НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
КАФЕДРА АВТОМАТИКИ ТА УПРАВЛІННЯ В ТЕХНІЧНИХ СИСТЕМАХ

«На правах рукопису»
УДК _____

«До захисту допущено»
Завідувач кафедри
_____ _____
(підпис) (ініціали, прізвище)
“___” _____20__ р.

Магістерська дисертація

зі спеціальності (спеціалізації)____126, Інформаційні системи та технолоігї, Інтегровані інформаційні системи

на тему:___Система віддаленого управління процесами та ресурсами майнінгових ферм

Виконав : студент __2__ курсу, групи ____ІА-72мп_____
(шифр групи)

Ващілін Сергій Миколайович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник__доцент кафедри АУТС, к.т.н, доцент Дорогий Я.Ю. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

Зміст

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

R&D (Research and Development) — сукупність етапів та робіт, що направлені на дослідження певної області, отримання знань, їх систематизація для подальшого практичного використання в рамках розробки технології, продукту тощо.

AMQP (Advanced Message Oriented Protocol) — відкритий протокол для передачі повідомлення між компонентами певної системи.

REST (Representational State Transfer) — архітектурний стиль, який поширений при створенні програмного інтерфейсу WEB-застосунків. Такий підхід дає можливість отримати доступ до даних, а саме — читати, змінювати, створювати — стандартні CRUD-операції.

GPU (Graphical Processing Unit) — апаратний пристрій для обробки графічної інформації.

CPU (Central Processing Unit) — центральний процес для обробки будь-якої інформації. Основа усього комп’ютерного обладнання.

ASIC (Application-specific integrated circuit) — інтегральна схема, яка спеціалізується на вирішення конкрентної задачі і не можу буде використана за межами своїх задач.

VNC (Virtual Network Computing) — система віддаленого доступу до робочого столу комп’ютера.

Криптовалюта — цифрова валюта, створення та контроль за якою відбувається з допомогою криптографічних методів.

ВСТУП

Актуальність даної магістерської дисертації полягає в тому, що з розвитком ринку криптовалюти росте потреба у автоматичній підтримці апаратного забезпечення для майнінгу. Це необхідно для того, щоб можна було з легкістю запускати програмне забезпечення для майнінгу на десятках, сотнях і тисячах ферм, що в свою чергу скорочує час для налаштування інфраструктури та пришвидшення отримання прибутку від криптовалют.

Об’єктом дослідження є система віддаленого управління процесами та ресурсами майнінгових ферм, автоматизація процесів налаштування необхідного програмного забезпечення.

Предметом дослідження є створення системи для автоматизації майнінгових ферми для компанії “АКРОНІМ СІСТЕМС”.

Метою даної магістерської дисертації є створення програмного комплексу для управління процесами майнінгу. Зараз практично не існує універсального програмного забезпечення, яке б могло задовольнити повністю вимоги майнінг процесів, тому розробка подібної системи це фактично унікальна можливість в рамках цієї предметної області. Така системи повинна вирішити ряд проблем, а саме:

– віддалений доступ до ферм;

– кросплатформеність та доступ з WEB-мережі;

– можливість групування ферм та налаштування доступу для користувачів;

– можливість віддаленого запуску програмного забезпечення для майнінгу без фізичного доступу;

– збір та класифікація статистики роботи кожної з ферм;

– запуск віддалено різноманітних команд, які безпосередньо керують віддаленою фермою;

Для досягнення мети магістерської дисертації було поставлено наступні задачі:

– розглянути та дослідити інструменти для розробки системи;

– провести R&D (Research and Development) в області автоматизації процесів криптомайнінгу;

– розробити гнучку архітектуру системи та реалізувати “шаблон” застосунку для перевірки концепції;

– провести перегляд архітектури, внести корективи та розробити систему відносно проведених досліджень;

– впровадити систему в існуючу інфраструктуру для майнінгу та провести закрите тестування ранньої версії.

В рамках магістерської дисертації дослідження предметної області, інструментів, проблем проводилось з використанням загальнонаукових та спеціальних методів, а саме:

– метод абстрагування — розробка високоабстрактної архітектури на рівні логічних та функціональних компонентів системи;

– метод індукції та дедукції — виявлення мережевих проблем під час розробки конпепції та врахування їх в архітектурі системи;

– метод порівняння — огляд аналогів дозволив покращити системи з урахування недоліків та переваг;

– метод моделювання — розробка конпеції на ранніх стадіях створення системи;

– аналіз та синтез — проведення дослідження мережевих протоколів, можливих рішень та розробка високотехнічних рішень цих проблем;

Розробка система породила ряд важливих проблем, які можуть вплинути на працездатність застосунку в майбутньому. Перша проблем –

це вибір стеку технологій для розробки, це найважливій крок, оскільки необхідно обрати кросплатформений інструмент, друга проблема – яким чином проектувати першу в своєму роді систему, третя проблема – розробка зручного та кросплатформеного графічного інтерфейсу.

Для найбільш ефективного процесу розробки було використано наступний підхід – перше проектування застосунку, швидка розробка концепції, друге проектування застосунку з урахуванням помилок та зауважень, безпосередньо розробка застосунку. Завдяки такому ітераційному підходу було прийнято рішення використовувати мікросервісну архітектуру, оскільки програмний комплекс є достатньо складним за структурою.

Найпроблемнішою частиною затосунку є вибір протоколів комунікації. Розробка конпепції базувалась на протоколі HTTP і виклику REST API застосунку, через що велика кількість запитів не оброблювалась через довге очікування в часі. Крім того, після проведення огляду концепції було виявлено проблему NAT'ів, яку також необхідно було долати в рамках P2P взаємодії.

Третя ітерація розробки дала продуктивний результат – внутрішні компоненти системи спілкуються за протоколом AMQP через брокер повідомлень RabbitMQ, що в свою чергу зменшує затримки та робить систему більш стабільною. Мережі проблеми було вирішено використанням протоколів STUN/TURN, які описані в наступних розділах.

Практична цінність системи віддаленого керування процесами та ресурсами майнінгових ферм полягає у впровадженні високого рівня автоматизації налаштування майнінгових ферми. Це дозволило зменшити час для встановлення необхідного програмного забезпечення та пришвидшити запуск процесів майнінгу.

Наукова цінність системи полягає у вирішенні проблеми з обходом NAT'ів, які в свою чергу не дозволяли встановити пряме P2P підключення. Оскільки на даний момент не існує рішень, які дозволяють це вирішити, то рішення представлене в рамках магістерської дисертації має наукову цінність в області дослідження мережі Інтернет.

1 ОГЛЯД РІШЕНЬ ТА ТЕХНОЛОГІЙ

1.1 Технологія контейнеризації та Docker

Оригінальная технологія контейнеризації Linux називається Linux Containers або LXC. LXC — це специфічний метод віртуалізації на рівні операційної системи і призначений для того, щоб запускати безліч ізольованих застосунків на одному хості.

Для розуміння терміну "контейнер", слід розібратись в тому, що таке контрольні групи та простір імен Linux — функції ядра ОС UNIX, які створюють бар'єри між контейнерами та іншими процесами, які запущені на хост-машині. Простір імен дозволяє створювати оболочку для набору системних ресурсів та надають ці ресурси процесу. В свою чергу, контрольні групи займаються ізоляцією та використанням ресурсів, а саме — центральним процесором, оперативною та постійною пам'яттю. Наприклад, якщо певний застосунок використовує занадто багато ресурсів, достатньо розмістити його в контрольній групі, щоб обмежити використання ресурсів.

Якщо узагальнити, то простір імен займається ізоляцією ресурсів для одного процесу, а контрольні групи — ресурсами для набору процесів.

Контейнери відділяють застосунки від системи, що в свою чергу дозволяє запускати повноцінні програмні комплекси на легких дистрибутивах Linux без зайвого програмного забезпечення.

Оскільки операційна система відділена від контейнерів, то усі контейнери можна переміщати на будь-який інший Linux хост чи сервер, який відтримує технологію контейнеризації.

Docker — відкрита платформа для розробки, розгортання та використанням застосунків. Docker контейнери змінили повністю підходи до розгортання застосунків. Фактично це відкрило нову еру в сфері IT. Завдяки Docker контейнерам можна розгортати , копіювати, переносити та робити резервні копії інформації швидше та легше у порівняння з віртуальними машинами.

1.2 Підпрограмне забезпечення (Middleware)

Підпрограмне забезпечення, проміжне програмне забезпечення, middleware — термін, який визначає шар чи комплекс технологічного програмного забезпечення для встановлення взаємодії між різними застосунками, системами чи компонентами. Даний термін є достатньо абстрактним, тому може визначити різні рівні програмних комплексів, наприклад: CMS (Content Management System), СУБД, веб-сервіси тощо. Крім того, до підпрограмного забезпечення також включають:

– моніторинг транзакцій;
– виклик віддалених процедур;
– орієнтоване на обробку повідомлень;

1.3 Брокер повідомлень

Вид підпрограмного забезпечення, який орієнтується на забезпечення обміну повідомленнями [1]. Перш за все, такі програмні інструменти призначені для реалізації відкладеного обміну повідомленнями. Брокери повідомлень реалізують парадигму черг повідомлень за щаблоном видавець-підписник. Брокери повідомлень надають асинхронний протокол передачі даних, тому відправник та отримувач повідомлення не повинні взаємодіяти з чергою повідомлень

одночано. Розміщені в черзі повідомлення зберігаються до тих пір, поки отримувач не отримає ці повідомлення. Реалізацією даного middleware є протокол AMQP та сервіси, побудовані на його основі — RabbitMQ, ActiveMQ, ZeroMQ та інші.

1.3.1 Передача повідомлень

Передача повідомлень — це концепція взаємодії в рамках програмних комплексів, яка була сформована з початком розвитку WEB-технологій. В даному випадку повідомлення це елементарна одиниця для передачі даних — агрегує інформацію в будь-якому форматі, дає можливість вказувати специфічні заголовки [2].

Концепція обміну повідомленнями постійно розвивається і на фоні розвитку створюється поняття віддаленого виклику процедур або RPC (remote procedure call). Ідея RPC полягає в розширенні функціоналу передачі керування та даних всередині програми, яка виконується на одній хост-машині, на передачу керування та даних через мережу. Засоби RPC призначені для полегшення організації розподілених обчислень і створення розподілених клієнт-серверних системи. Віддалений виклик процедур найбільш ефективний в застосунках, в яких існує інтерактивний зв’язок між віддаленими компоненатми з невеликим часом відповіді та відносно малою кількістю передачі даних.

Характерними особливостями виклику віддалених процедур є:

– асиметричність — завжди повинен бути ініціатор;

– синхронність — виконання процедури зупиняються у момент видачі запиту і оновлюється тільки після поверння з процедури, яка викликається.

На рисунку 1.1 зображено візуалізовано модель віддаленого виклику процедур. На рисунку Client stub і Server stub це компонети, які

використовуються для перетворення параметрів, які будуть передані з допомогою RPC.

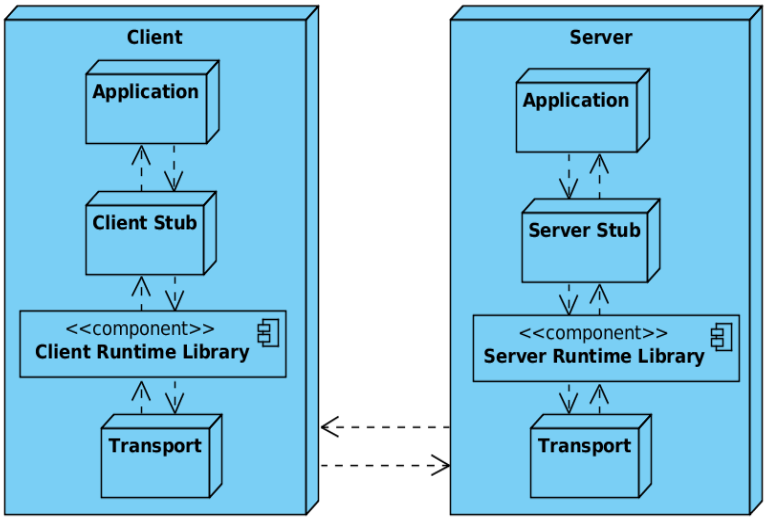


Рисунок 1.1 – Модель віддаленого виклику процедур (RPC)

1.3.2 Особливості роботи протоколу AMQP

Протокол AMQP (Advanced Message Queueing Protocol) забезпечує взаємодію між клієнтами та брокерами. Цей протокол створено для того, щоб через стандартизацію нада можливість різним застосункам та системам працювати незалежно від їхньої внутршіньої структури. AMQP — це достатньо широке поняття, яке визначає як організацію мереж, так і роботу брокерів повідомлень. Крім того, даний протокол відноситься до:

– маршрутизації і збереженню повідомлень брокерами і набору правил для визначення поведінки компонент;

– протоколу для реалізації зв’язків між клієнтами та брокерами, які виконують вище вказані дії.

Резюмуючи, протокол AMQP це набір чітко сформульованих правил та інструкцій, які дають можливість сформувати загальну основу для обробки всіх черг повідомлень.

AMQP протокол краще використовувати у випадкаха, коли необхідно забезпечити потребу у високоякісній та надійній відправці повідомлень між застосункам.

Таким чином, в контексті реальних задач AMQP підходить для:

– моніторингу;

– налаштування взаємодії між різними системами;

– забезпечення швидкої реакції сервера на повідомлення;

– передача ресурсоємких задач для подальшої обробки;

– забезпечення асинхронної взаємодії в рамках системи;

– підвищення надійності та безперебійності роботи застосунків;

Протокол AMQP включає в себе наступні поняття:

– брокер — це програмний інструмент, який реалізовує концепцію AMQP, приймає повдіомлення для їх подальшої маршрутизації, створює черги тощо;

– повідомлення — одиниця переданих даних (включаючи payload, заголовки тощо);

– consumer (споживач) — застосунок, який отримує повідомлення із черг;

– producer (виробник) — застосунок, який відправляє повідомлення в черги через exchange.

Протокол AMQP складається з наступних компонент:

– точка обміну (exchange) — частина брокеар, яка отримує повідомлення та направляє їх в чергу. В свою чергу точки обміну бувають різних видів, а саме:

a) direct exchange — доставляє повідомлення з черги використовуючи ключі маршрутизації. Ключ маршрутизації — це додаткові дані, які визначають, в яку чергу треба відправити конкретне повідомлення. Дуже часто такий тип

використовується для балансування навантаження алгоритмом Round robin;

b) fanout exchange — ігнорує ключі маршрутизації та відправляє повідомлення до всіх черг, що прив’язані до цього обмінника. Такі точки обміну використовуються для групової відправки типу broadcast;

c) topic exchange — використовується в шаблонах типу publisher-subscriber. В цьому випадку ключ маршрутизації використовується з прив’язкою черг до точки обміну;

– черга повідомлень — структура, з якою зв’язані повідомлення і звідки їх отримують споживачі;

– прив’язки — правила розповсюдження повідомлень із точок обміну в черзі.

1.4 Бази данних

1.4.1 Реляційні бази даних (RDMS)

Вид баз данних, в яких дані зберігається за принципом реляційної моделі даних. Дані таких баз пов’язані між собою відношеннями різнього ступеня і зберігаються у вигляді таблиць як на рисунку 1.2.

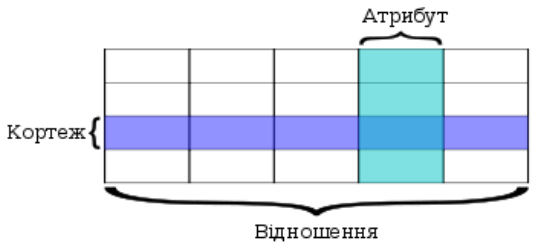


Рисунок 1.2 – Представлення таблиць в реляційних БД

В таблицях зберігається інформація про об’єкти, представлені в базі даних. В кожному стовпці таблиці зберігається певний тип даних, в кожній комірці – значення атрибуту. Кожен кортеж таблиці може бути

позначений унікальним ідентифікатором – первинний ключ, а кортежі з різних таблиць можуть бути пов’язані з допомогою зовнішніх ключів.

1.4.2 Time series бази даних (TSDB)

Це бази даних, які оптимізовані під роботу з так званими time series даними (часові ряди), масивами чисел, які згруповані по часу (даті чи проміжку дат).

TSDB дозволяють створювати, рахувати, оновлювати або видаляти часові ряди і організовувати їх, визначеним користувачем, способом. Сервери таких баз даних підтримують операції складання, перемноження та інші, над часовими рядами, щоб формувати нові ряди.

На даний момент існує велика кількість готових рішень (окремих СУБД), які можуть зберігати та працювати з часовими рядами, але оскільки багато рішень вже використовують реляційні бази даних, то деякі SQL рішення можуть оперувати time series даними за умови, якщо підтримуються одночасно великі двійкові об’єкти (BLOB) та користувацькі функції. На жаль, такі рішення зазвичай неефективні, тому зазвичай краще використовувати спеціалізовані рішення.

1.4.3 Бази даних типу “ключ-значення” (Key-value storage)

Сховища типу “ключ-значення” є одним із видів NoSQL баз даних, які були створені для вирішення проблем реляційних БД, наприклад, проблема низької швидкості обробки великого об’єму даних. Також, даний тип сховищ може іменуватись по-різному – документо-орієнтовані, атрибутно-орієнтовані, розподілені БД, розподілені хеш-таблиці і сховища типу “ключ-значення”.

1.4.4 Порівняльний аналіз баз даних

В таблиці 1.1 наведено порівняльну характеристики використаних баз даних.

Таблиця 1.1 – Порівняльна характеристики використаних баз даних

Реляційні БД	Time series БД	БД типу “ключ-значення”
Дані зберігаються в строго визначених таблицях з визначеними наборами атрибутів кожного кортежу та типами даних для кожного атрибуту	Дані зберігаються у вигляді часових рядів; часовий ряд складається з точок, по одній на кожне значення виміру певної метрики – кожна точка складається з часу (timestamp), назви метрики (measurement), не менше одного у форматі ключ-значення (де беспосередньо зберігається значення метрики).	Дані зберігаються у вигляді асоціативного масива, де під масивом слід розуміти звичайни масив, як структуру даних, бінарні дерева пошуку або хеш-таблиці.

Продовження таблиці 1.1

Реляційні БД	Time series БД	БД типу “ключ-значення”
Модель даних базується на основі	Достатньо проста модель даних.	Оскільки формат даних, які зберігаються,

Продовження таблиці 1.1

предметної області, а не на функціональних можлиовстях.	Фактично в кожному часовому ряді зберігаються значення одного типу і однієї семантики.	представляє собою, зазвичай, хеш-таблицю, то можна зберігати різні дані, різних типів без чітко визначених структур.
Можливість проведення нормалізації для уникнення дублікатів і створення відношень між таблицями.	Відсутні відношення між часовими рядами, що і не потрібно внаслідок специфіки таких баз даних.	Відсутні явно визначені відношення між значенням і ключами.

1.5 Огляд існуючих рішень

1.5.1 GUIMiner

Стартовий екран при запуску програми одразу дає можливість запустити CGMiner для пулу slush’s pool. Є можливість обрати будь який інший пул з випадаючого списку поля “Server”. Достатньо ввести користувача та пароль від акаунту пула, встановити OpenCL, обрати необхідні пристрої для майнінгу криптовалюти, а також (опціонально), додати опції для запуску майнера та натиснути кнопку “Start mining!”. На рисунку 1.3 зображено стартову сторінку запуску.

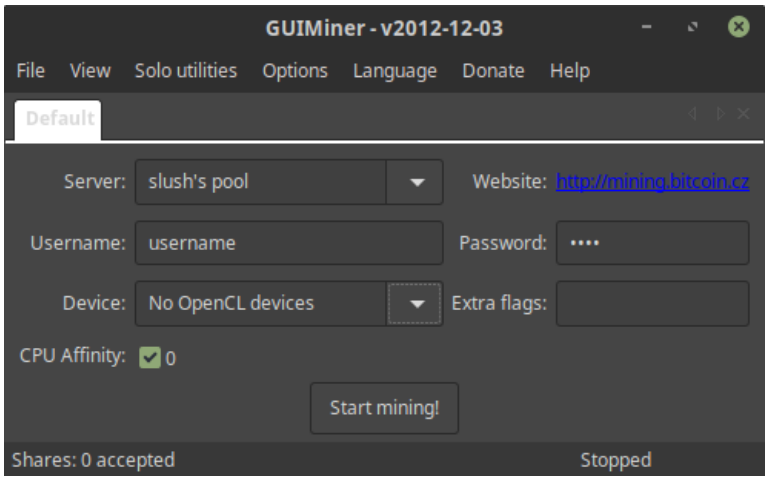


Рисунок 1.3 — Стартова сторінка для запуску майнерів

Такий самий порядок справедливий і для решти майнерів програми GUIMiner. Єдина відмінність полягає в тому, що для майнерів Cuda та Ufasoft не потрібно обирати пристрої для майнінгу. Також, слід зазначити, що CGMiner краще використовувати ASIC пристроїв, оскільки його реалізація оптимізована саме під такі комплектуючі.

