НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

На правах рукопису
УДК 004.056

До захисту допущено
В.о. завідувача кафедри ШІ
Олена ЧУМАЧЕНКО
«___» _____ 2022 р.

Магістерська дисертація

на здобуття ступеня магістра за спеціальністю 122 «Комп'ютерні науки»
на тему: «Агентна архітектура механізмів консенсусу в блокчейні»

Виконав:
студент 2 курсу, групи КІ-11мп
Лисов Богдан Сергійович

Керівник: завідувач кафедри ММСА,
к.т.н., доцент, Тимощук О.Л.

Рецензент: доцент кафедри
системного проектування
КПІ ім. Ігоря Сікорського
к.т.н., ст. н.с Кисельов Г.Д.

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

Київ
2022

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

Рівень вищої освіти — другий (магістерський)
Спеціальність (ОПП) — 122 «Комп'ютерні науки» («Системи і методи штучного інтелекту»)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри ШІ
Олена ЧУМАЧЕНКО
«___» _____ 2022 р.

ЗАВДАННЯ
на магістерську дисертацію студенту Лисову Богдану Сергійовичу

1. Тема дисертації: «Агентна архітектура механізмів консенсусу в блокчейні», науковий керівник дисертації Тимощук Оксана Леонідівна, к.т.н., доцент,, затверджені наказом по університету від «03» листопада 2022 р. № 4046-с

2. Термін подання студентом дисертації: 12.12.2022 р.

3. Об'єкт дослідження: транзакції блокчейну в залежності від механізму консенсусу.

4. Предмет дослідження: застосування методів штучного інтелекту та інтелектуальних агентів для підбору параметрів мережі блокчейн.

5. Перелік завдань, які потрібно розробити:
1) здійснити огляд технічної літератури за темою роботи;
2) дослідити актуальність обраної теми;
3) ознайомитись з доступними інструментами;
4) здійснити порівняльний аналіз наявних механізмів консенсусу, виявити їх переваги та недоліки;
5) розгорнути мережу блокчейн локально;
6) створити агента, який автоматизує роботу розгортання та збору метрик;
7) провести аналіз результатів та метрик;

8) зробити висновки з отриманих даних;

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:

7. Орієнтовний перелік публікацій:

1. Лисов Б.С. Тимощук О.Л. Агентна архітектура механізмів консенсусу в блокчейні 1-а Всеукраїнська Науково-практична конференція "Системний аналіз та інформатика", м. Київ, 22-29 листопада, 2022 року.

8. Дата видачі завдання: 30 серпня 2022 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	здійснити огляд технічної літератури за темою роботи	01.09.22 — 14.09.22	Виконано
2	дослідити актуальність обраної теми	15.09.22 — 22.09.22	Виконано
3	ознайомитись з доступними інструментами	23.09.22 — 30.09.22	Виконано
4	здійснити порівняльний аналіз наявних механізмів консенсусу, виявити їх переваги та недоліки	01.10.22 — 08.10.22	Виконано
5	розгорнути мережу блокчейн локально	09.10.22 — 16.10.22	Виконано Виконано
6	створити агента, який автоматизує роботу розгортання та збору метрик	17.10.22 — 01.11.22	Виконано
7	провести аналіз результатів та метрик	02.11.22 — 09.11.22	Виконано
8	зробити висновки з отриманих даних	10.11.22 — 17.11.22	Виконано

Студент Богдан ЛИСОВ

Науковий керівник дисертації Оксана ТИМОЩУК

РЕФЕРАТ

Магістерська дисертація: 100 с., 25 табл., 16 рис., 15 джерел, 1 додаток.

ІНТЕЛЕКТУАЛЬНІ АГЕНТИ, АГЕНТНА АРХІТЕКТУРА, БЛОКЧЕЙН, МЕХАНІЗМИ КОНСЕНСУСУ, ЦИФРОВІ ТРАНЗАКЦІЇ.

Об’єкт дослідження — транзакції блокчейну в залежності від механізму консенсусу.

Предмет дослідження — застосування методів штучного інтелекту та інтелектуальних агентів для підбору параметрів мережі блокчейн.

Мета роботи — розробка агентної архітектури для аналізу блокчейну Ethereum, роботи його транзакційного механізму в умовах обраного механізму консенсусу та порівняння цих механізмів у вигляді зібраних метрик часу проходження транзакції, кількості витрачених ресурсів та безпечності концепту.

В роботі проведено огляд різних механізмів консенсусу блокчейну в залежності від підібраних параметрів мережі за допомогою інтелектуальних агентів. Розглянуті сучасні методи штучного інтелекту в агентних архітектурах та їх роль у дослідженні світу криптовалют, і, в свою чергу, впливу криптовалют на довколишнє середовище.

Зібрано дані ефективності використання наборів протоколів, стимулів та ідей, які дозволяють мережі вузлів погоджувати стан блокчейну в залежності від налаштування мережі, що дозволяє оцінити доцільність використання того чи іншого механізму консенсусу. Застосовано штучний інтелект для проведення збору та аналізу метрик часу проведення транзакцій, їх стійкості до зламу, кількості використаних обчислювальних ресурсів і відповідно кількості витраченої електроенергії.

ABSTRACT

Master’s thesis: 100 p., 25 tab., 16 fig., 15 references, 1 appendix.

INTELLIGENT AGENTS, AGENT ARCHITECTURE, BLOCKCHAIN, CONSENSUS MECHANISMS, DIGITAL TRANSACTIONS.

The object of the research is blockchain transactions depending on the consensus mechanism.

The subject of the research is the application of artificial intelligence methods and intelligent agents to select the parameters of the blockchain network.

The purpose of the work is to develop an agent architecture for the analysis of the Ethereum blockchain, the operation of its transaction mechanism under the conditions of the selected consensus mechanism, and the comparison of these mechanisms in the form of collected metrics of transaction time, the amount of resources spent, and the safety of the concept.

The paper provides an overview of various blockchain consensus mechanisms depending on selected network parameters with the help of intelligent agents. Modern methods of artificial intelligence in agent architectures and their role in researching the world of cryptocurrencies, and the impact of cryptocurrencies on the environment, are considered.

Data on the effectiveness of using sets of protocols, incentives and ideas that allow a network of nodes to agree on the state of the blockchain depending on the configuration of the network have been collected, which makes it possible to assess the feasibility of using one or another consensus mechanism. Artificial intelligence was used to collect and analyze the metrics of transaction time, their resistance to hacking, the amount of computing resources used and, accordingly, the amount of electricity consumed.

ЗМІСТ

ВСТУП

Блокчейн технологія створила твердий фундамент для децентралізації проведення фінансових операцій без єдиного контролюючого органу за правилами мережі. З появою таких P2P платіжних систем як Bitcoin і Ethereum збільшується кількість людей, які зацікавленні в покупці та продажі цифрових активів. Інфраструктура не стоїть на місці: з'являються різні аналоги з додатковим фунціоналом, але всі вони підпорядковуються суворим стандартам.

Для валідації стану мережі існують механізми консенсусу, під якими мається на увазі досягнення загальної згоди її членами. Що стосується блокчейну Ethereum, процес є формалізованим, і досягнення консенсусу означає, що принаймні 66% вузлів у мережі погоджуються щодо глобального стану мережі.

До 15 вересня 2022 року Ethereum використовував механізм proof-of-work, але зрештою перейшов на proof-of-stake концепт. Операція переходу усунула потребу в енергоємному майнінгу, що натомість дозволило захистити мережу за допомогою накопиченого ETH. Це важливо, оскільки знижується кількість використаного ресурсу для проведення транзакції, що позитивно впливає на довколишнє середовище в порівнянні з минулими роками.

За допомогою штучного інтелекту, а саме інтелектуальних агентів та архітектур, що спираються на це поняття, можна протестувати та проаналізувати якість змін, навантаживши мережу і конфігуруючи її параметри.

Агентом вважаємо все, що може розглядатися як сприймаюче своє середовище за допомогою датчиків і впливає на це середовище за допомогою виконавчих механізмів [1].

Основним напрямком цієї роботи є застосування методів штучного інтелекту для налаштування мережі блокчейну і підбору найефективніших комбінацій параметрів з точки зору швидкості та задіяного ресурсу обчислень. Головною метою виступає збір та аналіз метрик. Такий аналіз надасть можливість зробити висновки, наскільки

безпечною та ефективною є децентралізована система, чи правильно обрано вектор розвитку сучасних блокчейнів. Висновок дозволить з більшою впевненістю інвестувати чи утриматись від інвестування в цю сферу, а також запропонувати краще рішення, якщо воно існує.

Магістерська дисертація містить чотири розділи.

Перший розділ магістерської дисертації присвячений огляду предметної області та детальнішого опису проблематики питання.

Другий розділ присвячено огляду теоретичних відомостей про механізми консенсусу. Детальний розбір наявних методів штучного інтелекту для рішення специфічної задачі.

У третьому розділі викладено практичну роботу із розгортання мережі локально за допомогою агенту та мануально, збір метрик, що є результатом роботи фінального продукту.

Четвертий розділ описує частину стартап-проєкту.

РОЗДІЛ 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ДОСЛІДЖЕННЯ

1.1. Актуальність блокчейну та його складових в сучасному світі

Цифрові активи мають величезне значення, адже навіть Mastercard та Visa — світові лідери, багатонаціональні корпорації з надання фінансових послуг — почали співпрацювати з криптовалютою [2]. Це відбувається через неможливість ігнорувати значущість їх популярності та кількість користувачів екосистеми. Лише на 15 липня 2022 року було зареєстровано 83,434,000 електронних гаманців [3].

Популярність та поширеність таких технологій призводить до масштабного використання ресурсів і, відповідно, впливу на джерела цих ресурсів. Найвідоміший вплив блокчейну на навколишнє середовище пов’язаний із споживанням енергії та, отже, можливим негативним впливом на клімат. Поточний стандартний процес перевірки транзакцій, заснований на алгоритмі підтвердження роботи, є «надзвичайно енергоємним», оскільки вимагає величезної кількості обчислювальних потужностей і, отже, електроенергії для роботи пов’язаних комп'ютерних розрахунки. Більш широке використання технології блокчейн могло б протидіяти зусиллям із пом’якшення кліматичних змін, оскільки електроенергія в усьому світі в основному виробляється з викопного палива. У 2016 році виробництво з горючих видів палива все ще становило 67,3% від загального світового валового виробництва електроенергії.

Приклад Bitcoin є особливо показовим. Порівняно з альтернативними методами оплати, біткоїн був у 20 000 разів більш енергоємним, ніж Visa. У 2019 році, за даними деяких аналітиків, енергія, споживана для кожної транзакції біткоїну, зросла до 635 кВт/год, що дорівнює електроенергії, яка могла б забезпечити енергією приблизно 21 домогосподарство США протягом 1 дня. Згідно з останніми дослідженнями, споживання електроенергії біткоїнами становить від 20 до 80 ТВт-год на рік, або приблизно 0,1-0,3% світового споживання електроенергії. Наприклад, Cambridge Bitcoin Electricity Consumption Index (CBECI) наводить цифру в 64 ТВт-год на рік

(CBECI, 2019), що перевищує річне споживання електроенергії Швейцарією або всіма електромобілями в усьому світі (58 ТВт-год) у 2018 році. Ці оцінки слід тлумачити з обережністю через методологічні проблеми, обмежену доступність даних і дуже різноманітні умови в галузі. Крім того, ці цифри все ще набагато нижчі, ніж інші кінцеві споживачі, такі як охолодження, яке спожило 2 020 ТВт-год електроенергії в 2016 році (Рисунок 1.1).

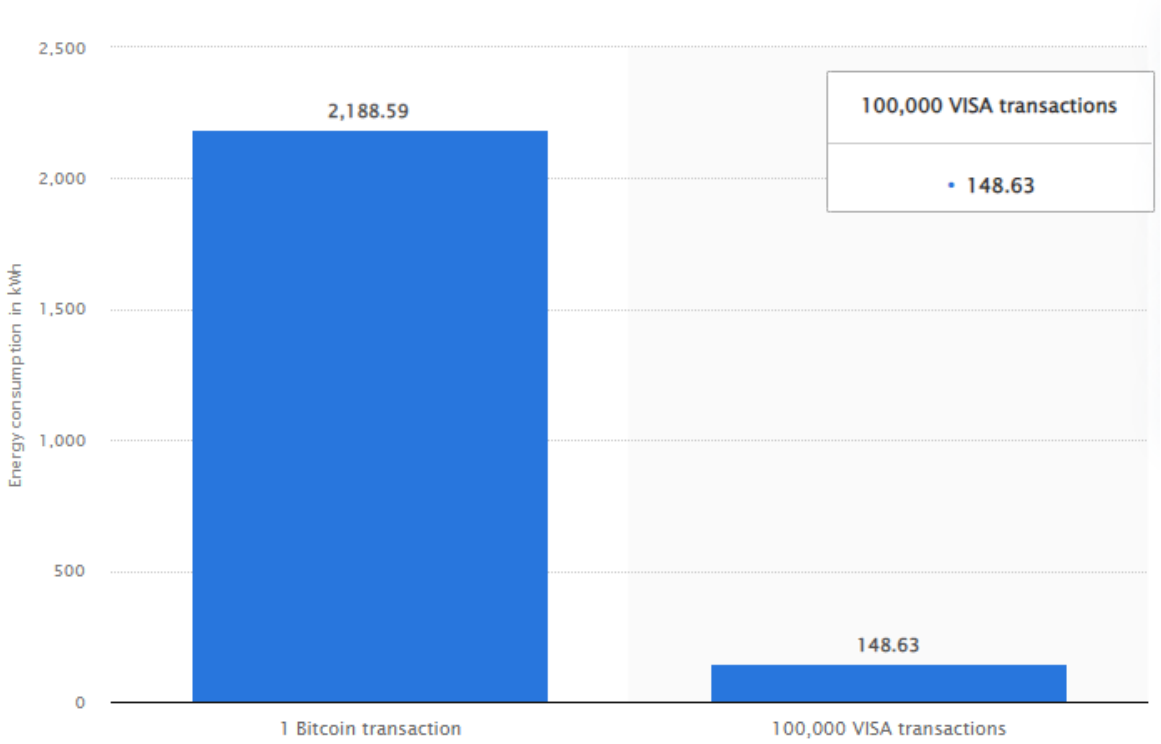

Рисунок 1.1 — Кількість витраченої електроенергії в кіловат-годинах на транзакцію

Хоча відкриття про те, що мережа біткоїн споживає понад 100 доларів електроенергії для транзакції, яку компанія, що займається кредитними картками, могла б провести за копійки, шокує, оцінка в 176 доларів виглядає занадто високою. Проблема, за словами Алекса де Фріса, голландського економіста, чий веб-сайт Digiconomist відстежує вуглецевий слід біткоїнів, полягає в тому, що витрати на електроенергію становлять 9 центів. «Я оцінюю, що в середньому майнери біткоїнів становлять 5 центів», — говорить він. «І це висока цифра. Багато хто майнить в країнах із наднизькими витратами за 3 або 4 центи». Цифра в 9 центів за кВт-год також

призведе до того, що рахунок за електроенергію для карбування кожної монети становитиме 35000 доларів США. З досвіду цього автора, фактичні витрати набагато менші. З кінця жовтня 2020 року до середини січня цього ж року біткоїн торгувався від 11000 до 35000 доларів США [4].

