 該服務建構於智能合約中，透過跨鏈技術結合 Oraclize (Provable) [12]服務，存取公有鏈上之數據於私有鏈。

- 實作 Binary Search 於智能合約中，提升使用者使用範圍查詢之效能

第二章 技術背景

本研究以探討去中心化數位貨幣交易記錄與查詢服務為目標，將以太坊區塊鏈作為開發平台，並以智能合約結合 Oraclize (Provable) 服務實作於區塊鏈跨鏈互操作性應用，將存在於公有鏈上之數位貨幣歷史交易儲存至私有鏈中，使用者可針對儲存於私有鏈中之歷史交易進行查詢。本章共分四小節，首先簡介區塊鏈之相關技術背景，接著說明智能合約的相關特性，最後則是介紹現有跨鏈技術的種類，以及 Oraclize (Provable) 服務是如何可信地與區塊鏈外部溝通。

2.1 區塊鏈

「區塊鏈」源自於中本聰 (Satoshi Nakamoto) 的比特幣，透過複雜的密碼學加密資料，再利用數學分散式演算法，成為「去中心化的系統」，也是作為比特幣的底層技術，或稱為「去中心化的分散式資料庫」。區塊鏈將舊有的資料庫重新改造，把資料分成不同的區塊，每個區塊透過特定的協議鏈結到上一區塊，將區塊互相接連呈現一完整的歷史數據。區塊鏈上的每筆交易，都能由區塊鏈的結構設計達到溯源的功能，並能針對每筆交易進行驗證。在區塊鏈的每一個區塊當中，皆會擁有一時間戳進行紀錄，以表示該區塊是由當時的時間所生成的，造就了不可篡改、不可偽造的系統。區塊鏈中所有參與的節點在沒有中心的情況下，節點們透過去中心化的共識機制、遵循著規則驗證訊息內容，並寫入時間戳後生成區塊，再將其廣播至所有節點。透過分散式的方式達成數據儲存、交易驗證、訊息傳遞即為區塊鏈的核心技術。

自從比特幣的出現後，區塊鏈的技術逐漸受到重視，公開透明、公平競爭。隨後世界便開始探索區塊鏈技術在各行各業中發展的可能性，因此，除了常見

的公有鏈以外，隨著應用場景的不同，更衍伸出適合企業、產業界使用的私有鏈與聯盟鏈。依照使用者對區塊鏈的權限程度不同，種類可略劃分為三種類型：

（一）公有鏈：任何人都可以訪問，發送、接收、驗證交易，並參與共識過程的區塊鏈。我們最熟悉的區塊鏈，大多屬於公有鏈，像是比特幣、以太幣等。

（二）私有鏈：區塊鏈的權限被一定程度地進行了限制，須受到授權才能成為節點，並非任何人都能參與。例如摩根大通（JP Morgan）引領的 Quorum [13] 就是私有鏈的代表之一。

（三）聯盟鏈：聯盟鏈與私有鏈相似，區塊鏈的開放程度與權限也是有所限制的，而授權的節點通常為企業與企業間有合約的關係等。舉例來說，Hyperledger Fabric [14]，以及 R3 Corda 皆是聯盟鏈的一種。

此外，區塊鏈上的交易是在於打包交易並出塊，通過驗證與確認交易是否有效，使交易順利完成，並讓區塊鏈中的所有節點進行更新、以及擁有相同的帳本內容，這樣打包交易並出塊的機制就是「挖礦」。在以太坊區塊鏈中，Ether（ETH）是其中的燃料，在區塊鏈上進行交易時，必須經由礦工運算打包後上鏈，使用者必須支付礦工該交易的手續費，其交易手續費是以 Gas 計算，並以 Ether 支付。於以太坊主網當中，每個區塊都有 800M 的大小限制（Block Gas Limit），而一般標準交易的 Gas Limit 約為 21,000，一個區塊約可以容納 380 筆交易。交易的 Gas Limit 會依據交易的內容而有所不同，每筆交易的發起方都須設置交易的 Gas Limit 和 Gas Price，不同的操作會產生不同的 Gas 成本，若是交易需花費的 Gas Limit 太高或是超出區塊鏈所設定的大小限制，則可能發生交易無法打包至區塊中，而滯留在交易池（Transaction Pool）的狀況。

整體而言，區塊鏈的應用技術不斷創新，仍有必要持續探索區塊鏈此一新興科技的潛能，在實務應用上，則必須解決區塊鏈的交易擁塞的情形、須支付的 Gas Fee 過高，以及區塊的 Gas Limit 的限制。有鑑於此，為了實現數位貨幣溯源的服務，本研究將實驗環境建立於私有鏈當中，為解決原區塊鏈上發生交易

擁塞的情況、需支付 Ether 作為交易手續費的情形、提升打包交易並出塊的速度、以及自定義其區塊大小以增加每筆同步交易的數量，以進行數位貨幣溯源的探討與實驗。

2.2 智能合約

以太坊區塊鏈在比特幣區塊鏈的基礎上，衍伸了一個具開源性與開發性高的區塊鏈平台，且支援開發多元化區塊鏈分散式應用程式，稱之為智能合約（Smart Contracts），並透過其原生貨幣以太幣（Ether）執行去中心化的運行。以太坊區塊鏈也具有圖靈完備（Turing Complete）的特性，使用者能透過 JavaScript 和 Python 等現有語言在以太坊虛擬機（Ethereum Virtual Machine，EVM）上開發分散式應用程式，或稱為 Dapps，並共同維護以太坊區塊鏈之完整性。

智能合約是區塊鏈中的一種特殊協議，其合約內容與代碼皆由程式所編寫而成，可用於發行客製化代幣、紀錄資料或是金融交易等功能。智能合約具有高安全性、高交易效率與可客製化等，我們也能透過智能合約儲存資料，且本身也能傳遞資料訊息到外界。智能合約就像是個可以被信任的第三方角色，能在沒有中介的情況下，進行交易。智能合約使得區塊鏈在未來能有更多的可能性。

ERC-20（Ethereum Request for Comment）是一個基於以太坊智能合約的一種 Token 標準協議（EIP，Ethereum Improvement Proposal），所有的 ERC-20 代幣都能於以太坊中進行交易、追蹤或是監測等。以太坊中的代幣交易皆須透過發起交易（Transaction）才能完成動作，而這些交易都會被記錄在區塊鏈上，使交易都具有可追溯性。

2.3 跨鏈技術

跨鏈技術重要的目標是在於保證資料的正確性。即傳遞一筆來自於區塊鏈 A 的資料，跨鏈技術判斷該資料是否來自於區塊鏈 A。為了驗證該筆資料，不同的跨鏈技術有不同的驗證方法。依據跨鏈技術對於第三方的依賴程度將該技術概分成兩種：（一）見證人模式（Notary）、（二）中繼模式（Relay），下列分別介紹此兩種跨鏈技術。

（一）見證人模式（Notary）：在見證人模式中，見證人會同時參與多個區塊鏈，且充當區塊鏈間都能共同信任的第三方公證人，作為中介不僅能進行數據收集，還能進行交易確認與資料驗證。在此機制中，它可以通過「單簽名」或「多簽名」的見證人機制來實現。舉例來說，當見證人需要傳遞區塊鏈 A 的資料至區塊鏈 B 時，見證人會負責將該資料寫入區塊鏈 B 中，即區塊鏈 B 的節點或是智能合約經過查驗得知該資料是由見證人簽章後所存入，便可相信資料確實來自於區塊鏈 A。

（二）中繼模式（Relay）：在中繼模式中，中繼者僅提供資料，以及驗證該資料所需的證據。亦即，中繼模式不依賴可信的第三方進行資料驗證，而是在取得發送的資料後自行驗證。例如：若區塊鏈是透過工作量證明（Proof-of-Work）達成共識（如比特幣），中繼者則需傳遞所有的區塊標頭（Block Header）合該交易的梅克爾證明（Merkle Tree），並由區塊鏈節點或是智能合約進行資料驗證。

2.4 Oraclize（Provable）服務

Oraclize（Provable）服務作為區塊鏈外部世界之資料提供者，能夠在區塊鏈上提供可信任的數據傳送服務，其目的是在區塊鏈建立一條可與外部溝通的橋樑，為了解決智能合約取得數據的限制，在保證可信的情況下，使得智能合

約具有取得區塊鏈外部數據的能力。此服務也不僅局限於以太坊區塊鏈，目前也可以使用在其他區塊鏈平台上，例如 Bitcoin [4]、EOS [15]、Rootstock [16]、Fabric [14]。

Oraclize (Provable) 服務在以太坊區塊鏈上部署了名為 `usingOraclize` 的智能合約，只需要在自己的智能合約當中繼承並使用，並依據 Web API 的使用方法進行調用即可。由 Oraclize (Provable) 服務之外部伺服器透過監聽其特定的 Event 來接收智能合約所發送的 Query 請求，調用該請求之 API 且處理完成後，再由 Oraclize (Provable) 服務將請求結果回傳至智能合約內之 `callback function`。此外，Oraclize (Provable) 服務也提供了五種資料來源，使得智能合約能夠取得區塊鏈外部資料，包括 URL 資料來源、Wolfram Alpha 資料來源、IPFS 資料來源、Random 資料來源、以及 Computation 資料來源等。

在跨鏈技術中，Oraclize (Provable) 服務也能作為傳遞訊息的第三方角色。在見證人模式中，我們能將 Oraclize (Provable) 服務作為完全可信任之見證人角色，完全相信其傳遞信息的正確性；在中繼模式中，Oraclize (Provable) 服務也能作為中繼者的身分，它也能提供資料的可靠證明 (Authenticity Proof) 服務，並透過驗證其傳遞之信息證據，確保資料的真偽。本論文將使用 Oraclize (Provable) 服務，並實作於見證人模式 (Notary)，將 Oraclize (Provable) 服務視為為可信任的第三方，完全相信由該服務傳遞之信息的正確性，作為公有鏈與私有鏈間之資料存取之跨鏈技術。

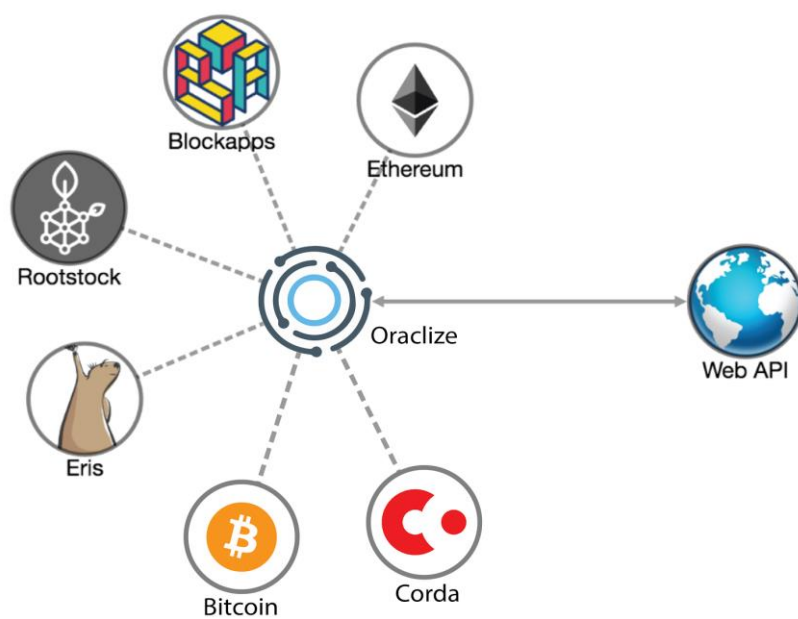


圖 2.1：Oraclize（Provable）服務示意圖

第三章 相關研究

本論文欲設計一建構於區塊鏈智能合約之數位貨幣交易記錄與查詢服務，以參考現有相關技術，並針對該服務之需求進行研究。本論文將以此作為出發點，進行該題目之探討與延伸，並開發該服務系統作為依據。本章共分為四小節，第一小節為 BigchainDB [8]，該論文則是針對分散式資料庫與區塊鏈的結合，以達成兩者之特性；第二小節為 EQL [9]，該論文針對區塊鏈查詢的功能，實作出一查詢語言 EQL (Ethereum Query Language)，此工具可針對區塊 (Block) 與交易 (Transaction) 進行查詢，但尚未完成智能合約內部數據的查詢；第三小節為 Etherscan [10]，是一個 Ethereum Block Explorer，可以查看區塊鏈上所有發生的交易及其交易狀態等；第四小節為透過現有的 API 技術 - Event Listener 監聽智能合約所發生的事件。

