**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Західноукраїнський національний університет**
**Факультет комп'ютерних інформаційних технологій**
Кафедра комп'ютерних наук

**БАРІДА Сергій Іванович**

# Інтелектуалізована програмна система обігу медичних рецептів на основі Blockchain / Intellectualized Blockchain-Based Software System for the Medical Prescriptions Circulation

спеціальність: 121 - Інженерія програмного забезпечення
освітньо-професійна програма - Інженерія програмного забезпечення

Кваліфікаційна робота

Виконав студент групи ІПЗм-21
С. І. Баріда

_____

Науковий керівник:
к.т.н., доцент, В. І. Манжула

_____

Кваліфікаційну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

**ТЕРНОПІЛЬ - 2022**

ГЛОСАРІЙ

1. *Технологія розподілених реєстрів* – система розподіленого зберігання та одночасної обробки та оновлення інформації на різних носіях у всіх учасників, яка дозволяє здійснювати обмін/зберігання практично будь-який інформації [1].

2. *Блокчейн* – спосіб реалізації мережі розподілених реєстрів, у якому дані про здійснені транзакції структуруються як ланцюг (послідовність) пов'язаних блоків транзакцій [2].

3. *Цифровий токен* – криптографічно захищене підтвердження прав його власника на отримання обіцяних йому цінностей або можливості виконання з його допомогою заздалегідь певних функцій.

4. *Криптовалюта* – децентралізована віртуальна валюта, заснована на математичних алгоритмах і захищена методами криптографії [3].

5. *Ethereum* – криптовалюта та платформа для створення децентралізованих онлайн-сервісів на базі блокчейну (децентралізованих додатків), що працюють на базі розумних контрактів. Реалізована як єдина децентралізована віртуальна машина [4].

6. *NFT* – невзаємозамінний токен, підвид цифрових токенів у мережі Ethereum. На відміну від інших токенів, кожен випущений NFT є унікальним [5].

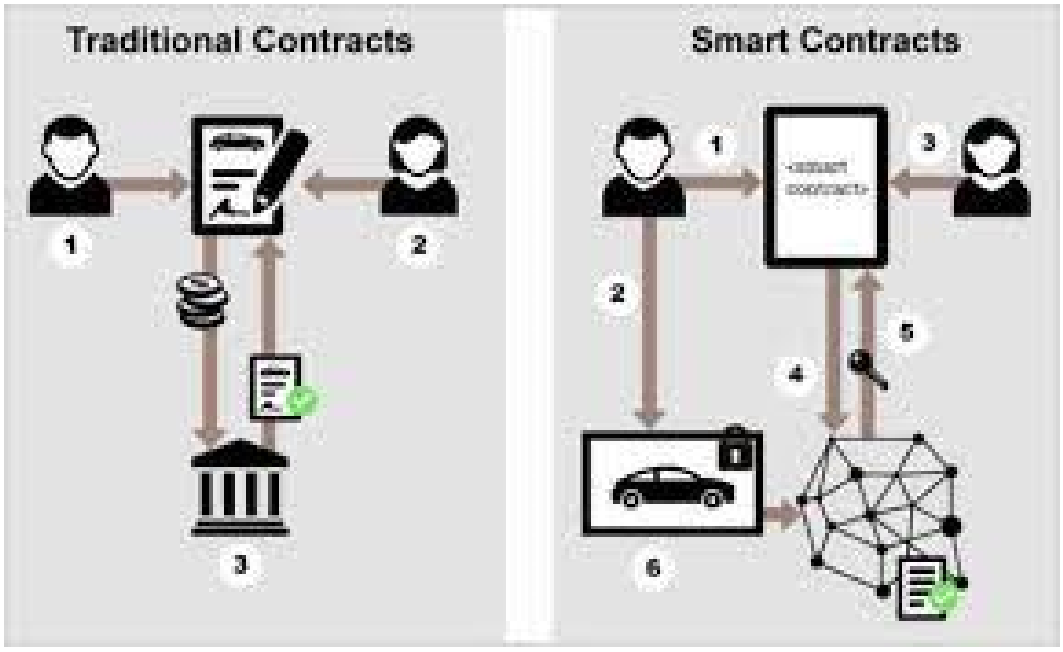
ЗМІСТ

# ВСТУП

**Актуальність**

Ритм життя сьогодні постійно прискорюється. Договори, угоди та контракти раніше укладалися в письмовій формі та вимагали присутності людей, що укладають договір. Укладаючи угоди один з одним, люди та організації приймали на себе зобов'язання виконати роботи, надати послуги, передати товари чи кошти. Однак нерідко прийняті за договором зобов'язання порушувалися. Потерпіла сторона для відновлення справедливостізверталася до суду та намагалася домогтися відновлення порушених прав. Навіть у разі відновлення порушених прав потерпілої сторони судом на це йшло багато часу, сил та грошей.

Смарт-контракт називається особлива програма в блокчейн-мережі, виконувана усіма вузлами і яка допомагає власникам криптовалют між собою взаємодіяти. Всі умови та положення цих контрактів записуються в блоковому ланцюжку. Ніхто не може обдурити, зламати чи підкупити користувача, порушивши тим самим умови смарт-контракт.




Сторони підписують розумний контракт, використовуючи аналогічні підписання відправки коштів у діючих криптовалютних мережах методи. Після

підписання сторонами контракт вступає в силу.

Актуальність використання смарт-контрактів укладається в тому, що, маючи безперешкодний доступ до об'єктів контракту, розумний контракт відстежує за виззначеними умовами досягнення або порушення пунктів і приймає самостійні рішення, ґрунтуючись на запрограмованих умовах.

Таким чином, основний принцип розумного контракту полягає у повній автоматизації та достовірності виконання договірних відносин між людьми. Розумні контракти, засновані на криптографії, здатні забезпечувати кращу безпеку, ніж традиційні контракти, що ґрунтуються на праві, та знизити інші транзакційні витрати, пов'язані з укладанням договорів та можливих судових витрат.

Смарт-контракт також дозволяє знизити транскордонні витрати під час укладання договорів, оскільки вони замінюють роботу посередників з платними послугами.

**Постановка завдання**

Метою випускний кваліфікаційної роботи є розробка веб-додатку для видачі рецептурних препаратів на основі медичних рецептів з використанням технології блокчейн.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1) провести огляд наукової літератури і аналогів;
2) вибрати блокчейн для створення смарт-контракту;
3) розробити смарт-контракт, яки здійснює випуск NFT;
4) розробити веб-додаток, що здійснює видачу медичних рецептів лікарем та на основі цих рецептів видачі препаратів аптеками з використанням технології блокчейн;
5) провести тестування реалізованого смарт-контракту та веб-систему.

У роботі здійснено аналіз наукової літератури на тему створення смарт-контрактів і рішення питань безпеки транзакцій, історія створення та приклади використання смарт-контрактів, а також розглянуто можливість використання NFT як носія медичного рецепту. Були вивчені аналоги в вигляді інших носіїв

медичних рецептів, і навіть схожі з продажу послуги, розглянуто вибір блокчейна. Розглянуто аспекти, пов'язані з механізмом роботи смарт-контрактів, EVM, відправкою транзакцій, викликом методу контракту, отриманням балансу.

Визначено функціональні і нефункціональні вимоги до системи, складені діаграми діяльності і варіантів використання.

Також описано проектування та реалізацію веб-системи для випуску медичних рецептів за допомогою технології блокчейн. Наведено зовнішній вигляд реалізованої веб-системи, описано взаємодію системи зі смарт-контрактом ти з користувачами. Проведено тестування реалізованої програмної системи.

# РОЗДІЛ 1
# АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТИ

## 1.1. Історія створення та використання смарт-контрактів

Вперше ідея смарт-контракту була запропонована 1994 р. Ніком Сабо (США) – вченим у сфері інформатики, криптографії та права. Він описав смарт-контракт як «цифрове представлення набору зобов'язань між сторонами, що включає протокол виконання цих зобов'язань».

Таким чином, смарт-контракт може бути визначений як договір між двома і більше сторонами про встановлення, зміну або припинення юридичних прав та обов'язків, у якому частина чи всі умови записуються, виконуються та/або забезпечуються комп'ютерним алгоритмом автоматично в спеціалізованою програмною середовище.

Незважаючи на те, що в надалі ідея смарт-контракту отримала широке поширення на хвилі зростання популярності криптовалют, смарт-контракти не обов'язково повинні бути пов'язані з технологією розподілених реєстрів, цифровими валютами або відсутністю посередника.

Питанням створення смарт-контрактів приділяється велика увага наукової літератури. У зокрема, в роботах [5, 6] докладно розглянуті питання функціональної моделі смарт-контракту на платформі Ethereum.

Найпоширеніша платформа для розгортання смарт-контрактів Ethereum. Проте зі швидким розвитком смарт-контрактів кількість атак також зростає. У роботах [7, 8] проаналізовано вразливості смарт-контрактів блокчейн-платформи Ethereum.

Тенденції розвитку смарт-контрактів, сучасні технології, механізми і основні платформи смарт-контрактів з підтримкою блокчейна, а також існуючі в даний час технічні та юридичні проблеми докладно описані в роботі [9].

Еволюція розробки децентралізованого та автономного рішення завдань з допомогою смарт-контрактів на основі блокчейна проаналізовано в роботі [10]. У документі також наведено дослідження досвіду організацій, які отримали вигоду від застосування смарт-контрактів.

У 1998 році цей термін був використаний для опису об’єктів на рівні служби управління правами системи The Stanford Infobus, яка була частиною Стенфордського проекту цифрової бібліотеки.

У 2014 році Віталій Бутерін, співзасновник платформи Ethereum, запропонував власну ідею щодо вдосконалення мережі Біткоїн. І вже за рік був запущений сам Ethereum, який є платформою для впровадження цих самих удосконалень і дозволяє створювати автономні контракти.

У якийсь момент Біткоїн також почав передавати більше інформації за допомогою протокола Script, але він значно відстає від смарт-контрактів, що є «повними за Тюрінгом» (тобто діють, як універсальний комп’ютер). Ethereum дозволив розробникам впроваджувати нові алгоритми та децентралізовані застосунки поверх блокчейну. Саме тому корпоративні контракти на блокчейні є наступним кроком у цифровій революції для бізнесу.