Тим не менш, біткоїн - це лише одна криптовалюта, яка є лише одним із застосувань блокчейну. Важливо знайти спосіб зменшити споживання енергії для перевірки транзакцій. Заміна оригінального механізму консенсусу, тобто алгоритму підтвердження роботи, на інші підходи (такі як алгоритми «підтвердження частки», «підтвердження повноважень» або «підтвердження вичерпаного часу») було запропоновано. Деякі криптовалюти та блокчейн-програми вже покладаються на ці альтернативи. Проте все ще потрібна ретельна оцінка кожного механізму та його енергетичного впливу та енергоефективності. Перехід на більш екологічні джерела енергії та розробка обчислень, які потребують менше енергії, є іншими варіантами, які слід. Також можна було б зменшити енергоспоживання біткоїна шляхом реструктуризації способу стимулювання обслуговування блокчейну.

Як і інші нові технології, блокчейн також викликає занепокоєння щодо електронних відходів (електронних відходів). Майнерам-конкурентам потрібне все більш ефективне обладнання для майнінгу, що призводить до швидкого старіння приблизно кожні 1,5 року. З моменту свого створення апаратне забезпечення для майнінгу біткоїнів вже перейшло від використання центральних процесорів до графічних процесорів, програмованих вентильних матриць та інтегральних схем для конкретних програм. Приблизні оцінки показують, що біткоїн створює 135 г електронних відходів за транзакцію, що в 30000 разів більше, ніж транзакція Visa [5][6].

15 вересня 2022 року сталася подія «The Merge», що означало фактично зміну механізму консенсусу в блокчейні Ethereum. Злиття було об’єднанням початкового рівня виконання Ethereum (мейнмережі, яка існувала з моменту створення) з його новим консенсусним рівнем підтвердження частки, Beacon Chain. Це усунуло потребу

в енергоємному майнінгу, а натомість дозволило захистити мережу за допомогою розставленого ETH. Це був справді захоплюючий крок у реалізації бачення Ethereum — більше масштабованості, безпеки та стійкості. Тому актуальність дослідження процесу переходу з одного механізму на інший є високою.

Протестувати систему глобально та порівняти результати досліджень не є можливим в масштабах одного дослідника, тому було прийнято рішення працювати з системою локально на машині, імітуючи реальне навантаження. Змінити значення параметрів мережі для знаходження найефективніших комбінацій може допомогти агентна архітектура.

З використанням показників продуктивності, які втілюють в собі критерії оцінки успішної поведінки агентів, можна визначити яким чином використовується обчислювальний ресурс, що в свою чергу призводить до споживання електроенергії

1.2. Постановка задачі дослідження

Основною задачею є створення агентної архітектури для дослідження впливу на ефективні показники мережі блокчейну механізмів консенсусу. Описати агентів, які будуть взаємодіяти з системою за певними правилами для збору інформації про витрачені зусилля на проведення транзакції.

Насправді раціональним агентом може бути будь-хто, хто приймає рішення, як особа, фірма, машина чи програмне забезпечення. Він виконує дію з найкращим результатом після розгляду минулих і поточних перцептів (перцептивних вводів агента в даному випадку). Система ШІ складається з агента та його середовища. Агенти діють у своєму середовищі. Середовище може містити інші агенти. Таким чином робота не обмежена конкретним методом чи алгоритмом для досягнення поставленої цілі.

1.3. Особливості предметної області

Кількома словами, блокчейн — це цифровий список записів даних, який постійно зростає. Такий список складається з багатьох блоків даних, організованих у хронологічному порядку, пов’язаних та захищених криптографічними доказами.

Перший прототип блокчейну датується початком 1990-х років, коли вчений Стюарт Хабер і фізик В. Скотт Сторнетта застосували криптографічні методи в ланцюжку блоків як спосіб захисту цифрових документів від підробки даних. Робота Хабера та Сторнетти, безсумнівно, надихнула на роботу Дейва Байєра, Хела Фінні та багатьох інших програмістів і ентузіастів криптографії, що зрештою призвело до створення біткоїна як першої децентралізованої електронної грошової системи (або просто першої криптовалюти). Білий документ про цю монету було опубліковано у 2008 році під псевдонімом Сатоші Накамото [7].

Незважаючи на те, що технологія блокчейну старша за біткоїн, вона є базовим компонентом більшості криптовалютних мереж, діючи як децентралізована, розподілена та загальнодоступна цифрова книга, яка відповідає за збереження постійного запису (ланцюжка блоків) усіх раніше підтверджених транзакцій.

Транзакції блокчейну відбуваються в одноранговій мережі глобально розподілених комп’ютерів (вузлів). Кожен вузол зберігає копію блокчейну та сприяє функціонуванню та безпеці мережі. Саме це робить біткоїн децентралізованою цифровою валютою без меж, стійкою до цензури та не потребує стороннього посередництва.

Як технологія розподіленої книги (DLT), блокчейн навмисно розроблений таким чином, щоб бути високостійким до модифікацій і шахрайства (таких як подвійне витрачання). Це правда, тому що блокчейн, як базf даних записів, не може бути змінений або підроблений без непрактичної кількості електроенергії та обчислювальної потужності — це означає, що мережа може нав’язати концепцію

«оригінальних» цифрових документів, створюючи кожну монету — дуже унікальна форма цифрової валюти, яку не можна копіювати.

Так званий консенсусний алгоритм Proof of Work дозволив створити біткоїн як візантійську відмовостійку систему, тобто його блокчейн здатний безперервно працювати як розподілена мережа, навіть якщо деякі учасники (вузли) представляють нечесну поведінку або неефективну функціональність. Алгоритм консенсусу Proof of Work є важливим елементом процесу видобутку біткоїнів.

Технологія блокчейну також може бути адаптована та реалізована в інших видах діяльності, таких як охорона здоров'я, страхування, ланцюг постачання, IOT тощо. Хоча він був розроблений для роботи як розподіленої книги (у децентралізованих системах), його також можна розгорнути в централізованих системах як спосіб забезпечення цілісності даних або зменшення операційних витрат. Тепер можемо перейти до визначення механізмів консенсусу, які диктують правила самоорганізації мережі — це протоколи, алгоритми чи інші комп'ютерні системи, які дозволяють криптовалютам працювати. Це системи угод, які визначають дійсність транзакцій і управління блокчейном. Існують різні типи механізмів консенсусу з різними перевагами та недоліками. Вони забезпечують довіру та безпеку в мережі блокчейн. Робота здебільшого розглядає два основні принципи: proof-of-work і proof-of-stake, оскільки саме вони є прикладними в Ethereum. Але будуть наведені також інші існуючі алгоритми для наочності, розглянуті їх плюси та мінуси у використанні.

1.4. Висновки до розділу 1

Тема являє собою матеріал актуального дослідження, адже описуються причини і результати події, яка трапилась на початку осені 2022 року. Важливо оцінити масштаби шкоди, яку наносять такі технологічні прориви, і попіклуватись, щоб прогрес завдавав якомога менших збитків довколишньому середовище, за умови неможливості звести шкоду до нуля.

Що стосується штучного інтелекту, то він стає зручним і ефективним помічником в дослідженні подібних процесів та їх технологічного впливу. Він допомагає мінімізувати людський фактор у винесені критичних рішень, адже правильно налаштована система надає змогу робити висновки вже з готових зібраних даних набагато швидше, аніж якби це було організовано лише за рахунок людської роботи.

РОЗДІЛ 2 ТЕОРЕТИЧНІ ВІДОМОСТІ

2.1. Агенти

Агентом є все, що може розглядатися як той, хто сприймає своє середовище за допомогою датчиків і впливає на це середовище за допомогою виконавчих механізмів.

Можна вважати базовими класами архітектур агентних систем три основні:

- деліберативні архітектури (deliberative architectures);
- реактивні архітектури (reactive architectures);
- гібридні архітектури (hybrid architecture).

Деліберативну архітектуру прийнято визначати як архітектуру агентів, що містять точну символічну модель світу та приймають рішення на основі логічного висновку. Пряме запозичення англомовного терміна «деліберативний» дає не надто милозвучну для української мови назву, але пропоновану Т.А. Гавриловою та В.Ф. Хорошевським назва для подібних агентів — «агенти, що базуються на знаннях» — теж не можна визнати дуже вдалим, оскільки агенти інших класів використовують певні знання, але виражені в іншій формі. Тому з метою компактнішого окреслення понять будемо користуватися терміном «деліберативна архітектура» [8].

Побудова деліберативних архітектур вимагає вирішення таких проблем, як побудова адекватного символічного опису реального світу (модель світу), що враховує складність процесів, що відбуваються в часі, і діючих об'єктів; організація логічного висновку з наявних знань, що має призводити до певних дій агентів. Прикладами систем, побудованих у цій архітектурі, є IPEM (Integrated Planning Execution and Monitoring), IRMA (Intelligent Resource-bounded Machine Architecture), AUTODRIVE, Homer.

Достоїнством деліберативних архітектур є можливість застосування строгих формальних методів та добре відпрацьованих технологій традиційного штучного

інтелекту, що дозволяють відносно легко представляти знання у символьній формі та переносити їх до агентної системи. У той самий час створення повної і точної моделі деякої предметної області реального світу, формалізація ментальних властивостей агентів і процесів міркування цих когнітивних структур становлять істотні труднощі для технічної реалізації.

Пошуки шляхів вирішення проблем, що виникають під час використання в агентних системах класичних методів ШІ, призвели до появи нового класу — реактивних архітектур. Основоположником цього напряму прийнято вважати Р. Брукса (R. Brooks), який так сформулював ключові ідеї біхевіористичного погляду на інтелект:

- інтелектуальна поведінка може створюватися без явного символьного уявлення знань;
- інтелектуальна поведінка може створюватися без явного абстрактного логічного висновку;
- інтелект є властивістю деяких складних систем, що раптово виникає.

У реальному світі інтелект не є експертною системою чи машиною логічного висновку, а інтелектуальна поведінка виникає як результат взаємодії агента із середовищем. Відомими прикладами таких архітектур є Pengi, архітектура ситуаційних автоматів, ABLE (Agent Behaviour Language).

Реактивний підхід дозволяє ефективно використовувати безліч досить простих сценаріїв поведінки агентів у рамках встановлених реакцій на певні події навколишнього середовища, але його обмеженість проявляється у практичній неможливості повного ситуативного аналізу всіх можливих активностей агентів. Тому в більшості проектів та діючих систем використовуються гібридні архітектури, спектр варіантів побудови яких досить широкий. Особливості побудови таких архітектур розглянемо з прикладу системи InteRRaP, що поєднує у собі властивості BDI-архітектур (Belief, Desire, Intentions) і багатошарових (layered) архітектур. В основі

BDI-архітектур лежить досить строга теоретична концепція, але вони недостатньо пристосовані для реального проектування ресурсно-обмежених та цілеспрямованих агентних додатків. Багатошарові архітектури добре підтримують моделювання різних рівнів абстракції, відповідальності та складності уявлення знань, але надто складні для формального призначення властивостей агентів та багато в чому залежать від інтуїції розробника.

Архітектура InteRRaP використовує ментальні категорії, визначені в BDI-теорії для опису знань агентів, цілей та станів (Рисунок 2.1).

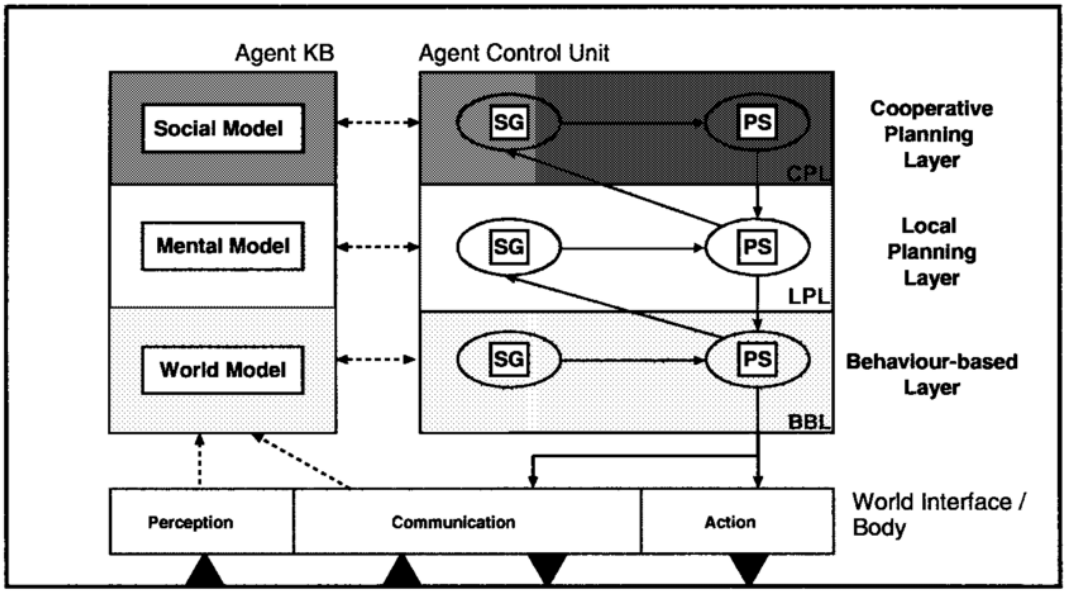

Рисунок 2.1 — Графічне уявлення архітектури InteRRaP

У цій архітектурі реалізовано три базові функції:

BR(P,B) = B` — функція ревізії переконань та абстрактних знань, що переводить поточний стан P агенту та його старе переконання B в нове переконання B`;

SG(B,G) = G` — функція розпізнавання ситуації та активації мети, що виводить нову мету G` з переконань агента B і його поточної мети G;

S(B,G,I) = I` — функція планування та складання розкладу, що виводить I` нових намірів, заснованих на переконаннях B, цілях G, обраних функцією SG, та поточної внутрішньої структури намірів даного агента [9].