На рисунку 1.4 зображено поля для майнінгу через Cuda. Такий майнер буде працювати тільки у випадку, якщо на фермі встановлено тільки відеокарти Nvidia, оскілька технологія розроблена саме в компанії Nvidia.

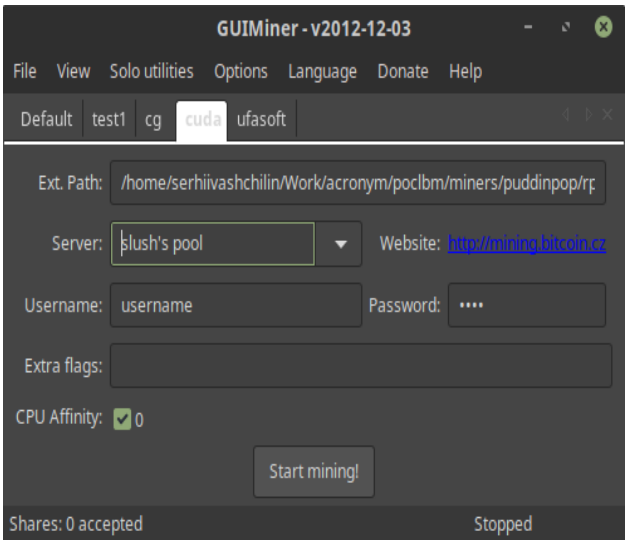


Рисунок 1.4 — Сторінка майнера на основі Cuda

На рисунку 1.5 зображено екран для майнінгу через Ufasoft — додаток спочатку спеціалізувався на майнінгу CPU, потім і для GPU.

Ufasoft майнер включає дві реалізації майнінгу для GPU — OpenCL та AMD CAL API. Рекомендується використовувати цей майнер виключно для CPU пристроїв, оскільки він також краще оптимізований під роботу з центральним обчислювальним пристроєм.

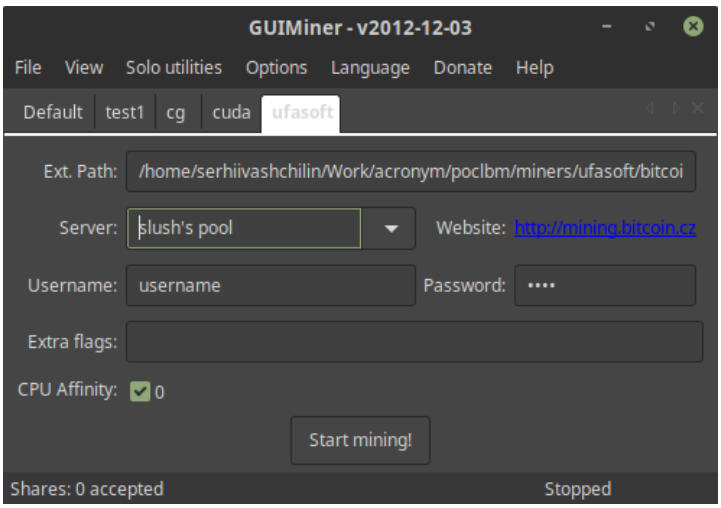


Рисунок 1.5 — Сторінка майнера на основі Ufasoft

Ця програма є достатньо простою, підтримує невелику кількість майнерів, проте більше орієнтована на роботу з Mining pool'ами. Ця програма пристосована для роботи лише з однією фермою, тому для великої кількості ферм ця програма буде доволі незручною

1.5.2 Awesome Miner

На даний момент цей застосунок для майнінгу криптовалюти є найпоширенішим для майнінгу у великих масштабах. Цей застосунок працює тільки для ОС Windows та має Beta-версію для UNIX подібних операційних систем. Також, має мобільний застосунок, що дозволяє завжди мати актуально інформацію про стан ферм, дохідність тощо, керувати фермами, що є достатньо зручним в епоху мобільних пристроїв. Відсутня повноцінна WEB-версія застосунку, для керування можна користуватись тільки десктоп версією.

Awesome Miner працює за принципом встановлення агентів на фермах та комунікації з клієнтом. Наявна також можливість групування, перегляд різноманітної статистики тощо.

Застосунок, розроблений в контексті магістерської дисертації, частково спирається на функціонал даного застосунку як прямого конкурента.

На рисунку 1.6 та 1.7 зображено головну сторінку застосунку де зображено список список ферм та інформація — температура, IP-адреса, хешрейт, криптовалюта та дохідність. Також, в нижній частині помжна побачити таби, в яких можна побачити інформацію про Pool'и, в яких відбувається майнінг на фермах.

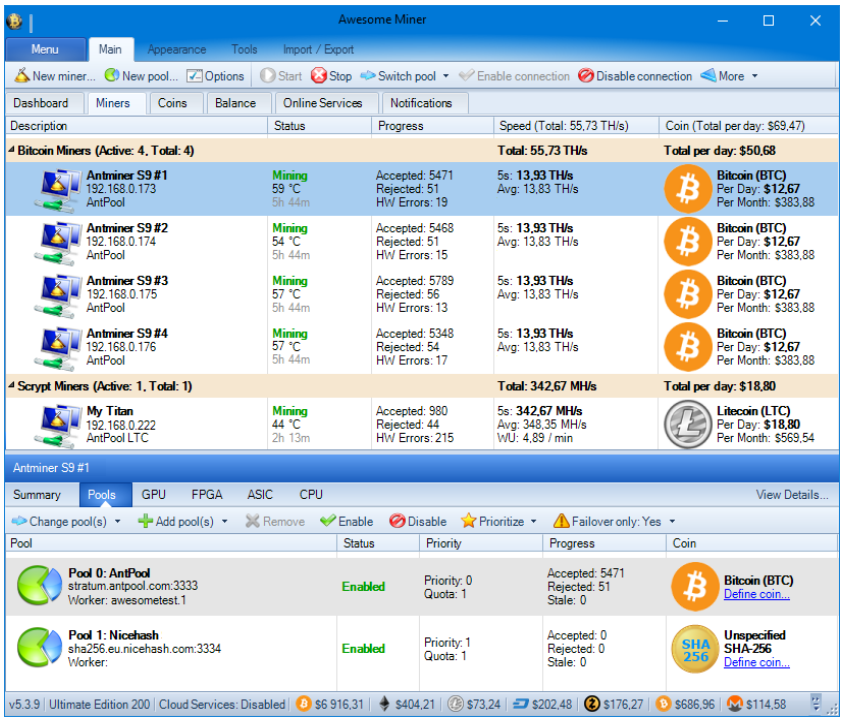


Рисунок 1.6 — Список ферм та інформація про них [3]

Крім достатньо зручного інтерфейсу управління фермами в застосунку Awesome Miner представлена окрема сторінка для відображення статистики — відношення хешрейт до температури, кількість GPU/ASIC/CPU та загальну кількість майнерів запущених на

фермах, дохідність на різні проміжки часу та відповідно графіки різноманітних залежностей.

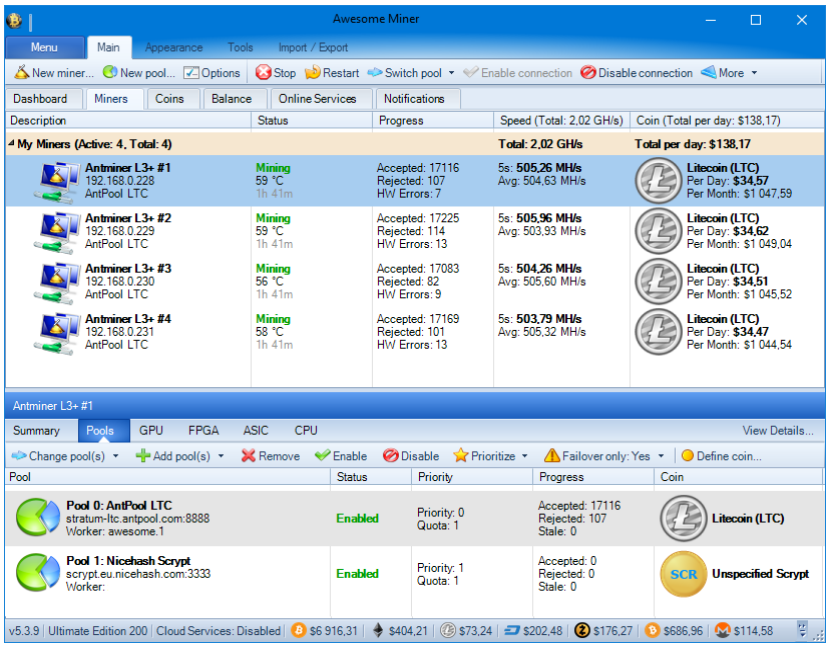


Рисунок 1.7 — Список ферм, які майнять криптовалюту Litecoin (LTC) [3]

На рисунку 1.8 зображено у вигляді графіка залежності температури від хешрейту ферм, дохідність, криптовалюту, яку майнять дані ферми, загальний хешрейт ферм тощо.

Крім того, для зручності, статистику можна виводити у двох форматах — таблиці та графіки. Графіки більш наочно можуть показати, як змінювалась певна характеристика ферм з часом, натомість в табличному форматі зручніше спостерігати за поточними значеннями ферм. На рисунках 1.8 та 1.9 зображені відповідно графік та таблиця.

На рисунку 1.10 зображено статистика по mining'у криптовалюти. Зазвичай такі статистичні дані необхідні для того, щоб можна було з легкістю відслідковувати дохідність кожної криптовалюти і визначати найдохідніші, щоб запускати ферми на видобуток саме цих.

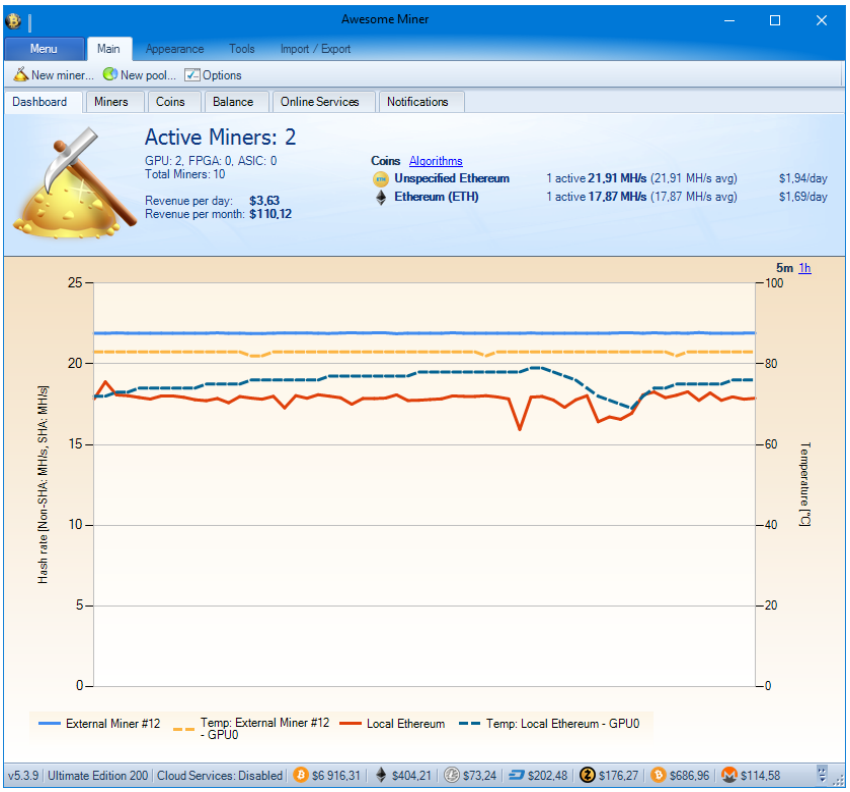


Рисунок 1.8 — Статистика для ферм (Graph mode) [3]

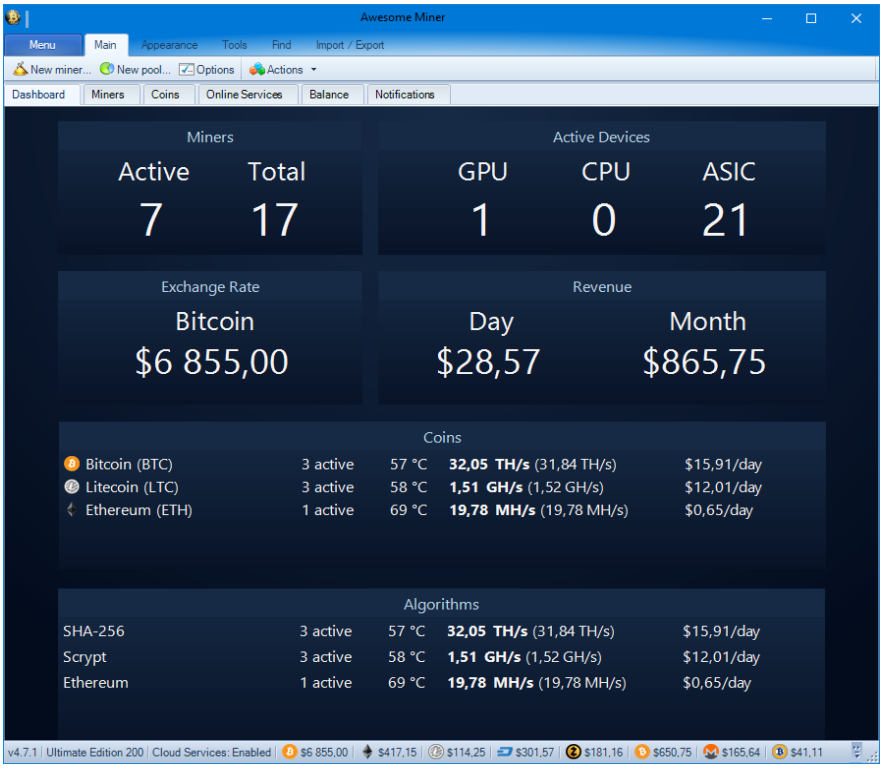


Рисунок 1.9 — Статистика для ферм (Table mode) [3]

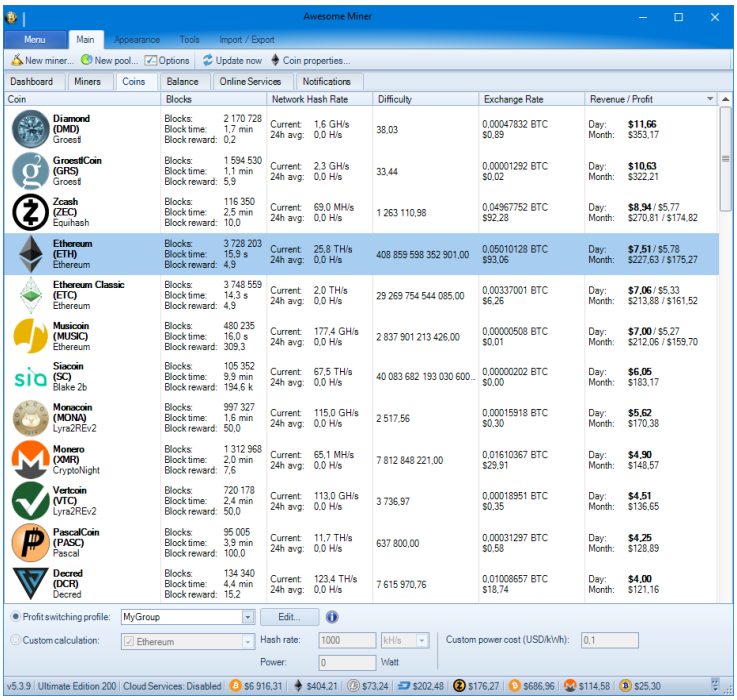


Рисунок 1.10 — Статистика по криптовалюті, яка добувалась фермами [3]

Як можна побачити, даний застосунок має доволі багатий функціонал, велика кількість статистичних даних та можливість часткового контролю ферм — запуск майнерів, збір статистика. Цього достатньо для просто процесу майнінгу на великій кількості ферм. Відсутній віддалений доступ до ферм та більш детальній контроль обладнання ферм.

Відсутність повноцінного багатоплатформного клієнту лімітує використання данного застосунку на ОС типу UNIX. На даний момент наявний UNIX агент, який дозволяє запускати майнери, але все ще не має можливості запускати клієнт.

1.5.3 MinerGate

Ще один застосунок для майнінгу MinerGate працює за принципом застосунку GUIMiner. Його необхідно встановити на фермі і запустити

відповідно цю програму, попередньо провівши авторизацію на сайті minergate.

На рисунку 1.11 зображено екран для виводу криптовалюти. Для того, щоб здійснити цю операцію, необхідно зареєструвати акаунт в гаманці під назвою Multi Wallet, а для криптовалюти ETH (Ethereum) достатньо ввести публічну адресу гаманця в мережі Ethereum (зазвичай це Parity). В такому підході є великий плюс — користувачу не потрібно реєструвати купу гаманців для кожної криптовалюти окремо, достатньо мати один акаунт на мульигаманці Freewallet і вводити лише нікнейм та пароль для виводу. Серед усієї різноманітності виводів криптовалюти це найпростіший варіант.

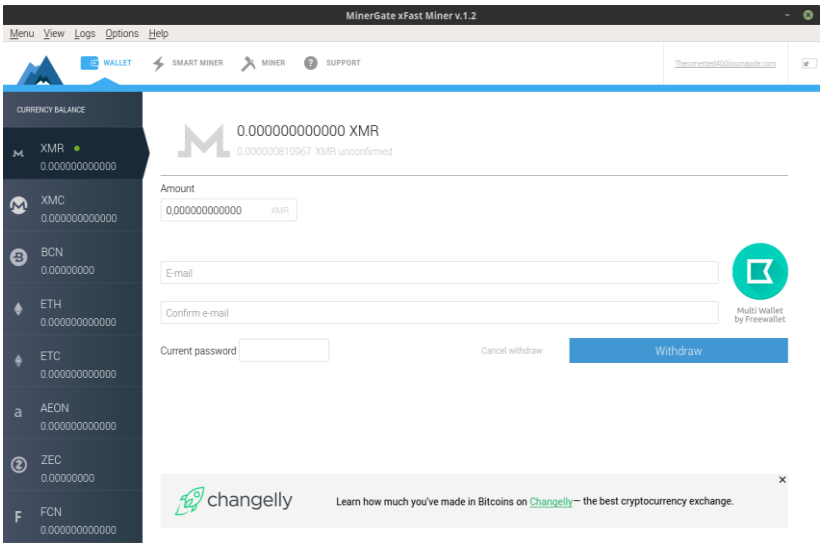


Рисунок 1.11 — Початковий екран при запуку програми

На рисунку 1.12 зображено вікно для запуску майнерів. Цей застосунок сам визначає, які є пристрої для майнінгу і вирішує, який найкраще використати на даний момент по хешрейту. При цьому, у випадку, якщо здійснюється майнінг на CPU, то майнер займає N-1 ядер процесора, що однозначно гарантує працездатність ферми без зависань.

На рисунку 1.13 зображено таблицю зі статистикою по кожній криптовалюті та статусу майнера на кожну з них.

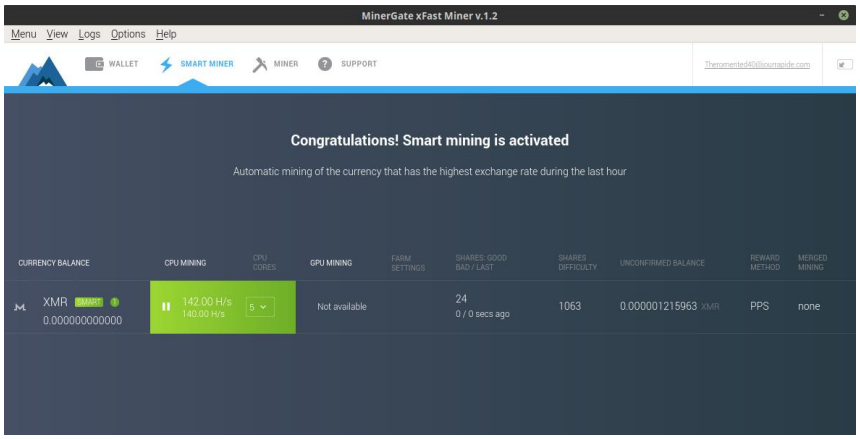


Рисунок 1.12 — Вікно для запуску майнерів

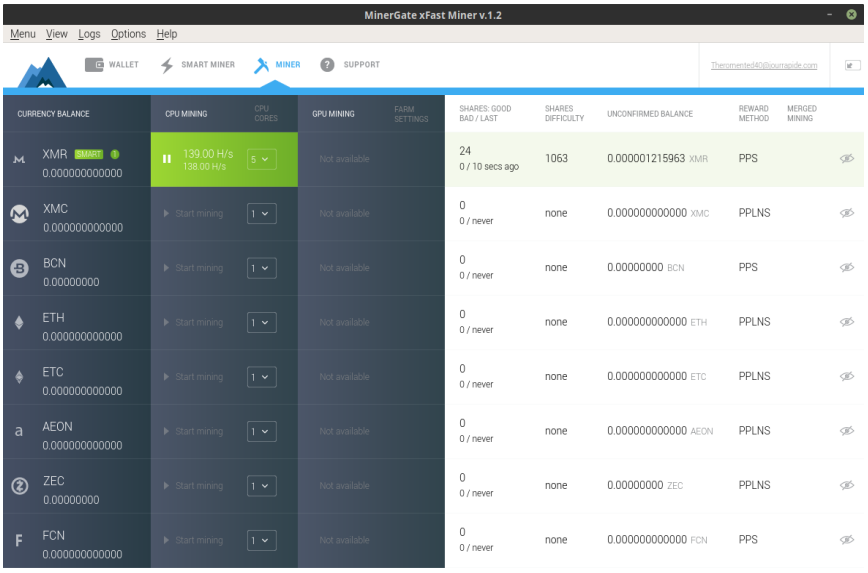


Рисунок 1.13 — Вікно з таблицею криптовалют і майнерів

Як можна побачити, застосунок MinerGate має приємний інтерфейс та виконує ті ж сам функції, що і програма GUIMiner, але при цьому

інтерфейс користувача кращий та зрозуміліший, користувачу не потрібно виконувати багато дій - достатньо зареєструватись і натиснути кнопку для майнінгу, а решту вже підбере програма на вигідних для користувачах умовах. Мається на увазі, що в залежності від оптимальної дохідності майнінг буде перемикатись автоматично на ту криптовалюту, дохідність від якої буде перемикатись.