3.1 BigchainDB

BigchainDB 是一個可用的去中心的資料庫。它的設計起源於分散式資料庫，加入了很多區塊鏈的特性，像是去中心控制、不可改變性、數字資產的建立和移動。BigchainDB 繼承了現代分散式資料庫的特性：吞吐量和容量都是與節點數量線性相關、功能齊全的 NoSQL 查詢語言，以及高效的查詢和許可權管理。因建構在已有的分散式資料庫上，它在程式碼層面也繼承了企業級的健壯性。圖 3.1 為 BigchainDB 之系統架構圖，根據現版本之設計，每個節點都有一 mongoDB 資料庫用於儲存鏈上資料之用，以此使用者皆能透過常規之 SQL 語言進行鏈上資訊 (Block、Transaction 等) 之搜索，即不包含數位貨幣之交易數量查詢功能。

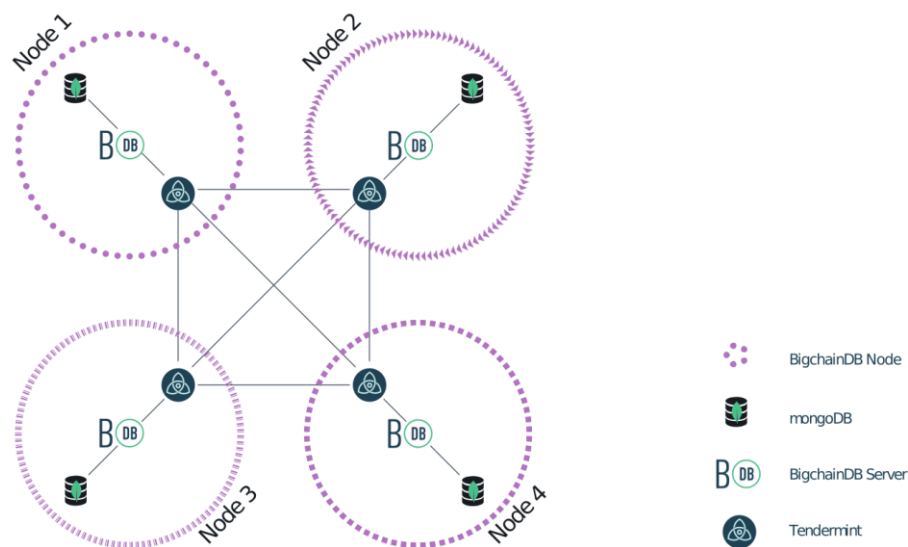


圖 3.1：BigchainDB 系統架構圖

3.2 Ethereum Query Language

此論文提出了以太坊查詢語言（EQL，Ethereum Query Language），一種允許用戶通過編寫類似 SQL 的查詢從區塊鏈中檢索信息的查詢語言。為區塊鏈查詢提供了豐富的語法，可以指定數據條件來查詢分散在多個記錄中的信息。該文獻之實作方法為先將儲存於區塊鏈之所有數據儲存於本地資料庫中，並提供一查詢工具，提供使用者更便利性查詢區塊鏈數據之平台。EQL 使從區塊鏈中搜索、獲取、格式化和呈現信息變得更加容易。本文獻雖然提供了使用者易於查詢區塊資訊的方法，但還缺乏取得數位貨幣之交易數量的功能。圖 3.2 是 EQL 對 Block 查詢的語法範例。

```
1 SELECT block.parent.number, block.hash,  
    block.timestamp, block.number,  
    block.amountOfTransactions  
2 FROM ethereum.blocks AS block  
3 WHERE block.timestamp BETWEEN date('2016-01-01')  
    AND now() AND block.transactions.size >10  
4 ORDER BY block.transactions.size  
5 LIMIT 100;
```

圖 3.2：EQL 語法範例

3.3 Etherscan

Etherscan，是一個 Ethereum Block Explorer，可以查看區塊鏈上所有發生的交易、交易狀態或是查詢 ETH 錢包餘額等功能。其中，Etherscan 也能瀏覽相關 ERC-20、ERC721 合約之代幣價格、相關交易以及代幣持有者等。該網站也提供了統計圖表和數據，進而分析供應量的增長、代幣價格的漲幅或是交易頻率等服務。雖然提供了使用者易於查詢區塊鏈資訊與數位貨幣交易的介面系統，但目前沒有提供區塊範圍查詢或是時間範圍查詢的功能。圖 3.3 為 Etherscan 使用者操作頁面。

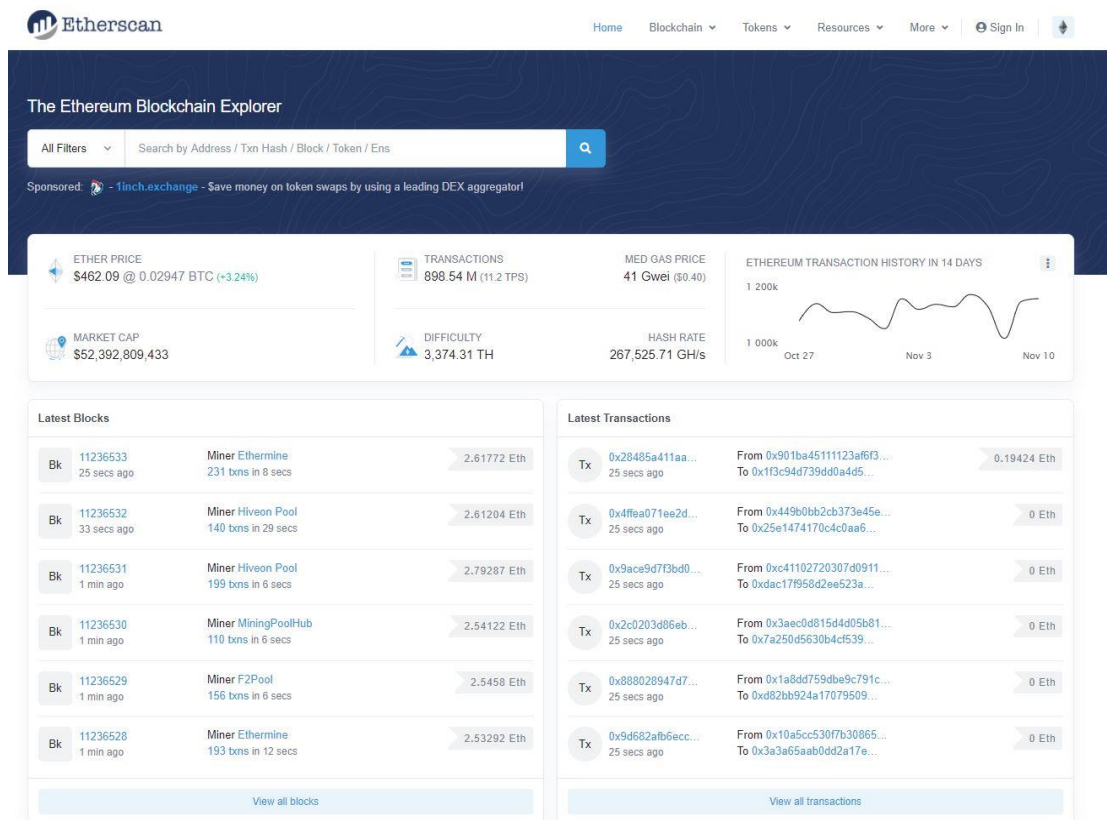


圖 3.3：Etherscan 使用者操作頁面

3.4 Event Listener

在區塊鏈的世界中，每一筆交易都必須等待礦工打包交易資料並寫入區塊後，交易才算完成。當我們與智能合約之間的互動所發出的事件稱之為 Event。而以太坊區塊鏈提供了智能合約 Event 的機制，讓使用者或是區塊鏈上其他的智能合約可以藉由監聽 Event 來確認交易是否完成。我們也能在 Event 中設定於外部監聽時欲取得的數據內容，同時也能針對特定的智能合約行為發送其特定的事件，以達到監聽智能合約或是數位貨幣交易的功能。雖然能透過此方法達成需求，但該方法只能針對尚未部署至區塊鏈上的智能合約，並修改其程式碼內容，且需有一外部伺服器負責監聽該智能合約所發送的事件內容，再將其內容儲存至資料庫中，以利後續搜索其事件內容。

第四章 系統實作

本研究依第三章所述之相關研究作為延伸，於以太坊區塊鏈平台之智能合約實作數位貨幣交易記錄與查詢服務，實踐合約導向程式語言 Solidity [16]建構資料儲存與資料搜索機制，使用者能直接對於該智能合約所儲存之數據進行搜索，並透過智能合約結合 Oraclize (Provable) 服務之跨鏈技術作為提供區塊鏈數位貨幣交易記錄與查詢之服務，利於後續搜索區塊鏈之實務應用發展之需求。

本章共分為三小節，第一小節闡述數位貨幣溯源服務之系統架構；第二小節介紹智能合約相關設計；第三小節說明 Web 後端設計。

4.1 實作系統架構

本實驗架構主要由以太坊公有鏈 (Main Chain) 與私有鏈 (Private Chain)、多位使用者、一個 Time Oracle、以及建立服務於私有鏈之 Control Tower 智能合約、及其 Token Tracer 智能合約、與使用 Oraclize (Provable) 服務於私有鏈環境下需使用之 Ethereum-Bridge 工具，以及提供使用者 Web UI 操作本服務，圖 4.1 為實作系統架構圖。

為實現數位貨幣記錄與查詢之服務於以太坊區塊鏈，使用 go-ethereum (geth) [17]工具建立數個節點 (enode)，其中一個節點架設 NodeJS [18]伺服器負責區塊鏈與使用者之溝通。利用前一小節所述之智能合約設計實作系統，Token Tracer 透過 Oraclize (Provable) 服務，並利用 Etherscan 提供之 API 取得相關歷史交易資訊，並將其資訊儲存至合約當中，以利使用者之後續查詢。

Time Oracle 為獨立於以太坊區塊鏈外的 NodeJS 伺服器，每隔一段時間，取得儲存於 Control Tower 智能合約中之 Token Tracer 合約位址，並觸發所有 Token Tracer 中之 Oraclize (Provable) 服務，且由外部之 Ethereum-Bridge 監聽由

Oraclize (Provable) 服務之 Event，調用該請求之 API 以取得公有鏈 (Main Chain) 上之交易紀錄，以確保 Token Tracer 同步歷史交易之功能不會中斷。