Базові функції в залежності від рівня ієрархії набувають своєї спрямованості, що відображає табл. 2.1.

Функція	Шар визначення поведінки	Шар локального планування	Шар кооперативного планування
BR	Генерація та розвиток переконань (модель середовища)	Абстракція локальних переконань (ментальна модель)	Встановлення моделей інших агентів (соціальна модель)
SG	Активація зразків поведінки	Розпізнавання ситуацій, потрібних локальним плануванням	Розпізнавання ситуацій, потрібних кооперативним плануванням
PS	Прямий зв'язок від ситуації до послідовності дій	Модифікація локальних намірів; локальне планування	Модифікація приєднаних намірів; кооперативне планування

Таблиця 2.1 — Базові функції в залежності від рівня ієрархії, їх спрямованість

Кожен рівень ієрархії містить два процеси, що реалізують функції SG і PS. Процес SG_i розпізнає ситуації, які цікавлять відповідний рівень, і виконує активацію мети. Процес PS_i реалізує перехід від цілей до намірів і дій, приймаючи на вході пари

«мета — ситуація», створені процесом SG_i, визначає плани досягнення мети і відстежує виконання кроків плану.

Якщо процес PS_i не компетентний для ситуації S, він посилає вимогу активації, що містить відповідну пару «ситуація — мета» SG_{i+1}, де опис ситуації обробляється за рахунок додаткових знань, доступних цьому процесу, для того щоб зробити відповідний опис мети. В результаті обробки ситуація S повертається назад до PS_i. Так працює механізм керування, заснований на компетентності.

Ситуація ж визначається як безліч формул

$$S \equiv S_B \vee S_L \vee S_C \qquad (2.1)$$

Де $S_B \subseteq WM$, $S_L \subseteq MM$, $S_C \subseteq SM$

Індекси позначають: B — шар визначення поведінки, L — шар локального планування, C — шар кооперативного планування. WM — модель середовища, MM — ментальна модель, SM — соціальна модель. Таким чином, ситуація описується в контексті станів зовнішнього середовища, цілей та намірів агента та його переконань про інших агентів.

Внутрішня організація архітектури InteRRaP представляється дуже логічною і переконливою, але така складна ієрархія функцій, процесів і моделей, природно, не піддається строгій формалізації, і працездатність системи багато в чому забезпечується інтуїцією та досвідом розробників.

Сфера застосування даної архітектури — завдання управління інтерактивними роботами, що виконують транспортні завдання в завантажувальному цеху. Роботи-агенти приймають завдання завантаження та розвантаження вантажівок і можуть вступати в конфлікти з іншими роботами, оскільки можуть блокувати один одного на під'їзних шляхах, можуть прагнути потрапити на площу, зайняту іншим роботом.

Неважко бачити, що розв'язуване транспортне завдання є дуже типовим для такого роду автоматизованих і автоматичних систем і передбачає дію однотипних

агентів-роботів у загальному середовищі, стан якого, хоча б у принципі, може бути точно визначено. Типи ситуацій також становлять кінцеву і досить обмежену множину [10].

У корпоративних системах управління (масштаб підприємства, організації) інтелектуальні агенти, які представляють ті чи інші об'єкти та суб'єкти системи управління, будуть свідомо нерівноправними, вони мають бути включені до жорсткої ієрархічної структури з певними правами та обов'язками. При цьому виникають проблеми розподілу глобальних завдань на підмножини агентів-виконавців.

Розглянутий приклад добре демонструє переваги та недоліки гібридних архітектур, які дозволяють гнучко поєднувати можливості різних моделей, але в більшості випадків сильно залежать від специфіки додатків, для яких вони розробляються. У таких архітектурах існує можливість появи неврахованих незалежних активностей агентів, що є неприйнятним для відповідальних корпоративних систем, оскільки непередбачувана поведінка може призвести до важких наслідків та дискредитувати МАС перед персоналом організації.

Становлення парадигми мультиагентного проектування інтелектуальних систем, що відбувається, призвело до появи останнім часом перших інструментальних систем, що підтримують програмування інтелектуальних агентів [11].

2.2. Консенсуси

Механізм консенсусу — це відмовостійкий механізм, який використовується в комп’ютерних і блокчейн-системах для досягнення необхідної згоди щодо одного значення даних або єдиного стану мережі між розподіленими процесами або системами з кількома агентами, наприклад, з криптовалютами.

Хоча proof-of-work забезпечує найбільшу криптовалюту — біткоїн, — це не єдиний спосіб запустити мережу криптовалюти. Найважливіші типи механізмів консенсусу, які використовуються сьогодні, поділяються на кілька основних типів:

- Proof-of-Work: за допомогою proof-of-work майнери змагаються один з одним, щоб перевірити наступний блок транзакції та отримати винагороду. Це дуже енергоємний механізм консенсусу, але забезпечує високий ступінь довіри.
- Proof-of-stake (PoS) — це консенсусний механізм, за допомогою якого ті, хто володіє найбільшою валютою мережі, підтверджують нові блоки. Це дозволяє здійснювати транзакції швидше та з меншими витратами. Він винагороджує тих, хто має найбільшу частку в мережі, за подальшу участь.
- Proof-of-Authority: не так поширено, але має унікальну форму. Він використовується в основному приватними компаніями або організаціями, які використовують блоки, створені перевіреними джерелами, які мають спеціальні дозволи на доступ до мережі. Гарантії ґрунтуються на репутації та авторитеті, а не на громадському консенсусі, як у випадку з іншими механізмами.
- Delegated Proof-of-Stake — це варіант PoS, у якому користувачі, які роблять ставки своїх монет, можуть голосувати за кількість делегатів для створення нових блоків.
- Proof-of-Capacity: валюти покладаються на доступний простір на жорсткому диску комп’ютера для децентралізованої перевірки блоків і процесу генерації.
- Proof-of-Activity: механізм консенсусу підтвердження активності є гібридом підтвердження частки та підтвердження роботи, у якому майнер прагне використовувати найкраще з обох систем.
- Proof-of-Elapsed Time: використовує випадковий таймер, який працює незалежно на кожному вузлі, щоб випадковим чином призначити перевірку блоку майнеру.
- Proof-of-Burn: за допомогою proof-of-burn консенсус досягається тим, що майнери періодично спалюють монети, тобто процес остаточного видалення або

виключення конкретної монети з обігу. Це підтверджує нові транзакції, одночасно запобігаючи інфляції [12].

Розглянемо детальніше перші три механізми.

2.2.1. Proof-of-work

Proof-of-work та майнінг тісно пов’язані ідеї. Причина, чому це називається «proof-of-work», полягає в тому, що мережа потребує величезної обчислювальної потужності. Блокчейни Proof-of-work захищені та перевірені віртуальними майнерами по всьому світу, які змагаються, хто першим розгадає математичну головоломку. Переможець отримує можливість оновити блокчейн останніми підтвердженими транзакціями та отримує винагороду від мережі заздалегідь визначеною кількістю криптовалюти.

Підтвердження роботи має деякі вагомі переваги, особливо для відносно простої, але надзвичайно цінної криптовалюти, як-от біткоїн. Це перевірений, надійний спосіб підтримувати безпечний децентралізований блокчейн. Оскільки вартість криптовалюти зростає, більше майнерів отримують стимул приєднатися до мережі, підвищуючи її потужність і безпеку. Через велику обчислювальну потужність для будь-якої особи чи групи стає непрактичним втручатися в блокчейн цінної криптовалюти.

З іншого боку, це енергоємний процес, який може мати проблеми з масштабуванням для розміщення величезної кількості транзакцій, які можуть генерувати блокчейни, сумісні зі смарт-контрактами, такі як Ethereum.

Розробники Ethereum з самого початку розуміли, що proof-of-work призведе до обмежень у масштабованості, які врешті-решт доведеться подолати — і справді, оскільки протоколи децентралізованого фінансування (або DeFi) на основі Ethereum різко зростають, блокчейну важко встигати, що призвело до різкого зростання комісії.

У той час як блокчейн біткоїн здебільшого має обробляти вхідні та вихідні транзакції біткоїнів, подібно до величезної чекової книжки, блокчейн Ethereum також

має обробляти величезну кількість транзакцій DeFi, смарт-контрактів стейблкойнів, карбування та продажів NFT, а також будь-які інновації, які пропонують розробники у майбутньому.

Їхнє рішення полягало в тому, щоб побудувати абсолютно новий блокчейн ETH2, який почав розгортатися в грудні 2020 року і був завершений у 2022 році. Оновлена версія Ethereum використовуватиме швидший і менш ресурсомісткий механізм консенсусу під назвою proof-of-stake. Криптовалюти, включаючи Cardano, Tezos і Atmos, використовують консенсусні механізми підтвердження частки — з метою максимізації швидкості та ефективності при зниженні комісій.

2.2.2. Proof-of-stake

У системі proof-of-stake стейкинг виконує функцію, подібну до майнінгу proof-of-work, оскільки це процес, за допомогою якого учасника мережі вибирають для додавання останньої партії транзакцій до блокчейну та отримання трохи криптовалюти в обмін.

Точні деталі залежать від проекту, але загалом блокчейни підтвердження частки використовують мережу «валідаторів», які вносять — або «ставлять» — свою власну криптовалюту в обмін на можливість перевірити нову транзакцію, оновити блокчейн і заробити винагороду.

Мережа обирає переможця на основі кількості крипто, яку кожен валідатор має в пулі, і тривалості часу, протягом якого він там був — буквально винагороджуючи учасників, які інвестували найбільше.

Коли переможець підтвердить останній блок транзакцій, інші валідатори можуть засвідчити, що блок є правильним. Після досягнення порогової кількості атестацій мережа оновлює блокчейн.

Усі валідатори, що беруть участь, отримують винагороду у власній криптовалюті, яка зазвичай розподіляється мережею пропорційно долі кожного валідатора.

Стати валідатором — це велика відповідальність і вимагає досить високого рівня технічних знань. Мінімальна сума криптовалюти, яку валідатори повинні зробити, часто є відносно високою (для ETH2, наприклад, це 32 ETH), і валідатори можуть втратити частину своєї ставки за допомогою процесу, що називається скороченням, якщо їх вузол виходить з мережі або якщо вони перевіряють «поганий» блок транзакцій.

Але навіть якщо це звучить як надто велика відповідальність, ви все одно можете брати участь у стейкінгу, приєднавшись до пулу стейкінгів, яким керує хтось інший, — і отримувати винагороду за криптовалюту. Цей процес часто називають делегуванням, і інструменти, пропоновані біржами Coinbase, можуть зробити його простим.

Споживання енергії є однією з основних відмінностей між двома механізмами консенсусу. Оскільки блокчейни з підтвердженням частки не вимагають від майнерів витрачати електроенергію на дублюючі процеси (змагаючись розв’язати ту саму головоломку), підтвердження частки дозволяє мережам працювати зі значно меншим споживанням ресурсів.

Обидва механізми консенсусу мають економічні наслідки, які штрафують збої в мережі та перешкоджають зловмисникам. У якості підтвердження роботи штрафом для майнерів, які подають недійсну інформацію або блоки, є безповоротна вартість обчислювальної потужності, енергії та часу. На підтвердження участі криптокошти валідаторів служать економічним стимулом діяти в найкращих інтересах мережі. У випадку, якщо валідатор приймає поганий блок, частина його поставлених коштів буде «урізана» як штраф. Сума, яку можна скоротити для валідатора, залежить від мережі.

2.2.3. Proof-of-authority

Proof-of-Authority (PoA) — це сімейство консенсусних алгоритмів, яке забезпечує високу продуктивність і відмовостійкість. У PoA права на створення нових блоків

надаються вузлам, які довели свої повноваження. Щоб отримати ці повноваження та право генерувати нові блоки, вузол повинен пройти попередню автентифікацію.

Переваги консенсусу PoA порівняно з іншими типами консенсусу, які вимагають підтвердження витрачених обчислювальних ресурсів (Proof-of-Work) або наявної «частки» (Proof-of-Stake):

- Високопродуктивне обладнання не потрібно. У порівнянні з консенсусом PoW, консенсус PoA не вимагає від вузлів витрачати обчислювальні ресурси для вирішення складних математичних завдань.
- Інтервал часу, за який генеруються нові блоки, є передбачуваним. Для консенсусу PoW і PoS цей час змінюється.
- Висока швидкість транзакцій. Блоки генеруються авторизованими мережевими вузлами в послідовності через встановлений інтервал часу. Це збільшує швидкість перевірки транзакцій.
- Толерантність до скомпрометованих і шкідливих вузлів, якщо 51% вузлів не скомпрометовано. Apla реалізує механізм заборони для вузлів і засоби відкликання прав на створення блоків.

Атаки типу «відмова в обслуговуванні» у цьому консенсусі мають вигляд: зловмисник надсилає велику кількість транзакцій і блоків на цільовий вузол мережі, намагаючись порушити його роботу та зробити його недоступним.