2 ТЕОРЕТИЧНИЙ ОПИС РІШЕННЯ

2.1 Проблема віддаленого доступу

2.1.1 Система VNC (Virtual Network Computing)

Дана система використовує протокол RFB (Remote Framebuffer) [4] – протокол прикладного рівня. Призначений для віддаленого доступу до графічного робочого місця комп’ютера. Цей протокол працює на рівні кадрового буфера, тому може використовуватись для різноманітних графічних систем (X11, Windows, Macintosh тощо).

Система VNC створена на основі клієнт-серверної взаємодії. В даному випадку, сервер надає доступ до екрану комп’ютера, на якому застосунок встановлено. Клієнт, в свою чергу, це програма, яка отримує зображення екрана із сервера.

RFB протокол працює наступним чином – сервер відправляє клієнту невеликі прямокутники, які закодовано певним способом. При цьому вони доповнюються інформацією про те, де клієнт повинен їх намалювати. Керування робочим місцем виконується шляхом передачі натиснень клавіш на клавіатурі або миші.

Взаємодія клієнта та сервера в системі VNC відбувається в три етапи (з допомогою протоколу RFB):

– етап “рукостискання”, під час якого відбувається узгодження однієї з версій протоколу (3.3, 3.7, 3.8), визначення типу авторизації (відсутня, VNC автентифікація, Transport layer security або TLS)

– етап ініціалізації, під час якого клієнт та сервер обмінюються повідомлення для ініціалізації клієнта та сервера;

– звичайна взаємодія за певними правилами, визначеними протоколом. Фактично відбувається обмін звичайними повідомленнями

VNC система працює за принципом передачі змін екрану, тобто, після передачі початкового стану екрану передаються тільки пікселі, які змінились. Такий підхід показує гарний результат при незначних змінах на екрані – наприклад, при зміні положення курсора або набору текста з допомогою клавіатури. Але у випадках частої зміни пікселів, наприклад, перегляд відео, канал передачі завантажується дуже сильно, що може негативно вплинути на пропускну здатність каналу в цілому. Для вирішення даної проблеми використовуються різні види кодувань даних, які упаковують, зменшують розмір наборів змін пікселів екрану.

Види кодувань, які використовуються системою VNC [5]:

– Raw encoding – пікселі передаються в порядку зліва-направо, зверху-вниз;

– CopyRect – використовується в тих випадках, коли необхідна область зображення уже присутня в копії кадрового буфера клієнта. В такому випадку достатньо вказати координати прямокутника, який необхідно скопіювати з копії кадрового буфера. Даний тип кодування найчастіше використовується у випадках, коли користувач переміщує вікно по екрану та коли вміст вікна прокручується;

– Hextile – вид кодування, при якому прямокутник розбивається на невеликі квадрати, кожен з яких може мати власне представлення, що дозволяє кодувати повторювані області (RLE) та використовувати палітри;

– TRLE (Tiled Run-Lenght Encoding) – це вид кодування довжин серій, який комбінує в собі плитки (tiles), палітри (palettization). Даний вид стискання працює наступним чином: певна прямокутна область (частина екрану або сам екран) ділиться на плитки розміром 16x16 пікселів зліва-направо, зверху-вних. Якщо ширина прямокутної області не ділиться на 16, то ширина останньої плитки менша, що справедливо і для висоти плитки.

– ZRLE – це вид кодування довжин серій, схожий на TRLE, але при цьому використовує Zlib для стискання даних. В даному випадку, всі плитки представляються в порядку зліва-направо, зверху-вниз, як в TRLE, але розміри плиток 64x64 пікселів. Якщо прямокутна область на ділиться націло на 64, то справедливе правило відносно останньої плитки як і для ZRLE.

Система VNC є open source системою, тому достатньо поширена для розробки різноманітних застосунків, оскільки існує велика кількість бібліотек для різним мов програмування.

2.1.2 Обгрунтування вибору системи VNC

Вибір на користь VNC було зроблено за рахунок того, що ця система є Open Source рішенням. Наявна велика кількість сторонніх бібліотек для розробки різноманітних застосунків. Крім того, є можливость зміни налаштувань серверу для з’єднань з віддаленим робочим місцем для підвищення стабільності і захищенності каналів передачі. Також, для системи VNC достатньо просто створювати програмні Middleware для підтримки P2P з’єднань у випадку використання IPv4.

VNC також є кросплатформнним рішенням, що в свою чергу дає можливість створювати програмне забезпечення з малою кількістю залежностей, що значно спрощує розробку та підтримку рішення.

Переваги системи VNC:

– робоче місце, на якому запущено VNC сервер не потребує фізичного монітору;

– використання протоколу TCP, що значно зменшує втрати пакетів в мережі;

– cross-platform система;

– VNC може використовувати SSH/VPN тунелі для покращення безпеки підключення та обходу NAT;

– паролі та секрети не передаються через мережу;

Недоліки:

– передаються цілі зображення через мережу, що в свою чергу значно збільшує витрати пропускної здатності мережі;

– через використання примітивних кодувань, які можна зустріти в половині VNC клієнтів передача великої кількості змін на екрані може зумовити великі затримки для промальовки картинок і передачі пакетів;

2.2 Мережеві проблеми

2.2.1 Механізм NAT

Network Address Translation (Перетворення мережевих адрес) [6] [7] – механізм, який працює в мережах TCP/IP, який дозволяє перетворювати IP адреси з одного адресного простору в інший. Цей механізм має наступні переваги:

– економія IP адрес – можливість трансляції декількох IP адрес з одного адресного простору (внутрішня мережа) в одну IP адресу публічну. Наприклад, таким чином працюють корпоративні мережі;

– обмеження доступу ззовні в середину мережі за NAT'ом – в цьому випадку хости мають доступ ззовні внутрішньої мережі, але

отримати доступ ззовні всередину неможливо без конфігурації правил брандмауера або правил трасляції;

– можливість маппінгу портів – використовується у ситуації, коли певний застосунок працює на порту 80 (HTTP-сервер) і необхідно підмінити зовнішній порт на будь-який інший.

На даний момент реалізовано декілька концепцій технології NAT, а саме:

– статичний NAT – однозначний мапінг внутрішньої IP-адреси до зовнішньої, це дозволяє отримати доступ до хоста у внутрішній мережі ззовні;

– динамічний NAT – мапінг внутрішньої IP-адреси хоста на випадкову IP-адресу з-поміж пулу адрес;

– перегружений NAT – мапінг внутрішніх ІЗ-адрес на одну єдину зовнішню IP-адресу, але з різними портами;

Існує декілька видів NAT'ів, оскільки для подолання їх необхідні різні механізми [4]:

– симетричний NAT (динамічний) - в даному випадку внутрішній IP адресі та порту у відповідність ставиться випадкова унікальна публічна пара IP адреси та порту. На рисунку 2.2 зображено тип;

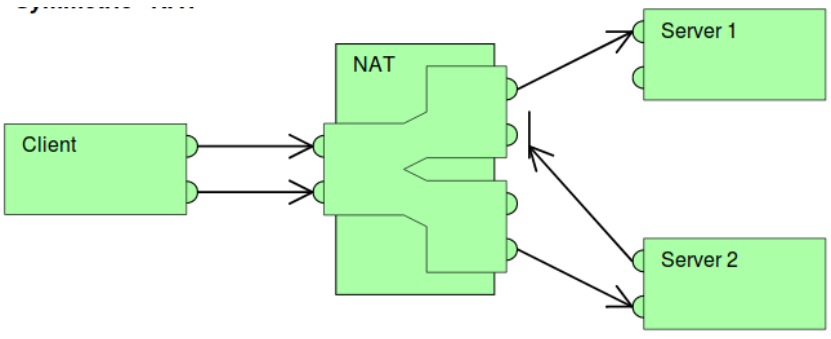
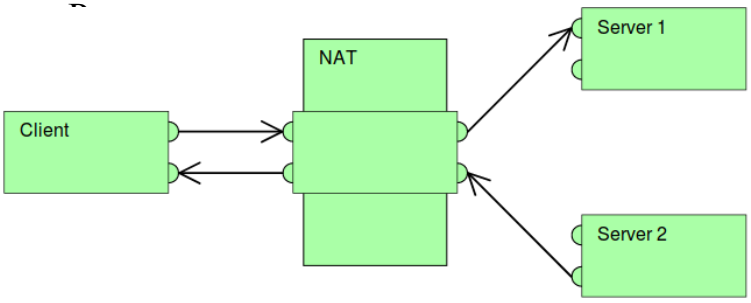


Рисунок 2.2 – Симетричний NAT [6]

– Cone NAT або NAT повного конуса (статичний) – якщо хтось із зовнішньою мережі намагається відправити пакет даних на хост, що

знаходиться у внутрішній мережі за NAT'ом і йому стоїть у відповідність IP адреса, то достатньо знати номер порта, на який потрібно відправити пакет. В цьому випадку порт зовнішній порт відкритий для доступу від будь-яких адрес. На рисунку 2.3 зображено схему відповідний тип NAT;



ок 2.3 – NAT повного конуса [6]

– Restricted cone NAT або NAT обмеженого конуса (динамічний) – даний тип блокує всі пакети відправлені із зовнішнього хоста внутрішньому до тих пір, поки внутрішній хост не відправить пакет на IP адресу зовнішнього хоста. На рисунку 2.4 зображено схематичне зображення роботи даного типу NAT'у;

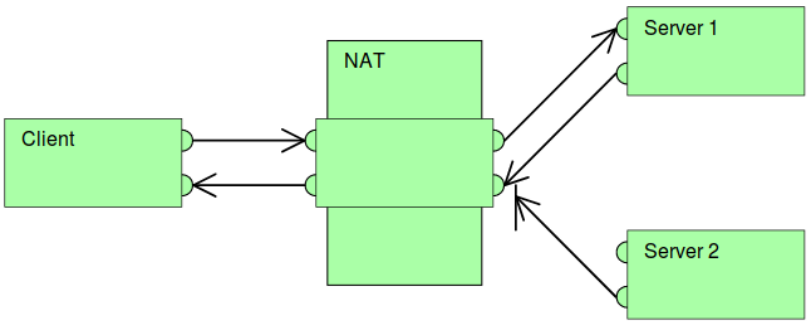


Рисунок 2.4 – NAT з портом обмеженого конуса [6]

– Port-Restricted cone NAT або NAT з портом обмеженого конуса (динамічний) – в цьому випадку блокуються всі пакети надіслані ззовні на хост за NAT'ом, якщо перед цим хост із внутрішньої мережі не надіслав назовні пакет на IP адресу зовнішнього хоста. На рисунку 2.5 зображено схематичне відображення роботи даного типу NAT'у;

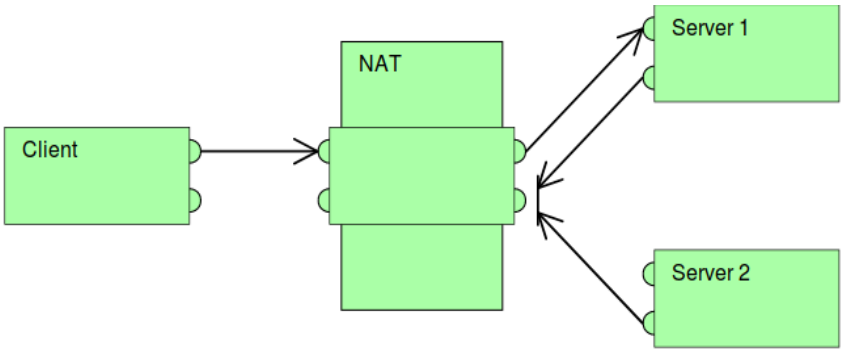


Рисунок 2.5 – NAT з портом обмеженого конусу [6]

Технологія NAT використовується практично усіма Інтернет провайдерами в основному для економії адресного простору IPv4. Також, при розробці певних застосунків виникають наступні проблеми:

– з однієї зовнішньої IP адреси може йти багато запитів з великої кількості хостів внутрішньої мережі;

– необхідно налаштувати специфічні правила для трансляції для доступу специфічних застосунків, що інколи буває достатньо важко або неможливо, особливо в корпоративних мережах;

– деякі протоколи дуже чутливі до наявності NAT в мережі, через що можуть працювати некоректно або не працювати взагалі (FTP тощо).

Для обходу NAT існує декілька підходів, а саме STUN (Session Traversal Utilities for NAT) та IPv6, що буде розглянуто в наступних розділах.

2.2.2 STUN (Session Traversal Utilities for NAT) протоколу

STUN [8] – протокол прикладного рівня моделі OSI, який дозволяє хосту, що знаходиться за NAT'ом, визначити свою зовнішню IP-адресу, спосіб трансляції адреси та порт ззовні. Цей протокол корисний при встановленні з'єднань на основі UDP протоколу (передача голосу, зображень).

Даний протокол працює тільки з трьома із чотирьох основних типів NAT'ів: NAT повного конуса, NAT з портом обмеженого конуса, NAT з портом обмеженого конуса.

Як тільки клієнт визначив зовнішню адресу, він може його передати хосту з яким потрібно ініціювати з'єднання. Якщо під час встановлення з'єднання трапляються Full Cone NAT, то будь-яка із сторін може почати спілкування. У випадку Restricted Cone NAT та Port Restricted Cone NAT обидва ініціатори повиннні почати передачу даних разом.

STUN дозволяє клієнту отримати транспортну адресу (IP та порт), яка може бути корисна для прийому пакетів від певних вузлів. Але не всі адреси, отримані через протокол STUN, можуть бути доступні усім вузлам в мережі. Отримані адреси працюють в залежності від топології мережі.

2.2.3 TURN (Traversal using relay NAT)

Протокол, який дозволяє хосту, що знаходиться за NAT'ом отримувати вхідні дані через TCP/UDP підключення. Така можливість дуже корисна для хостів, які є приймальною стороною, за симетричним NAT'ом. Фактично TURN це специфічна реалізація STUN протоколу для TCP підключень.

2.2.4 P2P (Peer-To-Peer) мережа

Однорангова мережа [9] [10] - тип логічної комп’ютерної мережі, в якій кожен учасник рівноправний. В такій мережі немає чітко виділеного сервера, тому в такій мережі кожний вузол відіграє одночасно як роль сервера, так і роль клієнта.

Така організація мережі дозволяє без затримок з боку сервера встановити підключення та обмінювати пакетами даних. В рамках даної магістерської дисератації це дозволить зробити систему більш стабільною, оскільки одночасна підтримка великої кількості підключень вимагає потужних серверів, а це в свою чергу великих витрат.

2.3 Mining

Це процес або діяльність направлена на створення певних структур даних, які називаються блокам для структури даних блокчейн, для підтримки мережі, на яких базуються криптовалютні платформи. Отримання нагороди за майнінг відбувається наступним чином — за кожне створення блоку майнер отримає встановлену винагороду. Також, цей процес підтримує емісію криптовалюти [11].

Функціонально суть майнінгу полягає в тому, що проводиться серія обчислень з перебором певних параметрів для знаходження хеш-значення, яке задовольняє певним властивостям. При чому кожна криптовалюта використовує різні методи обчислень і зазвичай час обчислень дуже великий — все це справедливо для доведення виконаної роботи (Proof Of Work, PoW).

2.3.1 Майнінг ферма

Майнінг ферма — це особливий вид серверних потужностей, які оснащені обладнанням для видобутку криптовалюти, який виник

внаслідок постійного підвищення складності процесу майнінгу, що вимагає велику кількість ресурсів (технічних, енергетичних тощо).

Фактично, майнінг ферма представляє з себе приміщення з великою кількістю комп'ютерного обладнання, які вирішують задачі для створення блоків.

Основними характеристиками майнінг ферм:

– hash rate — швидкість вирішення математичної задачі та вимірюється величинию "хеш за секунду" H/s;

– кількість використаної електроенергії — це найважливій показник для майнінгових ферм, оскільки бувають випадки, коли обладнання потребує настільки багато електроенергію, що майнити криптовалютуту стає не рентабельною справою;

На даний момент існує декілька видів обладнання, яке використовується для майнінгу, а саме:

– CPU-майнінг — найперша та найдоступніша версія майнінгу, яка використовує для обрахунку криптографічних задач CPU. Через те, що складність алгоритмів для криптовалют виросла, використовувати CPU для майнінгу стало практично неможливим;

– GPU-майнінг — використанням графічних процесорів для майнінгу. За рахунок особливою структури графічних процесорів підвищилась швидкість ферм в цілому.

– FPGA-майнінг — вид майнінгу з використанням модернізованих графічних процесорів з низьким енергоспоживанням.

– ASIC-майнінг — майнінг з використанням спеціального обладнання, яке створено виключно для майнінгу і для роботу з криптовалютою. Ефективність такого обладнання в рази перевищує ефективність GPU або FPGA.

На даний запуск майнінг ферми вимагає великих ресурсів, інвестицій, і на даний момент набирає популярність вид майнінгу в пулах (Mining Pool).

2.3.2 Mining Pool

Майнінг пул — це мережа об’єднаних потужностей усіх ферм користувачів, які здають в оренду власні обчислювальні потужності в межах мережі. При такому майнінгу, весь дохід пулу рівномірно розділяється між усіма учасника пулу в залежності від того, скільки обчислювальних ресурсів кожен користувач вирішив надати для пула. В контексті пулу, кожен користувач отримає так звані “шари” (від англ. слова share), які і визначають частку доходу, яку отримає користувач за майнінг в пулі. Концепцію майнінгу пулів була створена і втілилась в життя, коли для кожного користувача з обладнанням для майнінгу криптовалюти генерація блоку в мережі почала складати декілька століть. Саме в цьому полягає основна користь майнінг пулів для майнерів, оскільки майнінг пул швидше згенерує блок і розподілить нагороду між усіма.

Розподіл нагороди за блок може відбуватись за декількома алгоритмами:

– Proportional — пропорційна модель, при якій нагорода з блок чітко ділиться пропорційно надісланій кількості шар від кожного майнера;

– PPLNS (Pay Per Last N Shares) — виплата розраховується за кількістю шар, які надіслані не за час, що пройшов між двома згенерованими блоками, а за фіксовану кількість певних часових проміжіків, якій називаються shift (шіфти). Виплата відбувається після того, як пул знайде черговий блок. Якщо блок довгий час не знаходиться, то виплатим в даному випадку будуть росту, а якщо

блоки будуть знаходитись за короткі проміжки часу, то виплата за кожний окремий блок будуть знижуватись. Наприклад, за 5 годин пул знайшов 2 блоки. Стандарта нагорода за 1 блок в мережі Bitcoin складає 25 BTC. Отже, майнер отримає 25 BTC * 2 / 100, тобто 0.5 BTC. Якщо ж за 10 годин не було знайдено жодного блоку, то дохід майнера складе всього 0.25 BTC.

– PPS (Pay Per Share) — фіксовані виплати за кожну прийняту пулом шару. В цьому випадку в пулі визначено фіксовану ціну за шару. Ця ціна вираховується на основі нагороди за блок, яка розділена на поточну складність мережі і множиться на кількість шар користувача зі складністю 1.

3 АРХІТЕКТУРА ТА ОБҐРУНТУВАННЯ ВИБОРУ КОМПОНЕНТ

Система віддалено управління процесами та ресурсами майнінгових ферм розроблена з використанням мікросервісної розподіленої архітектури, технології контейнеризації та асинхронної взаємодії. Використані підходи дають можливість спростити підтримку та реалізацію нового функціоналу.