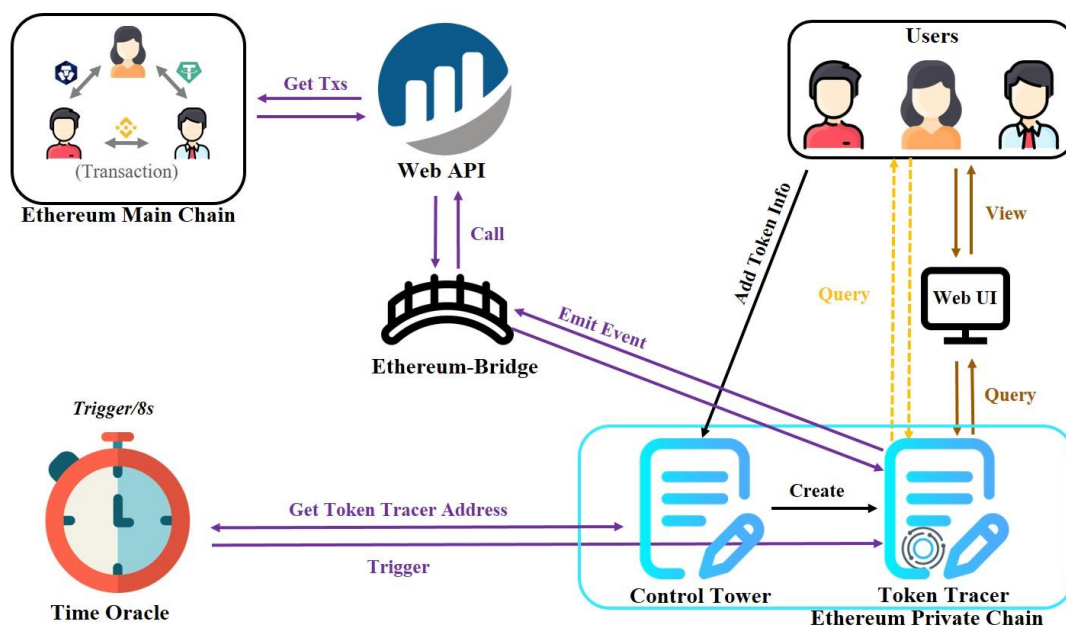


圖 4.1：實作系統架構圖

4.2 智能合約設計

本節共分為五部分，第一部分說明本實作之智能合約架構；第二部分介紹 Oraclize (Provable) 服務與 Ethereum-Bridge 之使用方式；第三部分說明擷取之交易資料處理方法；第四部份描述 Query 數位貨幣歷史交易紀錄之功能；第五部分為實作 Binary Search 之實驗結果。

4.2.1 智能合約架構

本實驗設計之智能合約架構主要有一 Control Tower 智能合約，與多個 Token Tracer 智能合約，而所有的 Token Tracer 智能合約皆由 Control Tower 建立而成，如圖 4.2 所示。以下將說明兩智能合約的關聯性與功能：

- (一) Control Tower：使用者透過此智能合約新增欲追蹤之數位貨幣資訊（如 USDT），在其新增後，Control Tower 將自動生成一 Token Tracer 智能合

約，同時，將使用者新增之數位貨幣資訊紀錄儲存於 Token Tracer，而 Token Tracer 也擁有 Oraclize (Provable) 服務，並透過 Oraclize (Provable) 服務取得該數位貨幣之歷史交易紀錄，再將其交易紀錄儲存於 Token Tracer 中。Control Tower 也將儲存所有已建立之 Token Tracer 合約位址。

(二) Token Tracer：此智能合約由 Control Tower 建立而成。Token Tracer 結合 Oraclize (Provable) 服務，於本智能合約建立時所記錄之數位貨幣資訊，透過設定其 Oraclize (Provable) 服務，存取其特定數位貨幣（如 USDT）歷史交易。

透過此智能合約設計架構，我們將每一種數位貨幣之歷史交易紀錄分別儲存於獨立的智能合約中，使得使用者能夠更容易取得與查詢特定之數位貨幣之歷史交易紀錄。

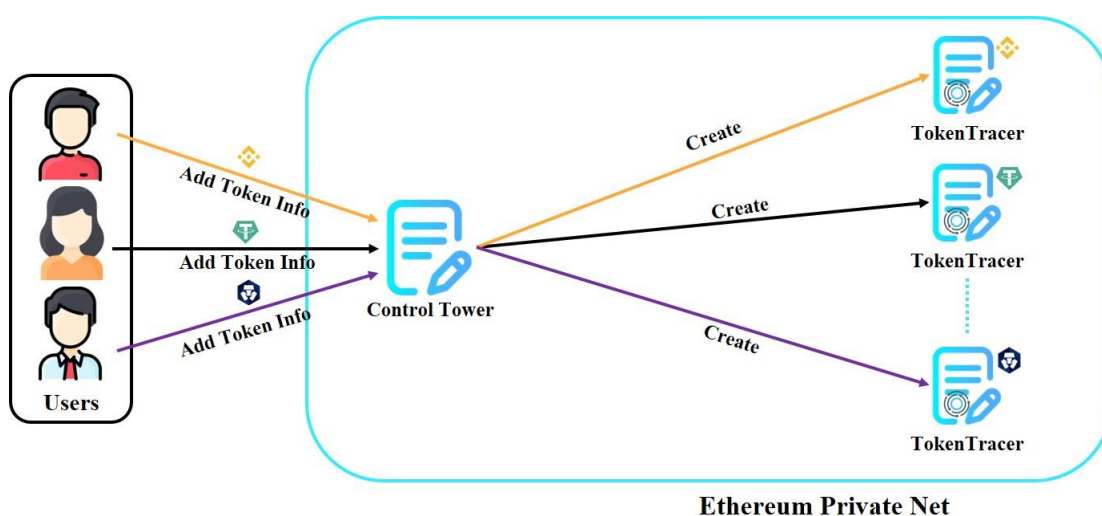


圖 4.2：智能合約設計架構圖

4.2.2 Oraclize (Provable) 服務與 Ethereum-Bridge 介紹

4.2.2.1 Oraclize (Provable) 服務

Oraclize (Provable) 服務是一個兼容以太坊智能合約的數據傳送服務，能夠自定義智能合約，在區塊鏈與真實世界之間建立一可信的通道，在保證可信的情況下，使其可透過 API 取得以太坊外部真實世界的資訊。

```
1.      // oraclize results
2.      function __callback(bytes32 myid, string memory _result) public {
3.          if (msg.sender != provable_cbAddress()) revert();
4.
5.          oraclizeIsRunning = false;
6.          // 檢查是否有回傳值
7.          if (bytes(_result).length != 0) {
8.              savingTx(_result);
9.          }
10.     }
11.
12.     // call oraclize
13.     function traceTx() payable public {
14.         uint gasLimit = 50000000;
15.         oraclizeIsRunning = true;
16.
17.         // 設定為每次 Oraclize 取得的交易筆數[:Count]
18.         string memory apiStr1 = "json(https://api.etherscan.io/api?module=logs&action=getLogs&fromBlock=";
19.         string memory apiStr2 = "&toBlock=latest";
20.         string memory apiStr3 = "&address=0x";
21.         string memory apiStr4 = "&topic0=0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef";
22.         string memory apiStr5 = "&apikey=HTI3IX924Z1IBXIIN4992VRAPKHJI149AX).result[";
23.         string memory apiStr6 = "][transactionHash, blockNumber, timeStamp, topics, data]";
```



```

24.         string memory apiUr1 = string(abi.encodePacked(apiStr1, uint2str(sync
           cBlockHeight), apiStr2, apiStr3, Parser.parseAddrressToString(tokenContract)
           , apiStr4, apiStr5, uint2str(syncIndex), ":", uint2str(syncIndex + 50), apiS
           tr6));
25.         provable_query("URL", apiUr1, gasLimit);
26.     }

```

上述程式碼為 Token Tracer 的部分片段，以下將針對（一） Oraclize （Provable）服務結合 API 之使用方法、（二）Token Tracer 存取交易紀錄方式，進行說明：

（一） Oraclize （Provable）服務結合 API 之使用方法：於 Token Tracer 中，我們將透過觸發 tracerTx 函式以調用 Oraclize （Provable）服務的 provable_query 功能，並使用由 Etherscan 所提供之 Logs APIs [18]，存取特定之數位貨幣交易紀錄；如程式碼第 18 至 24 行所示，並參考圖 4.3 Etherscan API 範例，是為組成 API 之過程，透過變數 syncBlockHeight 設定存取之起始區塊高度，並依區塊高度由小到大進行存取交易紀錄；如程式碼第 18 行所示，利用 Oraclize （Provable）服務提供之內建 JSON Parser；如程式碼第 24 行所示，在本次實作中，我們每次將利用變數 syncIndex 設定每次存取之交易紀錄筆數（在本實作中，每次將存取 50 筆交易紀錄），並設定欲取得之交易紀錄內容（transactionHash、blockNumber、timeStamp、topics 與 data），最後再其交易紀錄透過 JSON 資料格式回傳至 Token Tracer 中。

Get Event Logs from block number 379224 to 'latest' Block, where log address = 0x33990122638b9132ca29c723bdf037f1a891a70c and topic[0] = 0xf63780e752c6a54a94fc52715dbc5518a3b4c3c2833d301a204226548a2a8545

```

https://api.etherscan.io/api?module=logs&action=getLogs
&fromBlock=379224
&toBlock=latest
&address=0x33990122638b9132ca29c723bdf037f1a891a70c
&topic0=0xf63780e752c6a54a94fc52715dbc5518a3b4c3c2833d301a204226548a2a8545
&apiKey=YourApiKeyToken

```

設定區塊高度範圍

數位貨幣位址

欲取得之交易類型

圖 4.3：Etherscan API 範例

（二）Token Tracer 存取交易紀錄方式：透過 Oraclize （Provable）服務所取得之交易紀錄，將會由 Ethereum-Bridge 調用 Token Tracer 中的 __callback 函

式將交易紀錄傳入並儲存（如程式碼第 3 至 13 行所示）；如程式碼第 8 行所示，取得之回傳結果為 JSON 資料格式，需再透過 savingTx 函式進行資料剖析，並轉換其資料型態再儲存於 Token Tracer 中。因此，有了 Oraclize（Provable）服務，當以太坊內部需要外部資訊時，可以透過 API 使用，並寫好相對應之__callback 函式即可。

4.2.2.2 Ethereum-Bridge

本實驗環境為以太坊私有鏈，我們必須透過 Ethereum-Bridge [19]作為 Oraclize（Provable）服務對外部世界溝通的橋樑。啟動 Ethereum-Bridge 後，它則會開始監聽鏈上 Oraclize（Provable）服務發出的特定 Event。作為橋樑的角色，每當它接收到 Event 時，於區塊鏈外部執行該 API 請求，並將最終結果傳回該智能合約中。