Механізм PoA дозволяє захиститися від цієї атаки:

- Оскільки мережеві вузли проходять попередню автентифікацію, права на генерування блоків можна надати лише тим вузлам, які можуть протистояти DoS-атакам.
- Якщо вузол недоступний протягом певного періоду, його можна виключити зі списку перевіряючих вузлів.

Згідно з консенсусом PoA, атака 51% вимагає від зловмисника отримати контроль над 51% вузлів мережі. Це відрізняється від атаки 51% для консенсусних

типів Proof-of-Work, коли зловмиснику потрібно отримати 51% обчислювальної потужності мережі. Отримати контроль над вузлами в дозволеній мережі блокчейн набагато важче, ніж отримати обчислювальну потужність.

Наприклад, у мережі консенсусного типу PoW зловмисник може збільшити обчислювальну потужність (продуктивність) контрольованого сегмента мережі, таким чином збільшуючи контрольований відсоток. Це не має сенсу для консенсусу PoA, оскільки обчислювальна потужність вузла не впливає на рішення мережі блокчейну.

2.3. Висновки до розділу 2

На перший погляд переваги «proof-of-stake» є очевидними, та в роботі порівняно кількісні характеристики. Це дасть змогу впевнено стверджувати про доцільність використання того чи іншого механізму консенсусу.

Також для збору метрик буде використано агентну архітектуру, щоб здійснити перевірку та перевірити комбінації параметрів автоматизовано.

РОЗДІЛ 3 ОПИС ПРОГРАМНОГО ПРОДУКТУ

3.1. Використані інструменти

Для того, щоб розгорнути локальну версію блокчейну були використані наступні технології: Puppeth, Vagrant, PHP [13][14][15].

Створення приватного блокчейну зараз не є тривіальним завданням. Щоб розгорнути мережу, потрібно багато часу та зусиль. Хоча з'являються сервіси, які значно спрощують це завдання, але ми хочемо контролювати приватну мережу та мати повну можливість запускати та підтримувати її всередині. Зробити це на 100% вручну було б надзвичайно трудомістким, якби не зручний інструмент від групи Ethereum під назвою puppeth.

Puppeth — це майстер CLI, який допомагає створити нову мережу Ethereum аж до генезису, завантажувальних вузлів, підписувачів, ethstats, крана, інформаційної панелі тощо, без клопоту, який зазвичай потребує налаштування всіх цих служб одну за одною. Puppeth використовує ssh для підключення до віддалених серверів і створює свої мережеві компоненти з контейнерів докерів за допомогою docker-compose. Користувач керується процесом за допомогою майстра командного рядка, який автоматично виконує важку роботу та конфігурує топологію.

Vagrant — це програмний продукт із відкритим вихідним кодом для створення та підтримки портативних віртуальних середовищ розробки програмного забезпечення, який допоможе нам з розташуванням відповідних вузлів мережі. Він використовує «Provisioners» і «Providers» як будівельні блоки для керування середовищами розробки. Провайдери — це інструменти, які дозволяють користувачам налаштовувати конфігурацію віртуальних середовищ. Puppeth і Chef є двома найпоширенішими провайдерами в екосистемі. Постачальники — це служби, які Vagrant використовує для налаштування та створення віртуальних середовищ.

Підтримка віртуалізації VirtualBox, Hyper-V і Docker поставляється з Vagrant, тоді як VMware і AWS підтримуються через плагіни.

Для контролювання і автоматизації процесу було обрано варіант написання свого агенту та агентної архітектури, використовуючи мову програмування PHP.

3.2. Деталі розробки програмного продукту

Вирішення задачі полягало в декількох етапах, тому опишемо їх детально.

Перший етап — створення шести віртуальних машин, щоб запустити на них програму та додати в спільну мережу. Сконфігуровано та налаштовано порти, які будуть приймати і з яких будуть віддаватись команди та дані.

Другий етап полягає у створенні вузлів блокчейну Ethereum на кожній віртуальній машині. Для цього було застосовано майстра CLI Puppeth, який дав готовий інтерфейс. Також програма згенерувала свої унікальні адреси в мережі. Коли програми були готові до отримання вказівок, а акаунти ініціалізовані — зроблено додавання їх у мережу.

Третій етап — створення такої ж мережі, але з proof-of-stake механізмом консенсусу. Для цього розподілення було 3 master вузлів та 3 slave вузлів.

Етап номер чотири заключає в собі написання інтелектуального агенту, який автоматично створює віртуальні машини, розгортає мережу з певним механізмом консенсусу, надсилає запити на мережу, перевіряє можливу кількість операцій на секунду та збирає метрику витраченого ресурсу: комп’ютерного (розрахункового) та приблизно кількість витраченої електроенергії.

Для того, щоб результат був більш наочним — було додано ще огляд proof-of-authority механізму.

Пройдемось по деталям реалізації.

Копіюємо код (Рисунок 3.1).

```
mkdir my_project; cd my_project
git clone https://github.com/swagger/improved hi_puppeth1
git clone https://github.com/swagger/improved hi_puppeth2
git clone https://github.com/swagger/improved hi_puppeth3
git clone https://github.com/swagger/improved hi_puppeth4
git clone https://github.com/swagger/improved hi_puppeth5
git clone https://github.com/swagger/improved hi_puppeth6
```

Рисунок 3.1.

Замінюємо IP-адресу другого клонів, перейшовши в папку hi_puppethN і змінивши поле IP-адреси на 192.168.10.1N замість 192.168.10.10.

Далі відкриваємо деякі порти на віртуальних машинах, змінивши останній розділ Homestead.yaml кожного клону, ось так (Рисунок 3.2):

```
ports:
    - send: 8545
      to: 8545
    - send: 30301
      to: 30301
    - send: 30302
      to: 30302
    - send: 30303
      to: 30303
    - send: 30304
      to: 30304
    - send: 30305
      to: 30305
    - send: 30306
      to: 30306
```

Рисунок 3.2.

Нарешті, запускаємо vagrant up; vagrant ssh для завантаження кожної машини та SSH до неї.

Puppeth запускає допоміжні програми та вузли Ethereum у контейнерах Docker, тому також потрібен Docker. Усі інші передумови буде втягнуто Puppeth через сам Docker (Рисунок 3.3).

```
sudo add-apt-repository -y ppa:ethereum/ethereum
sudo apt-get update
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    software-properties-common \
    ethereum \
    docker.io \
    docker-compose
```

Рисунок 3.3.

На головній машині (поза віртуальними машинами) створюємо нові облікові записи Ethereum у папці, де ми запускаємо наш проект (Рисунок 3.4).

```
$ mkdir node1 node2
$ geth --datadir node1 account new
INFO [05-20|10:27:20] Maximum peer count                       ETH=25
Your new account is locked with a password. Please give a password. Do
Passphrase:
Repeat passphrase:
Address: {aba88be2dc16eaed464e3991eed5a1eaa5e7b11b}
$ geth --datadir node2 account new
INFO [05-20|10:27:35] Maximum peer count                       ETH=25
Your new account is locked with a password. Please give a password. Do
Passphrase:
Repeat passphrase:
Address: {655a6ea9950cdf9f8a8175fda639555f17277bdf}
```

Рисунок 3.4

Тепер віртуальні машини запущені, а облікові записи ініціалізовано. Коли віддалені сервери все ще працюють, у новій вкладці на вашому хості запускаємо Puppeth за допомогою команди puppeth.

Перше, що він запитає, це назва мережі. Це корисно для ідентифікації різних блокчейнів, якщо використовується декілька на локальній машині. Тут використовується «puptest» (Рисунок 3.5).

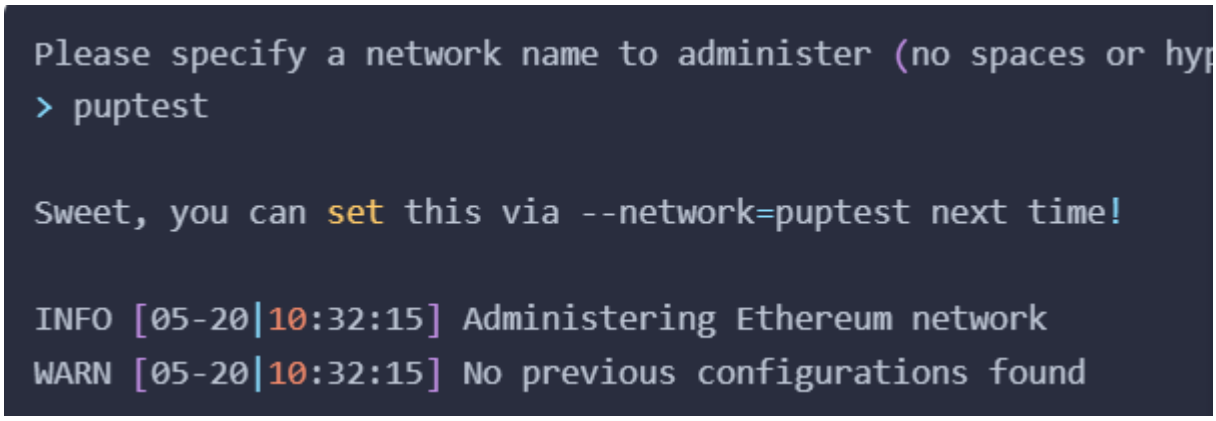

Рисунок 3.5.

Puppeth запитає парольну фразу ключа SSH на випадок, якщо SSH використовується для підключення до сервера. Якщо ні, він запитає пароль. Пароль SSH за замовчуванням для користувача vagrant на відповідній віртуальній машині.

Вихідні дані в кінці повторюють стан «справності» віддаленого сервера. Оскільки в ньому немає запущених служб, він просто перерахує IP-адресу.

Повторимо процес для інших віртуальних машини, щоб всі з’явилися на екрані стану справності.

Щоб запустити блокчейн, налаштовуємо новий файл генезису. Файл генезису - це файл, з якого будується перший (генезисний) блок і на якому зростає кожен наступний блок (Рисунок 3.6).

```
How many seconds should blocks take? (default = 15)
> 10

Which accounts are allowed to seal? (mandatory at least one)
> 0xaba88be2dc16eaed464e3991eed5a1eaa5e7b11b
> 0x655a6ea9950cdf9f8a8175fda639555f17277bdf
> 0x

Which accounts should be pre-funded? (advisable at least one)
> 0x655a6ea9950cdf9f8a8175fda639555f17277bdf
> 0xaba88be2dc16eaed464e3991eed5a1eaa5e7b11b
> 0x

Specify your chain/network ID if you want an explicit one (default
>
INFO [05-20|11:25:55] Configured new genesis block
```

Рисунок 3.6.

Puppeth розгортає ці компоненти в окремих докер-контейнерах за допомогою інструменту docker-compose.

Вибираємо перший раніше доданий сервер. Потім ми додаємо порт для розгортання цього програмного забезпечення, а потім називаємо домен, через який матимемо доступ до програми. Нарешті, створюємо простий «секрет» для доступу до API програми. Потім Docker бере на себе роботу та створює для нас програмне забезпечення.

Кінцевим результатом має бути ще один вихід на екран здоров’я, але цього разу таблиця матиме інший запис: у ній також буде перераховано програму ethstats із деталями її конфігурації:

Під час відвідування URL-адреси homestead.test:8081 у браузері відображається статистика (Рисунок 3.7).

```
Which server do you want to interact with?
 1. vagrant@192.168.10.10
 2. Connect another server
> 1

Which port should ethstats listen on? (default = 80)
> 8081

Allow sharing the port with other services (y/n)? (default =
>
INFO [05-20|11:43:32] Deploying nginx reverse-proxy
Building nginx
Step 1/1 : FROM jwilder/nginx-proxy
 ---> e143a63bea4b
Successfully built e143a63bea4b
Recreating puptest_nginx_1

Proxy deployed, which domain to assign? (default = 192.168.1
> homestead.test

What should be the secret password for the API? (must not be
> internet2
Found orphan containers (puptest_nginx_1) for this project.
Building ethstats
Step 1/2 : FROM puppeth/ethstats:latest
```

Рисунок 3.7.

Далі потрібно розгорнути принаймні один вузол, щоб ця програма почала щось показувати (Рисунок 3.8).

```
INFO [05-20|11:44:23] Starting remote server health-check      server=vagrant@192.168.10.10
+----------------------+---------------+----------+----------------------+----------------+
|        SERVER        |    ADDRESS    | SERVICE  |        CONFIG        |     VALUE      |
+----------------------+---------------+----------+----------------------+----------------+
| vagrant@192.168.10.10 | 192.168.10.10 | ethstats | Banned addresses     | []             |
|                      |               |          | Login secret         | internet2      |
|                      |               |          | Website address      | homestead.test |
|                      |               |          | Website listener port | 8081           |
|                      |               |          | -------------------- | -------------- |
|                      |               | nginx    | Shared listener port | 8081           |
+----------------------+---------------+----------+----------------------+----------------+
```

Рисунок 3.8.

Завантажувальний вузол — це вузол, який служить лише першою точкою з'єднання, через яку вузол Ethereum з'єднується з іншими вузлами. По суті, це ретранслятор інформації, який допомагає вузлам підключатися.

Вибрано варіант 2, щоб розгорнути завантажувальний вузол (Рисунок 3.9).

```
Which server do you want to interact with?
 1. vagrant@192.168.10.10
 2. vagrant@192.168.10.11
 3. Connect another server
> 1

Where should data be stored on the remote machine?
> /home/vagrant/mychain

Which TCP/UDP port to listen on? (default = 30303)
>

How many peers to allow connecting? (default = 512)
>

How many light peers to allow connecting? (default = 256)
>

What should the node be called on the stats page?
> booty
```

Рисунок 3.9.

Docker створить вузол і запустить його. Місце для зберігання даних на віддаленій машині є довільним. Було обрано домашній каталог користувача vagrant.