Враховуючи велику кількість проблем і задач, які потрібно реалізувати в рамках системи, було прийнято рішення розробити систему з наступними компонентами:

– P2P та VNC клієнти (для підключення до віддаленого робочого місця);

– API бекенд (авторизація, аутентифікація, реєстрація та системні дані);

– Minio файловий сервер (для збереження файлів та бінарних файлів для запуску майнерів);

– Jump сервер (для додатково функціоналу в рамках p2p клієнтів);

– агент, який встановлюється на кожній фермі (хост-машині) – забезпечує виконання всіх команд для запуску певних програм для майнінгу, а також збір інформації для побудови статистичних графіків;

– сервіс сповіщень (сповіщення на електронну пошту та популярні месенджери);

– Telegram-бот (для відправки сповіщень в Telegram канал);

– кросплатформений UI клієнт (десктоп версія для UNIX/Windows операційних систем та WEB-клієнт);

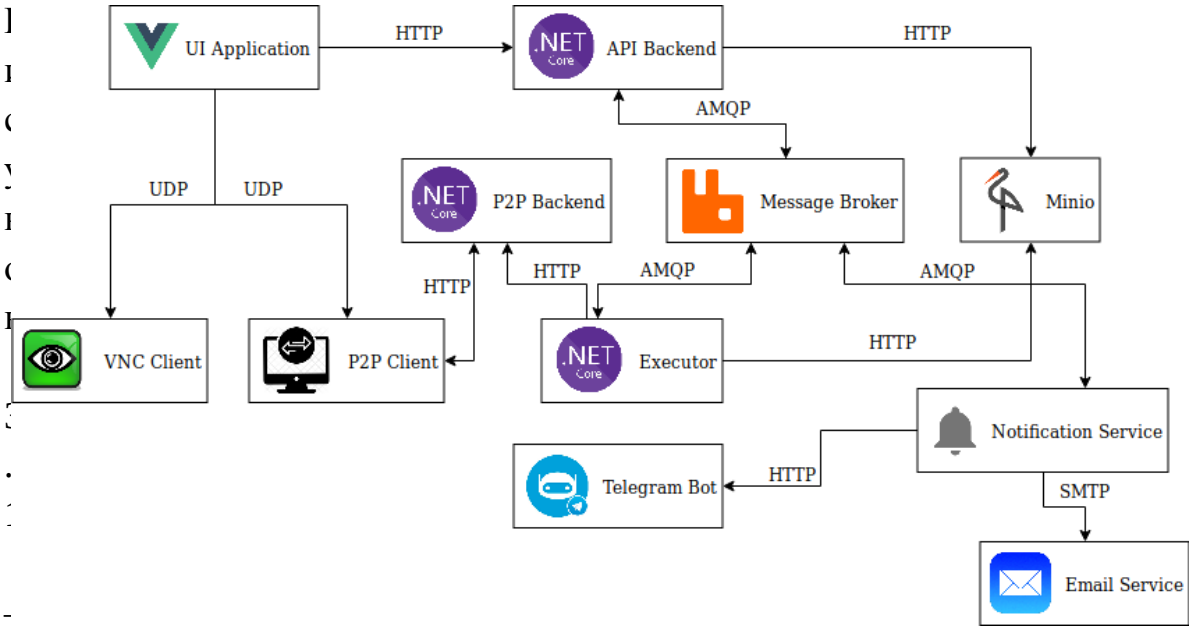
P2P клієнти працюють з використанням протоколу UDP та STUN, додатково використовуючи Jump сервер, у випадках, коли не спрацьовує пряме P2P з’єднання – наявність певних типів NAT’ів, які неможливо

обійти. Його основна функція полягає в тому, щоб встановити підключення між двома хостами.

Для комунікації між агентом, сервісом сповіщень та API сервером використовується брокер повідомлень RabbitMQ (AMQP протокол), який забезпеченує асинхронну взаємодію всі частин та використовує механізм Remote Procedure Call.

В решті випадків використовується звичайна комунікація засобами HTTP протоколу (REST API, Telegram Bot, Minio).

На рисунку 3.1 зображено високорівневу архітектуру застосунку за принципом мікросервісної модульної побудови. Стрілки позначають напрям взаємодії та протокол спілкування – це важливо, оскільки у випадку реалізації системи протоколи достатньо різні і принципова відрізняються один від одного.



Абстрактна архітектура системи

3.1 Компонента графічного інтерфейсу

На рисунку 3.1 компонента графічного інтерфейсу це UI Application, для реалізації якої було використано технології VueJS та ElectroJS, завдяки яким вдало було розроблено кросплатформенний варіант

застосунку, а також WEB-версію. На рисунку 3.2 можна побачити початковий екран застосунку для входу в систему. Користувач повинен ввести свій нікнейм та пароль для авторизації і синхронізації сесійних та постійних даних.

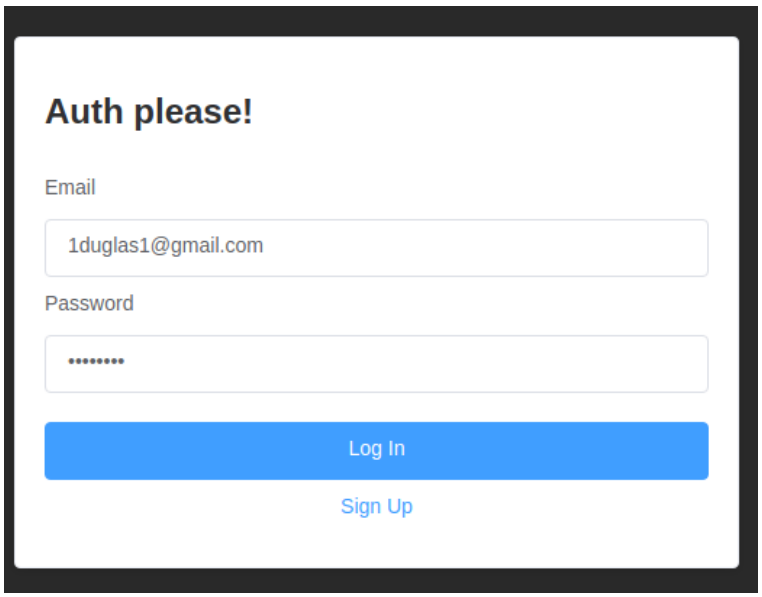


Рисунок 3.2 – Скрін входу в систему

На рисунку 3.3 зображено скрін форми реєстрації нового пристрою. Серед полей необхідно ввести назву ферми та встановити пароль для доступу до пристрою. Під типом тут слід розуміти, які комплектуючі встановлені для майнінгу – GPU або ASIC, оскільки одночасно обидва типи комплектуючих не можуть працювати в рамках однієї ферми. Перед цим користувач повинен встановити на кожній фермі агентський застосунок, щоб можна було підключатись та моніторити систему.

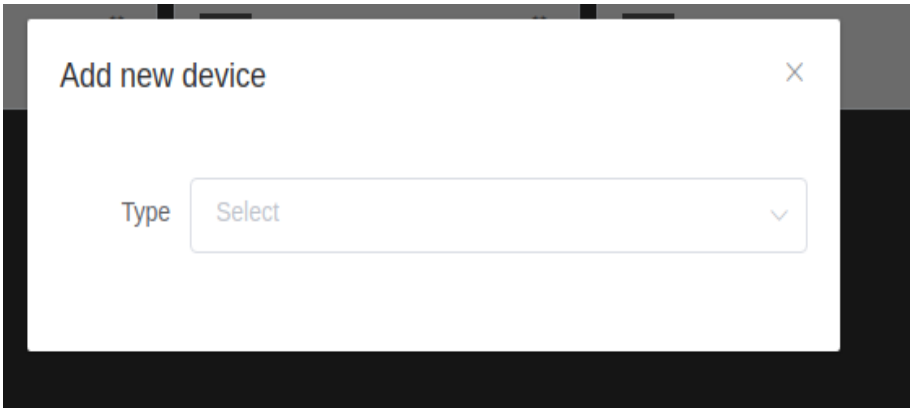


Рисунок 3.3 – Скрін додавання нової ферми до акаунту користувача

На рисунку 3.4 зображено скрін з додатково зображеним полем з кнопками, які дозволяють виконати певні операції, а саме: створити групу для групування пристроїв і надання можливостей управління групами для інших учасників, що в свою чергу дає можливість делегувати управління великою кількістю ферм різним людям, додати пристрій або додати ферму – реєстрація ферми в рамках акаунту користувача, редагувати профіль, відповідно, відредагувати користувацькі дані акаунту та вихід.

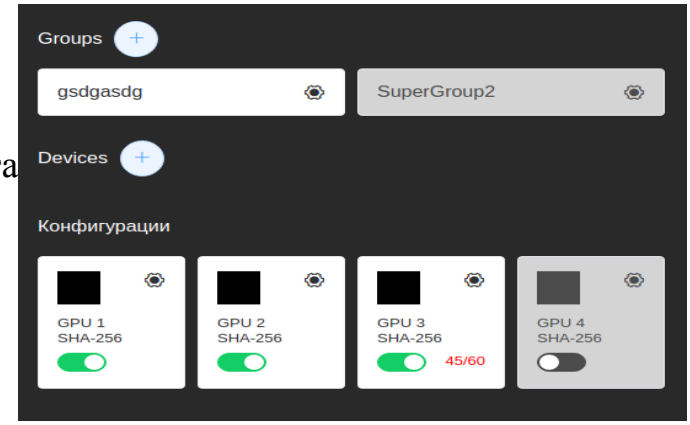


Рисунок 3.4 – Скрін для створення груп, додавання нових хостів (ферм) та список наявних пристроїв для майнінгу на фермі

На рисунку 3.5 зображено список відеокарт в певній групі. На скріні можна побачити номери гаманців на які відправляється видобувана криптовалюта, а також поточні алгоритми для обрахунку задач майнінгу.

Також в правому кутку панель інструментів для керування усіма фермами та налаштуванням аліасів (синонімів) для криптовалюта та іншого.

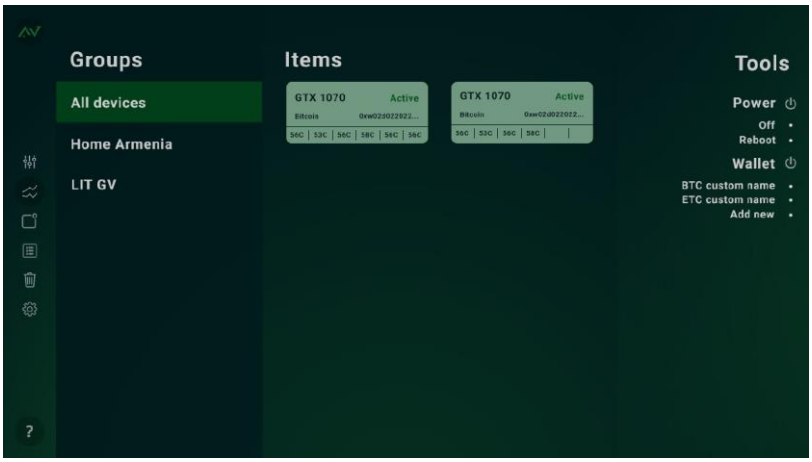


Рисунок 3.5 – Скрін зі списком відеокарт на фермі

3.2 Брокер повідомлень

Підпрограмне забезпечення або брокер повідомлень – це AMQP клієнт, в даному випадку RabbitMQ, який зображено на рисунку 6.1 як елемент під назвою Message Broker. Це один із центральних засобів та протоколів для внутрішньосервісної взамодії усіх компонентів системи, який використовується для відправки та прийому повідомлень в асинхронному режимі, комунікації агентських машин з цілою системою, відправкою команд для виконання на цільових хостах та агрегації повідомлень, що вміщують в собі статистичну інформацію про стан кожної ферми, що зареєстрована в системі.

Основною метою використання такого специфічного програмного комплексу було зменшення затримок при комунікації з іншими компонентами системи і реалізації принципів “відправив та забув”, “відправив та дочекався”.

У випадку принципу “віпдравив та дочекався” раціонально використати механізм RPC RabbitMQ, оскільки всі з’єднання не будуть обриватись мережею через високий час очікування відповіді.

Структура черг має наступний формат:

– черги для статистики – формуються динамічно в залежності від кількості екземплярів, які записують дані для статистики;

– черги для комунікації з агентами – формується статичний пул черг, до яких підключаються агенти з ферм. Це нобхіжно для балансування навантаження і затримки між чергами та приймальними сторонами;

– черги для сповіщень – окремі черги, які необхідні для комунікації із мікросервісом, що відповідальний за відправку сповіщень по каналам електронної пошти та Telegram месенджера;

3.3 Віддалений доступ до ферм

Для реалізації віддаленого доступу було обрано дві технології: VNC (Virtual Network Computing) та BitTorrent протокол (сімейство P2P). VNC система, через відкритість, простоту та велику кількість бібліотек на різних мовах програмування, дозволяє достатньо легко реалізувати віддалений доступ до ферми у вигляді віддаленого робочого столу в контексті окремого застосунку, а не пропрієтарного рішення. Аргументи на користь VNC приведені в розділах 2.1.1 та 2.1.2. Ці елементи зображені на рисунку 3.1 як VNC Client, P2P Client, P2P Backend.

BitTorrent протокол використовується для створення більш децентралізованої системи. Водночас, це дозволяє збільшити стабільність подібного роду застосунку, але і породжує ряд проблем: проблема обходу NAT’ів та трансляції підключень VNC. Перша проблема вирішується за допомогою STUN/TURN протоколів, які описані в розділах 2.2.2, 2.2.3. VNC не володіє функціоналом проходу NAT’ів, тому було необхідно

знайти рішення, яке дозволить поєднати можливості віддаленого доступу та вирішить купу мережевих проблем. В контексті таких проблем і було вирішено розробити зв'язку VNC + BitTorrent. Підключення VNC транслюється через BitTorrent протокол з допомогою перекодування бітів, які приходять в певному форматі на клієнт BitTorrent протоколу, у формат зрозумілий для VNC. Емпірично було доведено, що такий підхід не вносить значних затримок при передачі кадрів віддаленого робочого місця в мережі.

Також, існують варіанти, коли BitTorrent клієнт не має можливості встановити з'єднання напряму з фермою, тому в такому випадку передбачено наявність Jump серверу, який є посередником для підключення між двома вузлами. Такий самий принцип використовується при віддаленому доступі засобами TeamViewer. Такий метод доступу є найлегшим у порівняння з варіантом P2P мережі. Але при цьому, система стає більш залежною від центрального серверу з'єднань. У випадку великої кількості підключень постає проблема постійного доступу до серверу і необхідно проектувати схеми вертикального і горизонтального масштабування, щоб уникнути падіння серверів та простою. Саме тому застосунок має гібридну архітектуру для підключень до віддалених вузлів.

3.4 Сервер збереження даних користувача

Оскільки дана система це в більшій мірі WEB-затосунок, було розроблен API (Application Programming Interface) за допомогою підходу REST (Representational State Transfer), який забезпечує доступ до інформаційних ресурсів системи, користувача. Фактично ця частина застосунку є так званим медіатором або посередником системи, який трансляє потрібні запити в потрібні частини системи. Ця компонента зображена на рисунку 3.1 як API Backend.

В цій частині застосунку повно реалізовано доменну модель системи з елементами авторизації засобами JWT (Json Web Token), збереження сесійних та постійних даних користувача, взаємодію з рештою компонентів системи. Крім того, графічни інтерфейс користувача повністю спирається на REST API цієї частини системи для відображення даних у вигляді форм та графічних компонентів. Такий підхід де-факто стандарт в розробці WEB-застосунків.

3.5 Агентський застосунок

Концепція застосування агентів як сервісів є достатньо широким архітектурним підходом для організації взаємодії різних віддалених середовищ в комп'ютерних системах – від застосунків для моніторингу серверів і до програм, які керують кластерами та виконують певні вказівки адміністратора. На рисунку 3.1 агентська компонента зображена як Executor.

В контексті розробленої системи агентський застосунок являє собою широкопрофільне програмне забезпечення, яке вміє виконувати команди, змінювати стан віддаленого робочого місця, проводити аналіз та збір статистики у вигляді часових рядів. Реалізація подібного функціоналу дає такі можливості:

– єдиний сервіс, який виконує широкий спектр задач;

– не потрібно встановлювати велику кількість специфічного програмного забезпечення та інтегрувати з різними клієнтами;

– збір різноманітної статистики(час роботи, дохідність, потужність тощо) про роботу ферм;

– віддалений доступ для ручного налаштування;

Таким чином вирішується велика кількість проблем пов'язаних зі спецілізованим процесом майнінгу, для якого достатньо мало розроблено подібних рішень.

3.6 Сервіс сповіщень

Притримучиюсь мікросервісної архітектури, було розроблено окремий сервіс для сповіщень. Необхідність в такому підході полягає в тому, що простіше підтримувати та вдосконалювати функціонал сповіщень в рамках невеликого проекту. Розгортання такого сервісу займає дуже мало часу, як і його оновлення. На рисунку 3.1 даний сервіс зображено як Notification Service. Оскільки в рамках системи декілька компонентів можуть незалежно відправляти сповіщення, то було доцільним розробляти цей сервіс як окремий застосунок

На даний момент в сервісі проведено інтеграцію з електронною поштою та Telegram-ботом, але при цьому сервіс готовий до інтеграції будь-яких каналів сповіщень. Протокол комунікації з цим сервісом – AMQP реалізований засобами брокеру RabbitMQ. Для даного сервісу найбільше підходить принцип “відправив та забув”, тому на всі повідомлення, відправлені в черги, відповіді не очікуються.

3.7 Файлове сховище

Кожний застосунок потребує місце для збереження статичних файлів, завантажених файлів та іншого. Найвідоміший формат взаємодії затосунку з файловою системою – це зберігати усі файли локально в директорії з проектом. Такий підхід має певні проблеми: з точки зору організації файлової системи та розгорнутого застосунку, формально, є всесторонній доступ до цих файлів, при чому вони не структуровані, або доводиться реалізовувати специфічний функціонал, який цим займається; необхідність резервних копій та їх архівування – в даному випадку усі бекапи будуть займати місце файлової системи застосунку, що в свою чергу може принести неочікувані проблеми з відсутністю вільного місця. Сучасні практики диктують, що файлова система для збереження великої кількості файлів повинна знаходитись окремо і необхідно

використовувати спеціальні програмні комплекси для взаємодії з такими файловими системи через секретні ключі. На рисунку 3.1 компоненту системи зображено як Minio.

Дана система була побудована на основі сервісу Minio, який має вбудований механізм авторизації та взаємодії з файловою системою серверу, на якому встановлено сервіс. Також, даний сервіс має достатню кількість бібліотек для розробки на різних мовах програмування, що спрощує інтеграцію з файловою системою до рівня виклику методів певних інтерфейсів і передачі кілька параметрів. Крім того, в сервісі присутній графічний інтерфейс, через який можна взаємодіяти з усіма файлами, при чому їх можна структурувати в так званих кошиках або англійською “buckets”. Великою перевагою даного сервісу є наявність CLI (Command Line Interface) застосунку, завдяки чому можна з легкістю описувати задачі резервних копій для планувальника Crontab.

3.8 Розробка системи

Враховуючи архітектурні підходи, які наведені в розділі 4, розробка системи супроводжувалась використанням новітніх підходів для розробки програмного комплексу з підтримкою масштабованості, простого оновлення та впровадження нового функціоналу. Також, розробка система супроводжувалась моделюванням UML діаграм — діаграми компонентів, розгортання, варіантів використання, ER, класів доменної моделі та функціональної частин.

3.8.1 Вимоги до системи

Аналіз існуючих рішень показав, що на даний момент універсальних засобів та інструментів керування процесом mining’у, практично не існує, а ті що віддалено намагаються вирішити проблеми в контексті магістерської дисертації мають досить мало функціоналу, мало інтерактивних можливостей та зазвичай вимагають фізичного доступу до

ферми для керування або ж встановлення іншого програмного забезпечення для віддаленого керування.

Більшість систем подібного характеру не можуть нормально відображати статистику усіх ферми на яких запущено програму-майнер, неможливість об’єднання велику кількість ферм в групи для зручного керування, а також відсутність одночасного перемикання певних налаштувань на усій множині наявних обчислювальних ресурсів.

В рамках магістерської дисертації до розроблюваної системи було поставлено ряд вимог, а саме:

– кросплатформність та наявність WEB-інтерфейсу;

– робота без блокуваня – кожен запит не повинен затримувати роботу системи, тому більшість операцій проводиться в асинхронному режимі;

– інтеграція широкого спектру програмного забезпечення для майнінгу для користувацького вибору;

– можливість групування усіх ферм та наявність ACL (Access Control List) для надання різного рівня доступу до груп;

– обов’язковий доступ до віддаленого робочого місця (ферми);

– наявністю сповіщень в різні канали за вибором користувача.