4.2.3 資料處理方式

因透過 Oraclize（Provable）服務所取得之交易紀錄為 JSON 資料格式，因此，Token Tracer 繼承了「jsmnSol」功能（此工具為 GitHub 上之開源專案，並以此進行資料處理，且為 Solidity 語言所開發之 JSON parser） [20]，使用它進行交易內容的剖析處理，並根據其交易內容之型態進行轉換，再將其儲存於 Token Tracer 當中，以利後續使用者進行交易查找。

```
1.    bytes32[] transactionHash;
2.    address[] sender;
3.    address[] receiver;
4.    uint[] value;
5.    uint[] blockNumber;
6.    uint[] timeStamp;
7.
```

```

8.      // 取得已儲存之交易筆數
9.      uint public transactionCount;
10.
11.     mapping(bytes32 => bool) private isExist;
12.
13.     function savingTx(string memory json) internal {
14.         uint returnValue;
15.         JsmnSolLib.Token[] memory tokens;
16.         uint actualNum;
17.
18.         // (returnValue, tokens, actualNum) = JsmnSolLib.parse(json, 9);
19.         (returnValue, tokens, actualNum) = JsmnSolLib.parse(json, 450);
20.
21.         // 迴圈設定每次 Oraclize 取得之交易筆數
22.         for (uint i = 0; i < 50; i++) {
23.             JsmnSolLib.Token memory a = tokens[1 + 8*i];
24.             bytes32 _transactionHash = Parser.parseStringTo32Bytes(JsmnSolLib
                b.getBytes(json, a.start, a.end));
25.
26.             // 避免重複紀錄同一筆交易
27.             if (!isExist[_transactionHash] && _transactionHash != "") {
28.                 isExist[_transactionHash] = true;
29.                 transactionHash.push(_transactionHash);
30.                 JsmnSolLib.Token memory b = tokens[2 + 8*i];
31.                 uint _blockNumber = Parser.parseHexToUint256(JsmnSolLib.getBytes(
                    json, b.start, b.end));
32.                 blockNumber.push(_blockNumber);
33.                 JsmnSolLib.Token memory c = tokens[3 + 8*i];
34.                 timeStamp.push(Parser.parseHexToUint256(JsmnSolLib.getBytes(
                    json, c.start, c.end)));
35.                 JsmnSolLib.Token memory d = tokens[6 + 8*i];
36.                 sender.push(parseAddr(Parser.subString(JsmnSolLib.getBytes(j
                    son, d.start, d.end), 26, 66)));
37.                 JsmnSolLib.Token memory e = tokens[7 + 8*i];
38.                 receiver.push(parseAddr(Parser.subString(JsmnSolLib.getBytes(
                    json, e.start, e.end), 26, 66)));
39.                 JsmnSolLib.Token memory f = tokens[8 + 8*i];
40.                 value.push(Parser.parseHexToUint256(JsmnSolLib.getBytes(json
                    , f.start, f.end)));

```

```

40.
41.          // 更新下回開始搜尋之 blockNumber, 需避免同一 block 有多筆交易
42.          syncBlockHeight = _blockNumber;
43.          syncIndex = 0;
44.      } else if (_transactionHash == "") {
45.          syncIndex = 0;
46.          break;
47.      }
48.  }
49.  if (transactionCount == transactionHash.length) {
50.      // 同一 block 有超過 50 筆交易, 從後 50 筆開始找, 以此類推
51.      syncIndex += 50;
52.  }
53.  transactionCount = transactionHash.length;
54.  }

```

上述程式碼為 Token Tracer 的部分片段，以下將說明 JSON 資料格式的剖析處理與儲存之過程：如圖 4.4 所示，此為回傳之其一交易紀錄內容（其資料型態為 JSON 格式），而內容包含 transactionHash、blockNumber、timeStamp、topics（topics[0]為交易類型、topics[1]為交易發送方、topics[2]為交易接收方）與 data 等資料內容。如程式碼第 22 至 39 行所示，利用「jsmnSol」功能剖析其 JSON 資料，該功能乃利用字串切割之方式讀取其資料內容，故以圖 4.4 為例，我們欲取得之 transactionHash 為該內容之第一個字串片段、blockNumber 為第二個資料片段，以此類推，交易發送方與接收方則為字串切割後之第六個與第七個字串片段。下一步，我們將資料切割後之字串內容進行資料型態轉換並儲存。如圖 4.5 所示，該將為透過 Oraclize（Provable）服務取得之交易內容，經資料處理過後，交易內容的排序將由區塊的高度，並由小到大儲存於 Token Tracer。最後，如程式碼第 42 行所示，Token Tracer 將更新下一次存取之起始區塊高度。

「區塊範圍搜索」、「時間範圍搜索」、「交易發送方位址查詢」與「交易接收方位址查詢」等功能：

- (一) 檢索所有交易記錄：在此功能中，為避免每次回覆之交易筆數過多、等待回覆時間太長，如下方程式碼所示，設定其每次至多回傳 100 筆交易紀錄，並同時回傳該次搜尋所查詢之 checkPoint，以利下一回合可由該 checkPoint 繼續往下搜尋。其回傳之交易內容可參考圖 4.5 所示，其內容包含 Transaction Hash、Block、Time、From、To 與 Quantity 等交易內容。。

```
1. // 取得查詢交儲存之交易結果 (100 txns)
2. function token_query(uint checkPoint) public view returns(uint _checkPoint,
   bytes32[] memory _transactionHash, address[] memory _sender, address[] memory
   _receiver, uint[] memory _value, uint[] memory _blockNumber, uint[] memory _tim
   estamp) {
3.     uint count = 100;
4.     if (checkPoint + 100 > transactionCount) {
5.         count = transactionCount - checkPoint;
6.     }
7.
8.     _transactionHash = new bytes32[](count);
9.     _sender = new address[](count);
10.    _receiver = new address[](count);
11.    _value = new uint[](count);
12.    _blockNumber = new uint[](count);
13.    _timestamp = new uint[](count);
14.    for (uint i = 0; i < count; i++) {
15.        _transactionHash[i] = transactionHash[checkPoint + i];
16.        _sender[i] = sender[checkPoint + i];
17.        _receiver[i] = receiver[checkPoint + i];
18.        _value[i] = value[checkPoint + i];
19.        _blockNumber[i] = blockNumber[checkPoint + i];
20.        _timestamp[i] = timeStamp[checkPoint + i];
21.    }
22.    _checkPoint = checkPoint + count;
23. }
```

(二) 區塊範圍搜索與時間範圍搜索：在此功能中，根據使用者所輸入之區塊範圍或時間範圍進行搜索，並將符合條件（使用者可針對該區塊範圍內之特定發送方或接收方位址進行過濾）之結果回傳。以下程式碼以區塊範圍搜索為例，如程式碼第 5 行所示，該 Array.findUpperBound 函式乃是採用 Binary Search 之方法，取得其使用者欲搜索之最低區塊高度於 blockNumber 陣列中之 checkPoint，其後再根據其他搜索條件進比對；如程式碼第 8 行所示，為避免回覆之交易筆數過多、等待回覆時間太長，設定每查找 100 筆交易後，即回傳搜尋結果。其回傳之交易內容可參考圖 4.5 所示，其內容包含 Transaction Hash、Block、Time、From、To 與 Quantity 等交易內容。

```
1. function token_queryBlock(uint startBlock, uint endBlock, address _from, address _to, uint checkPoint) public view returns(uint _checkPoint, bytes32[] memory _transactionHash, address[] memory _sender, address[] memory _receiver, uint[] memory _value, uint[] memory _blockNumber, uint[] memory _timestamp) {
2.     uint size;
3.     uint[] memory matchIndex = new uint[](transactionHash.length);
4.     if (checkPoint == 0) {
5.         checkPoint = Arrays.findUpperBound(blockNumber, startBlock);
6.     }
7.
8.     for (_checkPoint = checkPoint; _checkPoint < transactionCount && _checkPoint < checkPoint + 100; _checkPoint++) {
9.         if (endBlock >= blockNumber[_checkPoint]) {
10.             // Query for Sender
11.             if (msg.sender == _to && sender[_checkPoint] == _from) {
12.                 matchIndex[size] = _checkPoint;
13.                 size++;
14.             }
15.             // Query for Receiver
16.             else if (msg.sender == _from && receiver[_checkPoint] == _to) {
17.                 matchIndex[size] = _checkPoint;
18.                 size++;
19.             }
20.         }
21.     }
22. }
```

```

20.          // Query for Both
21.          else if (_from == sender[_checkPoint] && _to == receiver[_check
    Point]) {
22.              matchIndex[size] = _checkPoint;
23.              size++;
24.          }
25.          // Only query for block
26.          else if (msg.sender == _from && msg.sender == _to) {
27.              matchIndex[size] = _checkPoint;
28.              size++;
29.          }
30.          } else {
31.              break;
32.          }
33.      }
34.
35.      _transactionHash = new bytes32[](size);
36.      _sender = new address[](size);
37.      _receiver = new address[](size);
38.      _value = new uint[](size);
39.      _blockNumber = new uint[](size);
40.      _timestamp = new uint[](size);
41.      for (uint i = 0; i < size; i++) {
42.          _transactionHash[i] = transactionHash[matchIndex[i]];
43.          _sender[i] = sender[matchIndex[i]];
44.          _receiver[i] = receiver[matchIndex[i]];
45.          _value[i] = value[matchIndex[i]];
46.          _blockNumber[i] = blockNumber[matchIndex[i]];
47.          _timestamp[i] = timeStamp[matchIndex[i]];
48.      }
49.  }

```

(三) 交易發送方位址查詢與交易接收方位址查詢：在此功能中，根據使用者所輸入之交易發送方位址或交易接收方位址進行搜索，並將符合條件之結果回傳。如程式碼第 4 行所示，此查詢的方法為從紀錄中的第一筆資料進行比對，為避免回覆之交易筆數過多、等待回覆時間太長，設定每查找

100 筆交易後，即回傳搜尋結果。其回傳之交易內容可參考圖 4.5 所示，其內容包含 Transaction Hash、Block、Time、From、To 與 Quantity 等交易內容。

```
1. function token_queryAccount(address _from, address _to, uint checkPoint) public
   view returns(uint _checkPoint, bytes32[] memory _transactionHash, address[] me
   memory _sender, address[] memory _receiver, uint[] memory _value, uint[] memory _
   blockNumber, uint[] memory _timestamp) {
2.     uint size;
3.     uint[] memory matchIndex = new uint[](transactionHash.length);
4.     for (_checkPoint = checkPoint; _checkPoint < transactionCount && _check
       Point < checkPoint + 100; _checkPoint++) {
5.         // Query for Sender
6.         if (msg.sender == _to && sender[_checkPoint] == _from) {
7.             matchIndex[size] = _checkPoint;
8.             size++;
9.         }
10.        // Query for Receiver
11.        else if (msg.sender == _from && receiver[_checkPoint] == _to) {
12.            matchIndex[size] = _checkPoint;
13.            size++;
14.        }
15.        // Query for Both
16.        else {
17.            if (sender[_checkPoint] == _from && receiver[_checkPoint] == _t
               o) {
18.                matchIndex[size] = _checkPoint;
19.                size++;
20.            }
21.        }
22.    }
23.
24.    _transactionHash = new bytes32[](size);
25.    _sender = new address[](size);
26.    _receiver = new address[](size);
27.    _value = new uint[](size);
28.    _blockNumber = new uint[](size);
```

```

29.         _timestamp = new uint[](size);
30.         for (uint i = 0; i < size; i++) {
31.             _transactionHash[i] = transactionHash[matchIndex[i]];
32.             _sender[i] = sender[matchIndex[i]];
33.             _receiver[i] = receiver[matchIndex[i]];
34.             _value[i] = value[matchIndex[i]];
35.             _blockNumber[i] = blockNumber[matchIndex[i]];
36.             _timestamp[i] = timeStamp[matchIndex[i]];
37.         }
38.     }

```

4.2.5 Query 實驗結果

本實驗對 Token Tracer 中實作 Binary Search 演算法提升使用者查詢效能進行測試。我們將測試環境運行在 Amazon 雲端平台上，測試的雲端虛擬機器為 Amazon EC2 t3.xlarge。t3.xlarge 的硬體規格為 4 個虛擬 CPU 與 16GB 的記憶體。