Вузол печатки - це вузол, який може служити майнером нових блоків. Розгорнемо його (Рисунок 3.10).

```
Which server do you want to interact with?
 1. vagrant@192.168.10.10
 2. vagrant@192.168.10.11
 3. Connect another server
> 1

Where should data be stored on the remote machine?
> /home/vagrant/mychainsealer

Which TCP/UDP port to listen on? (default = 30303)
> 30301

How many peers to allow connecting? (default = 50)
>

How many light peers to allow connecting? (default = 0)
>

What should the node be called on the stats page?
> sealer

Please paste the signer's key JSON:
> {"address":"655a6ea9950cdf9f8a8175fda639555f17277bdf","crypto":{"ci

What's the unlock password for the account? (won't be echoed)
>

What gas limit should empty blocks target (MGas)? (default = 4.700)
>
```

Рисунок 3.10.

Усі значення за замовчуванням, крім місця для зберігання даних і назви вузла. Для JSON беремо вміст із файлу, який було створено раніше під час створення нових облікових записів Ethereum. Повний вміст цього файлу вставляємо сюди, і Puppeth запитає пароль, щоб розблокувати цей гаманець. Відтоді все інше знову автоматично.

На екрані працездатності вузол має відображатися як робочий, і він має з’являтися на екрані Ethstats під назвою, яку йому було дано (Рисунок 3.11).

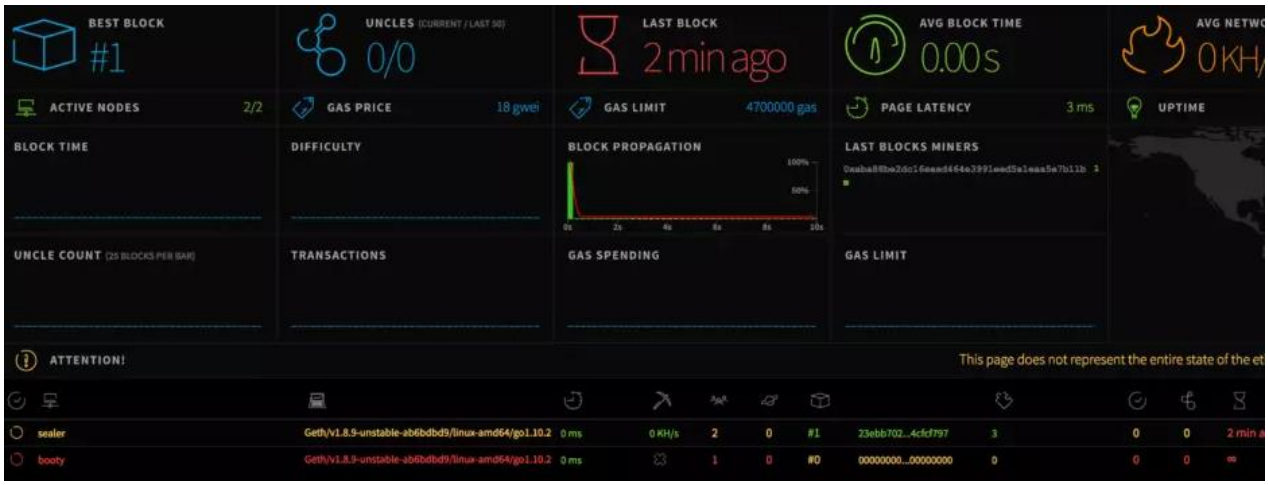

Рисунок 3.11.

Далі повторимо процес для іншої віртуальної машини. Даємо цьому вузлу іншу назву та використовуємо інший файл сховища ключів. Іншими словами, встановлюємо інший обліковий запис, який було створили, як печатку у цьому вузлі. Вузли тепер працюватимуть і майнитимуть разом.

Щоб мати можливість легко надсилати Ether і спеціальні токени, розгорнемо власну версію MyEtherWallet за допомогою Puppeth.

Виберемо Wallet у списку компонентів для розгортання та заповнимо параметри (Рисунок 3.12).

```
Which port should the wallet listen on? (default = 80)
> 8083

Allow sharing the port with other services (y/n)? (default = yes)
> no

Where should data be stored on the remote machine?
> /home/vagrant/wallet

Which TCP/UDP port should the backing node listen on? (default = 30303)
> 30304

Which port should the backing RPC API listen on? (default = 8545)
>

What should the wallet be called on the stats page?
> wallet
```

Рисунок 3.12.

Нарешті, Puppeth пропонує «Інформаційну панель», веб-інтерфейс, який поєднує в собі всі інтерфейси, які було запущено (Рисунок 3.13).

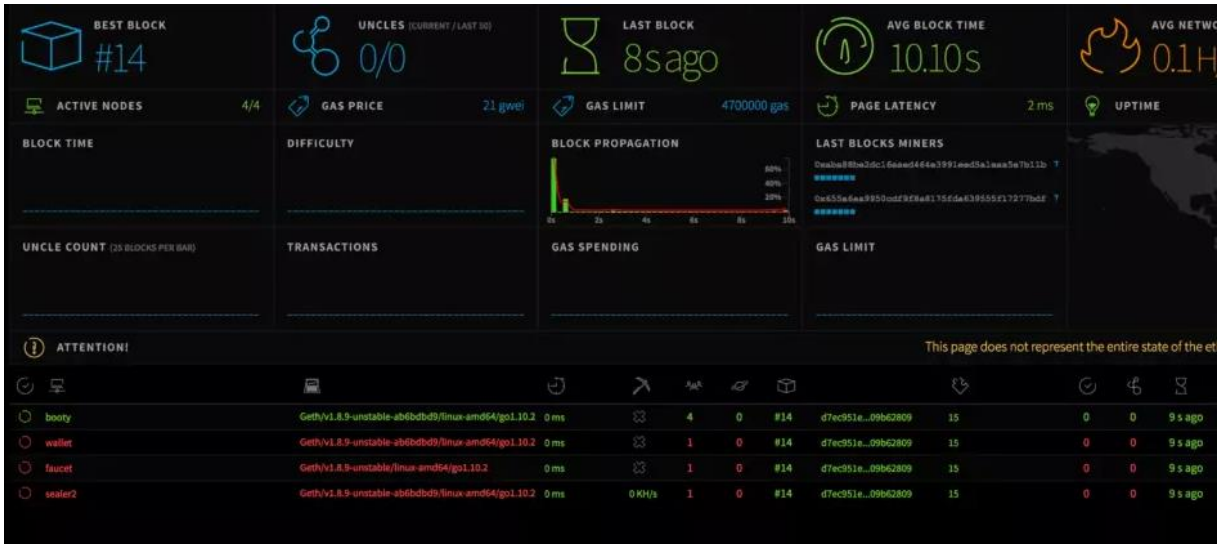

Рисунок 3.13.

Аналогічно додаємо ще 4 вузли.

Як можемо помітити, процес створення і розгортання локальної мережі достатньо довгий та нетривіальний, тож було написано агента, який може робити цей процес автоматично та навантажувати отриману систему (Рисунок 3.14).

Агент виконує всі дії, що вказано вище, взаємодіючи з системою та віртуальними машинами. Також він налаштовує мережу та реагує на відповідь від системи. Також на старті в якості параметра задається механізм консенсусу, яким чином мають створитись вузли. Коли мережа готова приймати запити — агент починає навантажувати її великою кількістю запитів, починаючи зі 100 запитів на секунду, підвищуючи це число на 100 через хвилину, якщо мережа впоралась.

Коли стрес тестування завершено — робиться запит на ethstats для збору метрик.

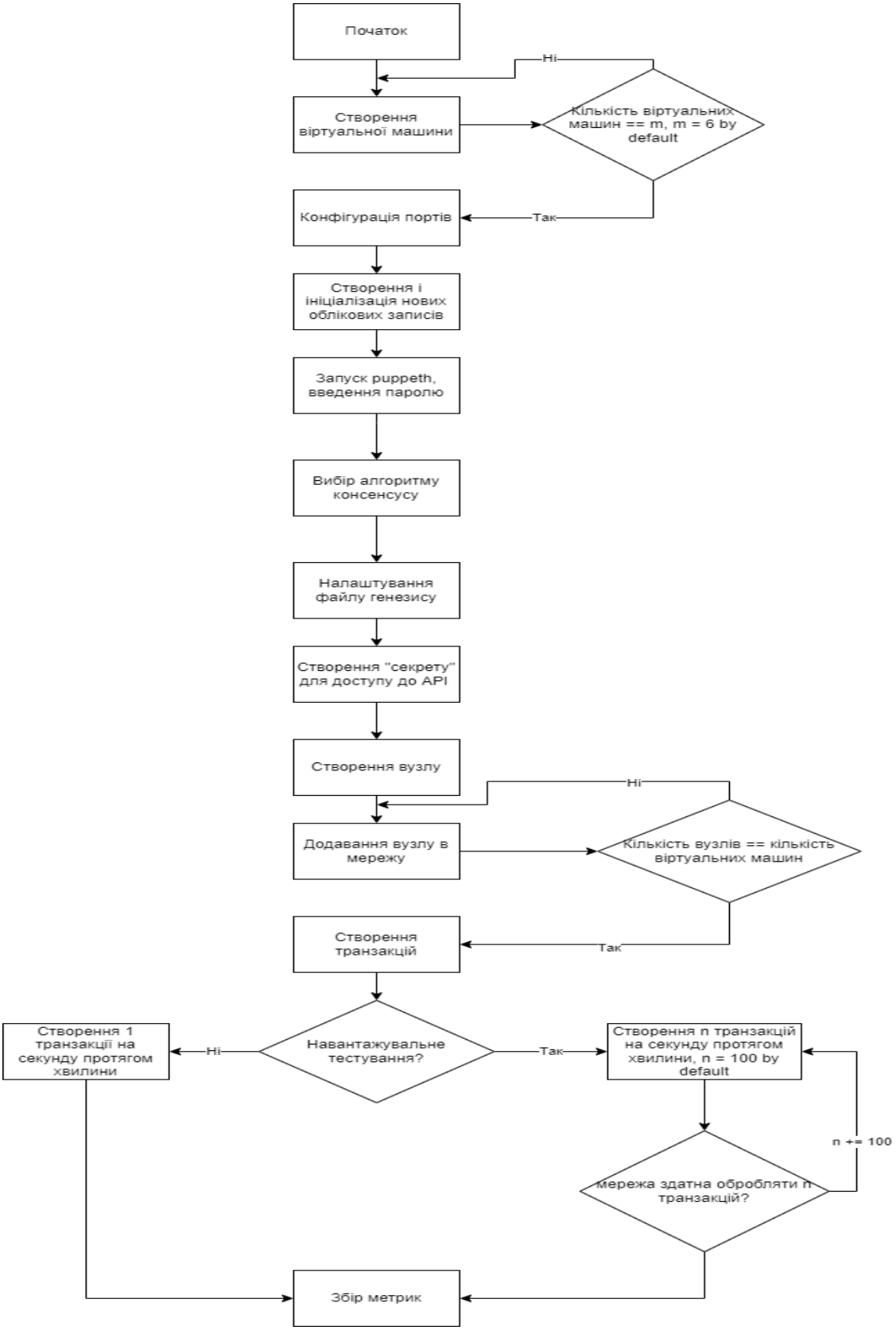

Рисунок 3.14 — Архітектура агента

3.3. Аналіз отриманих даних

Ефективність трьох консенсусних протоколів порівнюється на основі наступних параметрів:

- Transactions Per Second (TPS)
- Electricity usage per transaction

Зібрані метрики було оформлено у вигляді таблиць для наочності. Інструмент для моніторингу стану мережі Ethstats одразу показував кількість транзакцій на секунду для кожного алгоритму (Табл. 3.1).

Таблиця 3.1 — Кількість транзакцій на секунду в залежності від механізму консенсусу

Механізм консенсусу	Кількість транзакцій на секунду
PoA	2380
PoW	395
PoS	4613

Фінальним порівнянням виступили результати метрик, отримання яких власне і ставилось за завдання: кількість витраченої електроенергії (Табл. 3.2). Ethstats дозволяє отримати ці дані, надіславши запит на його інтерфейс.

Таблиця 3.2 — Кількість електроенергії для проведення однієї транзакції в залежності від механізму консенсусу

Механізм консенсусу	Кількість використаної електроенергії (КВт-годин/транзакція)
PoA	1.1783
PoW	1.2321
PoS	0.0201

3.4. Висновки до розділу 3

Таким чином можемо сказати, що proof-of-stake концепт дійсно є значно енергоефективним та перехід на нього є доцільним, оскільки шкоди навколишньому середовищу завдається менше. Мінусом цього підходу є більша складність в реалізації та відсутність довготривалих випробувань в реальних умовах, тобто практично не доведено стійкість до зовнішніх або внутрішніх атак, лише описано теоретично.

РОЗДІЛ 4 РОЗРОБКА СТАРТАП-ПРОЕКТУ

Останні роки довели, що стартапи сильніші, ніж більшість думала. Гнучкий та інноваційний підхід були ключовими, коли світ зіштовхнувся з такими проблемами, як віддалена робота, масштабні зміни в галузі та ринку, а також абсолютно нова реальність.

Здатність стартапів швидко змінюватися та адаптуватися і є ключовою властивістю даного виду бізнесу, не кажучи вже про те, що багато стартапів продовжували рости та розширювати свої команди.

Стартап — це бізнес-структура, що розвивається та працює на основі інновацій, створена для вирішення проблеми шляхом надання нової пропозиції в умовах надзвичайної невизначеності.

Власне, стартап — це бізнес, який:

- швидко росте;
- порушує ринок або галузь (щось нове, що змушує конкурентів удосконалюватись);
- вирішує проблему;
- працює в умовах надзвичайної невизначеності.

Багато підприємців і відомих бізнес-магнатів визначають стартап як культуру та менталітет побудови бізнесу на основі інноваційної ідеї для вирішення критичних проблем.

Одне, що відрізняє стартапи від інших компаній — це зв’язок між їхнім продуктом та його попитом. Стартапи мають продукти, орієнтовані на невикористаний ринок. Підприємці-початківці знають ідеальну стратегію, щоб створити продукт, який хоче ринок, а також охопити й обслуговувати їх усіх. Це викликає швидке зростання.