Динаміка використання смарт-контрактів у мережах Ethereum наведено на рисунку 1.1

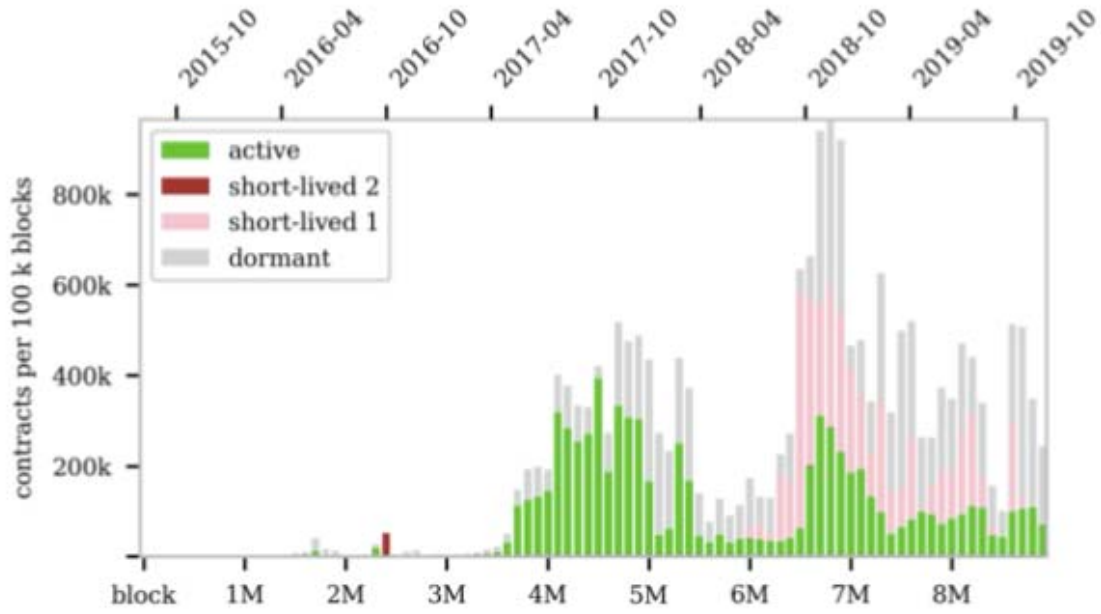



Рис. 1.1. Розгортання активних, короткочасних та неактивних контрактів у мережі Ethereum упродовж всього часу

Як це не дивно звучить сьогодні, першою країною в світі, яка

законодавчо закріпила смарт-контракти, стала Республіка Білорусь. В 2017 Республіка Білорусь видала Указ про розвиток цифрової економіки, ставши першою країною, яка легалізувала smart contracts. Білоруський юрист Денис Алєйніков вважається автором юридичного концепту розумних контрактів, представлених в цьому указі. У Декреті від 21 грудня 2017 року № 8 «Про розвиток цифрової економіки» було узаконено біржі криптовалют, оператори обміну криптовалют, майнінг, смарт-контракт, блокчейн, токени тощо. Відповідно до зазначеного Декрету під смарт-контрактом розуміється програмний код, призначений для функціонування в реєстрі блоків транзакцій (блокчейні) в цілях автоматизованого вчинення та (або) виконання угод або вчинення інших юридично значущих дій. Виходячи з буквального тлумачення визначення смарт-контракту, викладеного в Декреті, можна зробити висновок про те, що білоруський законодавець пішов шляхом віднесення смарт-контракту до програмного коду, котрий допомагає виконувати умови класичного договору. Таким чином, це лише певна умова про особливості виконання договору, але не різновид такого.

В 2018, Сенат США зазначив у своєму звіті: «Попри те, що смарт-контракти можуть звучати по-новому, концепція вкорінена в основному договірному праві. Зазвичай судова система вирішує договірні суперечки та забезпечує виконання умов, але також часто існує інший метод арбітражу, особливо щодо міжнародних операцій. За допомогою смарт-контрактів програма застосовує контракт, вбудований у код».

Ряд американських штатів прийняли законодавство щодо використання розумних контрактів, такі як Аризона, Невада, Теннессі та Вайомінг. Концепція смарт-контракту знайшла своє відображення і в законодавстві штату Арізона (США). По інший бік океану смарт-контракт визначений як комп'ютерна програма, що запускає свою дію при настанні певних умов, яка заснована на розподіленому децентралізованому колективному та відтворюваному реєстрі даних та може служити для зберігання та виконання інструкцій щодо передачі майнової цінності в такому реєстрі. Буквальне

тлумачення визначення знову приводить нас до думки про те, що смарт-контракт все ж таки є комп'ютерною програмою, яка лише полегшує виконання класичного договору, але не є самим договором.

Однією з найбільш популярних платформ, які реалізують можливість виконання смарт-контрактів є Ethereum. Ethereum гарантує неможливість зміни кінцевого стану мережі після виконання деякої логіки смарт-контракту і його виконання буде відповідати саме тій логіці, яка була в ньому прописано початково.

Появі ідеї смарт-контракту передувало створення вендінгової кавомашини – процес придбання товару в ній втілює угоду, відповідно до якої будь-хто може придбати продукт. за наперед визначеною ціною, при цьому механізми безпеки автомата влаштовані таким чином, що вартість злому вища за вартість продуктів і накопичених автоматом готівки коштів.

Сучасним прикладом ідеї смарт-контракту можна, можливо назвати формат роботи компанії Uber. Агрегатори відіграють роль посередника та арбітра, який забезпечує виконання угоди між водієм таксі та клієнтом: клієнт висловлює згоду сплатити поїздку по вартості, заздалегідь визначеної системою-посередником (агрегатором), а водій, в свою черга, зобов'язується виконати послугу по перевезенні клієнтадо заздалегідь певного місця.

З використанням технології розподілених реєстрів виконання смарт-контрактів відбувається автоматично, що дає додаткові можливості для скорочення витрат учасників відносин, виникаючих при висновку угоди і виконанні її умов. Реалізовані через смарт-контракти багатосторонні взаємодії дозволяють зменшити витрати на проведення операцій та контроль за ними, збільшити швидкість виконання операцій та зменшити ризики, пов'язані з несумлінними діями сторін, максимально скоротити або повністю виключити посередників з угоди.

Практика використання смарт-контрактів на сьогоднішній день зводиться в основному до часткової автоматизації окремих аспектів угод, таких як обмін цифровими активами, наприклад, обмін грошових коштів на майнові права.

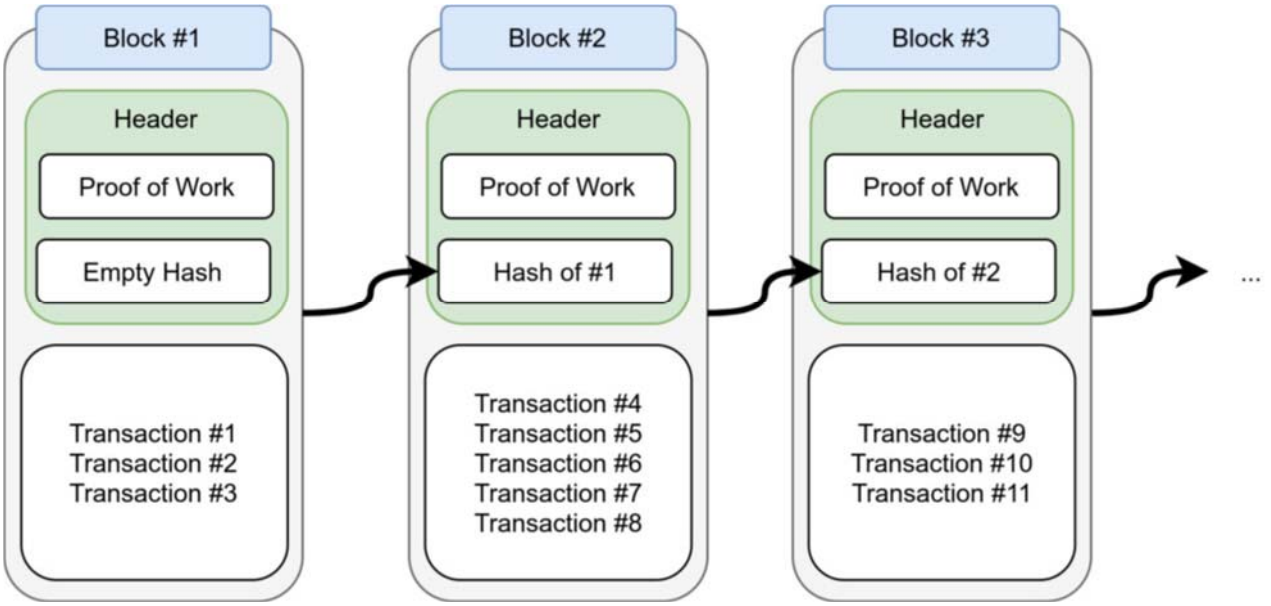
Однак, дуже мабуть, що по мірі розвитку інфраструктури та платформ на основі технології розподілених реєстрів смарт-контракти перестануть бути лише доповненням до паперової версії документа і стануть основним гарантом виконання зобов'язань сторін при укладанні угод, забезпечивши перехід до цифрових контрактів без необхідності їх підтвердження паперовими документами.

Смарт-контракт може оновити дані в блокчейні у відповідності до заданих правил на початку – наприклад, перекласти цифрові активи від одного учасника іншому. Подібні дії можуть бути зроблені частково або повністю самовиконуваними і самодостатніми.

## 1.2. Концепція смарт-контрактів

Смарт-контракти – це комп’ютерні алгоритми, призначені для укладання та підтримки контрактів. Такий контракт пишеться у кодовій формі та виконуються на блокчейні.

Блокчейн є розподіленим реєстром даних і ведеться та керується мережею комп’ютерів. Простими словами, блокчейн дозволяє смарт-контрактам виконувати складні правила та обмінювати активи, уникаючи послуг посередника. Щоб отримати повну картину, звернімось до прикладу на крипто ринку.




Трейдер знаходить покупця для одного зі своїх предметів. Тепер трейдер

та покупець підписують контракт із зашифрованою в ньому логікою.

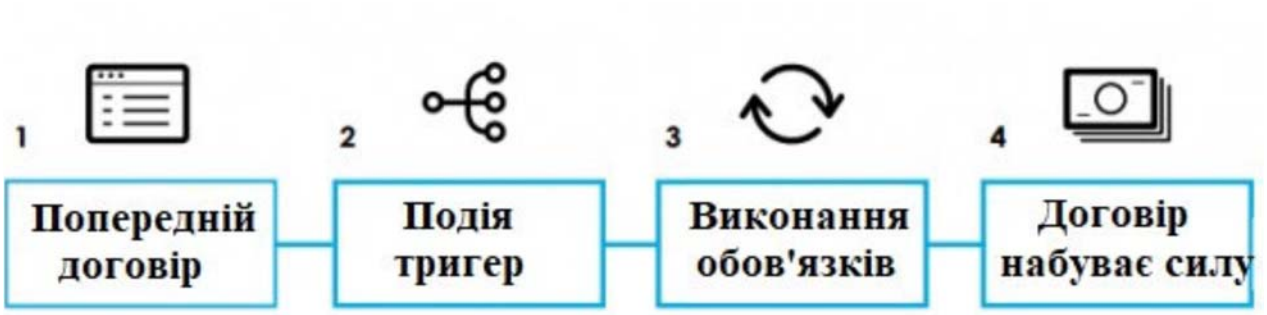
Як тільки покупець отримує свій товар, його платіж автоматично перераховується торговцю. Жодна зі сторін не повинна нічого робити: процес повністю автоматизований за допомогою коду.

Вони просто погоджуються з умовами та ставлять свої цифрові підписи в алгоритмі.

Інші нюанси також можна закодувати в смарт-контракт. Наприклад, якщо товар надійшов пошкодженим, торговець отримує лише частину суми. Отже, якщо стандартний юридичний договір буде включати умови відносин, що забезпечуються законом, контракт на основі блокчейну забезпечує взаємозв’язок між усіма залученими сторонами за допомогою коду та криптографії.

1.2.1. Принцип роботи смарт-контрактів

На відміну від юридичного договору, смарт-контракт (поки) не має юридичної сили. Якщо хтось вирішить змінити зміст контракту, це побачать усі учасники мережі – ноди. Коли смарт-контракт виконує закодовану в ньому операцію (наприклад, переказує платіж), ця інформація оновлюється для кожної сторони. В результаті кожна сторона може стежити за процесом виконання контракту.




Реалізація смарт-контракту

Ноди перевіряють такі контракти як блоки. Якщо смарт-контракт викликає сумнів (у разі зловмисної ноди), він може не потрапити до реєстру.