在此，我們將探討一般搜索與 Binary Search 下查詢不同區塊範圍時，觀察資料回傳時間的變化。對於此測試，我們預先儲存 100000 筆資料於智能合約當中，其儲存資料之區塊高度為 1 的有 10 筆資料、高度為 2 的有 10 筆資料，以此類推，最高的區塊高度為 10000 的有 10 資料。實驗總共進行了七種搜索測試，分別為區塊高度為 10、50、100、500、1000、5000、10000。預儲存之資料格式內容可參照圖 4.5。

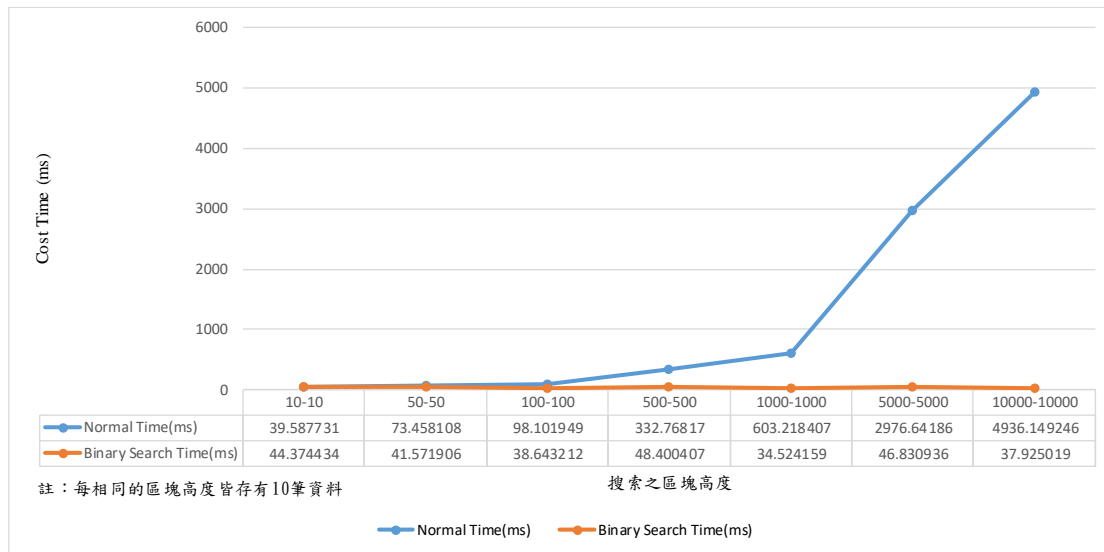


圖 4.6：比較 Binary Search 搜索效能

從圖 4.6 中可以看到使用一般搜索方式時，其查詢資料之回傳時間，在隨著搜索區塊高度的增加，回傳時間也隨之增加；然而，加入 Binary Search 搜索方法後，其實驗結果顯示，資料回傳時間不隨著搜索區塊高度的增加而增長。

4.3 Web 後端設計

本實驗也提供網站介面使其使用者容易操作其去中心化數位貨幣交易記錄與查詢服務。使用者可於該網頁中進行區塊範圍、時間範圍、交易發送方、交易接收方或是特定數位貨幣等歷史交易紀錄之查詢功能。為此，我們將透過 Express（NodeJS Web 應用程式架構）撰寫 Backend Blockchain APIs 對其 Token Tracer 所儲存之歷史交易紀錄進行查詢，並將查詢結果呈現於前端中。

透過使用者於前端所設定之查詢條件，我們透過 jQuery 方式取得網頁資訊，再將其內容傳遞至後端，並調用 API 與 Token Tracer 進行溝通以取得欲查詢之結果。為避免搜尋之結果筆數過大，可能造成區塊鏈負荷過大或是查詢等待時間過長等，因此，設定為每 2 秒取得新的查詢結果，而每次查詢結果至多回傳 100 筆資料，並透過 AJAX（Asynchronous JavaScript and XML）技術自動更新前端之查詢結果，藉此優化使用者使用體驗。

第五章 系統操作介面與使用流程

為了讓使用者容易操作其去中心化數位貨幣交易記錄與查詢服務，本研究也為此 Dapp（Decentralized Application）設計其介面，提升使用時之便利性。

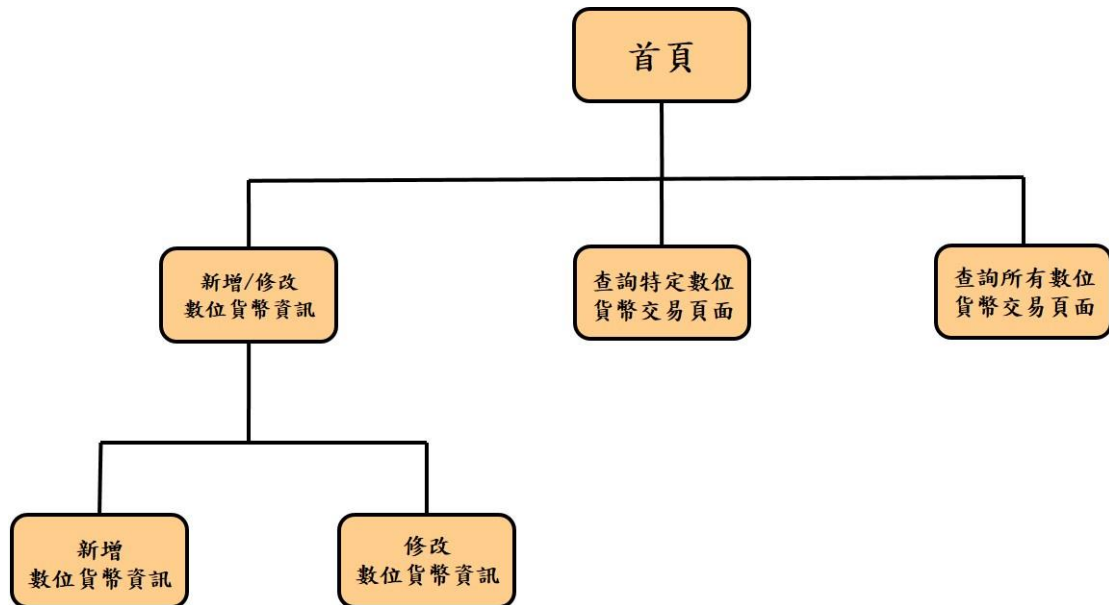


圖 5.1：網頁架構圖

以下將分別介紹「數位貨幣交易記錄與查詢服務」功能，實作示範新增數位貨幣之合約位址與相關資訊、修改數位貨幣之相關資訊、查找特定數位貨幣之交易紀錄以及查找所有已新增之數位貨幣之交易紀錄的實作結果做說明：