Головною метою даної магістерської дисертації було створення зручного програмного комплексу для оркестрації майнінгових ферм, зручного графічного інтерфейсу, вирішення мережевих проблем пов’язаних з NAT’ами.

3.8.2 Нефункціональні вимоги

В рамках розробки проекту магістерської дисертації було виділено нефункціональні вимоги на основі наступних критеріїв [12]:

– група runtime:

а) доступність (availability);

b) надійність (reliability);

c) вимоги до часу збереження даних (durability);

d) масштабованість (scalability);

e) вимоги до зручності використання (usability);

f) вимоги до безпеки (security);

g) вимоги до конфігурованості (configurability);

h) вимоги до продуктивності (performance);

i) обмеження (restrictions);

– Група design time:

a) вимоги до повторного використання реалізації або компонентів застосунку чи системи (reusability);

b) вимоги до розширюваності (extensibility);

c) вимоги до переносимості (portability)

d) вимоги до підтримки системи або застосунку (supportability);

e) вимоги до автоматичного та ручного тестування (testability);

3.8.2.1 Вимоги групи Runtime

Вимоги до доступності:

– час неперервної роботи системи повинен складати 24x7;

– мінімальний час простою системи повинен складати не більше 10 хвилин в рік без урахувань часу для оновлення системи;

Вимоги до надійності:

– агентський застосунок у разі непередбаченого вимкнення повинен автоматично перезапускатись. Час перезапуску повинен складати не більше 1 хвилини;

– серверні частини застосунку повинні здійснювати резервне копіювання усіх даних не рідше 3 разів на добу;

– система повинна бути продубльована з використанням “гарячого” та “холодного резервів” та повинен бути налаштований процес балансування навантаження;

– у випадку падіння серверної частини, система повинна автоматично перемикнутись на холодний резерв без простою;

Вимоги до довгострокового збереження даних:

– для збереження данних повинні використовуватись реляційна СУБД PostgreSQL та NoSQL база даних InfluxDB;

– time series дані повинні зберігатись не менше року;

– дані користувача не мають часових обмежень у збереженні;

Вимоги до масштабованості:

– горизонтальне масштабування системи повинне бути реалізоване з допомогою збільшення кількості контейнерів застосунку з використанням оркестратора;

– вертикальне масштабування повинне використовуватись у випадку, коли кількість контейнерів при горизонтальному масштабуванні дійшло до фізичного ліміту;

– будь-який вид масштабування не повинен привести систему до простою, тобто, ніяким чином не повинен відображатись на працездатності системи.

Вимоги до зручності використання:

– встановлення агентського застосунку повинно виконуватись з мінімальною кількістю кроків (2-4 кроки);

– агентський застосунок повинен автоматично налагоджувати роботу ферм – встановлювати необхідне програмне забезпечення для моніторингу ферм, підключатись до серверів;

– запуск програмного забезпечення для майнінгу повинен проводитись без технічних деталей з пропозиціями для конфігурації та однією кнопкою запуску;

– інтерфейс користувача клієнтської частини не повинен містити елементів командного рядку;

Вимоги до безпеки:

– віддалене підключення повинно бути захищене паролем;

– всі паролі для доступів повинні зберігатись у зашифрованому вигляді всередині бази даних;

– усі паролі повинні бути представлені у вигляді хеш-значення довжиною не менше 16 символів;

– усі публічні ключі повинні бути зашифровані засобами VeraCrypt;

– усі паролі повинні бути зашифровані засобами KeePass та VeraCrypt;

– з боку інфраструктури доступ повинен бути організований з урахуванням прав доступу адміністратора та використанням груп та користувачів на рівні операційної системи;

– повинна технічно вимагатись примусова зміна первинного паролю користувача при вході до інфраструктури;

– зміна паролей повинна проводитись не рідше ніж 3 місяці на рік;

Вимоги до конфігурованості:

– налаштування кластеру Kubernetes для розгортання повинно бути реалізовано засобами Ansible та TerraForm;

– розгортання системи повинно здійснюватись засобами Kubernetes з описаними конфігураційними файлами розгортання сервісів;

Вимоги до продуктивності:

– система повинна підтримувати не менше 5 тисяч віддалених підключень на день (в даному випадку розглядаються підключення з використанням jump серверу);

– система повинна підтримувати не менше 5 тисяч користувацьких сесій на день;

– час для запуску майнінгу повинен складати не менше 2-х хвилин;

– вибірка для статистичних даних по фермах повинна складати не більше 5 секунд;

– система повинна вміти оброблювати до 50 тисяч повідомлень різного роду на день;

Обмеження:

– мінімальна пропускна здатність мережі датацентру, в якому повинно бути розгорнути система , повинна складати не менше 1 Гбіт/с;

– мінімальна кількість оперативної пам’яті для кожного серверу в рамках кластеру повинна складати 32 Гб;

– для зберігання даних необхідно встановити 2 RAID 1 масива на HDD з об’ємом не менше 1 Тб;

3.8.2.2 Вимоги групи Desing Time

Вимоги до повторного використання:

– система повинна бути розроблена з урахуванням принципів SOLID, DRY, KISS для запобігання дублювання;

– система повинна бути розроблена з використанням мікросервісної архітектури;

Вимоги до розширюваності:

– система повинна проектуватись та розроблятись з урахуванням принципів SOLID, DRY, KISS;

– доставка оновлень повинна відбуватись в час підтримки;

Вимоги до переносимості:

– система повинна працювати з використанням платформи Docker

– система повинна працювати на сімействах операційних систем UNIX, Windows, Mac OS;

Вимоги до підтримки системи або застосунку:

– в системі повинен бути реалізований механізм журналювання помилок з можливістю графічного доступу для аналізу;

– критичні збої в системі повинні надсилати email, СМС сповіщення до людей відповідальних за підтримку;

Вимоги до автоматичного та ручного тестування:

– автоматичне тестування клієнтського застосунку повинне проводитись засобами Selenium, мовою C#;

– автоматичне тестування інших компонентів системи повинне бути реалізовано у вигляді Unit-тестів, функціональних тестів, інтеграційних тестів та повинні проганятись Continuous Integration системою перед кожним оновленням та зміною версій;

3.8.3 Функціональні вимоги

В рамках магістерської дисертації функціональні вимоги описані засобами UML у вигляді діаграми використання або діаграми прецедентів (англ. Use Cases diagram). Такий варіант опису фукціональних вимог дає можливість більш наочно побачити кількість ролей в системі та відповідно функціональними можливостями кожного з них.

3.8.3.1 Ролі користувачів

В таблиці 3.1 зображено ролі, які система надає для користувачів.

Таблиця 3.1 – Доступні ролі в рамках системи

Роль	Доступ
User	Авторизація, додавання ферм, запуск майнерів, створення груп

Продовження таблиці 3.1

Роль	Доступ
System admin	Додавання майнерів, майнінг

	пулів, монет, створення форм та опцій для майнерів
Admin	Зміна налаштувань для майнінгу в групі та фермі, зміна швидкості обертів кулерів на фермах
Super admin	Надання адмінських прав іншим користувачам, видалення груп, додавання гаманця, видалення гаманця, видалення ферми з групи, створеня шаблонів

3.8.3.2 Варіанти використання

В системі наявно 21 варіант використання, а також їх графічне представлення в додатку Д1. Натомість в рамках магістерської дисертації описано 8 основних варіантів використання, які суттєво впливають або описують систему. Далі неведено детальні пояснення кожного з них.

1) UC-01. Sign in.

Дійові особи: User, Admin, System admin, Super admin.

Мета: увійти в систему

Тригер: користувач вирішує авторизуватись в системі і заходить на сторінку авторизації.

Передумови: користувач зареєстрований в системі та вводить правильний нікнейм та пароль.

Результат: користувача авторизовані в системі.

Основний потік подій зображено в таблиці 3.2.

Таблиця 3.2 – Основний потік подій для варіанту використанням UC-01

№	Дійова особа	Крок
1	Користувач	Заходить на сторінку авторизації
2	Користувач	Вводить нікнейм та пароль акаунту
3	Система	Перевіряє чи існує користувач з таким нікнеймом та чи правильно введено пароль
4	Система	Підтвердження авторизації
5	Користувач	Переходить на головну сторінку

2) UC-02. Sign Up.

Дійові особи: User, Admin, System admin, Super admin.

Мета: зареєструватись у системі.

Тригер: користувач вирішує зареєструватись в системі та переходить на сторінку реєстрації.

Передумови: користувач не зареєстрований в системі та вводить унікальні і правильні дані.

Результат: користувач зареєстрований в системі.

Основний потік подій зображено в таблиці 3.3.

Таблиця 3.3 – Основний потік подій для варіанту використанням UC-02

№	Дійова особа	Крок
1	Користувач	Заходить на сторінку реєстрації

2	Користувач	Вводить нікнейм, електронну пошту, пароль для створення акаунту

Продовження таблиці 3.3

№	Дійова особа	Крок
2	Користувач	Вводить нікнейм, електронну пошту, пароль для створення акаунту
3	Система	Перевіряє унікальність (нікнейм, електронна пошта) та правильність даних (пароль).
4	Система	Реєструє користувача та відправляє сповіщення на пошту
5	Користувач	Переходить на головну сторінку

3) UC-03. Add rig.

Дійові особи: User, Admin, System admin, Super admin.

Мета: зареєструвати ферму в рамках акаунта користувача.

Тригер: користувач вирішує запустити майнінг на своїй фермі.

Передумови: користувач авторизований в системі.

Результат: користувач додав до акаунту нову ферму.

Основний потік подій зображено в таблиці 3.4.

Таблиця 3.4 – Основний потік подій для варіанту використанням UC-03

№	Дійова особа	Крок
1	Користувач	Встановлює агент на фермі

2	Користувач	Копіює ідентифікатор та пароль для доступу до ферми
3	Користувач	Заходить на сторінку Devices та натискає кнопку “+” для додавання нової ферми
4	Користувач	Вводить скопійований ідентифікатор та пароль доступу у форму
5	Система	Перевіряє чи доступна ферма з заданим ідентифікатором

Продовження таблиці 3.4

№	Дійова особа	Крок
6	Система	Агент перевіряє пароль при підключенні, здійснюється прив’язка ферми до користувача
7	Користувач	Спостерігає оновлення списку ферм

4) UC-04. Start mining on rig.

Дійові особи: User, Admin, System admin, Super admin.

Мета: запустити майнінг на фермі.

Тригер: користувач вирішує запустити майнінг на своїй фермі.

Передумови: користувач авторизований в системі, користувач додав до акаунту нову ферму.

Результат: майнінг запущено.

Основний потік подій зображено в таблиці 3.5.

Таблиця 3.5 – Основний потік подій для варіанту використанням UC-04

№	Дійова особа	Крок
1	Користувач	Обирає потрібну ферму
2	Користувач	Натискає кнопку “Start mining”
3	Користувач	Користувач обирає необхідні опції для майнера, або обирає запропоновані
4	Система	Запускає майнер на фермі
5	Користувач	Система інформує користувача про успішний запуск майнера на фермі

5) UC-05. Remote connect to rig

Дійові особи: User, Admin, System admin, Super admin.

Мета: підключитись віддалено до ферми.

Тригер: користувач вирішує змінити налаштування ферми вручну.

Передумови: користувач авторизовано, доадно нову ферму.

Результат: користувач віддалено підключився до ферми.

Основний потік подій зображено в таблиці 3.6.

Таблиця 3.6 – Основний потік подій для варіанту використанням UC-05

№	Дійова особа	Крок
1	Користувач	Обирає потрібну ферму
2	Користувач	Натискає кнопку “Remote connect”
3	Користувач	Користувач вводить пароль до ферми

4	Система	Перевіряє правильність вводу пароля та ініцює VNC підключення
5	Користувач	Користувача підключено та має змогу здійснювати віддалені операції

6) UC-06. Create group.

Дійові особи: User, Admin, System admin, Super admin.

Мета: створити групу для групування ферм.

Тригер: користувач вирішує згрупувати ферми в одну логічну одиницю.

Передумови: користувач авторизований в системі, користувач додав до акаунту мінімум 1 ферму, введено унікальну назву групи.

Результат: користувач створив групу.

Основний потік подій зображено в таблиці 3.7.

Таблиця 3.7 – Основний потік подій для варіанту використанням UC-06.

№	Дійова особа	Крок
1	Користувач	Заходить на сторінку Groups

Продовження таблиці 3.7

№	Дійова особа	Крок
2	Користувач	Натискає кнопку “+”
3	Користувач	Заповнює форму створення групи, вводячи назву групи.
4	Система	Перевіряє чи не створена ідентична група в рамках акаунту.
5	Система	Сповіщає користувача про

		створення нової групи.

7) UC-07. Change mining settings of group/rig.

Дійові особи: Admin, Super admin.

Мета: змінити або створити налаштування майнінгу для групи або ферми.

Тригер: користувач вирішує змінити налаштування для майнінгу, змінити криптовалюту тощо.

Передумови: користувач авторизований в системі, користувач додав до акаунту мінімум 1 ферму, користувач створив групу з однією фермою.

Результат: користувач змінив налаштування та запустив новий майнер на фермі або в групі ферм.

Основний потік подій зображено в таблиці 3.8.

Таблиця 3.8 – Основний потік подій для варіанту використанням UC-07.

№	Дійова особа	Крок
1	Користувач	Заходить на сторінку Devices
2	Користувач	Обирає групу або ферму
3	Користувач	Натискає кнопку “Mining settings”
4	Користувач	Змінює відповідні параметри
5	Система	Перевіряє правильність введених даних відповідно до можливих налаштувань майнерів

Продовження таблиці 3.8

№	Дійова особа	Крок
6	Система	Запускає майнер на фермі з новими

		налаштуваннями

8) UC-11. Create flightsheet.

Дійові особи: Super admin.

Мета: створити шаблон налаштування для майнінгу.

Тригер: супер адміністратор вирішує створити шаблон налаштувань.

Передумови: користувач авторизований в системі, користувач є супер адміністратором групи.

Результат: успішне створення шаблону

Основний потік подій зображено в таблиці 3.9.

Таблиця 3.9 – Основний потік подій для варіанту використанням UC-07.

№	Дійова особа	Крок
1	Користувач	Заходить на сторінку Groups
2	Користувач	Натискає на кнопку “Create flightsheet”
3	Користувач	Заповнює необхідні дані для форми та вводить ім’я шаблону
4	Система	Перевіряє назву шаблону в рамках групи та правильність введених даних
5	Система	Сповіщає користувача про успішне створення шаблону

3.8.4 Послідовності взаємодії

В додатках Д3 та Д4 відображено послідовності взаємодії клієнта з системою в технічому контексті.

Додаток Д3 відображає послідовність віддаленого підключення – яким чином запит переходить між компонентами системи та яким чином ініціюється підключення. Послідовність кроків наступна:

– користувач здійнсює запит на підключення у компоненти API;

– компонента API оброблює запит, знаходить інформацію (1.1) та повертає відповідь, в який інкапсульована інформація про підключення до ферми (1.2);

Далі можливі два варіанти розвитку подій – пряме P2P підключення або підключення з використанням Jump серверу. Така умова відображена блоком alt (від англ. Alternative):

1)користувач здійснює пряме P2P підключення до ферми (2);

2)користувач здійснює запит на підключення в рамках якого Jump сервер знаходить ферму (3), повертає відповідь (3.1) та встановлює підключення (4, 4.1)

Завершальним етапом взаємодії є встановлення VNC сесії (5).

Додаток Д4 відображає узагальнену послідовність запуску команди на майнінг обладнанню:

– користувач відправляє запит до компоненти API про запуск команди (1) ;

– компонента API відправляє команду в чергу брокера (1.1) та повертає відповідь про успішність відправки (1.2);

– комонента Rig/Agent дістає з черги команду (2) та виконує її (3);

– після виконання Rig/Agent повертає в окрему чергу відповідь (3.1);

– API компонента дістає результат виконання команди (4) та відправляє сповіщення користувачу про виконання команди (5).

3.8.5 Доменна модель

Розробка доменної моделі є одним із найголовнішим етапів розробки програмного застосунку. На цьому етапі відбувається осмислення термінології області криптовалюти та її подальшу реалізацію у вигляді об’єктної моделі. Представлення доменної моделі зображено в наступних додатках:

– Додаток Д6 — ER-модель;

– Додаток Д5 — діаграма класів доменної моделі;

В рамках системи є дві основні сутності — Ферма (англ. Rig) та Користувач (англ. User). На основі досліджень доменної моделі на кожній фермі повинно запускатись спеціалізоване програмне забезпечення з певними налаштуваннями. В даному випадку за це відповідають сутності — Miner, MiningSettings, MiningPool, Wallet. Сутність Miner інкапсулює необхідно інформацію про програмне забезпечення для майнінгу — вказано сумісність з операційними системами, шлях до бінарного файлу, назва файлу. Сутність MiningSettings в свою чергу зберігає більш розширений список налаштувань із прив’язкою до конкретної ферми. Такий підхід дозволяє розуміти, який майнер з якими налаштуваннями запущений на даний момент, щоб мати можливість відобразити це з боку клієнтського застосунку.

Сутність MiningPool виражає одну опцію для запуску майнерів, оскільки можуть запускатись у двох режимах — індивідуально та в пулах. Майнінг пули підтримуються не усіма майнерами, тому завдяки цієї сутності можна прослідковувати зв’язок підтримки майнерами певних опцій.

Сутність Wallet також є важливою саме для користувача, оскільки в ній інкапсульовано інформацію про адресу гаманця куди буде надходити винагорода за майнінг.

Сутність MiningDevice відображає кожен окремий обчислювальний пристрій, який використовується на фермі — GPU, CPU чи ASIC. Це необхідно для того, щоб збирати статистичну інформацію — температуру, хешрейт, вольтаж, оберти кулерів тощо. В рамках програмного комплексу закладались вимоги щодо можливості керування та відображення низькорівневних властивостей фрем.

В рамках доменної моделі також реалізовано спрощену модель ACL — Access Control List, що забезпечети розмежування користувачів по правам в групах. Групування є однією з головних функціональних можливостей, оскільки дасть змогу в один клік запустити на десятках ферм програмне забезпечення для майнінгу. Отже, список доступу реалізовано на основі наступний сутностей — Role, Group, GroupUser.

Сутність ролі дає можливість агрегувати певну кількість ролей за замовчуванням — користувач, адміністратор, супер адміністратор. При цьому закладається можливість розширення — достатньо буде додати нову роль і описати необхідні правила в рамках застосунку. Сутність груп відповідає за агрегацію великої кількості ферм в одному логічному домені. Саме в цьому випадку використовується ACL, щоб можна було призначити адміністраторів, а власнику лише спостерігати за процесами. Сутність GroupUser фактично є доповненою реалізацією підходу розвязочної таблиці у вигляді окремої сутності. Ця сутність дає можливість прив’зати користувача до ролі в рамках групи.

Сутність FlightSheet — дає можливість створення шаблонів налаштувань для майнерів. Це корисно у випадку, коли є велика кількість ферм, створюється FlightSheet і використовується як MiningSettings на всі ферми.

Також, в доменній моделі присутні допоміжні сутності — Coin, OperationalSystem, Event, ResetPassword. Це, відповідно до порядку, криптовалютна монета, операційна система, подія, скидання паролю.

Сутності Event та ResetPassword необхідні для збереження подій, коли, наприклад, користувач забув свій пароль від акаунту в системі, а також багато інших, які будуть реалізовані в майбутньому.

Таким чином, можна зробити висновок, що в рамках магістерської дисертації доменна модель пророблена достатньо детально та з розрахунком на майбутній потенційний функціонал, який тільки покращить систему і дасть можливість швидше взяти нішу криптовалютного ринку.

В рамках Entity Relation діаграми було відображено модель таблиць реляційної бази даних. Дана діаграма є абстрактним представленням, що в свою чергу означає, що реалізація на конкретній СУБД може варіюватись в рамках типізації.

Таблиці іменуються в множині, щоб розуміти, що зберігається множина даних. Також, для реалізації ООП наслідування було використано шаблон проектування Single Table Inheritance [13] [14] [15]. Наведено, приклад. На рисунку 3.6 зображено приклад об’єктної моделі.