4.1. Опис ідеї проекту

В межах підпункту було проаналізовано і подано у вигляді таблиць:

- зміст ідеї;
- можливі напрямки застосування;
- основні вигоди, що може отримати користувач товару;
- чим відрізняється від існуючих аналогів та замінників.

Перші три пункти подані у вигляді таблиці (Табл. 4.1) і дають цілісне уявлення про зміст ідеї та можливі базові потенційні ринки, в межах яких потрібно шукати групи потенційних клієнтів.

Таблиця 4.1 — Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Дана комплексна система дозволяє розгортати мережу блокчейну локально автоматично, навантажувати систему і збирати корисні метрики.	Тестування блокчейну, коригуючи код під свої потреби.	Швидкий доступ до локальної версії блокчейну
	Оцінка нових механізмів консенсусу шляхом приєднання його до системи	Швидка можливість оцінки нового самописного механізму консенсусу і порівняння з іншими.

Аналіз потенційних техніко-економічних переваг ідеї (чим відрізняється від існуючих аналогів та замінників) порівняно із пропозиціями конкурентів передбачає:

- визначення переліку техніко-економічних властивостей та характеристик ідеї;

- визначення попереднього кола конкурентів (проектів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та проводиться збір інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до визначеного вище переліку;
- проводиться порівняльний аналіз показників: для власної ідеї визначаються показники, що мають:
    - гірші значення (W, слабкі);
    - аналогічні (N, нейтральні) значення;
    - кращі значення (S, сильні) (Табл. 4.2).

Таблиця 4.2 — Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Технікоекономічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Block.stats	Ethereum.stats			
1.	Форма виконання	Надавання послуг	Надавання послуг	Надавання послуг		+	
2.	Собівартість	Низька	Висока	Висока			+
3.	Функціонал	Широкий	Широкий	Широкий	+		

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності.

4.2. Технологічний аудит ідеї проекту

В межах даного підрозділу було проведено аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару). Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових (Табл. 4.3):

1. За якою технологією буде виготовлено товар згідно ідеї проекту?
2. Чи існують такі технології, чи їх потрібно розробити/доробити?
3. Чи доступні такі технології авторам проекту?

Таблиця 4.3 — Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології реалізації	Наявність технологій	Доступність технологій
1	Розгортання мережі блокчейну локально автоматично, навантажування системи і збір корисних метрик	Використання мови програмування PHP	Наявна	Доступна
		Використання мови програмування C#	Відсутні	Недоступні
		Використання мови C++	Відсутні	Недоступні
Обрана технологія реалізації ідеї проекту: мова програмування Node.js та Golang.				

За результатами аналізу таблиці зроблено висновок щодо можливості технологічної реалізації проекту. Технологічним шляхом реалізації проекту було обрано таку технологію, як PHP через доступність та безкоштовність.

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Спочатку було проведено аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (Табл. 4.4).

Таблиця 4.4 — Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	50
2	Загальний обсяг продаж, грн/ум.од	10000
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі, %	20%

За результатами аналізу таблиці 4.4 було зроблено висновок, що ринок є привабливим для входження.

Надалі були визначені потенційні групи клієнтів, їх характеристики та сформовано орієнтовний перелік вимог до товару для кожної групи (Табл. 4.5).

Таблиця 4.5 — Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Точна та швидка генерація рекомендацій цільовій аудиторії	Власник бізнесу	Велика кількість даних	Простота використання, висока точність
Підбір рекомендацій	Кінцевий користувач	Цікавить простота у використанні, низька ціна підтримки системи	Швидкість створення, низька ціна

Після визначення потенційних груп клієнтів було проведено аналіз ринкового середовища: складено таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (Табл. 4.6, 4.7).

Таблиця 4.6 — Фактори загроз

№ / п	Фактор	Зміст загрози	Можлива реакція компанії
	Конкуренція	Вихід на ринок продуктів з кращими характеристиками	Передбачити додаткові переваги власного програмного продукту (ПП) для того, щоб повідомити про них саме після виходу на ринок конкурентів. Вдосконалення технічних моментів власного продукту. Обрати нову цільову аудиторію і зосередитися на ній: зниження цін.
	Зміна потреб користувачів	Користувачам необхідний сервіс з більшим/новим функціоналом.	Розроблення гнучкої архітектури програмного забезпечення для легшого впровадження нового функціоналу

Таблиця 4.7 — Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Гнучкі ціни	Зменшення ціни товару задля збільшення попиту	Введення власних гнучких цін
2	Поява нових методів локального розгортання мережі	З’являться нові методи, що будуть швидше, та більш ефективно розгортати мережу локально	Покращити ПП додаванням нового функціоналу, розширення можливостей

Надалі було проведено аналіз пропозиції: визначили загальні риси конкуренції на ринку (Табл. 4.8).

Таблиця 4.8 — Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
1. Тип конкуренції- чиста	Існує величезна кількість конкурентів на ринку.	Якісно провести рекламу.
2. За рівнем конкурентної боротьби - міжнародний	Компанії-конкуренти з інших країн	Створити основу ПП таким чином, щоб можна було легко переробити даний ПП для використання у галузях інших країн.

3. За галузевою ознакою - міжгалузева	Продукт може використовуватись для різних галузей	Постійне вдосконалення продукту.
4. Конкуренція за видами товарів: - товарно-видова	Конкуренція між видами ПП, їх особливостями.	Створити ПП, враховуючи конкурентів
5. За характером конкурентних переваг - неціновa	Вдосконалення технології створення ПП, щоб собівартість була нижчою	Удосконалення моделі. Використання більш дешевих технологій для розробки, ніж використовують конкуренти, але тільки якщо ці технології відповідають необхідним вимогам якості.
6. За інтенсивністю - не марочна	Бренд присутній, але його роль незначна	Реклама, участь у конференціях, семінарах.

Було проведено аналіз конкуренції у галузі за моделлю М. Портера (Табл. 4.9).

Таблиця 4.9 — Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Поста чальники	Клієнти	Товаризамінники
	Навести перелік прямих конкурентів	Визначити бар’єри входження в ринок	Визна чити фактори сили постахальників	Визначити фактори сили споживачів	Фактори загроз з боку замінників

	Amazon Personalize	Наявність вже існуючих рішень	-	Контроль якості продукту	Наявність більш широкого функціоналу, зручнішого інтерфейсу та авторитет
Висновки:	Досить інтенсивна конкурентна боротьба з іншими гравцями	Є можливості виходу на ринок, але є і конкуренти.	-	Клієнти диктують умови роботи на ринку: зручний інтерфейс	Необхідно випускати ПЗ не гірше, ніж у конкурентів та розширяти функціонал.

За результатами аналізу було зроблено висновок про можливість роботи на ринку з огляду на конкурентну ситуацію.

Цей висновок був врахований при формулюванні переліку факторів конкурентоспроможності у наступному пункті. На основі аналізу конкуренції, проведеного в таблиці, а також із урахуванням характеристик ідеї проекту (Табл. 4.2), вимог споживачів до товару (Табл. 4.5) та факторів маркетингового середовища (Табл. 4.6, 4.7) визначається та обґрунтовується перелік факторів конкурентоспроможності.

Аналіз оформлено у (Табл. 4.10).

Таблиця 4.10 — Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування
1	Ціна	Один із факторів для вибору продукту клієнтом.
2	Якість	Один із факторів для вибору продукту клієнтом.
3	Зручність роботи з програмою	Дозволяє користувачу легко працювати з програмою

За визначеними факторами конкурентоспроможності (Табл. 4.10) проведено аналіз сильних та слабких сторін стартап-проекту (Табл. 4.11).

Таблиця 4.11 — Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні						
			–3	–2	–1	0	+1	+2	+3
1	Ціна	15					*		
2	Якість	10			*				
3	Зручність роботи з програмою	15					*		

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (Табл. 4.12) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (Табл. 4.11).

Перелік ринкових загроз та ринкових можливостей було складено на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення.

Таблиця 4.12 — SWOT-аналіз стартап-проекту

Сильні сторони: Якість Простота використання Висока швидкодія	Слабкі сторони: Дуже насичений ринок, мала кількість функціоналу, відсутня кросплатформеність.
Можливості: насичення ринку новим підходом до розгортання мережі; різноманітна клієнтура, вдосконалення системи	Загрози: Конкуренція

На основі SWOT-аналізу було розроблено альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок.

Визначені альтернативи були проаналізовані з точки зору строків та ймовірності отримання ресурсів (таблиця 4.13).

Таблиця 4.13 — Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	PR, просування бренду	50%	6 місяців
2	Перехід на безкоштовне розповсюдження	75%	3 місяців
3	Партнерство для об’єднання продукції	65%	2 місяці

Після аналізу було обрано альтернативу №2.

4.3. Аналіз ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: було проведено опис цільових груп потенційних споживачів (таблиця 4.14).

Таблиця 4.14 — Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Підприємці	Висока	Високий	Сильна	Просто
2	Великі компанії	Середня: велика конкуренція і можливість власних веб-відділів	Високий	Сильна	Складно
3	Маленькі компанії.	Низька	Низький	Слабка	Середня
Які цільові групи обрано:1,2,3					

За результатами аналізу потенційних груп споживачів було обрано цільові групу, для яких буде запропоновано даний товар, та визначено стратегію охоплення ринку - стратегію диференційованого маркетингу (компанія працює з декількома сегментами).

Для роботи в обраних сегментах ринку сформовано базову стратегію розвитку (Табл. 4.15).

Таблиця 4.15 — Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Постійне оновлення і покращення продукту	Ринкове позиціонування на індивіуальних користувачів	Швидкодія, якість продукту	Концентрований маркетинг

Наступним кроком обрано стратегію конкурентної поведінки (Табл. 4.16).

Таблиця 4.16 — Визначення базової стратегії конкурентної поведінки

№ п/п	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
1	Ні.	Компанія буде шукати нових споживачів та забирати існуючих у конкурентів	Буде копіювати, удосконалювати та створювати свої унікальні пропозиції	Зайняття конкурентної ніші

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту (Табл. 4.5), а також в залежності від обраної базової стратегії розвитку (Табл. 4.15) та стратегії конкурентної поведінки (Табл. 4.16) розроблено стратегію позиціонування (Табл. 4.17), що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 4.17 — Визначення стратегії позиціонування

п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартаппроекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
1	Легкість розуміння, зручний інтерфейс, надійний, швидкий, точний та достовірний ПП для генерації рекомендацій.	Стратегія диференціації	Позиція на основі порівняння фірми з товарами конкурентів; Відмінні особливості споживача	Економія часу; Зручність застосування; Практичність та точність результату

Результатом виконання підрозділу стала узгоджена система рішень щодо ринкової поведінки стартап-компанії, яка визначає напрями роботи стартап- компанії на ринку.

4.4. Розроблення маркетингової програми стартап-проекту

Сформовано маркетингову концепцію товару, який отримає споживач. Для цього підсумовано результати попереднього аналізу конкурентоспроможності товару (таблиця 4.18). Концепція товару — письмовий опис фізичних та інших характеристик товару, які сприймаються споживачем, і набору вигод, які він обіцяє певній групі споживачів.

Таблиця 4.18 — Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Швидкість отримання результату	Швидка видача рекомендацій наступної пропозиці	Необхідно покращити швидкість навчання моделі для видачі рекомендацій наступної пропозиці
2	Зручність застосування	Нативна підтримка мови програмування Python	Розробка зручного прикладного програмного інтерфейсу
3	Практичність та точність результату	Користувач отримує точні (з малою похибкою розбіжності) результати рекомендацій.	Користувач на виході роботи ПП отримує модель та прогноз, котрі відповідають необхідним показникам варіативності та точності.

Розроблено трирівневу маркетингову модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (таблиця 4.19).

1-й рівень. При формуванні задуму товару вирішується питання щодо того, засобом вирішення якої потреби і / або проблеми буде даний товар, яка його основна вигода. Дане питання безпосередньо пов'язаний з формуванням технічного завдання в процесі розробки конструкторської документації на виріб.

2-й рівень. Цей рівень являє рішення того, як буде реалізований товар в реальному/ включає в себе якість, властивості, дизайн, упаковку, ціну.

3-й рівень Товар з підкріпленням (супроводом) — додаткові послуги та переваги для споживача, що створюються на основі товару за задумом і товару в реальному виконанні (гарантії якості , доставка, умови оплати та ін).

Таблиця 4.19 — Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Генерація палітри кольорів зображення за допомогою інтелектуальних систем</td></tr>
<tr><td rowspan="3">II. Товар реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх/Тл/Е/Ор</td></tr>
<tr><td>1. Індивідуальний підхід.
2. Низька ціна.
3. Простота у використанні.</td><td>1.Нм
2.Нм
3.Нм</td><td>1.Технологічна
2.Економічна
3.Технологічна</td></tr>
<tr><td colspan="3">Якість: тестування фірмами аудиторами</td></tr>
</table>

	Пакування: відсутнє
	Марка: ROSO

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів (таблиця 4.20). Аналіз проведено експертним методом.

Таблиця 4.20 — Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	2000$	3700$	У всіх трьох груп високий рівень доходів	900$--

Наступним кроком є визначення оптимальної системи збуту, в межах якого було прийняте рішення (таблиця 4.21).

Таблиця 4.21 — Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Канал нульового рівня	Продаж	0(напряму)	Власна

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (таблиця 4.22).

Таблиця 4.22 — Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
	Інтеграція API у клієнтській системі	Інтернет	Низька ціна, простота використання, універсальність	Показати переваги рішення над конкурентам, виділити ключові особливості	Створення сайту продукту, розповсюдження інформації про продукт на спеціалізованих ресурсах.

Було визначено, що придбання продукту буде проводитись через мережу Інтернет або при безпосередньому спілкуванні із представниками компанії. Розповсюдження інформації про продукт буде проводитись виключно через Інтернет, адже аудиторія даного продукту активно користується всесвітньою мережею.

Результатом підрозділу стала ринкова (маркетингова) програма, що включає в себе концепції товару, збуту, просування та попередній аналіз можливостей ціноутворення, спирається на цінності та потреби потенційних клієнтів, конкурентні переваги ідеї, стан та динаміку ринкового середовища, в межах якого впроваджено проект, та відповідну обрану альтернативу ринкової поведінки.