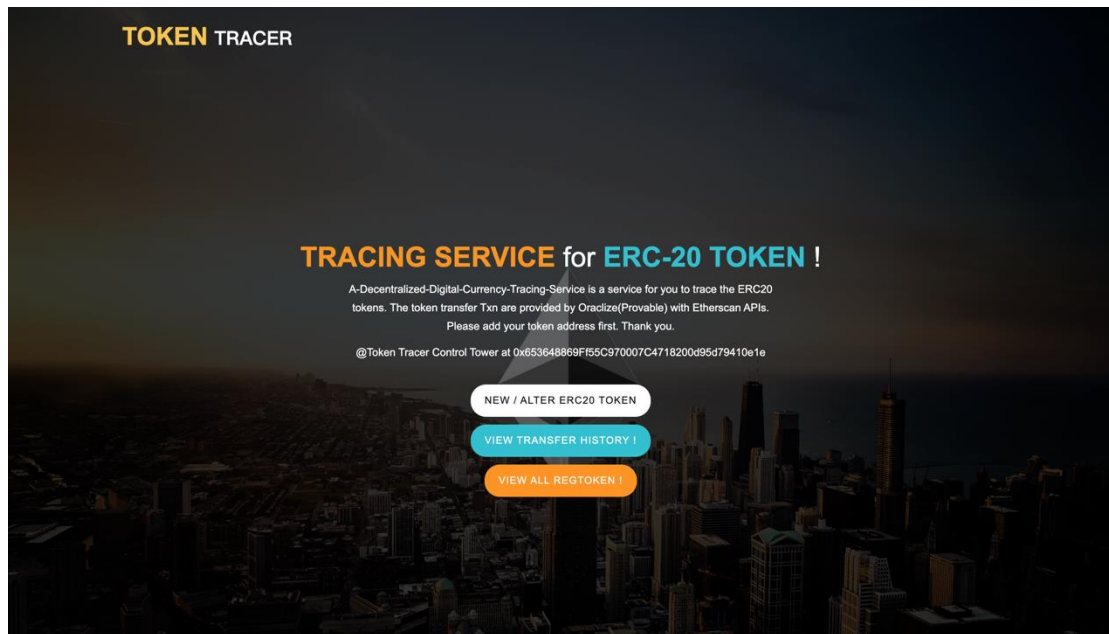


圖 5.2：數位貨幣交易記錄與查詢服務之首頁畫面

5.1 新增數位貨幣之合約位址與相關資訊

本節將說明使用數位貨幣交易記錄與查詢服務之流程，其註冊之操作流程簡述如下：

(一) 透過首頁之「NEW / ALTER ERC-20 TOKEN」之按鈕，進入註冊服務主頁，如圖 5.3。

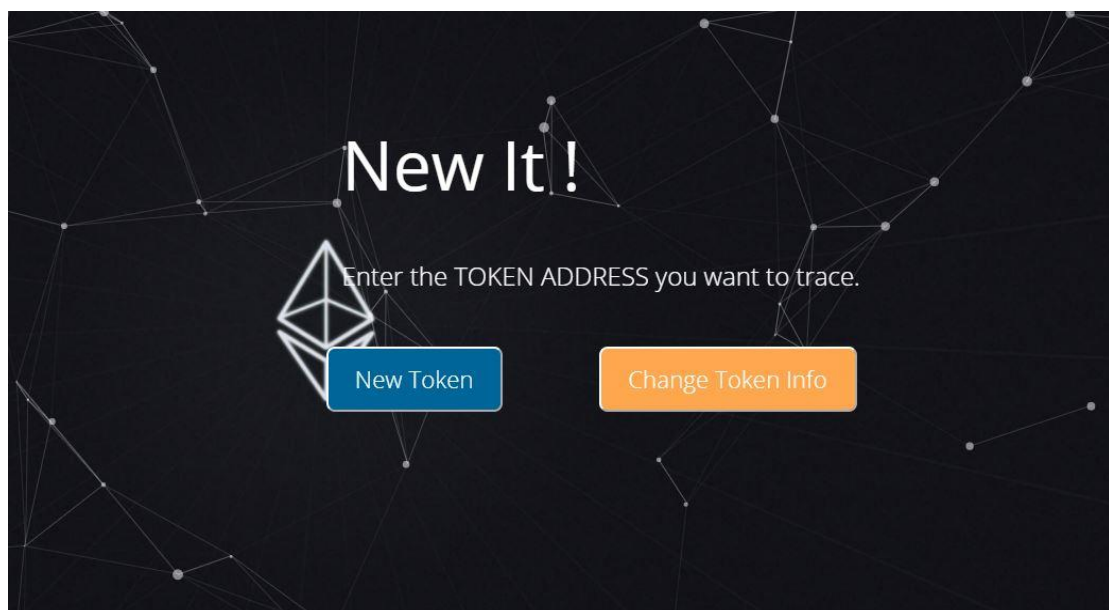
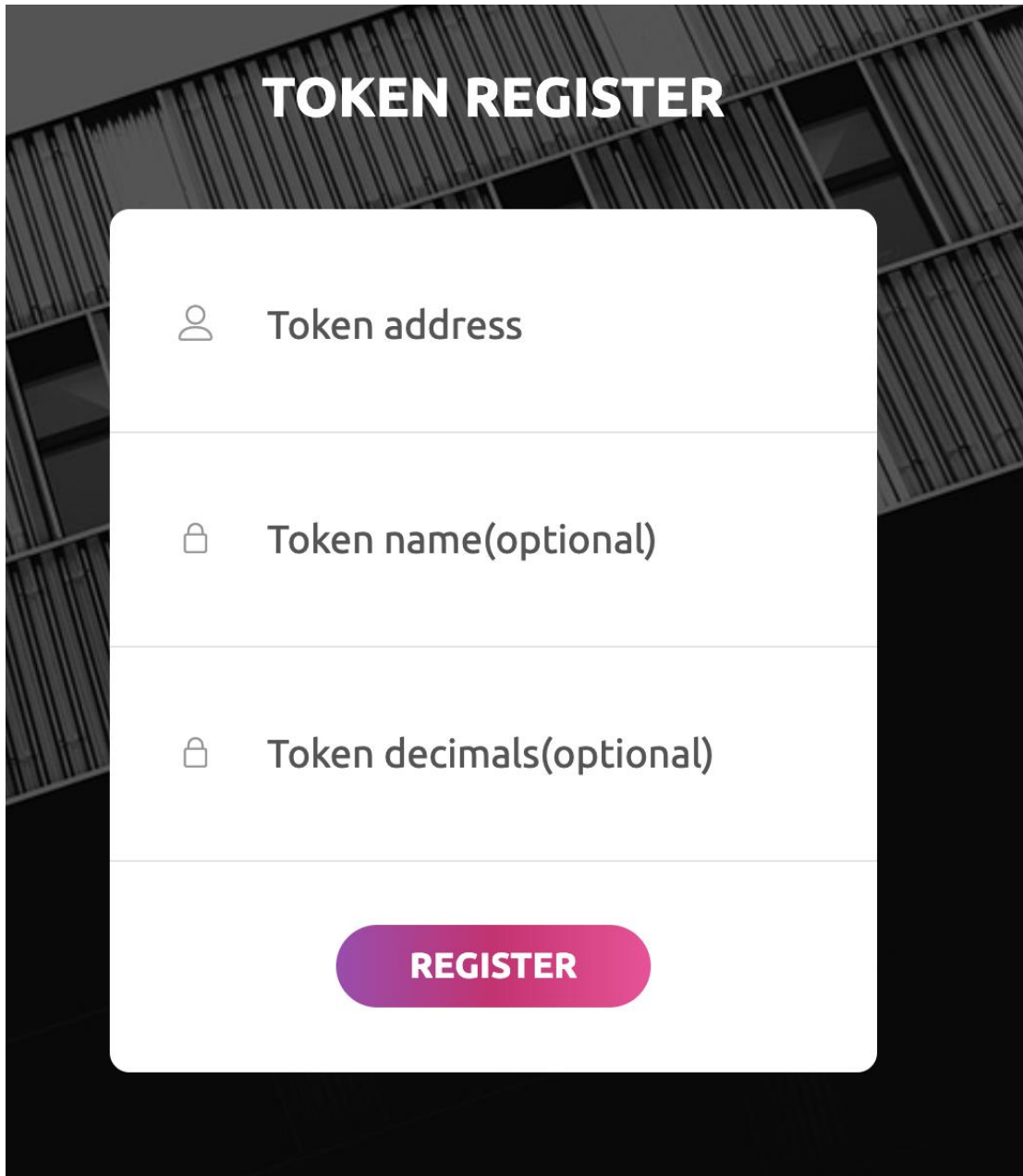



圖 5.3：新增／修改數位貨幣資訊主頁


(二) 按下註冊主頁上之「New Token」之按鈕，進入代幣註冊頁面，如圖 5.4 所示。




TOKEN REGISTER



Token address



Token name(optional)



Token decimals(optional)

REGISTER

圖 5.4：新增數位貨幣資訊頁面

(三) 填寫欲進行數位貨幣交易記錄與查詢服務之 ERC-20 代幣之合約位址、代幣名稱和代幣最小單位並送出，其完成新增代幣歷史交易記錄與查詢服務之畫面如圖 5.5 所示。