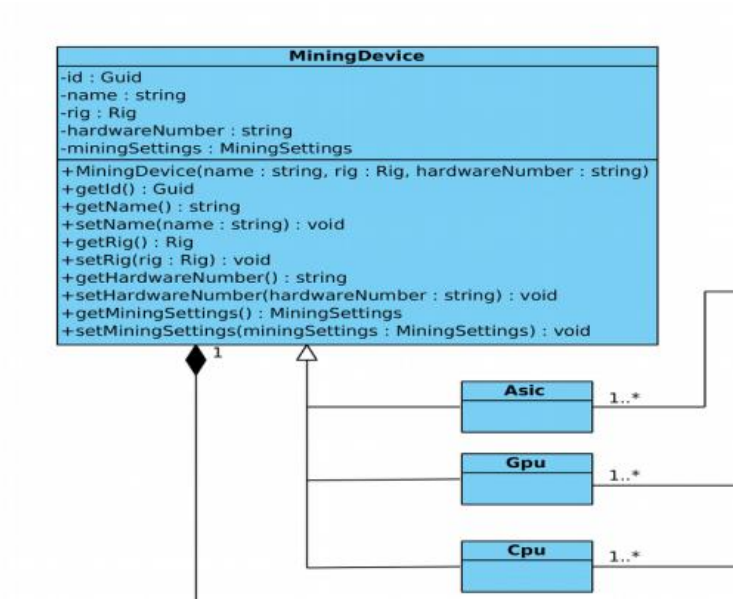


Рисунок 3.6 — Наслідування класів Asic, Gpu, Cpu від абстрактного класа Mining Device

Це достатньо зручно, оскільки при даному підході відпадає потреба у дублюванні коду. Але при цьому розробник при використанні ORM має вибір — створювати під кожну сутність окрему таблиці з повторенням властивостей, чи вмістити все в одну. Останній варіант і представляє собою вище описаний підхід.

В рамках ER діаграми даний підхід реалізовується наступним чином: в таблицю додається нове поле — дискримінатор (англ. Discriminator). В цьому полі інкапсулюється системне значення ORM — назва сутності. Фактично кожен кортеж в таблиці представляє собою окрему сутність, об'єкт якої може змінюватись в залежності від дискримінатор. З точки зору Entity Relation моделі та ООП це найкращий варіант. Реалізацію можна побачити на рисунку 3.7.

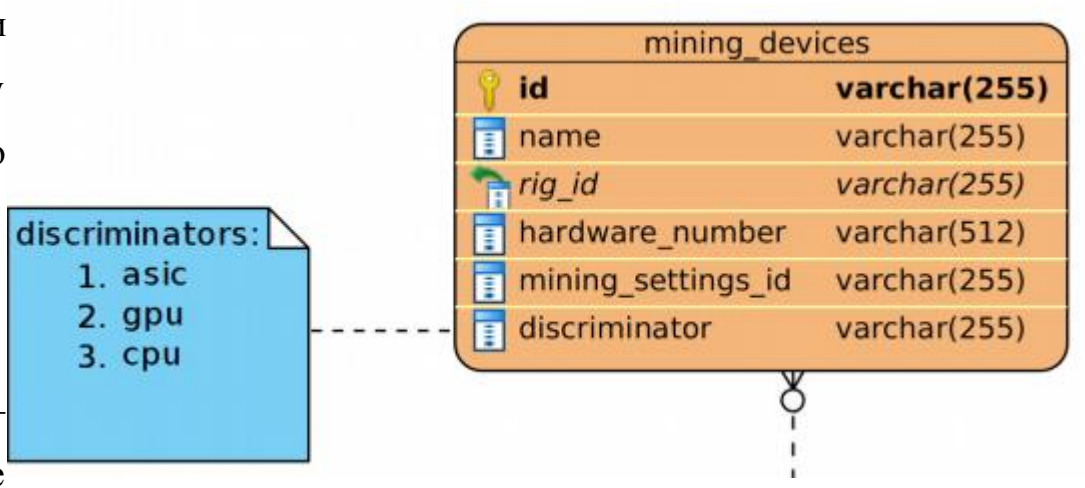


Рисунок 3.7 — Реалізація Single Table Inheritance в ER моделі

Отже, в даному розділі було наведено короткий опис розроблених діаграм, які синтезовані на основі опрацювання доменної моделі. На основі даного існує ціла методика розробка програмного забезпечення, яка називається — Domain-Driven Design (DDD) — набір принципів, схем, які дають можливість створити оптимальні об'єктні системи. Суть полягає у створенні програмних абстракцій, які і називаються моделями

предметних областей програмного забезпечення. Крім того, в рамках доменної моделі, крім зберігання даних, інкапсулюється бізнес-логіка, яка відповідає за зв’язок області використанням з програмни кодом. З цього твердження також випливаються два підходи в реалізації бізнес-логіки:

– rich domain model — багато доменна модель, коли вся бізнес логіка реалізована в рамках сутностей. В такому випадку, шар, який відповідає за збереження даних, також займається функціональним обробленням цих даних;

– anemic domain model — анемічка модель даних, коли бізнес-логіка релізована за рамками доменної моделі. В такому випадку, усі сутності займаються лише представленням даних в об’єктному вигляді всередині коду проекту.

Якщо коротко, то DDD — це набір правил, побажань, які дозволяють побудувати оптимально архітектуру рішення, створити розширювану модель даних та спростити розуміння програмних комплексів в цілому.

3.8.6 Програмні компоненти системи

Крім розуміння доменної моделі, зажди потрібно мати осяжну картину поточного високорівневого стану системи у вигляді абстрактного представлення архітектури та інтерфейсів. Таку можливість надають, так звані діаграми компонентів. В рамка магістерської дисертації ця діаграма представлена в додатку Додаток Д2. Перед читанням цієх діаграми, слід розібратись в нотації цього типу UML діаграм. На рисунку 3.8 наведено нумерацію всіх елементів діаграми компонетів. Нижче наведено опис кожного з них:

1)компонент — функціональний елемент систем. Компонент надає та використовує поведінку через інтерфейси, а також може використовувати інші компоненти;

2)інтерфейс, який надається — елемент, який надає певний виклик, який реалізується компонентом, і доступний для використання іншими елементами системи;

3)інтерфейс, який використовується (вимагається) — елемент, який необхідний для з'єднання з компонентами системи, які надають певні операції, що потребуються.

Після невеликих роз'яснень можна приступити до безпосереднього пояснення діаграми компонентів для системи автоматизації майнінгу.

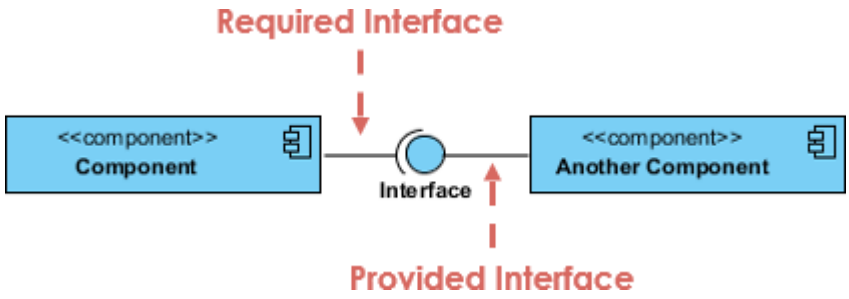


Рисунок 3.8 — Тестова діаграма компонентів для пояснення нотацій

Якщо розглянути додаток, то функціонально система складається з чотирьох основних функціональних підсистем:

– сервіс сповіщень (Notification Service) – мікросервіс, що відповідає за надсилання сповіщень. В свою чергу складається з компонентів для сповіщень по каналах Telegram та електронних листів. Надає у користування інтерфейс Notify. В даному випадку цей виклик є дуже високим рівнем абстракції, оскільки в рамках реалізація виклик відбувається з використанням протоколу AMQP, брокеру повідомлень та багато іншого. Детальна структура мікросервісу відображена в додатку Д7 (діаграма класів реалізації);

– API (Application Programming Interface) у вигляді Restfull Services – це одна із основних підсистем програмного комплексу. Тут зберігаються дані користувачів, наявні всі можливість для взаємодії

з усіма підсистемами та елементами. Складається з таких елементів: Authentication та AgentClient. Перший елемент відповідає за авторизацію та реєстрацію користувача і надає наступні інтерфейси у використанням – Login, Registration, а другий за комунікацію з агентами, які встановлені на фермах і лише потребує використанням інтерфейсу, який надають агентські застосунки. Сама підсистема API надає один великий інтерфейс – REST API, який використовується клієнтським застосунком для відображення даних та взаємодії із системою;

– інтерфейс користувача (UI Client) – графічний інтерфейс користувача, який агрегує в собі наступні функціональні елементи – VNCClient, UserInterface, P2PClient. VNCClient потрібен для встановлення віддаленого доступу до ферми і потребує інтерфейсу Connect від елементу VNCServer агентської підсистеми. UserInterface має лише одну залежність у вигляді REST API інтерфейсу підсистеми API. P2PClient має залежність у вигляді інтерфейсу Connect іншого елементу P2PClient агентської підсистеми та Establish Connection елементу Jump Server.

– агент (Agent) – найскладніша підсистема, яка агрегує в собі елементи Monitoring, Executor, VNCServer, P2PClient. Щодо останніх двох елементів – їхні інтерфейси описані в попередньому пункту. Елемент Monitoring надає інтерфейс Send Statistics, який використовується сервісом Statistics Writer, Executor – Run Miner, Run Command. Перший інтерфейс всередині підсистеми і в свою чергу потребує інтерфейс системи файлового сховища Get Miners Binary. Наступний інтерфейс виноситься для використання рештою сервісів;

– файлове сховище (File Storage) – сервіс для зберігання файлів, картинок тощо. Надає єдиний інтерфейс Get Miners binary.

3.8.7 Розгортання системи

Для такого роду систем важливим елементом життєвого цилку є планування та розробка стратегії розгортання в межах певної інфраструктури. В рамка магістерської дисертації було розроблено дві стратегії розгортання — низькорівнена модель кластеру на основі оркестратору Kubernetes та високорівнена модель розгортання в рамках інфраструктурного провайдера Azure Cloud. Це відповідно додатки Додаток Д8 та Додаток Д9.

Kubernetes [16] це технологія оркестрації докер контейнерів, яка дозволяє створювати HA (High Availability) застосунки використовуючи внутрішні механізми — внутрішня мережа, використовуючи мережі плагіни Flannel, WeaveNet тощо, спеціальні сервіси для розгортання контейнерів в рамках кластеру, збір інформації та моніторинг тощо. Налаштування Kubernetes кластеру з самого нуля це дуже складна справа, тому було обрано за основу сервіс Azure для розгортання. Також слід зазначити, що розгортання кластеру використовується принцип IaC або Ifrastructure as Code (інфраструктура як код) з використанням інструменту Terraform. В додатку А наведено приклад конфігураційного файлу, який описує інфраструктуру в рамках сервісу Azure. За більше детальними поясненнями слід звернутись до офіційної документації інструменту.

Відповідно до діаграми Додатку Д8 кластер складається з 6 серверів або нод, з яких дві — мастер-ноди. Мастер-ноди це особливі сервери, яку фактично займаються підтримкою кластером, вони розподіляють сервіси на певні віртуальні машини викиористовуючи задані умови, займаються балансування навантаження, реалізовують внутрішню мережу кластеру та DNS сервер для доступу до сервісів за доменним ім’ям.

Термінологія Kubernetes є дещо незвичною і може здатись складною на перший погляд, тому далі наведено список основних термінів:

– nodes — це віртуальна/фізична машина в рамках кластеру Kubernetes;

– pods — це група контейнерів зі спільними розділами, які запускаються як єдине ціле;

– replication Controller — це сервіс, який гарантує, що певна кількість реплів подів буде запущено в будь-який момент часу;

– services — сервіс, абстракція, яка визначає логічне об’єднання подів та політики доступу до них;

– volumes — це директорія, можливо, з даними всередині, яка доступна в контейнерах;

– labels — це пара “ключ-значення”, яка надаэться об’єктам.

– kubectl — інтерфейс командної стрічки для керування кластером.

На рисунку 3.9 зображено конфігурацію двох мастер-нод запропонованого кластеру [17].

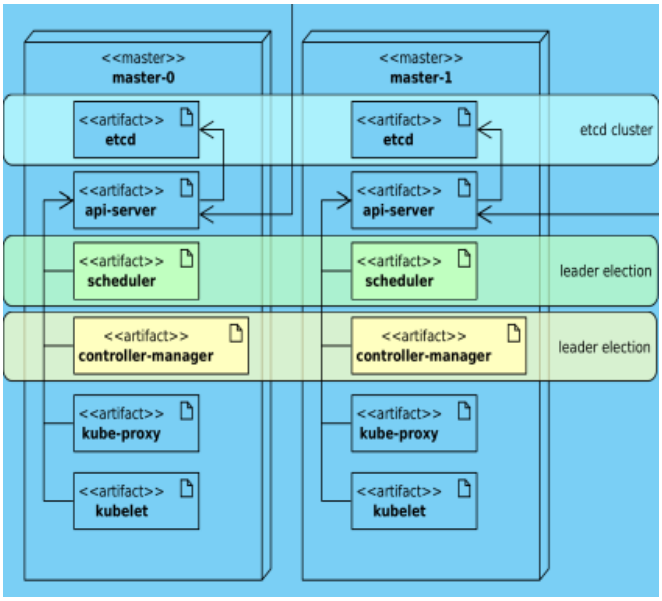


Рисунок 3.9 — Дві master-ноди для кластеру Kubernetes

Кожна мастер-нода складається з наступних елементів:

– etcd — це демон, в якому зберігається стан мастера. Це забезпечує надійне збереження конфігураційних даних;

– api-server — призначений для того, щоб бути простим CRUD серверм із вбудованою бізнес-логікою, яка реалізована в окремих компонентах або плагінах;

– scheduler — прив’язує незапущені поди (сервіси) до нод через спеціальні виклики сервісу api-server;

– controller-manager — серсів, який виконує решту функцій кластера. Наприклад, ноди знаходяться, керуються та контролюються засобами node controller. Ця сутність в може бути розділена на окремі компоненти;

– kube-proxy — це сервіс, який запускається на кожній ноді кластеру. Цей сервіс являє собою простий балансувальник навантаження;

– kubelet — сервіс, який керує pod’ами, контейнерами, образами, розділами тощо.

Після вияснення усіх компонентів та термінології інструменту Kubernete простіше пояснювати конфігурацію кластеру. Дві ноди кластеру зарезервовані під реляційну СУБД та time-series СУБД. Це дозволяю уникнути падіння нод або подів у випадках високого навантаження сервісів API, Jump Server тощо. Крім того, на цих нодах слід розгорнути File Storage сервіс Minio та мікросервіс Statistics Write, який в свою чергу постійно контактує із СУБД, тому логічно розгорнути його в рамках однієї ноди.

Решта сервісів — API, Notification Service, Jump Server, Haproxy, слід розгортати з дублюванням на двох нодах, що і показано на рисунку 3.10. Також слід зазначити, що сервіс Haproxy відіграє в даному випадку роль Web-сервера для відправки статики, а також балансування навантаження на різні частини застосунку.

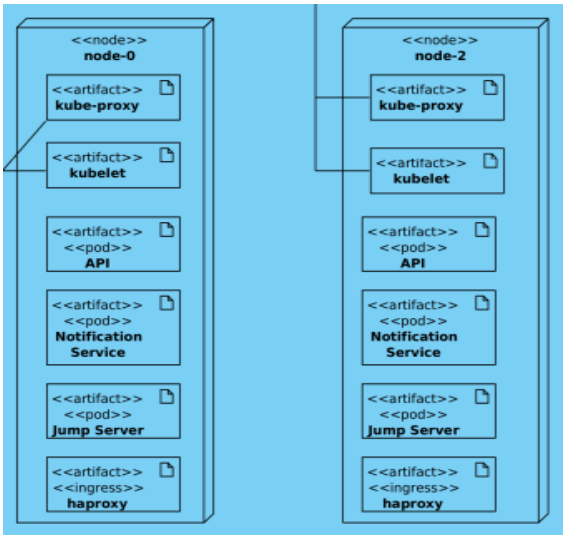


Рисунок 3.10 — Конфігурація нод кластеру з розгортанням API, Jump Server, Notification Service, Haproxy.

На основі отримах результатів простіше будувати стратегію розгортання кластеру в Azure Cloud. Перед розгортанням слід спочатку створити наступні ресурси:

– container registry — це реєстр для зберігання докер контейнерів;

– azure blob storage — це спеціалізований сервіс для зберігання статичних файлів. Цей сервіс необхідний для обладнання інструменту Minio і створення директорії з файлами без створення окремих віртуальних машин;

– AKS — Azure Kubernetes Service, спеціалізований сервіс для створення кластеру. Також, для створення кластеру, як згадувалось вище, застосовується інструмент Terraform і приклад конфігурації для створення AKS наведено в додатку А.

Отже, наведені вище стратегії розгортання застосунку мають право на життя, оскільки це не теоретичне описання, а практична модель, яка вже застосовувалась в інших проектах. Така модель розгортання зробить застосунок достатньо стійким до навантаження із вбудованими механізмами балансування навантаження.

4 ІНСТРУКЦІЯ КОРИСТУВАЧА

4.1 Робота програми

Графічний інтерфейс користувача розроблено з урахуванням сучасних підходів в розробці UI/UIX. При відвідуванні сторінки застосунку користувача зустрічає сторінка авторизації як на рисунку 4.1. Користувачу необхідно заповнити форму та натиснути кнопку Log In.

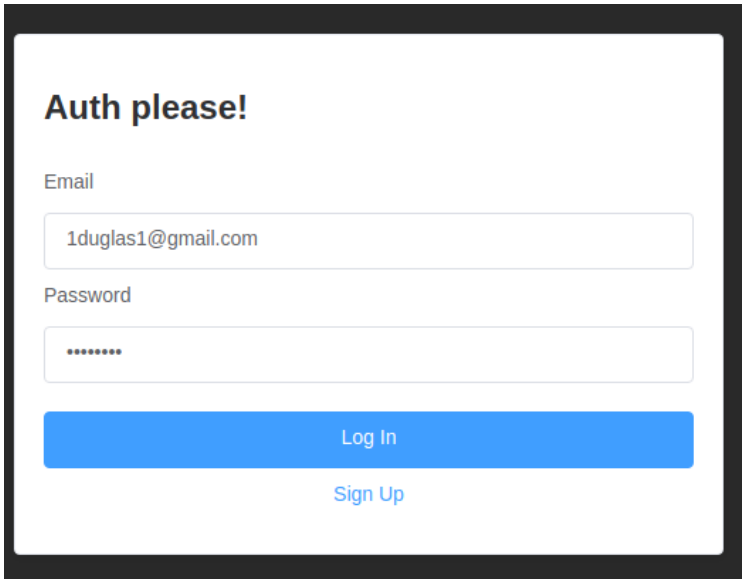


Рисунок 4.1 — Сторінка авторизації

У випадку, коли користувач ввів невірні дані, користувача буде сповіщенно про помилку як на рисунку 4.2. Через деякий час вікно сповіщення зникне. Користувач може в цей же час проводити повторну сесію авторизації.

Також, у користувача є можливість зареєструватись в системі. Форма реєстрації показана на рисунку 4.3. Форма має три поля — електронна пошта, нікнейм та пароль.

У випадку, якщо користувач ввів електронну пошту або нікнейм, який вже зареєстровано в системі, його буде сповіщено про помилку вікном як на рисунку 4.4.

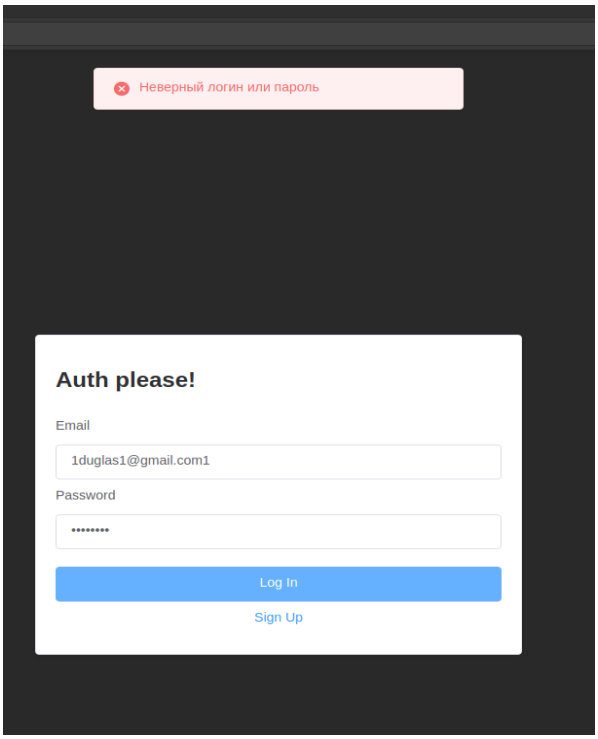


Рисунок 4.2 — Екран, коли введено неправильні дані про користувача

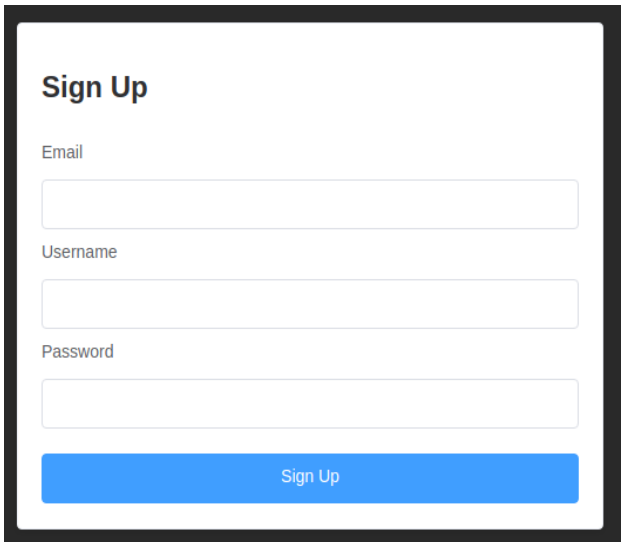


Рисунок 4.3 — Сторінка реєстрації

Рисунок 4.4 — Вікно сповіщення про помилку реєстрації

На даному етапі розвитку проекту, є деякі елементи, в яких може бути відсутня коректна інформація. Це все буде налагоджено в період альфа тестування та UAT (User Acceptance Tests) тестування.