Смарт-контракти можуть виконувати такі функції:

- керувати домовленостями між користувачами;
- функціонувати як рахунки з декількома підписами (оплата

надсилається лише тоді, коли всі ноди погодилися);

- зберігати інформацію.

Ілюстрація роботи смарт-контракту, як виглядає смарт-контракт в коді, наведено на рисунку 1.2.

```
contract BasicIterator

{

address creator; // reserve one "address"-type spot

uint8[10] integers; // reserve a chunk of storage for 10 8-bit unsigned integers in an array

function BasicIterator()

{

creator = msg.sender;

uint8 x = 0;

//Section 1: Assigning values

while(x < integers.length) {

integers[x] = x;

x++;

} }

function getSum() constant returns (uint) {

uint8 sum = 0;

uint8 x = 0;

//Section 2: Adding the integers in an array.

while(x < integers.length) {
sum = sum + integers[x];

x++;
}

return sum;

}
```

Рис. 1.2. Фрагмент коду контракта

### 1.2.2. Особливості смарт-контрактів

Існує три особливості, що характеризують функціональність розумних контрактів:

- детермінованість;
- обмеженість;

- ізольованість.


Детермінованість

Програму можна назвати детермінованою, якщо вона кожного разу повертає однакові результати на ввідні дані. Наприклад, якщо 2 + 2 = 4, тоді 2 + 2 завжди буде 4.

Обмеженість

З математичної точки зору, ми можемо зіткнутися з «проблемою збою»: неможливістю зрозуміти, чи може певна програма виконувати свою функцію протягом заданого часу. Алан Тюрінг, якого вважають батьком сучасної інформатики, дійшов висновку, що немає шансів дізнатись, чи може дана програма закінчитися за певний час чи ні.

Це серйозна проблема для розумних контрактів, оскільки вони повинні дійти певного результату у визначений термін. Таким чином, існують заходи, що гарантують можливість розірвання договору. Одним із рішень є комісія – в Ефіріумі – це gas. Коли кількість виконаних інструкцій досягає ліміт, контракт вичерпується. Інший приклад – це таймер: контракт розривається, коли досягається обмеження по часу.

Ізольованість

У блокчейні кожен може завантажити смарт-контракт. Однак ці

контракти можуть містити вірус або помилку. Щоб убезпечити весь блокчейн, украй важливо зберігати контракт ізольованим від усієї екосистеми. Існують спеціальні системи, які допомагають виконувати контракти в безпечний спосіб:

- віртуальна машина на Ефіріумі;
- докер, як у Hyperledger Fabric.

Приклад схеми роботи смарт-контракта для страхової компанії наведено на рисунку 1.3.

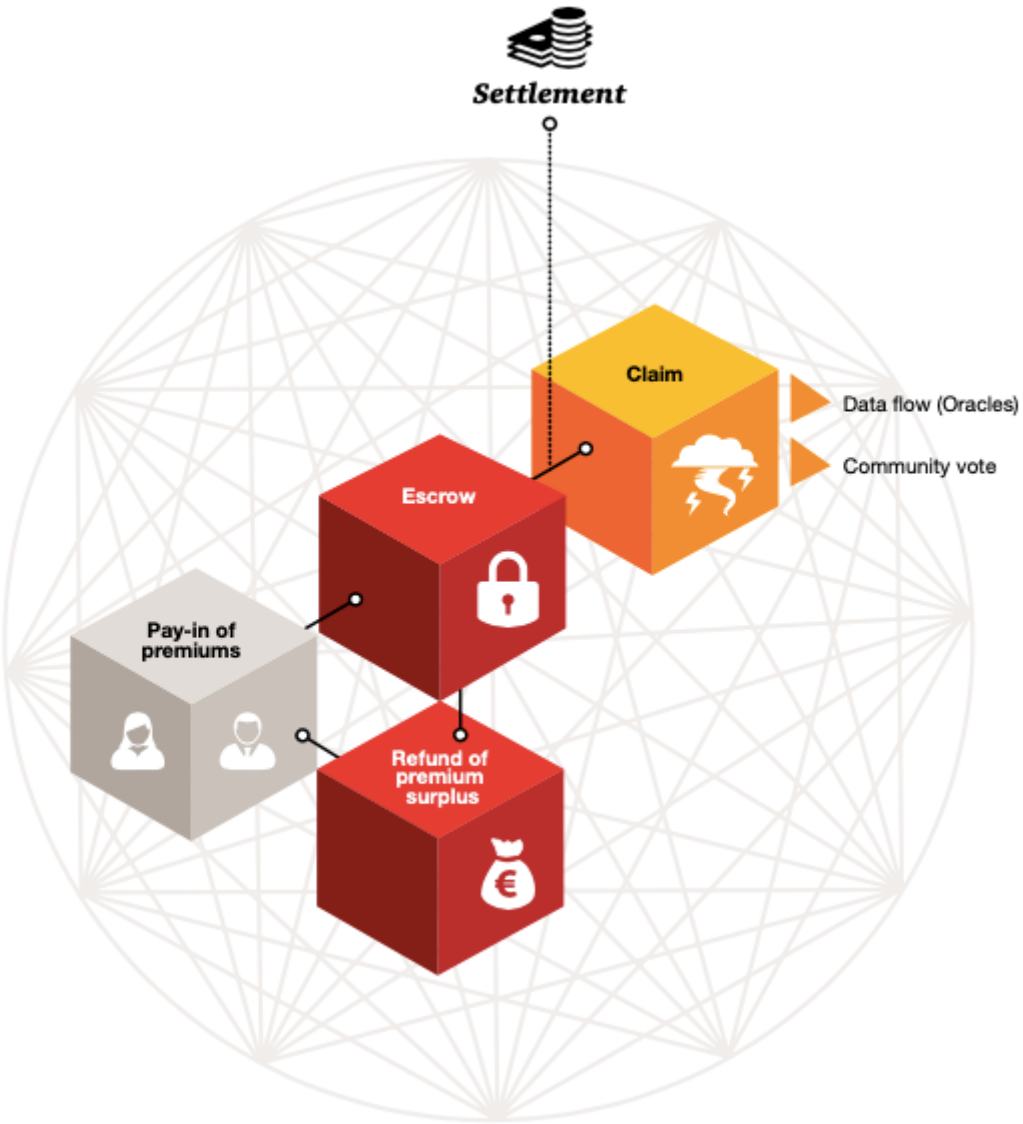



Рис. 1.3. Приклад роботи смарт-контракта для індустрії страхування (звіт PwC)

### 1.2.3. Види смарт-контрактів

До теперішнього часу відсутня загальноприйнята класифікація смарт-

контрактів, але з погляду виконання угод можуть бути виділено наступні види смарт-контрактів:

– контроль майнових відносин – володіння та проведення операцій з цифровими активами, включаючи криптовалюти і токени (Bitcoin,ETH, XRP та інші);

– фінансові сервіси – торгове фінансування, торгівля на біржі, участь в аукціонах та інше;

– кредитні зобов'язання – виконання зобов'язань з різних формам банківських кредитних продуктів в момент настання подій;

– соціальні послуги – процедури проведення голосувань, виборів, процеси страхування;

– організація управління доставкою і зберіганням товарів.

Контракти на блокчейні мають наступні атрибути:

•використання методів електронного підпису (як токен ЕЦП) на основі відкритих та приватних ключів, доступних двом або більше сторонам угоди.

•наявність приватного децентралізованого середовища (наприклад, Ефіріум), в якому пишуться ці контракти і яке підтримує ввід та вивід інформації для програм передбачення (спосіб пов’язати інформацію з реального та цифрового світів).

•предмет контракту та наявність інструментів, необхідних для його виконання (криптовалюти для розрахунку, програми передбачення тощо).

•точно описані умови його виконання, які сторони підтверджують підписом, а також надійність джерела цифрових даних.

Також розрізняють види смарт-контрактів за способом генерування: згенеровані користувачами та самими контрактами. На рис. 1.4 наведено динаміку генерування смарт-контрактів.

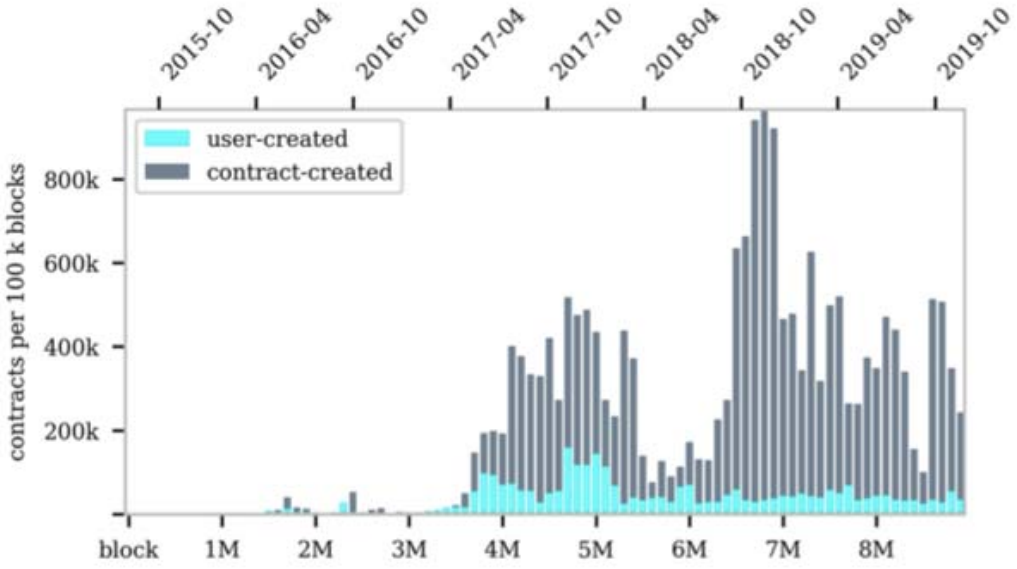



Рис. 1.4. Динаміка генерування смарт-контрактів за видами: згенеровані користувачем і самими контрактами (сірий колір).

1.2.4. Переваги смарт-контрактів:

Точність. Якщо контракт добре запрограмований, існує незначна ймовірність похибки.

Швидкість. Розумні контракти автоматизують процеси та не вимагають жодних операцій вручну.

Менше ризиків. Повторимось, що блокчейн – це розподілена база даних. Саме тому Смарт-контракти не тягнуть за собою ризики, витрати й інші проблеми юридичних угод.

Менше посередників. Такі контракти знімають або поступово зменшують роль постачальників довірчих послуг.

1.2.5. Приклади використання смарт-контрактів

Першим відомим кейсом використання Смарт-контракта у криптовалюті був проект Slock.it на базі мережі Ефіріум. Автори зазначали, що проект був таким собі Airbnb без посередників. Ви оплачуєте контракт ефіром і отримуєте доступ до квартири. На певний час, звісно.

Спільноті ідея зайшла і почали збирати на те гроші. Це було одне з перших ICO і перша ДАО (децентралізована автономна організація), вона так і звалась – The DAO. Але один хакер використав вразливість Ethereum, щоби 17

червня 2016 списати з рахунку The DAO мільйони доларів чужих грошей.

Інші кейси застосування Смарт-контрактів

Фінансовий сектор має найбільший попит на смарт-контракти. Цікаво, що близько половини всіх банківських злочинів вчиняються нинішніми чи колишніми працівниками. Тому банки постійно переглядають можливі рішення щодо усунення людського фактора, включаючи технологію блокчейн. Впровадження смарт-контрактів може спростити ці процеси, тому найбільші гравці розпочали розвиток у цій галузі.