4.5. Висновки до розділу 4

В даному розділі проведено підготовчий аналіз для впровадження розробленої системи в якості стартап проекту. Досліджено аналогічні конкурентні системи, встановлено сильні та слабкі сторони системи в порівнянні з ними. Також було досліджено можливі шляхи розповсюдження продукту та його ймовірну аудиторію, рівень доходів та ймовірну ціну продукту, що розробляється.

Було проведено аналіз потенційних ризиків і можливостей, а також розраховані основні фінансово-економічні показники проекту. Отримані результати кажуть про те, що реалізація проекту є доцільною. Було визначено сильні сторони проекту: зручність у використанні, ціна, якість та широкий функціонал. Серед слабких варто виділити повільне навчання. Варто відмітити можливість реклами продукту на спеціалізованих ресурсах із зазначенням сильних сторін проекту.

ВИСНОВКИ

У даній дисертації було спроєктовано агента, який автоматично вміє розгортати мережу блокчейну Ethereum з відповідним механізмом консенсусу та навантажувати систему великою кількістю запитів для отримання кількісних та якісних характеристик можливостей системи.

Поставлена задача є досить складною, оскільки аналіз великою мережі за рахунок екстраполяції локальних результатів може містити похибку. Експеримент проводиться на одній машині, імітуючи багато з'єднань, але реальна мережа набагато більша, з більшою кількістю складових як, наприклад, смарт контракти. Також на отримані результати впливає кількість вузлів та електронних гаманців у системі. Логічно, що реальні умови відрізняються від «лабораторних», але навіть з отриманих результатів можемо побачити різницю у споживанні ресурсів.

Офіційні цифри, які наводить головний сайт Ethereum, зі зменшенням кількості витраченої електроенергії кажуть про скорочення витрат на 99.8%. Тобто за словами розробників Ethereum, споживання має зменшитись у 50 разів. Отримані результати кажуть про зменшення споживання майже в 60 разів. Враховуючи похибку «стерильності» умов, можна зробити висновок, що офіційні цифри досить правдиві.

У рамках поточного дослідження агентна архітектура показала себе як життєздатна для проведення таких експериментів, збору метрик та налаштування мережі, автоматизувавши роботу та значно зменшивши втручання людини, залишаючи їй лише порівняння кінцевих чисел.

Практичне значення отриманих результатів каже про доцільність переходу криптовалют з їх блокчейнами на proof-of-stake концепт, що дозволить значно скоротити кількість використаної електроенергії. Особливо актуальним це є в наш час, коли подібний ресурс є обмеженим та дефіцитним в Україні через війну та атаки по цивільній інфраструктурі як, наприклад, електростанції. Кожна транзакція, виконана

за правилами proof-of-stake, на відміну від proof-of-work або proof-of-authority, дає змогу мати більше домівок зі світлом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Stuart J.Russel and Peter Norvig. Artificial Intelligence. A modern approach. Second edition. 2006. С. 75-440, 863-1010.

2. Mastercard [Електронний ресурс]. Режим доступу: https://www.mastercard.com/news/perspectives/2021/why-mastercard-is-bringing-crypto-onto-our-network/

3. Ресурс статистики світу блокчейну [Електронний ресурс]. Режим доступу: https://buybitcoinworldwide.com/blockchain-statistics/

4. Tully S. A new report highlights the incredibly high environmental cost of Tesla's Bitcoin investment. Fortune [Електронний ресурс]. Режим доступу: https://fortune.com/2021/06/18/a-new-report-highlights-the-incredibly-high-environmental-cost-of-teslas-bitcoin-investment/

5. Ethereum платформа [Електронний ресурс]. Режим доступу: https://ethereum.org

6. European Environment Agency [Електронний ресурс]. Режим доступу: https://www.eea.europa.eu/publications/blockchain-and-the-environment

7. Blockchain | Binance Academy [Електронний ресурс]. Режим доступу: https://academy.binance.com/en/glossary/blockchain?utm_campaign=googleadsxacademy&utm_source=googleadwords_int&utm_medium=cpc&ref=HDYAHEES&gclid=Cj0KCQiA7bucBhCeARIsAIOwr-9tIUaGyP-PHNHdzkFq1fFcuYq8zVMQa-33cik4v-dMml3sIshRUmQaAshOEALw_wcB

8. Т.А. Гаврилова, В.Ф. Хорошевський. Бази знань інтелектуальних систем, 2000. С.10-15.

9. Построение распределенных интеллектуальных информационных систем [Електронний ресурс]. Режим доступу: https://studfile.net/preview/6369046

10. Jörg P. Müller, Michael J. Wooldridge, Nicholas R. Jennings. Intelligent Agents III. Agent Theories, Architectures, and Languages, 1996, Proceedings. С. 103-128.

11. Onn Shehory, Arnon Sturm. Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks, 2014. С. 21-73.

12. Rosenberg E. What Is a Consensus Mechanism?. The Balance [Електронний ресурс]. Режим доступу: https://www.thebalancemoney.com/what-is-a-consensus-mechanism-5211399

13. Ethereum private network manager [Електронний ресурс]. Режим доступу: https://github.com/puppeth

14. The command line utility for managing the lifecycle of virtual machines [Електронний ресурс]. Режим доступу: https://www.vagrantup.com/

15. A popular general-purpose scripting language [Електронний ресурс]. Режим доступу: https://www.php.net/

ДОДАТОК А. ЛІСТИНГ ПРОГРАМНОГО КОДУ

```
class Homestead

 def Homestead.configure(config, settings)

   # Set The VM Provider

   ENV['VAGRANT_DEFAULT_PROVIDER'] = settings["provider"] ||= "virtualbox"


   # Configure Local Variable To Access Scripts From Remote Location

   scriptDir = File.dirname(__FILE__)


   # Prevent TTY Errors

   config.ssh.shell = "bash -c 'BASH_ENV=/etc/profile exec bash'"


   # Allow SSH Agent Forward from The Box

   config.ssh.forward_agent = true


   # Configure The Box

   config.vm.box = settings["box"] ||= "laravel/homestead"

   config.vm.box_version = settings["version"] ||= ">= 1.0.0"

   config.vm.hostname = settings["hostname"] ||= "homestead"


   # Configure A Private Network IP

   config.vm.network :private_network, ip: settings["ip"] ||= "192.168.10.10"
```

```
  # Configure Additional Networks
  if settings.has_key?("networks")
    settings["networks"].each do |network|
      config.vm.network network["type"], ip: network["ip"], bridge: network["bridge"] ||= nil
    end
  end

  # Configure A Few VirtualBox Settings
  config.vm.provider "virtualbox" do |vb|
    vb.customize ["modifyvm", :id, "--memory", settings["memory"] ||= "2048"]
    vb.customize ["modifyvm", :id, "--cpus", settings["cpus"] ||= "1"]
    vb.customize ["modifyvm", :id, "--natdnsproxy1", "on"]
    vb.customize ["modifyvm", :id, "--natdnshostresolver1", settings["natdnshostresolver"] ||=
"on"]
    vb.customize ["modifyvm", :id, "--ostype", "Ubuntu_64"]
    vb.customize ['modifyvm', :id, '--cableconnected1', 'on']
    if settings.has_key?("gui") && settings["gui"]
        vb.gui = true
    end
  end

  # Configure A Few VMware Settings
  ["vmware_fusion", "vmware_workstation"].each do |vmware|
    config.vm.provider vmware do |v|
```

```
    v.vmx["memsize"] = settings["memory"] ||= 2048
    v.vmx["numvcpus"] = settings["cpus"] ||= 1
    v.vmx["guestOS"] = "ubuntu-64"
    if settings.has_key?("gui") && settings["gui"]
        v.gui = true
    end
  end
end

# Configure A Few Parallels Settings
config.vm.provider "parallels" do |v|
  v.name = settings["name"] ||= "homestead-7"
  v.update_guest_tools = true
  v.memory = settings["memory"] ||= 2048
  v.cpus = settings["cpus"] ||= 1
end

# Standardize Ports Naming Schema
if (settings.has_key?("ports"))
  settings["ports"].each do |port|
    port["guest"] ||= port["to"]
    port["host"] ||= port["send"]
    port["protocol"] ||= "tcp"
```

```
  end
else
  settings["ports"] = []
end

# Default Port Forwarding
default_ports = {
  80   => 8000,
  443  => 44300,
  3306 => 33060,
  5432 => 54320
}

# Use Default Port Forwarding Unless Overridden
unless settings.has_key?("default_ports") && settings["default_ports"] == false
  default_ports.each do |guest, host|
    unless settings["ports"].any? { |mapping| mapping["guest"] == guest }
      config.vm.network "forwarded_port", guest: guest, host: host, auto_correct: true
    end
  end
end

# Add Custom Ports From Configuration
```

```
  if settings.has_key?("ports")
    settings["ports"].each do |port|
      config.vm.network "forwarded_port", guest: port["guest"], host: port["host"], protocol: port["protocol"], auto_correct: true
    end
  end

  # Configure The Public Key For SSH Access
  if settings.include? 'authorize'
    if File.exists? File.expand_path(settings["authorize"])
      config.vm.provision "shell" do |s|
        s.inline = "echo $1 | grep -xq \"$1\" /home/vagrant/.ssh/authorized_keys || echo \"\n$1\" | tee -a /home/vagrant/.ssh/authorized_keys"
        s.args = [File.read(File.expand_path(settings["authorize"]))]
      end
    end
  end

  # Copy The SSH Private Keys To The Box
  if settings.include? 'keys'
    settings["keys"].each do |key|
      config.vm.provision "shell" do |s|
        s.privileged = false
        s.inline = "echo \"$1\" > /home/vagrant/.ssh/$2 && chmod 600 /home/vagrant/.ssh/$2"
```

```
      s.args = [File.read(File.expand_path(key)), key.split('/').last]
    end
  end
end

# Copy User Files Over to VM
if settings.include? 'copy'
  settings["copy"].each do |file|
    config.vm.provision "file" do |f|
      f.source = File.expand_path(file["from"])
      f.destination = file["to"].chomp('/') + "/" + file["from"].split('/').last
    end
  end
end

# Register All Of The Configured Shared Folders
if settings.include? 'folders'
  settings["folders"].each do |folder|
    mount_opts = []

    if (folder["type"] == "nfs")
        mount_opts = folder["mount_options"] ? folder["mount_options"] : ['actimeo=1', 'nolock']
    elsif (folder["type"] == "smb")
```

```
      mount_opts = folder["mount_options"] ? folder["mount_options"] : ['vers=3.02', 'mfsymlinks']
    elsif (folder["type"] == "sshfs")
      mount_opts = folder["mount_options"] ? folder["mount_options"] : ['nonempty']
    end

    # For b/w compatibility keep separate 'mount_opts', but merge with options
    options = (folder["options"] || {}).merge({ mount_options: mount_opts })

    # Double-splat (**) operator only works with symbol keys, so convert
    options.keys.each{|k| options[k.to_sym] = options.delete(k) }

    config.vm.synced_folder folder["map"], folder["to"], type: folder["type"] ||= nil, **options

    # Bindfs support to fix shared folder (NFS) permission issue on Mac
    if (folder["type"] == "nfs" && Vagrant.has_plugin?("vagrant-bindfs"))
      config.vm.synced_folder folder["map"], "/mnt/vagrant", id: "vagrant", type: 'nfs'
      config.bindfs.bind_folder "/mnt/vagrant", folder["to"], owner: "vagrant", group: "vagrant", perms: "u=rwX:g=rwX:o=rD", 'create-as-user': true, 'create-with-perms': "u=rwX:g=rwX:o=rD", 'chown-ignore': true, 'chgrp-ignore': true, 'chmod-ignore': true, 'o': "nonempty"
    end
  end
end
```

```
# Install All The Configured Nginx Sites
config.vm.provision "shell" do |s|
    s.path = scriptDir + "/clear-nginx.sh"
end



if settings.include? 'sites'
  settings["sites"].each do |site|
    type = site["type"] ||= "laravel"

    if (site.has_key?("hhvm") && site["hhvm"])
      type = "hhvm"
    end

    if (type == "symfony")
      type = "symfony2"
    end

    if (type == "symfony-sulu")
        type = "sulu"
    end

    config.vm.provision "shell" do |s|
```

```
    s.name = "Creating Site: " + site["map"]
    s.path = scriptDir + "/serve-#{type}.sh"
    s.args = [site["map"], site["to"], site["port"] ||= "80", site["ssl"] ||= "443"]
  end

  # Configure The Cron Schedule
  if (site.has_key?("schedule"))
    config.vm.provision "shell" do |s|
      s.name = "Creating Schedule"

      if (site["schedule"])
        s.path = scriptDir + "/cron-schedule.sh"
        s.args = [site["map"].tr('^A-Za-z0-9', ''), site["to"]]
      else
        s.inline = "rm -f /etc/cron.d/$1"
        s.args = [site["map"].tr('^A-Za-z0-9', '')]
      end
    end
  end

end
```

```
config.vm.provision "shell" do |s|
  s.name = "Restarting Nginx"
  s.inline = "sudo service nginx restart; sudo service php7.2-fpm restart"
end

# Install MariaDB If Necessary
if settings.has_key?("mariadb") && settings["mariadb"]
  config.vm.provision "shell" do |s|
    s.path = scriptDir + "/install-maria.sh"
  end
end



# Configure All Of The Configured Databases
if settings.has_key?("databases")
    settings["databases"].each do |db|
      config.vm.provision "shell" do |s|
        s.name = "Creating MySQL Database: " + db
        s.path = scriptDir + "/create-mysql.sh"
        s.args = [db]
      end

      config.vm.provision "shell" do |s|
```

```
        s.name = "Creating Postgres Database: " + db
        s.path = scriptDir + "/create-postgres.sh"
        s.args = [db]
      end
    end
  end

  # Configure All Of The Server Environment Variables
  config.vm.provision "shell" do |s|
    s.name = "Clear Variables"
    s.path = scriptDir + "/clear-variables.sh"
  end

  if settings.has_key?("variables")
    settings["variables"].each do |var|
      config.vm.provision "shell" do |s|
        s.inline = "echo \"\nenv[$1] = '$2'\" >> /etc/php/7.2/fpm/php-fpm.conf"
        s.args = [var["key"], var["value"]]
      end

      config.vm.provision "shell" do |s|
        s.inline = "echo \"\n# Set Homestead Environment Variable\nexport $1=$2\" >>
/home/vagrant/.profile"
        s.args = [var["key"], var["value"]]
```

```
    end
  end

  config.vm.provision "shell" do |s|
    s.inline = "service php7.2-fpm restart"
  end
end

# Update Composer On Every Provision
config.vm.provision "shell" do |s|
  s.name = "Update Composer"
  s.inline = "sudo /usr/local/bin/composer self-update && sudo chown -R vagrant:vagrant /home/vagrant/.composer/"
  s.privileged = false
end

# Configure Blackfire.io
if settings.has_key?("blackfire")
  config.vm.provision "shell" do |s|
    s.path = scriptDir + "/blackfire.sh"
    s.args = [
      settings["blackfire"][0]["id"],
      settings["blackfire"][0]["token"],
      settings["blackfire"][0]["client-id"],
```

```
        settings["blackfire"][0]["client-token"]
      ]
    end
  end
end
end



<?php

namespace Laravel\Homestead;

use Symfony\Component\Console\Input\InputArgument;
use Symfony\Component\Console\Command\Command;
use Symfony\Component\Console\Input\InputInterface;
use Symfony\Component\Console\Output\OutputInterface;
use Symfony\Component\Yaml\Yaml;

class AddDatabaseCommand extends Command
{
  /**
   * @var array
   */
```

```
private $homesteadYamlContents;


/**
 * @var OutputInterface
 */
private $output;


/**
 * Configure the command options.
 *
 * @return void
 */
protected function configure()
{
    $this
        ->setName('add-db')
        ->setDescription('Add a new database to the Homestead.yaml file')
        ->addArgument('name', InputArgument::REQUIRED, 'the database name to be added');
}


/**
 * Execute the command.
 *
```

```
     * @param  \Symfony\Component\Console\Input\InputInterface  $input
     * @param  \Symfony\Component\Console\Output\OutputInterface  $output
     *
     * @return void
     */
    public function execute(InputInterface $input, OutputInterface $output)
    {
        $this->homesteadYamlContents = Yaml::parse(file_get_contents('./Homestead.yaml'));
        $this->output = $output;


        $name = $input->getArgument('name');


        $canAddItem = $this->canItemBeAdded($name);


        if($canAddItem) {
            $this->addItemAndSaveToYamlFile($name);
        }
    }


    /**
     * @param $name
     *
     * @return bool
```

```
 */
private function canItemBeAdded($name)
{
    $canAddItem = true;
    foreach($this->homesteadYamlContents['databases'] as $database) {
        if($database === $name) {
            $this->output->writeln(sprintf('<error>Database %s already defined.</error>', $name));
            $canAddItem = false;
            break;
        }
    }

    return $canAddItem;
}

/**
 * @param $name
 */
private function addItemAndSaveToYamlFile($name)
{
    $this->homesteadYamlContents['databases'][] = $name;

    file_put_contents('./Homestead.yaml', Yaml::dump($this->homesteadYamlContents));
```

```
        $this->output->writeln('<info>New database successfully added.</info>');
        $this->output->write('<info>Don\'t forget to re-provision your VM.</info>');
    }
}



<?php

namespace Laravel\Homestead;

use Symfony\Component\Console\Command\Command;
use Symfony\Component\Console\Input\InputArgument;
use Symfony\Component\Console\Input\InputInterface;
use Symfony\Component\Console\Output\OutputInterface;

class RunCommand extends Command
{
    /**
     * Configure the command options.
     *
     * @return void
     */
```

```
    protected function configure()
    {
        $this
            ->setName('run')
            ->setDescription('Run commands through the Homestead machine via SSH')
            ->addArgument('ssh-command', InputArgument::REQUIRED, 'The command to pass through to the virtual machine.');
    }

    /**
     * Execute the command.
     *
     * @param  \Symfony\Component\Console\Input\InputInterface  $input
     * @param  \Symfony\Component\Console\Output\OutputInterface  $output
     * @return void
     */
    public function execute(InputInterface $input, OutputInterface $output)
    {
        chdir(__DIR__.'/../');

        $command = $input->getArgument('ssh-command');

        passthru($this->setEnvironmentCommand().' vagrant ssh -c "'.$command.'"');
    }
```

```
  protected function setEnvironmentCommand()
  {
      if ($this->isWindows()) {
          return 'SET VAGRANT_DOTFILE_PATH='.$_ENV['VAGRANT_DOTFILE_PATH'].'
&&';
      }

      return 'VAGRANT_DOTFILE_PATH="'.$_ENV['VAGRANT_DOTFILE_PATH']."";
  }

  protected function isWindows()
  {
      return strpos(strtoupper(PHP_OS), 'WIN') === 0;
  }
}

<?php

namespace Laravel\Homestead;

use Symfony\Component\Process\Process;
use Symfony\Component\Console\Command\Command;
use Symfony\Component\Console\Input\InputOption;
```

```
use Symfony\Component\Console\Input\InputInterface;
use Symfony\Component\Console\Output\OutputInterface;

class UpCommand extends Command
{
  /**
   * Configure the command options.
   *
   * @return void
   */
  protected function configure()
  {
     $this
         ->setName('up')
         ->setDescription('Start the Homestead machine')
         ->addOption('provision', null, InputOption::VALUE_NONE, 'Run the provisioners on the
box.');
  }

  /**
   * Execute the command.
   *
   * @param  \Symfony\Component\Console\Input\InputInterface  $input
   * @param  \Symfony\Component\Console\Output\OutputInterface  $output
```

```
     * @return void
     */
    public function execute(InputInterface $input, OutputInterface $output)
    {
        $command = 'vagrant up';

        if ($input->getOption('provision')) {
            $command .= ' --provision';
        }

        $process = new Process($command, realpath(__DIR__.'/../'), array_merge($_SERVER, $_ENV), null, null);

        $process->run(function ($type, $line) use ($output) {
            $output->write($line);
        });
    }
}



require 'json'
require 'yaml'

VAGRANTFILE_API_VERSION = "2"
```

```
confDir = $confDir ||= File.expand_path(".")

homesteadYamlPath = confDir + "/Homestead.yaml"
homesteadJsonPath = confDir + "/Homestead.json"
afterScriptPath = confDir + "/after.sh"
aliasesPath = confDir + "/aliases"

require File.expand_path(File.dirname(__FILE__) + '/scripts/homestead.rb')

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  if File.exists? aliasesPath then
    config.vm.provision "file", source: aliasesPath, destination: "~/.bash_aliases"
  end

  if File.exists? homesteadYamlPath then
    Homestead.configure(config, YAML::load(File.read(homesteadYamlPath)))
  elsif File.exists? homesteadJsonPath then
    Homestead.configure(config, JSON.parse(File.read(homesteadJsonPath)))
  end

  if File.exists? afterScriptPath then
    config.vm.provision "shell", path: afterScriptPath
  end
```

```
end



---

ip: "192.168.10.10"

memory: 2048

cpus: 1

provider: virtualbox


folders:

  - map: .

    to: /home/vagrant/Code


sites:

  - map: homestead.test

    to: /home/vagrant/Code/Project/public


databases:

  - homestead


#variables:

#    - key: APP_ENV

#      value: local
```

```
# blackfire:
#     - id: foo
#       token: bar
#       client-id: foo
#       client-token: bar

ports:
   - send: 8545
     to: 8545
   - send: 30301
     to: 30301
   - send: 30302
     to: 30302
   - send: 30303
     to: 30303
   - send: 30304
     to: 30304
   - send: 30305
     to: 30305
   - send: 30306
     to: 30306
```