У випадку успішної авторизація, користувача перенаправляє на головну сторінку із графіками, які зображені на рисунку 4.5, 4.6 та 4.7.

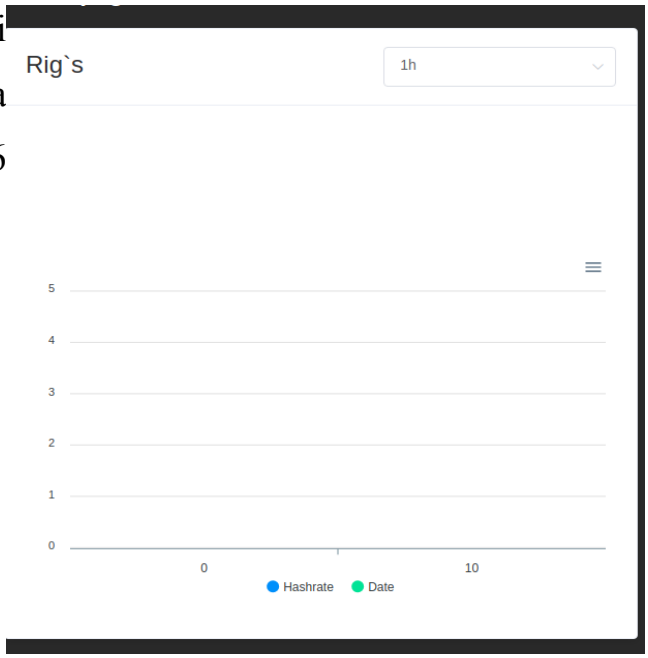


Рисунок 4.5 — Графік усіх ферм

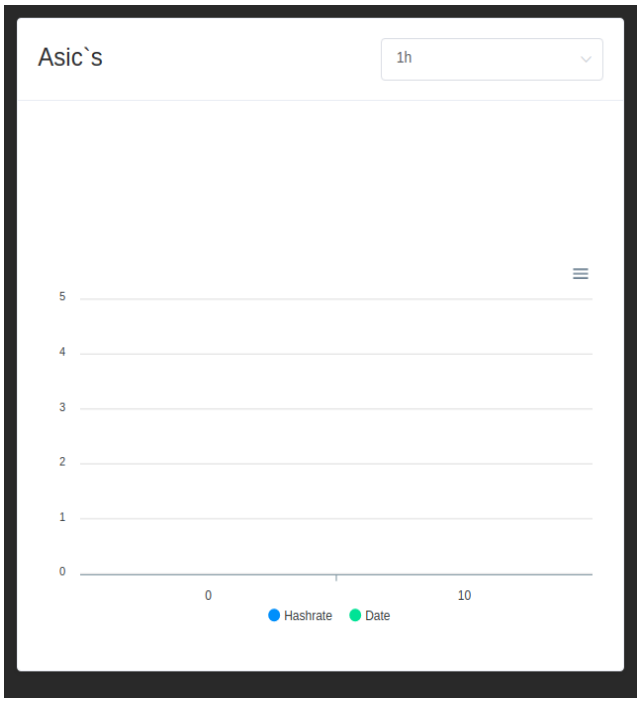


Рисунок 4.6 — Графік ASIC ферм

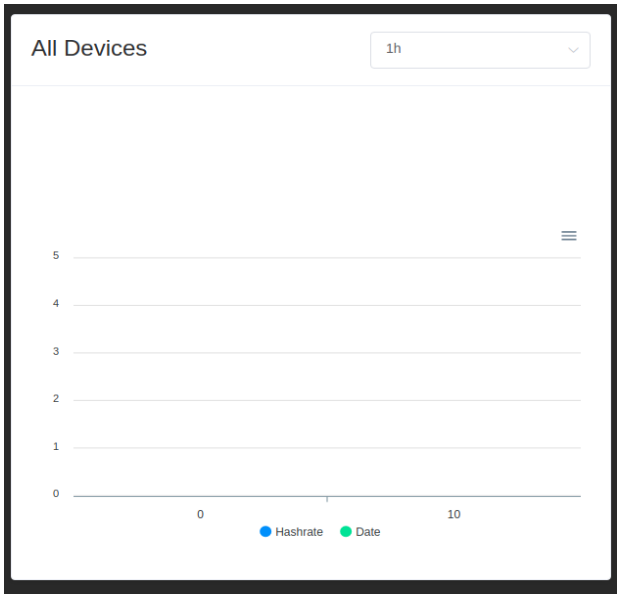


Рисунок 4.7 — Графік усіх майнінг пристроїв

Для навігації між інформаційними елементами передбачене меню з табами як на рисунку 4.8

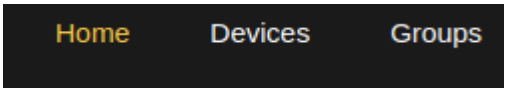


Рисунок 4.8 — Меню з головними табами застосунку

Якщо переміститись на таб “Devices”, користувач перейде на сторінку, яка складається з багатьох елементів. В рамках цього розділу всі елементи розділені на окремі скріни. Наприклад, на рисунку 4.9 зображено список ферм у вигляді скрол-списку з кнопкою додавання нової ферми. Також, кожен елемент списку можна обирати, що в свою чергу буде змінювати динамічно конфігурацію на клієнті.

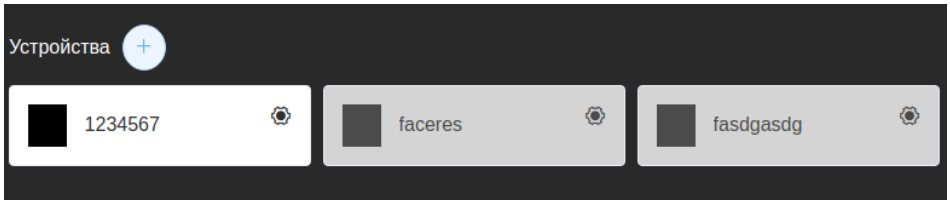


Рисунок 4.9 — Список ферм, які додані в рамках акаунту

Також, на рисунку 4.10 під кожну ферму на сторінці відображається список майнінгових пристроїв з інформацією про неї та контролом, який дозволяє відмикати майнінг на певних GPU/CPU/ASIC.

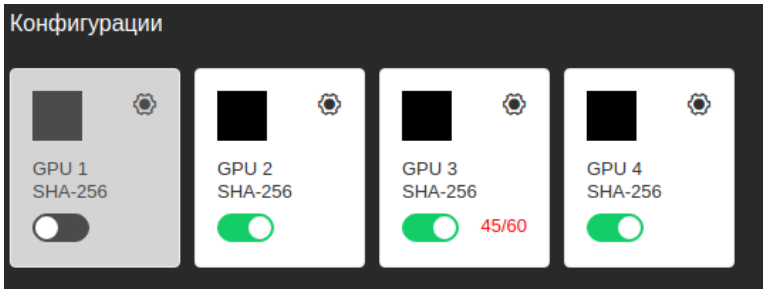


Рисунок 4.10 — Список пристроїв для майнінгу та їх конфігурація

На рисунку 4.11 зображено останню частину поточного екрану — це превью підключення до віддаленої ферми та інформативна таблиця.

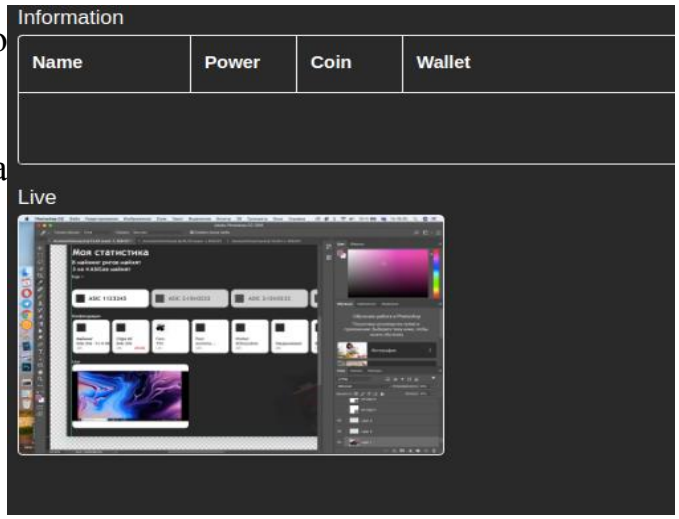


Рисунок 4.11 — Превью VNC підключення та таблиця з інформацією про пристрій для майнінгу

Для додавання нової ферми необхідно натиснути кнопку “+” навпроти “Devices” та обрати тип пристрою. Форма додавання зображена на рисунку 4.12.

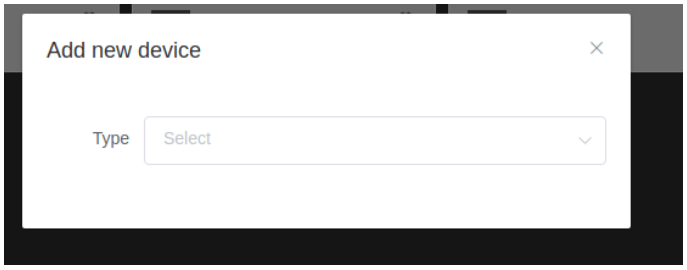


Рисунок 4.12 — Форма додавання нової ферми

Якщо перейти в меню на таб “Groups” перед користувачем також постане комбінований екран зі списком груп, списком ферм в групі та списком майнінгових пристроїв в групі. Все це зображено на рисунку 4.13.

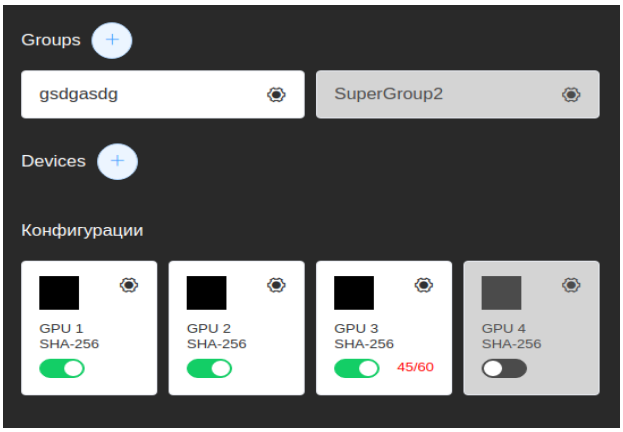


Рисунок 4.13 — Список груп та іншим елементів

Також, передбачене створення нових груп, що можна побачити на рисунку 4.14.

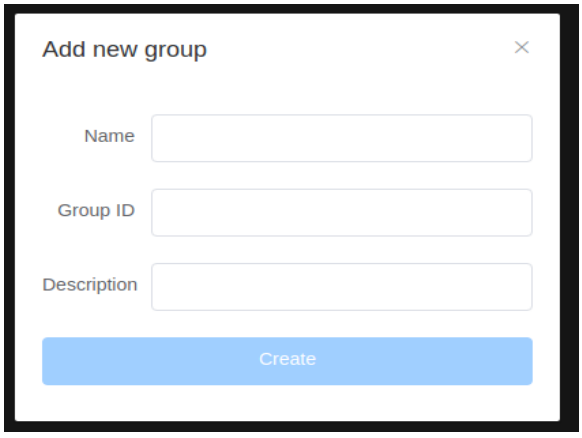


Рисунок 4.14 — Створення нової групи

Отже, розробка графічного інтерфейсу є невід'ємною частиною проекту даної магістерської дисертації. Використовуючи новітні підходи для розробки інтуїтивного UI/UIX було досягнуто непоганих результатів навіть на ранній стадії проекту.

4.2 Тестування системи

Тестування застосунку проводиться з використанням інтеграційного та модульного тестування. Тестами покриті не всі компоненти системи, а лише найбільш важливі – API та агентський застосунок.

Тестування програмного REST інтерфейсу дає можливість відслідковувати зміни JSON схем, які повертають контролери застосунку, логіку повернення даних в рамках шару доступу до даних, перевірку сервісів, що реалізовують бізнес-логіку.

Агентський застосунку тестується з використанням інтеграційних тестів та поділяється на дві частини – локальний тест та віддалений тест. Суть першого формату полягає в тому, щоб провести тестування функціоналу без затримок в мережі. Всі тести в даному випадку запускаються на одному хості. У випадку з віддаленими тестуванням відбувається запуск команд на віддаленому обладнанні – перевіряється статус відповідей, дані всередині відповідей. Таким формат дозволяє реалізувати швидкі тести (локальні) для перевірки щойно написаного функціоналу та довгі тести – перед розгортанням нового функціоналу в продуктовому оточенні.

Тестування проводиться з використанням бібліотеки NUnit.

Також, для тестування передбачені декілька основних тест кейсів, які чітко регламнтують роботу ядра системи:

1) Тест кейс ТС-01

Назва: Реєстрація користувача в системі

Опис: Користувач повинен ввести правильні та унікальні свої дані у форму реєстрації

Тестові дані:

- електронна пошта – test@test.com;
- нікнейм – nickname;
- пароль – password.

Очікуваний результат: користувач проходить реєстрацію та перенаправляється на головну сторінку із статистичними графіками

Кроки:

- перейти на сторінку авторизації;
- натиснути кнопку реєстрації;
- ввести дані у форму реєстрації;
- підтвердити реєстрацію натисненням відповідної кнопки на формі;

2) Тест кейс ТС-02

Назва: Додавання нового обладнання

Опис: Користувач повинен встановити агент на фермі та прив’язати обладнання до свого акаунту

Тестові дані:

- IP-адреса ферми – 10.75.6.11 (приватна мережа);
- тип ферми: GPU
- пароль доступу

Очікуваний результат: користувач встановив ферму, скопіював дані для входу, прив’язав ферму до акаунту

Кроки:

- встановити агент на фермі;
- в терміналі скопіювати ідентифікатор та пароль;
- перейти в клієнтський застосунок;
- перейти на сторінку пристроїв на натиснути кнопку додавання ферми;
- ввести у форму скопійовані ідентифікатор та пароль;

3) Тест кейс ТС-03

Назва: Виконання команди для отримання списку відеокарт на віддаленом хості

Тестові дані:

- майнер: claymore dual ethereum miner;
- виробник відеокарт на фермі: Nvidia;
- адреса гаманця Ethereum: використати будь-яку тестову адресу в мережі Rinkeby;
- решта опцій може бути обрана на власний розсуд;

Очікуваний результат: програма майнінгу повинна запуститись на фермі та почати майнити на тестовий гаманець користувача.

Кроки:

- обрати ферму на клієнті;
- натиснути кнопку для запуску майнера;
- у формі обрати вказаний майнер та внести довільні опції, що задовольняють тестовим даним;
- підтвердити вибір та запустити майнер;

5 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

5.1 Вступ

Ідеї та напрацьована інформація, що викладені в даній магістерській дисертації мають практичну та комерційну цінність для області криптовалют. Цільове призначення програмного комплексу — повна автоматизація процесів майнінгу. На даний момент існує лише одна система, яка має такі можливості, але з відсутністю віддаленого доступу до ферм.

5.2 Маркетинговий аналіз стартап-проекту

5.2.1 Опис ідеї проекту

Опис ідеї проекту наведено в таблиці 5.1

Таблиця 5.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
	1. Криптовалютна сфера	Віддалене керування в реальному часі без потреби витрачання ресурсів(часові, матеріальні, людські) на обслуговування кожної mining-ферми окремо
	2. Керування віддаленими системи	Повний збір метрик та інформації про стан ферм в режимі онлайн без

стороннього програмного забезпечення достатньо встановити застосунок		встановлення стороннього програмного забезпечення,

Продовження таблиці 5.1

Зміст ідеї	Напрямки застосування	Вигоди для користувача
	3. Моніторинг віддалених систем	Розрахунок доходності графічних процесорів та ASIC пристроїв на кожній ферми, а також ферми в цілому в режимі онлайн
		Можливість керування mining-фермами з допомогою команд ботів (Telegram)
		Відправка сповіщень про інциденти (збій в роботі ферми чи пристроїв), у форматі звітів про дохідність, кількість збоїв та час простою на певний проміжок часу у форматі email листів, повідомлень Telegram.
		Застосунок можна

		використовувати як в UNIX подібних ОС так і в Windows ОС, також присутня Web-версія застосунку.

5.2.2 Аналіз потенційних техніко-економічних переваг

Аналіз сильних, слабких та нейтральних сторін стартап-проекту визначено в таблиці 5.2. Перевагами системи є наявність віддаленого доступу до ферм, підтримка великою кількості майнінг-програм, зручний інтерфейс користувача, багатоплатформність, можливість моніторингу та наявність сповіщень.

Таблиця 5.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона	S (сил сторон
		Маг. проект	GUI Miner	Awesome Miner	MinerGate			
1	Зручність графічного інтерфейсу	+	-	+	+			
2	Автономний майнінг	+	+	+	-		+	
3	Кількість підтримуваних криптовалют	5	1	3	11			
4	Моніторинг	+	-	+	-			

	стану обладнання							
5	Тип майнінгу	C/GPU, ASIC	GPU, CPU	C/GPU, ASIC	GPU, CPU		+	
6	Автоматичний підбір обчислювальних алгоритмів	+	-	+	-		+	

Продовження таблиці 5.2

		(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона	S (сил сторон
		Маг. проект	GUI Miner	Awesome Miner	MinerGate	№	Техніко-економічні характеристики ідеї	(пот товари конк
6	Автоматичний підбір обчислювальних алгоритмів	+	-	+	-		+	
7	Регулювання потужностей обладнання	+	-	+	-		+	
8	Захищеність	+	-	+	+			

9	Відсутність системи підписки (система після певного періоду здійснює майнінг на гаманці проекту)	+	-	-	-			
10	Віддалений доступ	+	-	+	-			

5.3 Технологічний аудит ідеї проекту

Список технологій реалізації програмного комплексу наведено в таблиці 5.3.

Таблиця 5.3 – Технологічна здійсненність ідеї проекту

№	Ідея проекту	Технології її реалізації	Наявність технології	Доступність технологій
1	Реалізація програмного забезпечення на базі фреймворку .NET Core 2.1	Фреймворк .NET Core 2.1 – ASP.NET Core, Nhibernate, Autofac	Технологія активно розвивається в області open source	Знаходиться у відкритому доступі для користувачів
2	Створення багатоплатформності графічного інтерфесу з використанням	Фреймворк VueJS у зв’язці з ElectronJS для десктоп застосунків та WEB.	Технологія доступна в open source	Доступна для розробників

	технології Javascript			
3	Збереження статистичних даних	Використання бази даних InfluxDB	Технологія знаходиться у відкритому доступі	Доступна для розробників та адміністраторів баз даних
4	Реалізація асинхронної взаємодії для уникнення затримок	Використання брокера повідомлень RabbitMQ на базі протоколу AMQP	Інструмент є open source рішенням	Доступна для розробників
5	Реалізація механізму сповіщень	Використання програмних інтерфейсів популярних сервісів	В залежності від сервісу може бути або відкритим, або	Доступна для розробників

Продовження таблиці 5.3

№	Ідея проекту	Технології її реалізації	Наявність технології	Доступність технологій
		для відправки електронних листів, Telegram ботів тощо	наполовину платним	

Основна мета у вибору технологій — багатоплатформність, що і було досягнуто. Система потребує специфічних технологій, накшталт реляційної СУБД, NoSQL СУБД, AMQP брокерів тощо.

5.4 Аналіз ринкових можливостей запуску стартап-проекту

На даний момент, через відсутність великої кількості рішень, або явного конкурента, дане рішення готове до використання вже в мінімально робочій версії продукту. Даний програмний комплекс може повністю взяти нішу для майнінгу криптовалюти. Це можна побачити з даних наведених в таблиці 5.4.

5.4.1 Аналіз попиту

В таблиця 5.4 наведено дані з попереднього огляду ринку криптовалют. Умовною одиницею є проміжок часу , на який перемикаються майнери користувача для оплати користування застосунком. Це значення може варіаюватись в залежності від навантаженості ферм і складає загалом від 5 хвилин до 20 хвилин.