Як доказ, консорціум R3 залучив більш як півсотні фінансових інституцій, а також провідні світові банки та розробляє платформу під назвою Corda. З допомогою Корди, оптимізованої для фінансового сектору, понад 15 банків-членів розробили прототип Смарт-контрактів для своїх операцій. Наприклад, Barclays використовує такі контракти для автоматизації платежів та зміни власника транзакції. HSBC і Bank of America замінили акредитиви на розумні контракти.

Розумні контракти можуть змінити спосіб ведення бізнесу. Вони внесуть радикальні зміни шляхом прискорення транзакцій, скорочення бюрократії та підвищення загальної сукупної ефективності. Багато галузей, таких як музика, мистецтво, фінанси, роздрібна торгівля, нерухомість, телекомунікації та ланцюги поставок, можуть отримати значну користь від використання смарт-контрактів. Однак справжній потенціал смарт-контрактів доки не доступний через обмеження інфраструктури: вона ще просто не існує.

## 1.3. Використання NFT в якості офіційного документа

Ключовим елементом цієї роботи є написання смарт-контракту для випуску NFT, містить в собі медичний рецепт. У насамперед розглянемо питання функціональної можливості застосування NFT для виписки, зберігання і передачі медичних рецептів.

Основними перевагами NFT як носій інформації є [11].

– Відносна анонімність. Публічна є інформація лише про гаманці в

блокчейні Ethereum, між якими здійснюється передача токена (або його емісія), тоді як персональні дані залишаються прихованим.

– Незамінність. Особливості платформи Ethereum натякають на, що неможливо створити точну копію NFT, таким чином повністю виключається можливість підробки рецепту (у відрізни та від документів з ЕЦП або УЕЦП, які можна, можливо просто скопіювати).
– Відстежуваність. Зважаючи на структуру блокчейну як такого, все операції залишаються збереженими в ньому, що в контексті використання в документах або медичних рецептів дозволяє виключити можливість копіювання, так як емісія скопійованого NFT буде проводитися не від імені уповноваженого спеціаліста, а від імені пацієнта
– Універсальність. NFT, що є результатом роботи смарт – контракту в блокчейні Ethereum може містити в собі будь-яку інформацію. В даний час широко поширене використання NFT як сховища для посилання на зображення на сторонніх платформах, проте це вичерпує його можливості. Він може виступати носіїв будь-якої інформації, а зберігання в собі лише посилання є мірою зменшення транзакційних витрат.

Далі розглянемо можливість використання NFT як носія медичного рецепту з погляду закону. В даний час багато країни і компанії всі активніше використовують блокчейн в цілому і в зокрема смарт-контракти і поступово вони набувають все більшої юридичної сили.

## 1.4. Огляд аналогів

Передача рецепту з допомогою NFT можна порівняти як концептуально, так і з погляду реалізації. Концептуально передачу медичногорецепту в вигляді токена в децентралізованою мережі можна, можливо закономірно порівняти з іншими методами передачі медичних рецептів.

Традиційний спосіб передачі на папері є багато в чому застарілим, тому

що має безліч «вузьких місць».

1. Низька надійність паперу як фізичного предмета. Вона можебути легко пошкоджено або втрачено.

2. Вразливість до підробки. Перевірка паперового рецепту в аптеках здійснюється виключно візуально, відповідно це несе великий ризик фальсифікації рецепт.

3. Анонімність як одна з причин уразливості до підробки. Реалізувавши підроблений рецепт, зловмисник зможе залишитися анонімним та його пошук у надалі буде утруднений.

4. Логістика. Також неможливо оперативно перемістити рецепт з одне місце в інше через те, що папір є фізичним об'єктом.

Таким чином, ми робимо висновок, що передача рецепту на папері є застарілим способом, що має безліч недоліків. Однак є і плюс, загальний для будь-якого паперового документообігу, що полягає в повної незалежності від наявності якої-небудь апаратури.

Також в даний час активно реалізується система «Електронний рецепт», що вже діє в Україні. Дана система дозволяє зберігати, виписувати та реалізовувати медичний рецепт в електронному вигляді. за бажанню користувача системи йому може бути видано паперова копія рецепт. Під неї вже розроблено нормативну базу і вже зараз нею користуються близько 700 аптек. Вона має весь необхідний функціонал, але при цьому є централізованою системою. Системи такого родумають ряд як переваг, так і недоліків.

Серед недоліків централізованих систем можна виділити підвищений ризик витоку персональних даних, так в випадку який-небудь витокуїї масштаб буде колосальним через організацію системи, проблеми масштабування: єдина система буде мати значні проблеми з обміном даних з огляду великої затримки, і навіть щодо низьку відмовостійкість. У системах такого роду вихід із ладу центрального вузла зазвичай тягне до повної втрати їх працездатності на час його ремонту.

У той же час децентралізовані системи, які включають себе технологію

блокчейн позбавлені багатьох цих недоліків. Відмовостійкість забезпечується структурою блокчейн, для висновку його з ладу або порушення роботи необхідний вихід з ладу порядку половини всіх вузлів мережі. Також у децентралізованій системі затримка в операціях не залежить від будь-якого центру і є приблизно однаковою для всієї мережі. Захист персональних даних забезпечує повсюдне шифрування дані. Їх витік малоймовірна і не може привести до втрат мільйонів персональних даних, як це відбувається регулярно із сервісами великих компаній.

З погляду реалізації можна порівняти систему передачі медичних рецептів через блокчейн з різними системами і сервісами сервісом для створення та продажу квитків на різні заходи у вигляді NFT. Прикладом такого сервісу є сервіс oveit, що дозволяє організаторам заходу випустити власні NFT-квитки та поширити їх через вбудований маGasин або через популярні маркетплейси, такі як opensea. Життєвий цикл квитка в таких сервісах схожий з життєвим циклом медичного рецепту: він створюється особою, яка має на це право, передається звичайним користувачем, а потім переходить в організацію, пов'язану з тим, хто випустив NFT. При цей сервіси такого родузазвичай мають веб-системи, а також мобільні програми на їх основі.

Такі сервіси використовують блокчейн Ethereum або інші EVM сумісні блокчейни, наприклад, Polygon, які реалізують аналогічні протоколи та функції.

Незважаючи на те, що NFT спочатку з'явилися як варіант токена в блокчейні Ethereum, не тільки Ethereum в даний час може підтримувати NFT. Реалізацію NFT також підтримує безліч інших блокчейнів, що підтримують роботу EVM, тому надалі говорячи про NFT в блокчейні Ethereum мають на увазі також інші блокчейни, сумісні з EVM. Таким чином, як блокчейн для використання в веб-додатку може використовуватися будь-який EVM-сумісний блокчейн, що дозволяє реалізувати протокол ERC-721, необхідний для створення NFT.

## Висновки до першого розділу

У розділі подано аналіз наукової літератури на тему створення смарт-контрактів та вирішення питань безпеки транзакцій, історія створення та приклади використання смарт-контрактів, а також розглянуто можливість використання NFT як носій медичного рецепту. Були вивчені аналоги у вигляді інших носіїв медичних рецептів, а також схожі по реалізації послуги, обгрунтовано вибір блокчейну.

## 2. Розділ 2

### 2.1. Надсилання транзакцій

Базовою операцією, яка може змінювати стан блокчейн Ethereum є транзакція. Найпростіша дія для зміни стану – надсилання коштів з адреси на адресу. Для цього в транзакції вказується адреса одержувача та сума, яка пересилається.

Вузол, на якому в результаті формується транзакція для відправки в мережі, намагається отримати доступ до приватного ключа відправника і підписати їмтранзакцію. Для того , щоб вузол успішно зміг виконати операцію, найчастіше, необхідно попередньо розблокувати відповідну адресу на ньому з використанням пароля доступу до приватного ключа.

Обробка кожної транзакції на вузлі, що включає їх у новий блок (при умові, що потім блок буде прийнятий решта частиною мережі Ethereum), споживає gas. Gas – поняття, введене в блокчейн Ethereum, щоб контролювати ресурси, які є споживані віртуальною машиною Ethereum (Ethereum Virtual Machine, EVM). Наприклад, обробка звичайної транзакції переказу коштів з адреси на адресу споживає 21 000 gas.

Передбачувана кількість gas (поле «gas»), яка може витратитися в ході обробки транзакції, множиться на ціну за одиницю gas. ("gasPrice") і віднімається з балансу адреси, з якої здійснюється транзакція. Якщоз'ясовується, що в адреси на балансі не вистачає засобів для обробки транзакції, то вона не включається в блок та не обробляється.

Gas витрачається не лише на обробку інформації, а й на її зберігання, тому він споживається за кожен байт, переданий як datав тіло транзакції.

Підсумкова кількість gas, витрачена під час включення транзакціїв блок, можна, можливо дізнатися в поле «gasUsed» при виклике методу getTransactionReceipt модуля eth.

Визначення адреси контракту

Оскільки блокчейн з технологічної точки зору – це децентралізована база даних, то як і більшість сучасних баз даних вона дозволяє зберігати не тільки дані, але і виконуваний код.

Спочатку передбачалося, що виконуваний код блокчейн повинен регулювати грошові потоки, так як основна мета блокчейн, як бази даних, передбачалася для зберігання записів про переміщення умовних одиниць, еквівалентних грошам. Наприклад, код міг визначати появу якоїсь суми на балансі рахунку, призначеного для накопичення отриманого підприємством прибутку, після чого він автоматично міг би ділити цей прибуток між акціонерами даного підприємства. Це було б схоже на те, як виконуються зобов'язання за договорами (англ. contracts). Саме тому код, керуючий грошовими потоками в блокчейн, погодилися називати смарт-контракти.

І блокчейн Bitcoin, і блокчейн Ethereum дозволяють користувачам заносити в блокчейн свій власний виконуваний код. Відмінності тільки в тому, де цей код в результаті зберігається, і наскільки універсальні алгоритми можуть бути реалізовані за допомогою тих команд, які доступні для програмування.

У блокчейн Ethereum набір доступних команд є повним, відповідно машини Тьюринга, що дозволяє складати досить складні смарт-контракти, в яких будуть і цикли, і умовні переходи та процедури. Користувач Ethereum може підготувати смарт-контракт і за допомогою спеціальної транзакції, помістити його в блокчейн.

Транзакція для реєстрації смарт-контракту вирушає без адреси отримувача. У поле даних (data) у такий транзакції знаходиться набір байт – скомпільований в вигляді байт-коду смарт-контракт. При цьому байткод складається з двох частин: заголовку і байт-коду для зберігання.

У заголовку зберігається набір інструкцій, які виконуються лише один раз при реєстрації смарт-контракту. Тут може бути початкова ініціалізація певних змінних, виконання конструктора смарт-контракту.

У байт-код для зберігання розташовується тіло самого смарт-контракту,

яке містить інструкції, що виконуються під час подальших звернень до цього смарт-контракту.

Потім транзакція включається до нового блоку одним із майнерів мережі Ethereum. При цьому майнер виконує ряд дій.

1. Ініціалізує нове сховище (storage), в якому зберігатимуться дані смарт-контракту (глобальні змінні, масиви, словникиі т.п.). Це сховище вибудовується з використанням структури даних Patricia Merkle Tree, тому воно називається Storage Tree.

2. Виконує всі дії із заголовка байт-коду, що призводить до початкового заповнення Storage tree.