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure("2") do |config|
 # The most common configuration options are documented and commented below.
 # For a complete reference, please see the online documentation at
 # https://docs.vagrantup.com.

 # Every Vagrant development environment requires a box. You can search for
 # boxes at https://vagrantcloud.com/search.
 config.vm.box = "base"

 # Disable automatic box update checking. If you disable this, then
 # boxes will only be checked for updates when the user runs
 # `vagrant box outdated`. This is not recommended.
 # config.vm.box_check_update = false

 # Create a forwarded port mapping which allows access to a specific port
 # within the machine from a port on the host machine. In the example below,
```

```
# accessing "localhost:8080" will access port 80 on the guest machine.
# NOTE: This will enable public access to the opened port
# config.vm.network "forwarded_port", guest: 80, host: 8080

# Create a forwarded port mapping which allows access to a specific port
# within the machine from a port on the host machine and only allow access
# via 127.0.0.1 to disable public access
# config.vm.network "forwarded_port", guest: 80, host: 8080, host_ip: "127.0.0.1"

# Create a private network, which allows host-only access to the machine
# using a specific IP.
# config.vm.network "private_network", ip: "192.168.33.10"

# Create a public network, which generally matched to bridged network.
# Bridged networks make the machine appear as another physical device on
# your network.
# config.vm.network "public_network"

# Share an additional folder to the guest VM. The first argument is
# the path on the host to the actual folder. The second argument is
# the path on the guest to mount the folder. And the optional third
# argument is a set of non-required options.
# config.vm.synced_folder "../data", "/vagrant_data"
```

```
# Provider-specific configuration so you can fine-tune various
# backing providers for Vagrant. These expose provider-specific options.
# Example for VirtualBox:
#
# config.vm.provider "virtualbox" do |vb|
#   # Display the VirtualBox GUI when booting the machine
#   vb.gui = true
#
#   # Customize the amount of memory on the VM:
#   vb.memory = "1024"
# end
#
# View the documentation for the provider you are using for more
# information on available options.


# Enable provisioning with a shell script. Additional provisioners such as
# Ansible, Chef, Docker, Puppet and Salt are also available. Please see the
# documentation for more information about their specific syntax and use.
# config.vm.provision "shell", inline: <<-SHELL
#   apt-get update
#   apt-get install -y apache2
# SHELL
```

```
end



#!/usr/bin/env bash


mkdir /etc/nginx/ssl 2>/dev/null

openssl genrsa -out "/etc/nginx/ssl/$1.key" 1024 2>/dev/null

openssl req -new -key /etc/nginx/ssl/$1.key -out /etc/nginx/ssl/$1.csr -subj
"/CN=$1/O=Vagrant/C=UK" 2>/dev/null

openssl x509 -req -days 365 -in /etc/nginx/ssl/$1.csr -signkey /etc/nginx/ssl/$1.key -out
/etc/nginx/ssl/$1.crt 2>/dev/null


block="server {

  listen ${3:-80};

  listen ${4:-443} ssl;

  server_name $1;

  root \"$2\";


  index index.html index.htm index.php;


  charset utf-8;


  location / {

      try_files \$uri \$uri/ /index.php?\$query_string;
```

```
}

location = /favicon.ico { access_log off; log_not_found off; }
location = /robots.txt  { access_log off; log_not_found off; }

access_log off;
error_log  /var/log/nginx/$1-error.log error;

sendfile off;

client_max_body_size 100m;

location ~ \.php$ {
    fastcgi_split_path_info ^(.+\.php)(/.+)$;
    fastcgi_pass unix:/var/run/php/php7.2-fpm.sock;
    fastcgi_index index.php;
    include fastcgi_params;
    fastcgi_param SCRIPT_FILENAME \$document_root\$fastcgi_script_name;
    fastcgi_intercept_errors off;
    fastcgi_buffer_size 16k;
    fastcgi_buffers 4 16k;
    fastcgi_connect_timeout 300;
    fastcgi_send_timeout 300;
```

```
        fastcgi_read_timeout 300;
    }

    location ~ /\.ht {
        deny all;
    }

    ssl_certificate     /etc/nginx/ssl/$1.crt;
    ssl_certificate_key /etc/nginx/ssl/$1.key;
}
"

echo "$block" > "/etc/nginx/sites-available/$1"
ln -fs "/etc/nginx/sites-available/$1" "/etc/nginx/sites-enabled/$1"
service nginx restart
service php7.2-fpm restart


#!/usr/bin/env php
<?php

$_ENV['HOME'] = getenv('HOME');
$_ENV['VAGRANT_DOTFILE_PATH'] =
homestead_path().DIRECTORY_SEPARATOR.'.vagrant';
```

```
if (file_exists(__DIR__.'/vendor/autoload.php')) {
  require __DIR__.'/vendor/autoload.php';
} else {
  require __DIR__.'/../../autoload.php';
}

function homestead_path()
{
  if (isset($_SERVER['HOME'])) {
     return $_SERVER['HOME'].'/.homestead';
  } else {
     return
$_SERVER['HOMEDRIVE'].$_SERVER['HOMEPATH'].DIRECTORY_SEPARATOR.'.homest
ead';
  }
}

$app = new Symfony\Component\Console\Application('Laravel Homestead', '2.1.7');

$app->add(new Laravel\Homestead\DestroyCommand);
$app->add(new Laravel\Homestead\EditCommand);
$app->add(new Laravel\Homestead\HaltCommand);
$app->add(new Laravel\Homestead\InitCommand);
```

```
$app->add(new Laravel\Homestead\MakeCommand);

$app->add(new Laravel\Homestead\ProvisionCommand);

$app->add(new Laravel\Homestead\ResumeCommand);

$app->add(new Laravel\Homestead\RunCommand);

$app->add(new Laravel\Homestead\UpCommand);

$app->add(new Laravel\Homestead\UpdateCommand);

$app->add(new Laravel\Homestead\SshCommand);

$app->add(new Laravel\Homestead\SshConfigCommand);

$app->add(new Laravel\Homestead\StatusCommand);

$app->add(new Laravel\Homestead\SuspendCommand);

$app->add(new Laravel\Homestead\AddSiteCommand);

$app->add(new Laravel\Homestead\AddDatabaseCommand);


$app->run();
```