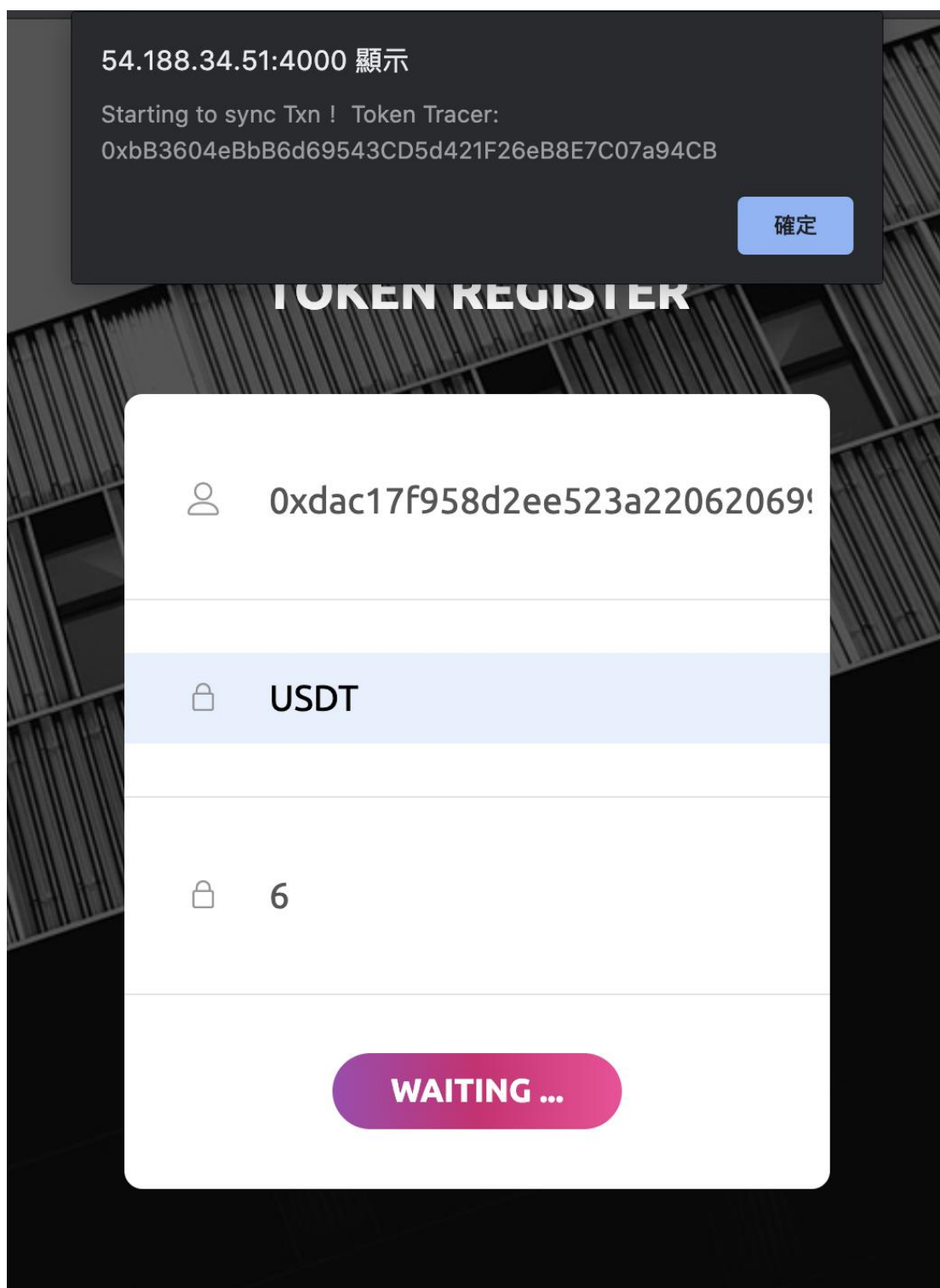


圖 5.5：成功新增數位貨幣資訊之畫面

(四) 新增完成後，可至交易查找之頁面進行確認，如圖 5.6 所示。

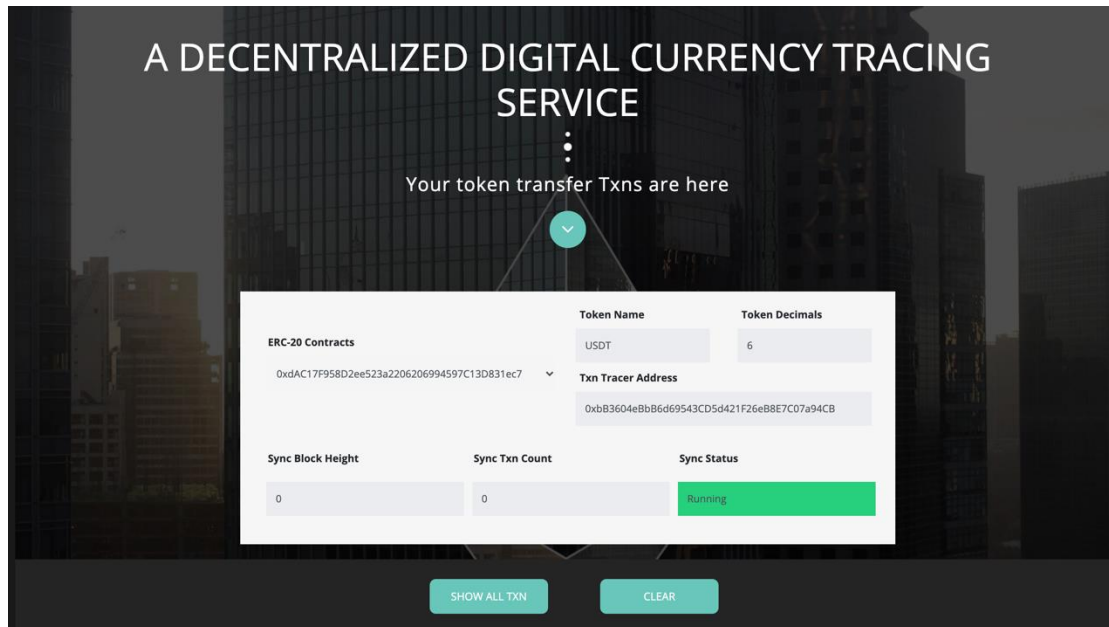


圖 5.6：查詢數位貨幣歷史交易頁面

5.2 修改數位貨幣之相關資訊

使用者完成註冊服務後，仍可針對代幣之名稱與單位進行變更，其操作流程簡述如下：

(一) 透過註冊主頁上之「Change Token Info」之按鈕進入後，顯示代幣資訊修改頁面，並將其內容更新後按下送出即可，如圖 5.7 所示。

Change Token Info

ERC-20 Tokens

0xdAC17F958D2ee523a22062069 ✓

Name

USDT

Decimal

6

CONFIRM

圖 5.7：修改數位貨幣資訊畫面

SHOW ALL TXN

CLEAR

Txn Hash	Block	Date Time	From	To	Quantity
0x51a23950874503...	4638568	Tue Nov 28 2017 23:...	0x369285008c1dCd...	0xC6CDE7C39eB2f0...	100
0x2c22ad3e367ee7...	4638577	Tue Nov 28 2017 23:...	0xC6CDE7C39eB2f0...	0x2b2c9153Acc4021...	10
0x4afc4e7f282e9de...	4638586	Tue Nov 28 2017 23:...	0x369285008c1dCd...	0xC6CDE7C39eB2f0...	99900
0x35348a160fd8b8...	4638663	Wed Nov 29 2017 00:...	0xC6CDE7C39eB2f0...	0x2b2c9153Acc4021...	10
Transaction Hash: 0x35348a160fd8b86adea49ff482fd9d0185d544206fa54c31bd0e51fd4391de Block Number: 4638663 Time: Wed Nov 29 2017 00:03:30 GMT+0800 (台北標準時間) From: 0xC6CDE7C39eB2f0F0095F41570af89eFC2C1Ea828 To: 0x2b2c9153Acc4021676C165DC3e3B1c7AD1C7849D Quantity: 10					
0xe544a6ac188ea9...	4638700	Wed Nov 29 2017 00:...	0x2b2c9153Acc4021...	0x9faf5515f177F3A8...	20
0xbc5f66220f56f24...	4638705	Wed Nov 29 2017 00:...	0xC6CDE7C39eB2f0...	0x2b2c9153Acc4021...	10
0xbf3037e883633fa...	4638713	Wed Nov 29 2017 00:...	0x2b2c9153Acc4021...	0x9faf5515f177F3A8...	10
0xf68518104b27f00...	4638894	Wed Nov 29 2017 01:...	0xC6CDE7C39eB2f0...	0x2b2c9153Acc4021...	100
0x6c809cb988fb88a...	4638924	Wed Nov 29 2017 01:...	0x2b2c9153Acc4021...	0x9faf5515f177F3A8...	100
0x76ea50785162d5...	4773369	Fri Dec 22 2017 06:1:...	0xC6CDE7C39eB2f0...	0x5A9cf70048dB6f7...	20
0xb7f200ca2bd1d2...	4805948	Wed Dec 27 2017 18:...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	0.1
0xdcfa773ee6469b2...	4806039	Wed Dec 27 2017 18:...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	0.11
0x17800301a62337...	4806074	Wed Dec 27 2017 18:...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	0.12
0x4ec148525312463...	4806105	Wed Dec 27 2017 19:...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	0.13
0x872f02e878628a7...	4806190	Wed Dec 27 2017 19:...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	0.13
0x183c82bd1ca377...	4806797	Wed Dec 27 2017 21:...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	1
0xf18c822657f1264...	4806958	Wed Dec 27 2017 22:...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	1.00001

圖 5.9：詳細的交易內畫面

(二) 區塊範圍篩選：可訂定其區塊範圍篩選交易紀錄，如圖 5.10 與圖 5.11 所示。

SHOW ALL TXN

CLEAR

Please enter the block height to query

fromBlock : 5000000

toBlock : 6000000

filter clear

Txn Hash	Block	Date Time	From	To	Quantity
0x51a23950874503...	4638568	Tue Nov 28 2017 23:...	0x369285008c1dCd...	0xC6CDE7C39eB2f0...	100
0x2c22ad3e367ee7...	4638577	Tue Nov 28 2017 23:...	0xC6CDE7C39eB2f0...	0x2b2c9153Acc4021...	10
0x4afc4e7f282e9de...	4638586	Tue Nov 28 2017 23:...	0x369285008c1dCd...	0xC6CDE7C39eB2f0...	99900
0x35348a160fd8b8...	4638663	Wed Nov 29 2017 00:...	0xC6CDE7C39eB2f0...	0x2b2c9153Acc4021...	10
0xe544a6ac188ea9...	4638700	Wed Nov 29 2017 00:...	0x2b2c9153Acc4021...	0x9faf5515f177F3A8...	20
0xbc5f66220f56f24...	4638705	Wed Nov 29 2017 00:...	0xC6CDE7C39eB2f0...	0x2b2c9153Acc4021...	10
0xbf3037e883633fa...	4638713	Wed Nov 29 2017 00:...	0x2b2c9153Acc4021...	0x9faf5515f177F3A8...	10
0xf68518104b27f00...	4638894	Wed Nov 29 2017 01:...	0xC6CDE7C39eB2f0...	0x2b2c9153Acc4021...	100
0x6c809cb988fb88a...	4638924	Wed Nov 29 2017 01:...	0x2b2c9153Acc4021...	0x9faf5515f177F3A8...	100

圖 5.10：訂定區塊範圍篩選之畫面

Txn Hash	Block	Date Time	From	To	Quantity
0x0881d7b0df1a2ef...	5000014	Tue Jan 30 2018 21:4...	0x532110357caf435...	0x05EE546c1a62f90...	217.515476
0x270210fc6287688...	5000015	Tue Jan 30 2018 21:4...	0x404c417f95Cbb77...	0x05EE546c1a62f90...	12000002.58
0x1a34b8c7e4d5b1...	5000015	Tue Jan 30 2018 21:4...	0x86940152DfbCcC1...	0x05EE546c1a62f90...	9259
0x33b81ab785f43b...	5000040	Tue Jan 30 2018 21:5...	0x876EabF441B2EE5...	0x404c417f95Cbb77...	1000000.29
0xb04476ccd9fb07c...	5000043	Tue Jan 30 2018 21:5...	0x876EabF441B2EE5...	0x404c417f95Cbb77...	700000.2
0x69084d85f394f09...	5000045	Tue Jan 30 2018 21:5...	0xA29114Bdc94a74...	0x876EabF441B2EE5...	64860
0x4546105326fabd...	5000056	Tue Jan 30 2018 21:5...	0x05EE546c1a62f90...	0xA29114Bdc94a74...	29455
0xe955d70079b3fd...	5000064	Tue Jan 30 2018 21:5...	0xA29114Bdc94a74...	0x876EabF441B2EE5...	29455
0xddb5f8544cdae4...	5000188	Tue Jan 30 2018 22:2...	0xa30D8157911ef23...	0x4a782b945500ac6...	327
0x6c0fd02f1f9c20a...	5000207	Tue Jan 30 2018 22:3...	0xa30D8157911ef23...	0x4a782b945500ac6...	2600.9299
0xc8f34419e30578...	5000217	Tue Jan 30 2018 22:3...	0x05EE546c1a62f90...	0xA29114Bdc94a74...	99990
0x00c0c4b4e8f9453...	5000241	Tue Jan 30 2018 22:3...	0xA29114Bdc94a74...	0x876EabF441B2EE5...	99990
0x9c0ae8b356ec41...	5000408	Tue Jan 30 2018 23:2...	0xa30D8157911ef23...	0xD363b77A1E2CC9...	33935.120908
0x1ccb35553f8a83a...	5000504	Tue Jan 30 2018 23:4...	0xa30D8157911ef23...	0x73C8A0410c21652...	3657.87786
0x1ccb35553f8a83a...	5000504	Tue Jan 30 2018 23:4...	0xa30D8157911ef23...	0x73C8A0410c21652...	3657.87786
0x1ccb35553f8a83a...	5000504	Tue Jan 30 2018 23:4...	0xa30D8157911ef23...	0x73C8A0410c21652...	3657.87786
0xec84c18137fa69a...	5004396	Wed Jan 31 2018 15:...	0x17Bc58b788808D...	0xa30D8157911ef23...	1690
0x71426e753e59bd...	5004495	Wed Jan 31 2018 15:...	0x05EE546c1a62f90...	0x9837a1c2E68759...	2990
0x6b165953657c01...	5004559	Wed Jan 31 2018 16:...	0x05EE546c1a62f90...	0xB2FA68E7aa90A3...	239990
0x2d9de3dc25f3334...	5004577	Wed Jan 31 2018 16:...	0xB2FA68E7aa90A3...	0x876EabF441B2EE5...	239990

圖 5.11：區塊範圍篩選結果之畫面

(三) 日期範圍篩選：可訂定其日期範圍篩選交易紀錄，如圖 5.12 與圖 5.13 所示。

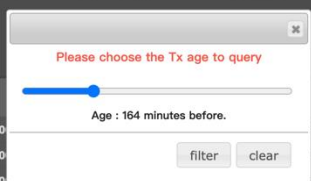
Txn Hash	Block	Date Time	From	To	Quantity
0x51a23950874503...	4773369	Fri Dec 22 2017 06:1...	0xC6CDE7C39eB2f0...	0x5A9cf70048dB6f7...	20
0x2c22ad3e367ee7...	4805948	Wed Dec 27 2017 18:...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	0.1
0x4afc4e7f282e9de...	4806039	Wed Dec 27 2017 18:...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	0.11
0x35348a160fd8b8...	4806074	Wed Dec 27 2017 18:...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	0.12
0xe544a6ac188ea9...	4806105	Wed Dec 27 2017 19:...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	0.13
0x5c5f66220f56f24...	4806190	Wed Dec 27 2017 19:...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	0.13
0xbf3037e883633fa...	4806797	Wed Dec 27 2017 21:...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	1
0xf68518104b27f00...	4806958	Wed Dec 27 2017 22:...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	1.00001
0x6c809cb988fb88a...	4806990	Wed Dec 27 2017 22:...	0xEF79e049D27F1d9...	0xD7C866d0D53693...	0.000002

圖 5.12：訂定日期範圍篩選之畫面

Txn Hash	Block	Date Time	From	To	Quantity
0xb7f200ca2bd1d2...	4805948	Wed Dec 27 2017 18...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	0.1
0xdcfaf773ee6469b2...	4806039	Wed Dec 27 2017 18...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	0.11
0x17800301a62337...	4806074	Wed Dec 27 2017 18...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	0.12
0x4ec148525312463...	4806105	Wed Dec 27 2017 19...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	0.13
0x872f02e878628a7...	4806190	Wed Dec 27 2017 19...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	0.13
0x183c82bd1ca377...	4806797	Wed Dec 27 2017 21...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	1
0xf18c822657f1264...	4806958	Wed Dec 27 2017 22...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	1.00001
0x8a62eaf1cc430d3...	4806990	Wed Dec 27 2017 22...	0xEF79e049D27F1d9...	0xD7C866d0D53693...	0.000002
0xdb573f4b290b45...	4807017	Wed Dec 27 2017 22...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	0.3
0x3ebe51bec060c...	4807045	Wed Dec 27 2017 22...	0xEF79e049D27F1d9...	0xD7C866d0D53693...	0.000002
0x5964ecec90b4d5...	4807062	Wed Dec 27 2017 22...	0xEF79e049D27F1d9...	0xD7C866d0D53693...	0.000002
0xfb2dd30718f8775...	4807082	Wed Dec 27 2017 22...	0x5A9cf70048dB6f7...	0xEF79e049D27F1d9...	0.4
0xf8bf62cb8ab0dfc...	4810661	Thu Dec 28 2017 13...	0xEF79e049D27F1d9...	0xD7C866d0D53693...	0.195827
0x90318f2cf8dda22...	4813389	Fri Dec 29 2017 00:2...	0xC6CDE7C39eB2f0...	0x876EabF441B2EE5...	1000