3. Формує в своїй копії дерева стану блокчейна (World State Tree) нову вершину, яка використовується для зберігання даних про смарт-контракт.

З вершиною World State Tree асоціюється адреса, яка називається адресою смарт-контракту. Оскільки у смарт-контракту немає власного приватного ключа, для призначення адреси смарт-контракту використовується адреса акаунту, який відправляє транзакцію на реєстрацію в мережу, а також номер цієї транзакції щодо інших транзакцій.

З описаного вище видно, що реєстрація контракту змінює дерево стану блокчейну (World State Tree), отже, зміниться його State Root, який буде відображено в новий блок. Блок поширюється по всіх вузлах мережі. При отриманні нового блоку вузол мережі обробляє транзакцію на реєстрацію смарт-контракту так само, як це робив майнер, включив її в блок. При цьому у вузлі зміниться спочатку Storage Root, а відповідно і State Root. Обчислений State Root повинен збігатися з відповідним полем у прийнятому блоці.

Таким чином, можна відзначити, що незважаючи на те, що код смарт-контракту зберігається спочатку в транзакції, як частина процесу майнінгу блоку він переміщається у World State Tree спочатку лише в одному вузлі, а потім на всіх пристроях в мережі блокчейн. Також виконання ініціалізації (заголовка байт-коду) відбувається на всіх вузлах мережі блокчейн, що забезпечує децентралізоване зберігання як коду, так і даних смарт-контракту.

## **2.2.** Виклик методу контракту

Для виконання коду смарт-контракту використовується віртуальна машина Ethereum (EVM). EVM є частиною практично будь-якого клієнта мережі Ethereum. Віртуальною її називають, оскільки вона реалізована на рівні програмного забезпечення, що дозволяє абстрагуватися від апаратного рівня комп'ютера, а значить гарантує, що на якому б реальному залізі не запускався код смарт-контракту, результат виконання буде один і той ж без необхідності перекомпілювати код.

Коли відбувається запуск смарт-контракту на виконання, на згадку EVM передається байт-код цього контракту, а також забезпечується доступ до його сховища (storage). Певні набори байт у байт-коді інтерпретуються в команди віртуальною машини і операнди до цим командам.

Якщо подивитися специфікацію Ethereum, то можна, можливо, побачити, що кожна команда EVM споживає якусь кількість ресурсів, які були прийнято вимірювати з використанням поняття «gas». Gas – це узагальнений підхід до споживання ресурсів, що виражає одночасно споживання процесорного часу і пам'яті. Концепція gas була введено,щоб вирішити кілька проблем:

Оскільки мається на увазі, що смарт-контракти виконуватимуть майнери під час випуску нових блоків, незважаючи на різні можливі апаратні платформи у майнерів, необхідно ввести якийсь універсальний механізм вимірювання ресурсів, витрачених на виконання смарт-контракт. Таким чином, замовник виконання смарт-контракту (відправник транзакції) зможе побудувати свої очікування щодо того, в якому обсязі робота обладнання майнера повинна бути компенсовано.

Обмежені обчислювальні ресурси майнера потребують можливості оцінки скільки виконання смарт-контракту може зайняти ресурсу, ще до того,як смарт-контракт буде запущено виконання. Завдяки тому, що відправник транзакції вказує максимальне кількість gas, яке він готовий сплатити, і яке, у свою чергу, характеризує кількість gas, споживане смарт-контрактом, майнер

може вирішувати – чи він візьметься завиконання даного смарт-контракт.

Різні команди споживають різну кількість gas. Спочатку, споживання gas в командах було асоційовано з припущенням того, яких ресурсів вимагатиме від майнера виконання цієї команди, чим більше ресурсів для обчислення чи зберігання потрібно, тим більше Gas споживатиметься командою. Так, наприклад, команди збереження даних у сховище смарт-контракту і команда створення нового смарт-контракту є найбільш важкими з цієї точки зору, оскільки вимагають від майнера виділення додаткової пам'яті, а, отже, додаткових ресурсів на довготривале зберігання цієї пам'яті.

Запуск смарт-контракту, який вже зареєстрований в блокчейн може виконуватися декількома способами. Якщо в ході виконання код смарт-контракту буде міняти дані у storage, то тоді потрібно відстеження даних змін у World State всіма вузлами мережі блокчейн. Це можливо лише через пересилку транзакції і наступним її включенням до нового блоку.

Якщо код контракту тільки обчислюватиме щось, базуючись на переданих йому параметрах або даних із storage, або, іншими словами, не змінює World State, то в цьому випадку можна використовувати локальний виклик смарт-контракту. Цей спосіб не вимагає оплати спожитого gas, але вимагає доступе до web3 модуля на вузлі Ethereum.

І в тому, і в другом випадку, в результаті, станеться завантаження повністю всього байт-коду контракту в віртуальну машину, після чого EVM почне його виконання. При цьому важливою деталлю є те, що поле data транзакції, зухвалою код контракту, передається на Вхід коду, завантаженому в EVM.

Більшість сучасних смарт-контрактів в блокчейн Ethereum написано мовою Solidity високорівневу мову програмування. Байт-код для EVM виходить в результаті компіляції вихідного коду Solidity. Процес компіляції автоматично готує набір команд для виконання EVM здатний обробити вхідні дані, що прийшли через поле data транзакції. У зокрема, за рахунок цього набору команд реалізується можливість мати у смарт-контракті кілька точок виклику окремих методів (функцій) одного і того ж Договору.

Ім'я функції та аргументів беруться з вихідного коду контракту. Але є ситуації, коли не завжди доцільно використовувати код повністю або коли взагалі немає можливості його використати. Натомість використовується опис інтерфейсів смарт-контракту (Application Binary Interface, ABI), репрезентуючий з себе JSON структуру, де перераховані ті сутності контракту на мовою Solidity, до яким можна, можливо звертатися в конкретному смарт-контракт. Більшість сучасних програм, що надають графічний інтерфейс користувача для операцій з блокчейному Ethereum вміють генерувати ABI з вихідного коду смарт-контракт. А практично будь-яка програмна бібліотека, що працює з Ethereum, дозволяє з використанням відомого ABI відправляти транзакції для викликуметодів смарт-контракт.

## **2.3.** Отримання балансу

Транзакції формують блоки, а потім між двома сусідніми блоками утворюється зв'язок по принципом «попередній» "наступний". Кожен блок фіксує набір змін, що виконуються транзакціями, тому блок можна контрольною точкою, що визначає «моментальний знімок» даних у блокчейн. Таким чином, у пам'яті блокчейн зберігаються всі «миттєві знімки» і можна простежити історію зміни будь-яких даних, наприклад балансу адрес. Баланс адреси у блокчейн Ethereum вимірюється в wei – найменша одиниця вимірювання, яка складає $10^{-18}$ від одного ETH.

Коли транзакція відправляється для перерахування коштів з одного рахунки на інший, то сума для переказу вказується у wei. Це досить незручно для людини, але для віртуальною машини Ethereum так простіше – не потрібно мати справу з дробовими значеннями, і, отже, з проблемами округлення.

## **2.4.** Використання мови JavaScript та бібліотеки React длярозробки веб-додатків

У справжнє час мова JavaScript домінує серед мов програмування для

розробки веб-застосунків. Згідно рисунку 1 мовою JavaScript для розробки веб-додатків користується близько 65% розробників, також для мови JavaScript доступно безліч фреймворків і пакетів бібліотек [12]. Розробка веб-додатків на мові JavaScript частіше всього передбачає використання скриптів, написаних мовою JavaScript для зміни або створення HTML і CSS елементів розмітки, а також для додавання власних об'єктів і способів інтеракції з користувачем на сторінку веб-системи.

Однак мова JavaScript дозволяє розробляти веб-додатки та без використання HTML і CSS, це ставиться можливим завдяки використанню сторонніх програмних платформ, таких як Node.js.

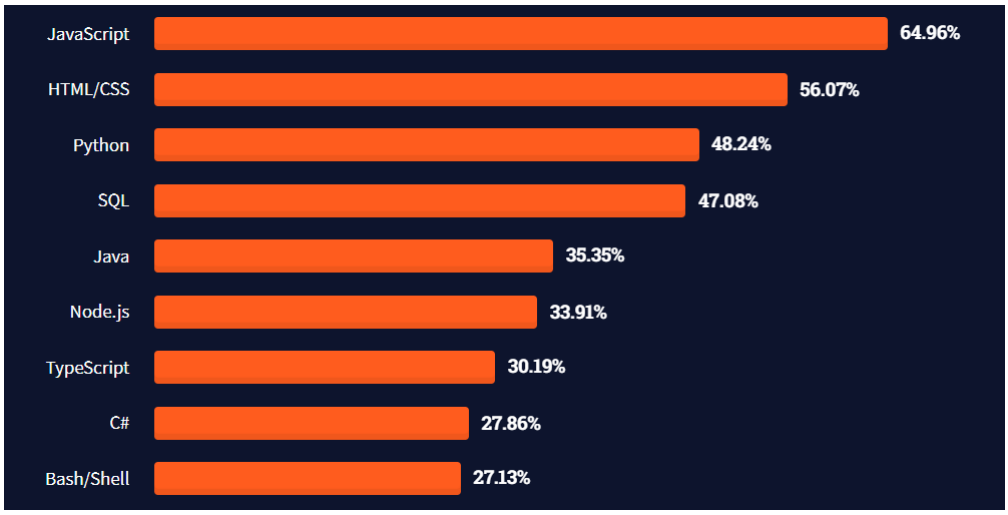



Рис. 1 – Статистика популярності мов програмування длярозробки веб-додатків

Однією з найбільш поширених бібліотек для JavaScript є React [13]. У справжнє час вона розробляється компанією Facebookта ентузіастами. React це бібліотека для JavaScript для створення користувачів інтерфейсів, має відкритий початковий код. Вона використовується во багатьох сайтах та веб-додатках, в в тому числі Instagram та Facebook.

Її ключові особливості включають функціональний підхід до структури та обробки елементів, а також життєвий цикл для кожного елемент. Таким чином завдяки можливості декомпозувати інтерфейс користувача на окремі модулі можна значно полегшити. розробку інтерфейсу, а завдяки наявності

життєвого циклу у кожного елемента є можливість працювати з ними окремо, не змінюючи вміст сторінки більший за необхідний. На рисунку 2 показаний прикладвиконання програми, написаною на мовою JavaScript з використанням бібліотеки React [14]. На ньому, незважаючи на його просто видно відмінні риси бібліотеки React: функціональний підхід до створення та зміни елементів і наявність життєвого циклу в особі функції render в якої відбувається відображення елемента на сторінку при досягненні їм необхідного стану в рамках життєвого циклу


Рисунок 2 – Приклад виконання програми на JavaScript звикористанням бібліотеки React

**2.5.** Використання пакету бібліотек web3.js для зв'язку веб-додатки і EVM

Мабуть, найбільш популярним способом для взаємодії міжвеб-додатками, написаними на мовою JavaScript і блокчейному Ethereum є пакет бібліотек web3.js. Крім різноманітних методів роботи з контрактами в мережі Ethereum web3.js дозволяє також взаємодіяти зі сторонніми програмами, як приклад «тонкий» гаманець MetaMask, що встановлюється як розширень для браузерів. Використання MetaMask дозволяє значно спростити взаємодія веб-додатку з блокчейному.

Висновки по другий главі

Розглянуто теоретичну частину питання, розглянуто аспекти, пов'язані з

механізмом роботи смарт-контрактів, EVM, відправкою транзакцій, викликом методу контракту, одержанням балансу.