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	3000 грн/ум.од
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер	Обмежень немає

Продовження таблиці 5.4

№	Показники стану ринку (найменування)	Характеристика
	обмежень)	
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по	80 %

	ринку), %	

5.4.2 Визначення потенційних груп клієнтів

Характеристики потенційних клієнтів стартап-проекту наведені в таблиці 5.5. Це можуть бути лише криптовалютні ентузіасти, або майнінг пули, які зацікавлені в повній автоматизації власниж потужностей.

Таблиця 5.5 – Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Автоматизувати процес майнінгу	Власники майнінгових ферм	Ріст та спад курсу криптовалют, законодавство різних країн	Стабільна робота, відмовостійкість, робота в режимі реального часу і наявність віддаленого доступу

5.4.3 Аналіз ринкового середовища

Сфера криптовалют є відносно новою і зародилась на початку 2010-х років, тому у більшості країн не розроблена відповідна законодавча база для регуляції даної області, тому необхідно враховувати низку інших загрозливих факторів. Фактори ризику та загроз наведені в таблиці 5.6.

Таблиця 5.6 – Фактори загроз

№	Фактори	Зміст загрози	Можлива реакція компанії

1	Законодавчий	Зміна законодавства країни у сфері криптовалют	Вихід на ринок інших країн з більш лояльними законами
2	Хакерство	Несанкціонований доступ до статистичних даних користувачів, а також доступ до ферм	Захищення каналів для передачі повідомлень в мережі
3	Ріст цін за електроенергію	Високі ціни на електроенергію можуть знизити прибутковість процесу майнінгу	Зменшити відсоток для майнінгу на гаманці проекту
4	Вихід на ринок нових конкурентів	З урахування помилок даного проекту можуть вийти рішення, які реалізують більшу кількість функціоналу та мають більше можливостей	Постійни розвиток проект і введення додаткового функціоналу

Навіть при наявності серйозних ризиків з боку інших держав, ринок криптовалют є популярним і його популярність все ще росте, та при цьому є багато ніш, які можна зайняти без жорсткої конкуренції. Низка таких можливостей наведена в таблиці 5.7.

Таблиця 5.7 – Фактори можливостей

№	Фактори	Зміст можливості	Можлива реакція компанії
1	Законодавчий	Відсутність законодавства у країни у сфері криптовалют	Можна самому обирати напрямок розвитку, оскільки законодавство нічого не забороняє

2	Напрямок, який динамічно	Багато ентузіастів, які прагнуть зайнятись	Запропонувати нові послуги та функції

Продовження таблиці 5.7

№	Фактори	Зміст можливості	Можлива реакція компанії
	розвивається	майнінгом криптовалюти	
3	Низькі ціни за електроенергію	Низькі ціни за електроенергію у деяких країнах є привабливим місцем для встановлення майнінгових ферм	Вийти на ринок даної країни
4	Невелика кількість конкурентів	Відсутність монопольного ринку дає можливість та час для розвитку успішного продукту	Моніторити потреби клієнтів і розвивати продукт

5.4.4 Аналіз пропозиції

Також, було розглянуто конкуренцію на ринку і результати наведені в таблиці 5.8.

Таблиця 5.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкуренто-спроможною)
Тип конкуренції - чиста	Немає чітко виокремленого	Сприятливо впливає на

	лідера серед конкурентів, та відповідно мала їх кількість	розвиток проекту
Рівень конкуретної боротьби - міжнародний	Сфера криптовалют зачіпає практично всі країни світу	Можливий швидкий вихід на світові ринки

Продовження таблиці 5.8

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкуренто-спроможною)
За галузевою ознакою - внутрішньогалузева	Даний проект актуальний лише у сфері криптовалюти	Слідкувати за трендами сфери криптовалют та відповідно розвивати потрібний функціонал
За видами товарів - товарно-родові	Задоволення потреб клієнтів за рахунок наявність максимальної автоматизації процесу майнінгу	Тримати рівень автоматизації не нижче поточного
За видами товарів - товарно-родові	Задоволення потреб клієнтів за рахунок наявність максимальної автоматизації процесу майнінгу	Тримати рівень автоматизації не нижче поточного

5.4.5 Аналіз умов конкуренції в галузі

Аналіз конкуренції за М. Портером дало більш детально інформацію про стан ринку і можливості конкурентів.

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Awesome Miner, MinerGate, GUIMiner	Цінність ідеї та її якісна реалізація	Немає	Споживачі прямо впливають на успішність	Конкуренти можуть стати дешевшими або стануть

Продовження таблиці 5.9

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
				продукту, оскільки здійснюється відсоткова оплата	надавати якісніші послуги
Висновки:	Конкурентів небагато, відсутність монополії, проте конкурентна боротьба значна за рахунок хорошої якості	Є можливість виходу на ринок, ідентичних продуктів немає і не передбачається	Немає постачальників	Так, реалізація потреб клієнтів вирішує успішність проекту	Достатньо важко предентувати на клієнтів, які вже використовують інший готовий продукт

	проектів конкурентів				

5.4.6 Перелік факторів конкурентоспроможності

Навіть при врахуванні конкурентів розроблена система має низку переваг, які наведені в таблиці 5.10.

Таблиця 5.10 — Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Галузь, яка тільки розвивається	Завдяки цьому фактору на ринку існує достатньо мало аналогів, які можуть виконувати поставлені задачі стабільно, тому гарна реалізація і повнота функціоналу дозволить швидко зайняти певну частку ринку.
2	Ідея	Ідея автоматизації на даному етапі розвитку індивідуальна і з часом планується задовольнити повністю всі потреби майнерів
3	Потреба в інвестиціях	Потреба в інвестиціях є, оскільки такого роду системи можуть розроблюватись доволі довгий час, якщо цим займатиметься

		один розробник.
4	Швидкий вихід на ринок	Достатньо випустити мінімальну базову працездатну версію продукту на ринок для його використання
5	Команда	Інтелектуальний рівень команди розробників дає можливість якісно та швидко розроблювати необхідний функціонал
6	Інноваційність	Впровадження нового рівня автоматизації, який забезпечує якісний ріст ефективності процесу майнінгу

5.4.7 Аналіз сильних та слабких сторін стартап-проекту

За наведеними факторами конкурентоспроможності в таблиці 5.10 було виконано порівняльний аналіз сильних та слабких сторін програмного комплексу (стартап-проекту) зі звязкою Awesome Miner, Miner Gate, який відображено в таблиці 5.11.

Таблиця 5.11 — Порівняльний аналіз сильних та слабких сторін системи віддаленого управління процесами та ресурсами майнінгових ферм

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні із системою управління процесами майнінгу						
			-3	-2	-1	0	+1	+2	+3
1	Галузь, яка тільки розвивається	15						+	
2	Ідея	18							+

3	Потреба в інвестиціях	10				+			
4	Швидкий вихід на ринок	20						+	
5	Команда	10					+		
6	Інноваційність	12							+

5.4.8 SWOT-аналіз

Складений SWOT-аналіз на основі вище отриманих даних відображено в таблиці 5.12.

Таблиця 5.12 — SWOT-аналіз стартап-проекту

Сильні сторони: – багатоплатформність; – широкий спектр функціоналу; – мала залежність від інвестицій; – невелика кількість сильних конкурентів; – невелика кількість сильних конкурентів; – зручний графічний інтерфейс користувача; – затребованість на ринку;	Слабкі сторони: – необхідно постійно впроваджувати новий функціонал; – в період спаду курсів криптовалюта потреба в програмному комплексі може зменшуватись; – вузьке коло клієнтів;

Продовження таблиці 5.12

Сильні сторони: – повна автономна робота; – підтримка великої кількості майнерів та криптовалют;	

– всесторонній моніторинг обладнання із сповіщеннями з їхнього приводу стану;	
Можливості: – здешевлення електроенергії; – послаблання законодавства в області криптовалюти в деяких країнах; – тенденції в розвитку криптовалют; – інвестиції; – висока зацікавленість потенційними клієнтами;	Загрози: – хакерство; – консервативність клієнтів на користь інших рішень, які вже пройшли “перевірку часом”; – заборона криптовалют на рівні законодавства; – велика вартість електроенергії;

5.4.9 Альтернативи ринкової поведінки

Визначення альтернатив було зроблено на основі часових термінів та ймовірності отримання ресурсів і відображено в таблиці 5.13

Таблиця 5.13 — Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива (орієнтований комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Введення меншої вартості використання програмного комплексу	Ймовірно	2 місяць

Продовження таблиці 5.13

№	Альтернатива (орієнтований комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації

2	Безкоштований пробний період	Дуже ймовірно	1 місяць

Найкраща альтернатива — безкоштовний пробний період для кожного користувача, який вирішить спробувати дану систему.

5.5 Розроблення ринкової стратегії проекту

Для розробки ринкової стратегії проекту необхідно спершу визначити стратегії охоплення ринку, а саме описати характеристи потенційних клієнтів стартап-проекту. Ця інформація наведена в таблиці 6.14.

5.5.1 Опис цільових груп потенційних клієнтів

В таблиці 5.14 наведено характеристики потенційних клієнтів.

Таблиця 5.14 - Характеристика потенційних клієнтів стартап проекту

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Криптовалютні ентузіасти	Висока	70%	Середня	Просто
2	Майнінг пули	Висока	85 %	Низька	Просто
3	Компанії, які займаються виключно майнінгом.	Висока	80%	Висока	Складно
Які цільові груп обрано: Майнінгу пули, криптовалютні ентузіасти					

За результатами аналізу потенційних клієнтів, було обрано дві цільові групи — майнінг пули та криптовалютні ентузіасти і на їх основі в таблиці 5.15 було обрано стратегію ринкового впливу.

5.5.2 Формування базових стратегій розвитку

Таблиця 5.15 — Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції	Базова стратегія розвитку
Пропонуванн системи майнінг пулам та криптовалютним ентузіастам	Диференційований маркетинг	Здатність пропонувати кращий та широкий спектр функціоналу	тегія диференціації

Базовою стратегією для стартап-проекту є стратегія диференціації, що означає надання програмному комплексу властивостей, які будуть відрізняти його від конкурентів.

5.5.3 Вибір стратегії конкуретної поведінки

В таблиця 5.16 наведено базові стратегії конкуретної поведінки.

Таблиця 5.16 — Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Ні	Залучати нових	Автомайнінг	Стратегія

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
	та забирати існуючих		вилику лідера

Важливим елементом виходу на ринок є широкий спектр функцій, завдяки чому може відпасти потреба у використанні стороннього програмного забезпечення.

5.5.4 Розробка стратегії позиціонування

На основі потреб споживачів з обраних сегментів, а також в залежності від обраної базової стратегії розвитку в таблиці 5.17 наведено визначені стратегії позиціонування.

Таблиця 5.17 — Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
Максимальна автоматизації процесів майнінгу Обрахунок доходності Моніторинг ферм	Стратегія диференціації	Розширення базового функціоналу Пропозиція майнінг-компаніям	Віддалений майнінг Автоматичний майнінг Швидкий майнінг

Віддалений доступ Автоперемикання майнінгу по вигідних курсах Сповіщення в різних каналах			Майнінг в один клік

5.6 Розробка маркетингової програми стартап-проекту

Першим кроком до формування маркетингової концепції товару, який отримає споживач. Для цього у таблиці 5.18 підсумовано результати попереднього аналізу конкурентоспромжності товару.

5.6.1 Формування маркетингової концепції

В таблиці 5.18 визначено ключові переваги концепції потенційного товару

Таблиця 5.18 – Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Запуска майнерів на великій кількості ферм	Гарантований автоматичний процес налаштування ферм та майнерів для видобування криптовалюти	Простота у користуванні (достатньо встановити агенти на фермах та скопіювати необхідну інформацію до клієнтського застосунку)
2	Моніторинг систем в	Відображення актуальних даних на	Швидка вибірка ключових актуальних даних

	режимі реального часу	момент перегляду Відображення статистики в різних форматах для кращого сприйняття	Сповіщення про виникнення певних умов на основі зібраних даних (умови задаються користувачем)
3	Віддалений доступ	Доступ в один клік	Вирішення мережевих проблем пов’язаних з NAT’ами через ще, можна без проблем отримати доступ до будь-якої ферми

На даному етапі необхідно розробити трирівневу маркетингову модель, де уточнюється ідея продукту, його фізичні складові, особливості процесу його надання. Ці дані наведено в таблиці 5.19.

Таблиця 5.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
Товар за задумом	Система віддалено управління процесами та ресурсами майнінгових ферм		
	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Дизайн 2. Високий рівень автоматизації 3. Кількість підтримуваних майнерів 4. Віддалений доступ до ферм		

Рівні товару	Сутність та складові
Товар за задумом	Система віддалено управління процесами та ресурсами майнінгових ферм
	Якість: стабільна робота ПЗ
	Назва: MiningViewer
	До продажу: допомога зі встановленням агентів та налагоджування віддаленого доступу
	Після продажу: реалізація додаткових функцій, підтримка, вирішення помилок
За рахунок чого потенційний товар буде захищено від копіювання: обфускація програмного забезпечення та приватний доступ до кодової бази	

5.6.2Визначення цінових меж

В даному розділі необхідно провести визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар. Визначена інформація вказана в таблиці 5.20.

Таблиця 5.20 – Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
—	0 - 25000 грн	500000 грн	0 - 15000 грн

5.6.3Визначення оптимальної системи збуту

Необхідно визначити оптимальну систему збуту, в межах якого приймається рішення. Розрахунки наведено в таблиці 5.21.

Таблиця 5.21 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Підписка, або перемикання майнерів на майнінг гаманців компанії	Створення проектної документації про програмний комплекс	Канал нульового рівня	Дистрибуція через Інтернет

5.6.4Розробка концепції маркетингових комунікацій

Останньою складового маркетингової програми є розробка концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів. Результати наведено в таблиці 5.22.

Таблиця 5.22 — Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення

	цільові клієнти			
Ведення приватної діяльності в області криптовалюти	Прямі офіційні	Послідовність в реалізації обраної позиції Доступність та об’єктивність інформації про фірму і товар Унікальність послуги Відповідність вимогам тендеру об’єктивність інформації про фірму і товар	Інформування цільової аудиторії про появу нового програмного рішення з акцентуванням на переваги Пояснення цільовій аудиторії відмінності та переваги в оплаті послуг	Раціоналістична стратегія реклами

Висновки

В результаті проведених досліджень, аналізу даних було проведено розробку стартап-проекту. Протягом процесу розробки стартап-проекту знайдено багато відповідей, які допомогли чіткіше визначити мету проекту та його цільову аудиторію, визначитись з методами монетизації та стратегією маркетингу для ринку криптовалюти та майнінгу.

Основними споживачами системи, що розроблена в рамках магістерської дисертації, є як великі компанії так і ентузіасти в області криптовалют.

У наш час попит на таку систему буде тільки зростати, тому позиція щодо створення системи управління процесами та ресурсами майнінгових ферм є дуже доцільною.

Перевагами проекту є:

– просто у використанні;

– кросплатформність;

– зручний інтерфейс користувача;

Унікальність проекту полягає в рівні автоматизації майнінг обладнання, що представлене фермами, збору статистичної інформації, сповіщенні користувача в різних каналах (email, Telegram).

ВИСНОВКИ

Як результат аналізу застосунків для майнінгу, було створено програмний комплекс, що враховує функціональні можливості аналогів та виправляє існуючі недоліки. Були розглянуті мережеві можливості віддаленого доступу, а також вирішено проблему обходу NAT'ів, що суттєво покращило роботу системи VNC, створено гнучку архітектуру програмного застосунку, реалізовано систему з використанням сучасних програмних інструментів, опрацьовано предметну область, де потенційно може бути використана система.

Експериментальним шляхом були обрані протоколи для реалізації віддаленого доступу — VNC, STUN/TURN, обрані сучасні технології розробки — .NET Core, RabbitMQ, Electron тощо, які дозволили в цілому створити стабільний застосунок для мінімально робочої версії. Головним критерієм для вибору протоколів була їхня універсальність та простота, а для інструментів — багатоплатформність та можливості для розробки. З допомогою ітераційної розробки було розроблено декілька робочих концепцій з використанням різних технологій, тому кінцевий вибір інструментів був зроблений на основі експериментальних результатів попередньої ітерацій розробки.

Для роботи системи віддаленого управління процесами та ресурсами майнінгових ферм обрано модель взаємодії “агент-сервер”, як і в багатьох програмних комплексах орієнтованих на моніторинг сервервних систем. Експериментальним шляхом була обрана асинхронна модель взаємодії на основі протоколу AMQP.

Система вимагає збереження широкого спектру даних, тому необхідно обрати декілька варіантів спеціалізованих сховищ. Для збереження даних користувача та інших постійних даних рекомендовано обрати реляційну модель збереження даних у вигляді реалізації PostgreSQL, для збереження великої кількості даних у вигляді часових рядів — модель NoSQL сховищ для Time series даних і було обрано базу даних InfluxDB. Ця база даних широко поширена в системах моніторингу, оскільки має вбудовано SQL-подібну мову для вибірок, швидко оброблює дані та має багато програмних бібліотек для роботи на різних мовах програмування.

Також, в рамках магістерської дисертації розроблено стартап-проект, який може мати практичну та комерційну цінність у випадку реалізації на ринку криптовалюти. Крім того, розробки стартап-проекту

допомогла осягнути проект в цілому та виділити найосновніші задачі та функціонал, необхідний для реалізації мінімально робочої версії.

Магістерська дисертація в повній мірі розкриває необхідність подібного роду програмних комплексів, оскільки сфера майнінгу зараз поширюється швидко і необхідність в обслуговуванні великої кількості обладнання тільки зростає.

ПЕРЕЛІК ПОСИЛАНЬ

[1] [Електронний ресурс] — 2017. Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%B5%D1%80%D0%

B5%D0%B4%D1%8C_%D1%81%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D0%B9

[2] [Електронний ресурс] — 2015. Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol

[3] [Електронний ресурс] — 2018. Режим доступу до ресурсу: https://www.awesomeminer.com/home

[4] [Електронний ресурс] — 2010. Режим доступу до ресурсу: https://en.wikipedia.org/wiki/RFB_protocol

[5] The remote Framebuffer Protocol [Електронний ресурс] / Tristan Richardson, John Levine — 2011. Режим доступу до ресурсу: https://tools.ietf.org/html/rfc6143

[6] [Електронний ресурс] — 2018. Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Network_address_translation

[7] Чи такий страшний Symmetric NAT [Електронний ресурс] // Habrhabr. — 2017. Режим доступу до ресурсу: https://habr.com/post/150298/

[8] [Електронний ресурс] — 2014. Режим доступу до ресурсу: https://en.wikipedia.org/wiki/STUN

[9] [Електронний ресурс] — 2018. Режим доступу до ресурсу: https://habr.com/company/tensor/blog/347534/

[10] [Електронний ресурс] — 2014. Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/%D0%9E%D0%B4%D0%BD%D0%BE%D1%80%D0%B0%D0%BD%D0%B3%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C

[11] Alan T.Norman Cryptocurrency mining: The ultimate guide to understanding Bitcoin, Ethereum, Litecoin, Monero, Zcash mining technologies: підручник . San Francisco, USA. 2017.

[12] Нефункціональні вимоги до програмного забезпечення. Частин 1 [Електронний ресурс] / Наталія Желнова // Habrhabr. — 2014. Режим доступу до ресурсу: https://habr.com/post/231961/

[13] [Електронний ресурс] — 2000. Режим доступу до ресурсу: https://martinfowler.com/eaaCatalog/singleTableInheritance.html

[14] [Електронний ресурс] — 2000. Режим доступу до ресурсу: https://martinfowler.com/eaaCatalog/concreteTableInheritance.html

[15] [Електронний ресурс] — 2000. Режим доступу до ресурсу: https://martinfowler.com/eaaCatalog/classTableInheritance.html

[16] Основи Kubernetes [Електронний ресурс] / Хроніков Александр // Habrhabr. — 2015. Режим доступу до ресурсу: https://habr.com/post/258443/

[17] Запуск повноцінного кластера на Kubernetes з нуля на Ubuntu 16.04 [Електронний ресурс] // Habrhabr. — 2018. Режим доступу до ресурсу: https://habr.com/post/348688/

Додаток А. Приклад конфігурації Terraform

```
resource "azurerm_resource_group" "test" {
```

```
  name     = "acctestRG1"
  location = "East US"
}

resource "azurerm_kubernetes_cluster" "test" {
  name                = "acctestaks1"
  location            = "${azurerm_resource_group.test.location}"
  resource_group_name = "${azurerm_resource_group.test.name}"
  dns_prefix          = "acctestagent1"

  agent_pool_profile {
    name            = "default"
    count           = 6
    vm_size         = "Standard_D1_v2"
    os_type         = "Linux"
    os_disk_size_gb = 30
  }

  service_principal {
    client_id     = "00000000-0000-0000-0000-000000000000"
    client_secret = "0000000000000000000000000000000000"
  }

  tags {
    Environment = "Production"
  }
}

output "client_certificate" {
  value                                                                 =
"${azurerm_kubernetes_cluster.test.kube_config.0.client_certificat
e}"
}

output "kube_config" {
  value = "${azurerm_kubernetes_cluster.test.kube_config_raw}"
}
```