圖 5.13：日期範圍篩選結果之畫面


(三) 交易發生時間篩選：可訂定其發生之時間，之於使用者當下的確切時間進行篩選，如圖 5.14 所示。



Txn Hash	Block	Date Time	To	Quantity	Token
0x452b940cee13...	1141500	Wed Dec 09 2020...	0x1b331fC29433...	46.25	USDT
0x9d910b8572b...	1141500	Wed Dec 09 2020...	0xd47140F6Ab73...	2521.759943	USDT
0x60baa6fc8a9a...	1141500	Wed Dec 09 2020...	0x5c9f3ffF6ee84...	5.683745	USDT
0x73ef67778b3d...	11415000	Wed Dec 09 2020...	0xB36c33a14b34...	155095.492078	USDT
0x3d13cce13614...	11415000	Wed Dec 09 2020...	0xDab86C6c2667...	2393.968249	USDT
0x53f0f64e8dd8...	11415000	Wed Dec 09 2020...	0x75e89d5979E4...	1247.038887	USDT
0xa6d7e5f344b7...	11415000	Wed Dec 09 2020...	0x4b3d21f41dC7...	50.37	USDT
0x82c9b56a27a8...	11415000	Wed Dec 09 2020...	0xE4e83Fe65447...	20	USDT

圖 5.14：交易發生時間篩選之畫面

(四) 發送方位址或接收方篩選：可針對其特定發送方或接收方之位址篩選交易紀錄，如圖 5.15 與圖 5.16 所示。



Txn Hash	Block	Date Time	Quantity
0x51a23950874503...	4638568	Tue Nov 28 20...	100
0x2c22ad3e367ee7...	4638577	Tue Nov 28 20...	10
0x4afc4e7f282e9de...	4638586	Tue Nov 28 20...	99900
0x35348a160fd8b8...	4638663	Wed Nov 29 2017 00...	10
0xe544a6ac188ea9...	4638700	Wed Nov 29 2017 00...	20
0xbc5f66220f56f24...	4638705	Wed Nov 29 2017 00...	10
0xbf3037e883633fa...	4638713	Wed Nov 29 2017 00...	10
0xf68518104b27f00...	4638894	Wed Nov 29 2017 01...	100
0x6c809cb988fb8a...	4638924	Wed Nov 29 2017 01...	100
0x76ea50785162d5...	4773369	Fri Dec 22 2017 06:1...	20
0xb7f200ca2bd1d2...	4805948	Wed Dec 27 2017 18...	0.1
0xdcfaf773ee6469b2...	4806039	Wed Dec 27 2017 18...	0.11
0x17800301a62337...	4806074	Wed Dec 27 2017 18...	0.12

圖 5.15：特定位址篩選之畫面

Txn Hash	Block	Date Time	From	To	Quantity
0x2c22ad3e367ee7...	4638577	Tue Nov 28 2017 23:...	0xC6CDE7C39eB2f0...	0x2b2c9153Acc4021...	10
0x35348a160fd8b8...	4638663	Wed Nov 29 2017 00:...	0xC6CDE7C39eB2f0...	0x2b2c9153Acc4021...	10
0xb5c5f66220f56f24...	4638705	Wed Nov 29 2017 00:...	0xC6CDE7C39eB2f0...	0x2b2c9153Acc4021...	10
0xf68518104b27f00...	4638894	Wed Nov 29 2017 01:...	0xC6CDE7C39eB2f0...	0x2b2c9153Acc4021...	100
0x76ea50785162d5...	4773369	Fri Dec 22 2017 06:1...	0xC6CDE7C39eB2f0...	0x5A9cf70048dB6f7...	20
0x90318f2cf8dda22...	4813389	Fri Dec 29 2017 00:2...	0xC6CDE7C39eB2f0...	0x876EabF441B2EE5...	1000
0x1dfbbda58a1b32...	4951210	Mon Jan 22 2018 16:...	0xC6CDE7C39eB2f0...	0x876EabF441B2EE5...	30000000
0xce14f6760cf25ac...	4988877	Mon Jan 29 2018 00:...	0xC6CDE7C39eB2f0...	0x876EabF441B2EE5...	30000000

圖 5.16：特定位址篩選結果之畫面

5.4 查找所有已新增之數位貨幣之交易紀錄

有別於前一小節之功能，使用者完成新增數位貨幣資訊後，使用者即可針對當前服務之所有數位貨幣之交易紀錄進行查找。透過首頁之「VIEW ALL REGTOKEN！」之按鈕，進入查詢頁面。使用者可對其「區塊範圍」、「日期範圍查」、「發送方位址」、「接收方位址」以及「數位貨幣種類」等功能進行交易篩選，以下將針對「數位貨幣種類」功能進行說明：

數位貨幣種類篩選：可針對其特定數位貨幣種類篩選交易紀錄，如圖 5.17 所示。

Txn Hash	Block	Date Time	From	To	Quantity	
0x51a239508745...	4638568	Tue Nov 28 2017 ...	0x369285008c1d...	0xC6CDE7C39eB...	10	
0x2c22ad3e367e...	4638577	Tue Nov 28 2017 ...	0xC6CDE7C39eB...	0x2b2c9153Acc4...	1	
0x4afc4e7f282e...	4638586	Tue Nov 28 2017 ...	0x369285008c1d...	0xC6CDE7C39eB...	995	
0x35348a160fd8...	4638663	Wed Nov 29 201...	0xC6CDE7C39eB...	0x2b2c9153Acc4...	1	
0xe544a6ac188e...	4638700	Wed Nov 29 201...	0x2b2c9153Acc4...	0x9faf5515f177F...	20	USDT
0xb5c5f66220f56f...	4638705	Wed Nov 29 201...	0xC6CDE7C39eB...	0x2b2c9153Acc4...	10	USDT
0xbf3037e88363...	4638713	Wed Nov 29 201...	0x2b2c9153Acc4...	0x9faf5515f177F...	10	USDT
0xf68518104b27...	4638894	Wed Nov 29 201...	0xC6CDE7C39eB...	0x2b2c9153Acc4...	100	USDT
0x6c809cb988fb...	4638924	Wed Nov 29 201...	0x2b2c9153Acc4...	0x9faf5515f177F...	100	USDT
0x76ea50785162...	4773369	Fri Dec 22 2017 0...	0xC6CDE7C39eB...	0x5A9cf70048dB...	20	USDT
0xb7f200ca2bd1...	4805948	Wed Dec 27 2017...	0x5A9cf70048dB...	0xEF79e049D27F...	0.1	USDT

圖 5.17：數位貨幣篩選結果之畫面

第六章 討論與結論

本章節統整本研究之研究成果，共分為三小節，先是針對實作中遭遇的限制提出說明；接著提出本研究整體的結論，以及本研究的研究貢獻；最後針對本研究之議題提出未來研究方向之建議。

6.1 討論

本研究目標在於實作以太坊區塊鏈之數位貨幣溯源服務。透過現有技術方案與相關文獻，根據溯源服務在實務上的可能作法進行研究與探討，再藉由實作與實驗，針對實務應用層面進行探討與評估，甚至找出問題並建議解決方案。下面我們探討針對區塊鏈數位貨幣交易記錄與查詢服務，其影響該服務之效率的可能因素、實作中遭遇的限制，以及實務應用中需要考量的難題：

- Etherscan API 之使用次數限制影響溯源效率：我們能透過 Etherscan API 取得以太坊公鏈上之數位貨幣歷史交易，但對於一般開發者而言，使用 API 有對於每個 IP 位址每秒只能使用五次 API 的限制。倘若該服務中已新增超過五種數位貨幣，則在同步交易的過程中，會發生無法同時取得所有已新增之數位貨幣之交易紀錄。
- Ethereum Bridge 處理 Oraclize (Provable) 服務之 Query 速度影響溯源效率：由於本實驗將數位貨幣交易紀錄與查詢服務建立於私有鏈之上，當使用 Oraclize (Provable) 服務之 Query 時，需搭配 Ethereum-Bridge 工具監聽其智能合約所發出之特定事件 Event，並作為中介者負責傳遞區塊鏈外的訊息。Ethereum-Bridge 並無多執行緒的多工處理能力，因此當它接收到事件時，只能將其放入佇列中，採先進先出的方式逐一處理。

6.2 結論

本研究之「去中心化數位貨幣交易記錄與查詢服務」是一個提供使用者查詢數位貨幣歷史交易紀錄的服務平台，可以保證其服務平台上所儲存之數據的可信度與正確性，並透過私有鏈上之所有節點共同維護其鏈上數據，以保證其公正性和安全性。

在實務應用上，隨著區塊鏈的演進與數位貨幣的發展趨勢，可能須面臨更透明化、更嚴謹之溯源服務，甚至是政府機關之區塊鏈監管機制。因此，未來在區塊鏈數位貨幣之溯源議題時，希望能持續改進，例如：（1）數位貨幣流向追蹤：透過資料分析與追查，評估各個錢包地址是否存有洗錢之動機等。（2）納入監理機制（採用中心化管理方式）：於每一種數位貨幣之智能合約中加入一監理者角色，並監聽該數位貨幣之所有交易內容；或是所有交易皆需經由監理者認可才能發送，藉此達到監管與治理的效果。

參考文獻

- [1] "Introduction To Smart Contracts," [Online]. Available:
<https://ethereum.org/en/developers/docs/smart-contracts/>.
- [2] "EIP-20: ERC-20 Token Standard," [Online]. Available:
<https://eips.ethereum.org/EIPS/eip-20>.
- [3] "What Is DeFi?," [Online]. Available: <https://www.coindesk.com/what-is-defi>.
- [4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system. White paper," 31 October 2008. [Online]. Available: <http://bitcoin.org/bitcoin.pdf>.
- [5] V. Buterin, "A next-generation smart contract and decentralized application platform. White Paper," 2014. [Online]. Available:
<https://github.com/ethereum/wiki/wiki/White-Paper>.
- [6] "Facebook Libra," [Online]. Available: <https://libra.org/en-US/>.
- [7] "R3 Corda," [Online]. Available: <https://www.corda.net/>.
- [8] "BigchainDB," [Online]. Available: <https://www.bigchaindb.com/>.
- [9] S. Bragagnolo, H. Rocha, M. Denker and S. Ducasse, "Ethereum Query Language," in *2018 IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, Gothenburg, Sweden, 2018.
- [10] "Etherscan," [Online]. Available: <https://etherscan.io/>.
- [11] "Introduction to SQL," [Online]. Available:
https://www.w3schools.com/sql/sql_intro.asp.
- [12] "Oraclize(Provable)," [Online]. Available: <https://provable.xyz/>.

- [13] "Quorum," [Online]. Available: <https://consensys.net/quorum/>.
- [14] "Hyperledger Fabric," [Online]. Available: <https://www.hyperledger.org/>.
- [15] "EOSIO," [Online]. Available: <https://eos.io/>.
- [16] "RSK," [Online]. Available: <https://www.rsk.co/>.
- [17] "Solidity," [Online]. Available: <https://solidity.readthedocs.io/en/latest/>.
- [18] "Go Ethereum," [Online]. Available: <https://geth.ethereum.org/>.
- [19] "Node.js," [Online]. Available: <https://nodejs.org/en/>.
- [20] "Etherscan - Logs APIs," [Online]. Available: <https://etherscan.io/apis#logs>.
- [21] "Ethereum-Bridge," [Online]. Available: <https://github.com/provable-things/ethereum-bridge>.
- [22] "jsmnSol," [Online]. Available: <https://github.com/chrisdotn/jsmnSol>.
- [23] E. Nyalety, R. M. Parizi, Q. Zhang and K.-K. R. Choo, "BlockIPFS - Blockchain-Enabled Interplanetary File System for Forensic and Trusted Data Traceability," in *2019 IEEE International Conference on Blockchain (Blockchain)*, Atlanta, GA, USA, 2019.