# РОЗДІЛ 3
# ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ

## 3.1. Проектування системи

### 3.1.1. Функціональні і нефункціональні вимоги

Аналіз предметний області дозволив виділити наступні функціональні і нефункціональні вимоги.

Функціональні вимоги до проектованої системі

- додаток повинен здійснити випуск NFT з медичним рецептом;
- випущені NFT повинні бути сумісні зі стандартом ERC-721 та коректно взаємодіяти зі стороннім ПЗ;
- для кожного типу користувачів програми повинна бути розроблено свої версія, яка реалізує необхідні для його функції.

Нефункціональні вимоги до проектованої системи

- додаток повинен бути реалізований на мові Solidity.
- додаток повинен використовувати протокол ERC-721.
- смарт-контракт повинен бути розгорнуть в тестовий мережі.
- кожна клієнтська частина веб-системи повинна бути незалежною одна від одної.

## **2.6.** Загальна концепція різних версій програми та їх взаємодій

У процесі виписування та реалізації рецепту є 3 основні актори: «Лікар», «Пацієнт», «Аптекар». на рисунку 3 показано діаграма діяльності для всього процесу створення і передачі медичного рецепту, яку має забезпечувати веб-система. Таким чином ми можемо сформувати вимоги до функціоналу для кожного клієнт-додатку веб-системи.

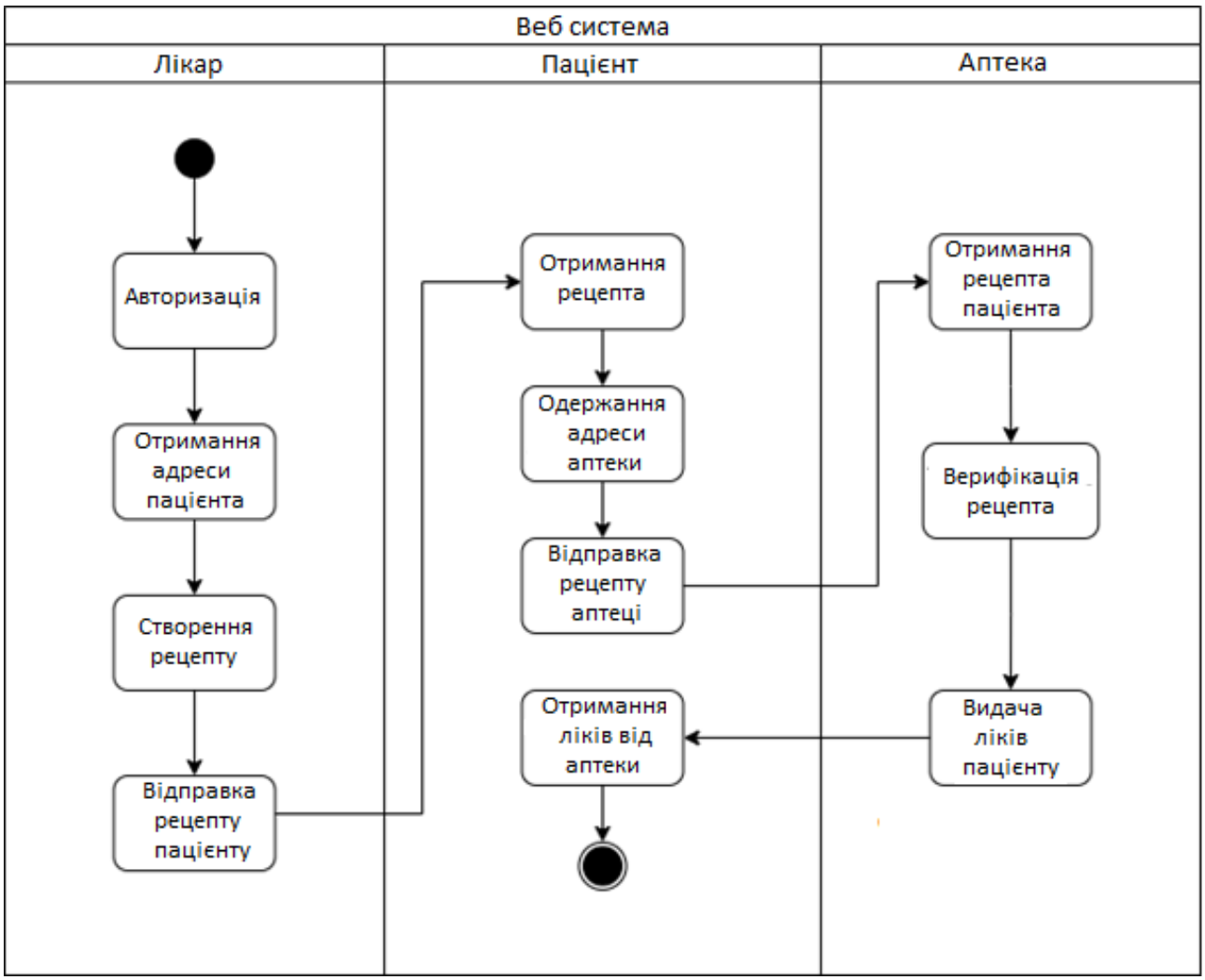



Рисунок 3.1 – Діаграма діяльності для обігу медичного рецепту у веб-системі

## 2.7. Сторінка веб-додатки «Сторінка лікарі»

На рисунку 4 показано діаграма варіантів використання для лікаря, використовує відповідну версію веб-системи.

Згідно з цією діаграмою, додаток має реалізовувати ряд варіантів використання.

1. «Сканувати QR-код зі рахунком пацієнта». Навівши камеру на QR код, що показується веб-додатком пацієнта, лікар отримує його адресу Ethereum-гаманець, на який надалі вироблятиме транзакцію рецепт-NFT.


Рисунок 4 – Діаграма варіантів використання веб-додатку для лікаря

2. "Виписати рецепт". Ядро функціоналу лікаря», під час виконання даного варіанту використання лікар заповнює поля рецепту, а потім відправляє його пацієнта. Для спрощення лікар може сканувати QR-код задресою пацієнта во час даного варіанти використання або заздалегідь.

3. «Відкликати рецепт». Даний різновид використання дозволяє відкликати рецепт, що знаходиться у пацієнта.

4. «Продовжити рецепт». Даний різновид використання дозволяє продовжити термін дії рецепту, що знаходиться у пацієнта.

5. «Переглянути виписані рецепти». Даний варіант використання користувач виконує в системі по замовчуванням, і він увімкнено в інші варіанти використання, такі як «Продовжити рецепт» та «Відкликати рецепт».

**2.8.** Сторінка веб-додатки «Сторінка пацієнта»

На рисунку 5 показано діаграма варіантів використання для лікаря, використовує відповідну версію веб-системи.

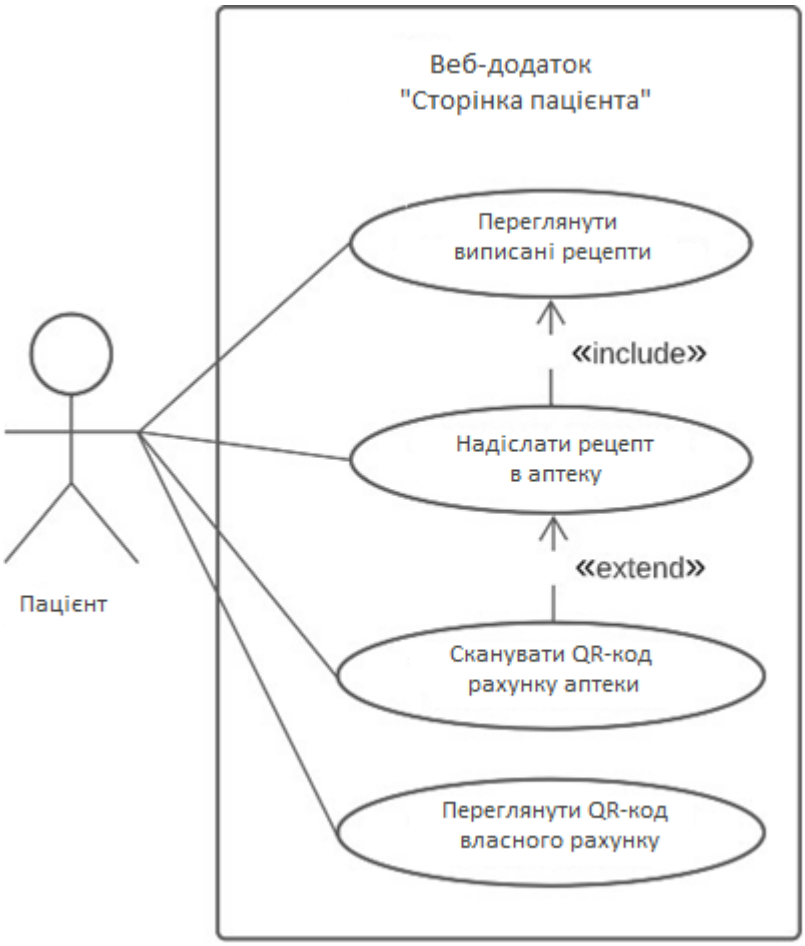



Рисунок 5 – Діаграма варіантів використання веб-системи дляпацієнта

Згідно з цією діаграмою, додаток має реалізовувати ряд варіантів використання.

1. «Сканувати QR-код з рахунком аптеки». Навівши камеру на QR код, що показується веб-додатком аптеки, пацієнт отримує її адресу Ethereum-гаманець, на який надалі вироблятиме транзакцію рецепт-NFT.

2. «Показати QR-код зі своїм рахунком». додаток виведе на екран QR-код, що зберігає в собі адреса Ethereum-гаманець пацієнта.

3. «Надіслати рецепт у аптеку». Даний різновид використання дозволяє

відправити пацієнтом рецепт в аптеку. У такому разі користувач повинен вказати адресу аптеки, що він повинен робити або самостійно, або при допомоги варіанти використання «Сканувати QR-код зірахунком аптеки».

4. «Продовжити рецепт». Даний різновид використання дозволяє продовжити термін дії рецепту, що знаходиться у пацієнта.

5. «Переглянути виписані рецепти». Цей варіант використання користувач виконує в системі за промовчанням, і він включений у варіант використання "Надіслати рецепт в аптеку».

### **2.9.** Сторінка веб-додатки «Сторінка аптеки»

на рисунку 6 показано діаграма варіантів використання для лікаря, використовує відповідну версію веб-системи.

Згідно з цією діаграмою, додаток має реалізовувати ряд варіантів використання.

1. «Показати QR-код зі своїм рахунком». Додаток виведе на екран QR-код, що зберігає в собі адреса Ethereum-гаманець аптеки.

2. «Позначити рецепт виданим ». Даний варіант використання дозволяє відзначити рецепт виданим та прибрати зі списку рецептів. При цьому він залишається зберігатися на гаманці аптеки для обліку.

3. «Переглянути надіслані рецепти». Цей варіант використання користувач виконує в системі за промовчанням, і він включений у варіант використання «Позначити рецепт виданим ».

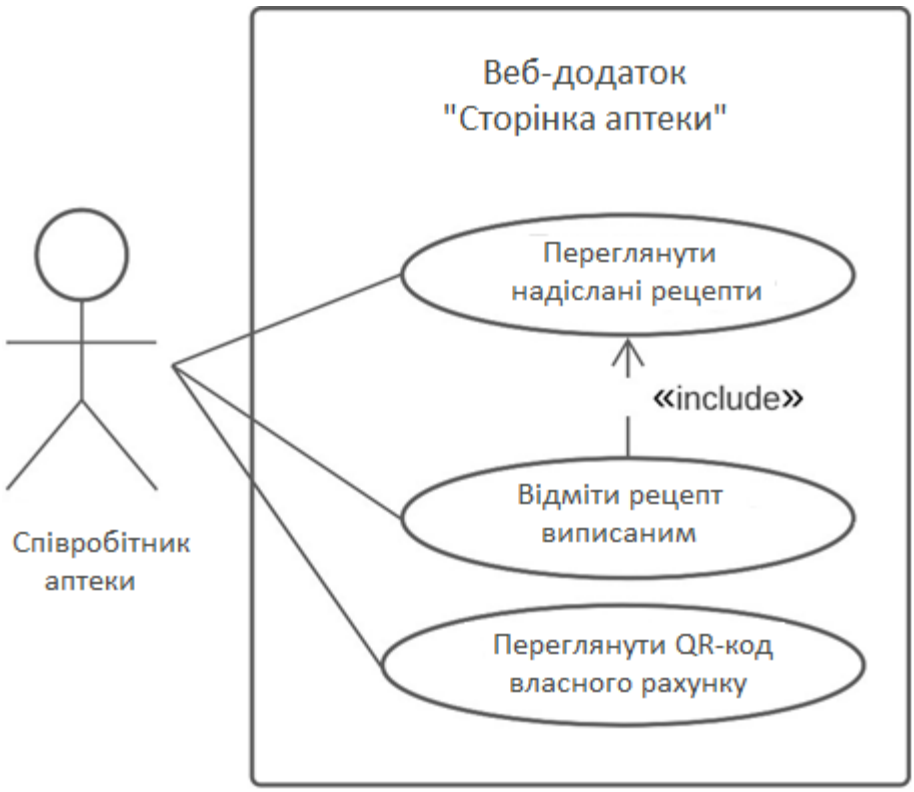
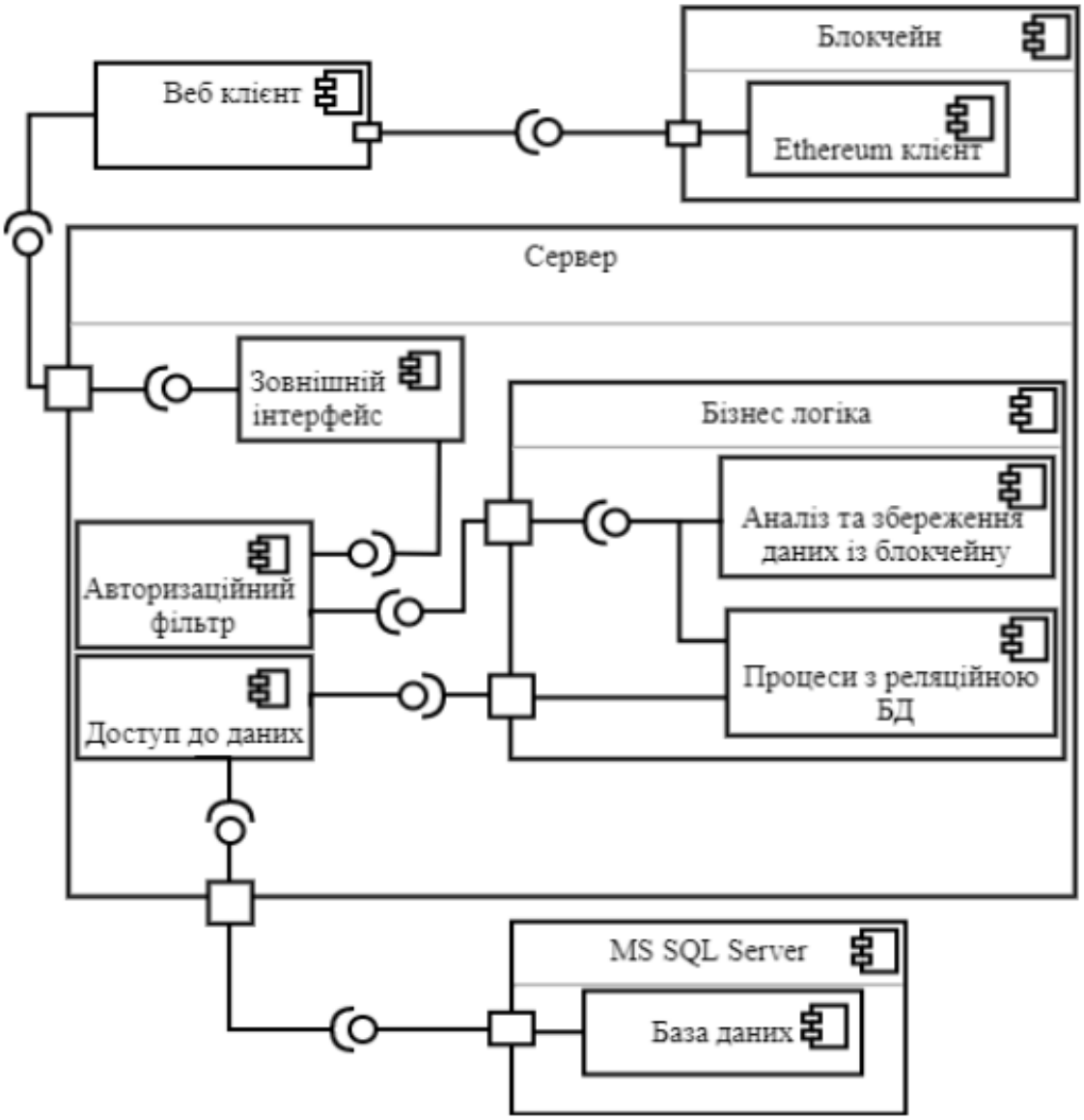



Рисунок 6 – Діаграма варіантів використання веб-системи дляпацієнта

## Висновки по третьою главі

У цьому розділі були визначені функціональні та нефункціональні вимоги до системи, описані діаграми варіантів використання для різних версій веб-системи.


Блокчейн
Веб клієнт
Ethereum клієнт
Сервер
Зовнішній інтерфейс
Бізнес логіка
Аналіз та збереження даних із блокчейну
Авторизаційний фільтр
Процеси з реляційною БД
Доступ до даних
MS SQL Server
База даних

# 4. РЕАЛІЗАЦІЯ

## **4.1.** Серовище виконання і програмні засоби реалізації

Як основну мову для розробки смарт-контракту було обрано мову Solidity версії 0.8.7.

Solidity – це нова мова програмування, призначена для написання програм, званих смарт-контрактами, які можуть запускатися віртуальною машиною Ethereum (EVM).

Solidity представляє собою сукупність угод з мережевих технологій, мови асемблера та веб-розробки.

Для розробки веб-системи було використано мову програмування JavaScript та бібліотеки React, web3.js, Bootstrap. В якості середовища розробки використовувалося IDE Visual Studio Code 2022.

Як блокчейн, в якому буде розташовуватися контракт, що забезпечує випуск і подальшу роботу з медичними рецептами був обрано тестову мережу EVM-сумісного блокчейну BNB Smart Chain. Його перевагами в порівнянні з блокчейном Ethereum є збільшене швидкодія і безкоштовне використання тестової мережі.

Для спрощення взаємодії між браузером і блокчейному було використано розширення MetaMask.

## **4.2.** Розробка і розміщення смарт-контракту

У лістингу 1 додатки А наведено вихідний код смарт-контракту «PrescriptionNFT» для створення NFT, який надалі передаватиметься між лікарем, пацієнтом та аптекою.

Даний код реалізує застосування стандарту ERC-721. ERC (Ethereum Request for Comments) – це офіційний протокол для внесення пропозицій по покращення мережі Ethereum; 721 – унікальний ідентифікаційний номер пропозиції. Початковий стандарту ERC-721 наведено в лістингу 2 додатки А.

Також у рамках використання стандарту ERC-721 необхідно включити до

нього бібліотеку SafeMath, яка забезпечує контракт можливістю використовувати безпечні математичні операції. Вихідний код бібліотеки SafeMath наведено в лістингу 3 додатки А.

Використання смарт-контракту можна порівняти з використанням класу в об'єктно-орієнтованому програмуванні Контракт позначається поля, які будуть заповнені при створенні NFT, а також функції,які використовують веб-системи.

Розміщення смарт-контракту здійснюється у тестовій мережі BNB Smart Chain за допомогою середовища розробки Remix та розширення MetaMask.На рисунку 7 показаний результат пошуку смарт-контракту на спеціалізованому сайті, що займається моніторингом стану блокчейну, на ньому видно результат розміщення смарт-контракту в мережі блокчейн BNB Smart Chain Testnet. Він публічний, має свій унікальний хеш та вписаний у блокчейн.

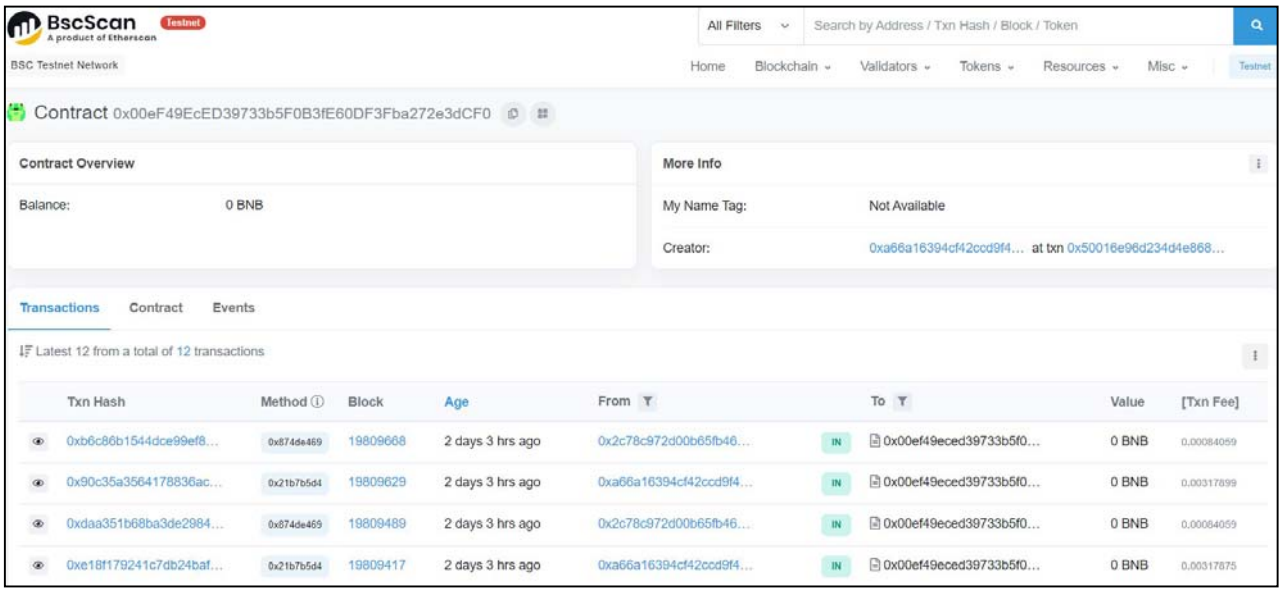



Рис. 7 Смарт-контракт, розміщений у мережі блокчейн BNB SmartChain Testnet

### 4.3. Взаємодія контракту і веб-додатків

Контракт «PrescriptionNFT» взаємодіє з веб-додатком у наступних

випадках.

При створенні лікарем рецепту та його відправленні пацієнту формується NFT. NFT з рецептів містить у собі інформацію про ліки, термін свого дії, а також адреса пацієнта. Це інформація не може бути змінена надалі ніким крім лікаря, якщо той захоче продовжити чи видалити рецепт. З технічного погляду важливо те, що лікар не перекладає NFT на свій гаманець, а виробляє його емісію в гаманець пацієнта.

Коли NFT переходить до пацієнта, веб-додаток отримує інформацію про це та відображає її у списку, використовуючи інформацію з NFT. Крім цього, пацієнт може перекласти цей NFT аптеці що, щоб отримати ліки.

Аптека при отриманні NFT від пацієнта насамперед перевіряє те, чи випущений NFT сертифікований лікарем. На цьому вичерпується взаємодія аптеки з Договором.

### **4.4.** Розробка різних версій веб-додатки

Зважаючи на наявність різних вимог до функціональності веб-додатку для різних користувачів були розроблені 3 версії веб-системи. Інтерфейс програми для лікаря представлений на рисунку 8. Вихідний код веб-додатка знаходиться в лістингу 1 додатка Б. Згідно з вимогами, розробленими на етапі проектування, лікар має можливість переглянути нещодавно виписані рецепти і такі рецептивін має можливість продовжити або скасувати їх.

Також лікар має можливість сканувати адресу пацієнта, використовуючи вбудовані функції бібліотеки React та браузера. У такому разі веб – додаток запам'ятає адреса і вставить його під час створення рецепт.

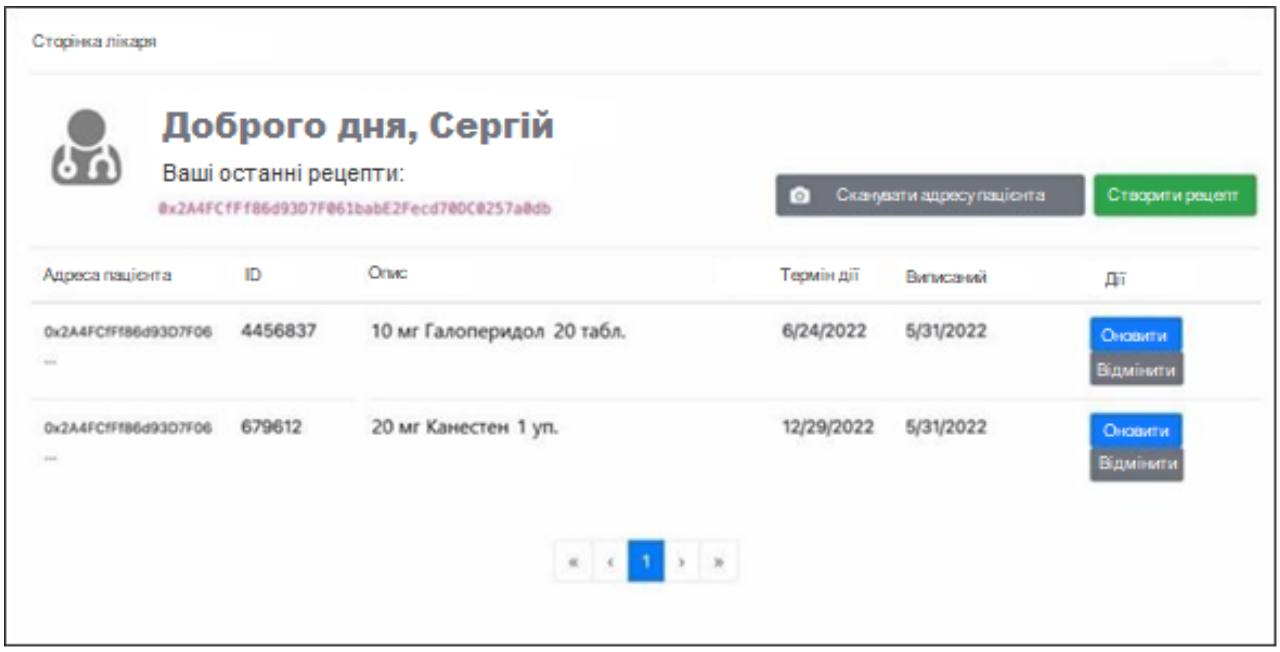



Рис. 8 – Інтерфейс веб-додатки для лікаря

Додаток для пацієнта також має свій унікальний інтерфейс, представлений на рисунку 9. Вихідний код веб-додатка знаходиться в лістингу 2 додатки В.

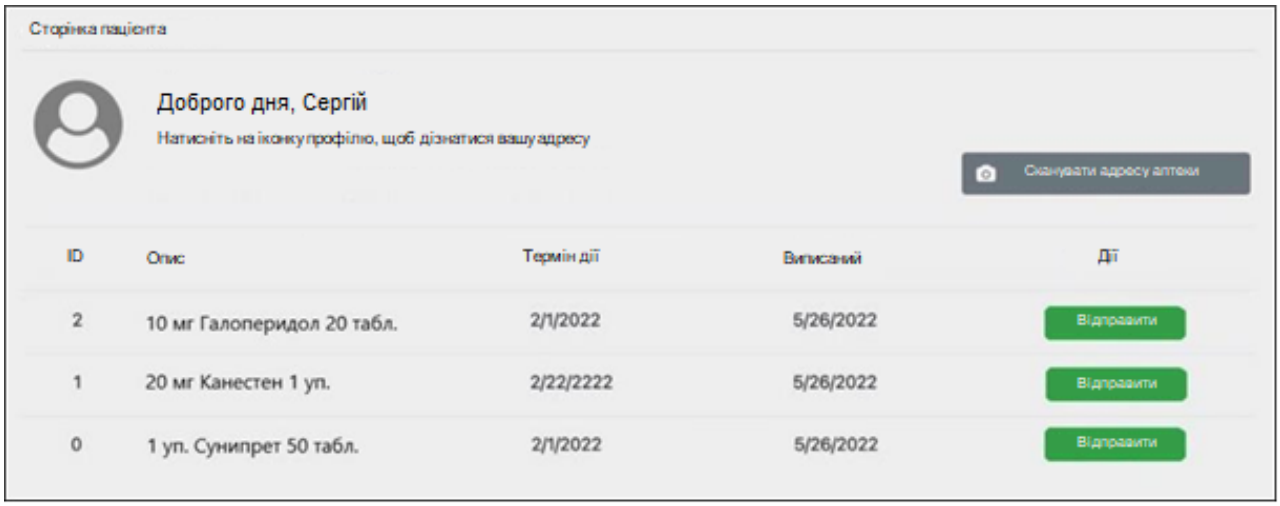



Рис. 9 – Інтерфейс веб-додатки для пацієнта

Згідно вимогам, описаними на етапі проектування пацієнт може лише переглянути свої рецепти, і відправити рецепт аптеці. Для спрощення відправки є можливість просканувати QR код з адресою гаманця аптеки.

Веб-додаток для аптеки має інтерфейс, показаний на рисунку 10.

Вихідний код веб-системи знаходиться в лістингу 3 додатка В.

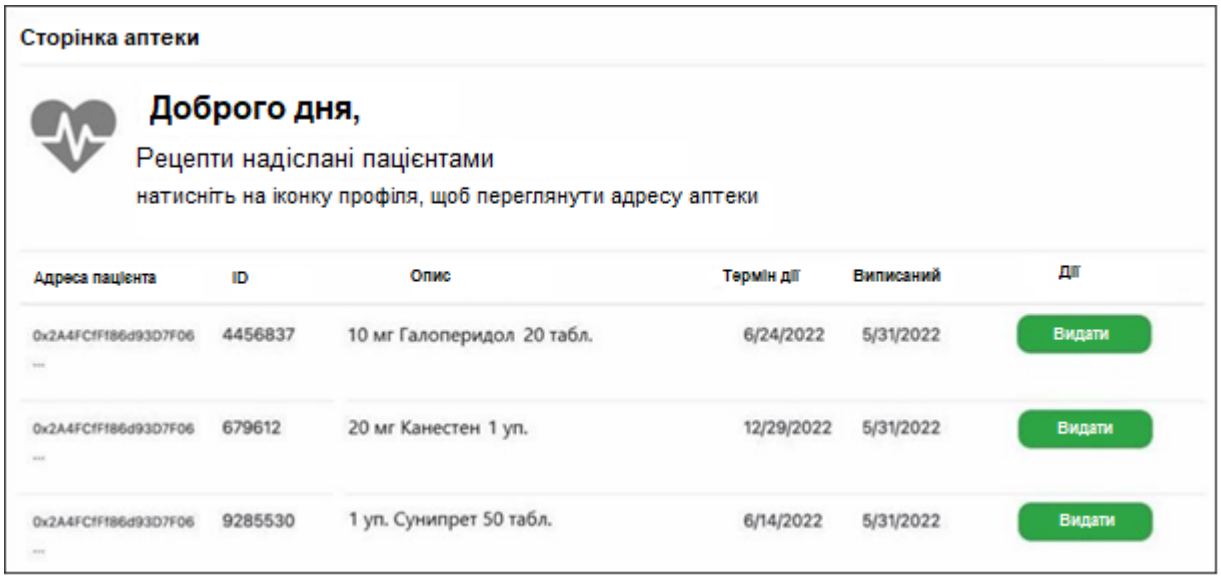



Рис. 10 – Інтерфейс веб-додатку для аптеки

Відповідно до вимог, описаних на етапі проектування веб-додаток для працівника аптеки, має можливість переглянути надіслані пацієнтами рецепти, а також кнопку «Видати», яка прибирає рецептз даного список.

## Висновки по четвертою главі

У ході виконання роботи поставлене завдання розробити веб-додаток для видачі рецептурних препаратів виконано. У главі представлений зовнішній вигляд реалізованого веб-додатку, описано його взаємодію зі смарт-контрактом та з користувачем.

**Функціональне тестування**

Тест №1. Мета: Зайти до веб-додатку «Сторінка лікаря». Дія: зайти на веб-сторінку програми «Сторінка лікарі».

Очікуваний результат: відкрився інтерфейс веб-додатки «Сторінка лікарі».

Тест пройдено? Так.

Тест №2. Ціль: Просканувати QR-код пацієнта в веб-додатку «Сторінка лікарі».

Дія: проведено транзакція.

Очікуваний результат: Веб-додаток "Сторінка доктора" просканувала QR-код пацієнта.

Тест пройдено? Так.

Тест №3. Ціль: Створити рецепт і Надіслати його пацієнту в веб-додатку "Сторінка лікаря".

Дія: Натиснута кнопка "Створити рецепт", поля були заповнені даними, натиснута кнопка «Надіслати рецепт».

Очікуваний результат: Рецепт додався до списку відправлених до додаток «Сторінка лікаря» і з'явився у додатку «Сторінка пацієнта».

Тест пройдено? Так.

Тест №4. Ціль: Створити ще 2 рецепту і Надіслати їх пацієнту в веб-додатку «Сторінка лікарі».

Дія: Натиснута кнопка "Створити рецепт", поля були заповнені даними, натиснута кнопка «Надіслати рецепт» 2 рази.

Очікуваний результат: Рецепти додалися до списку відправлених до

додатку «Сторінка лікарі» і з'явились в додатку «Сторінка пацієнта».

Тест пройдено? Так.

Тест №5. Ціль: Надіслати рецепти аптеці в додатку «Сторінкапацієнта».

Дія: Натиснута кнопка "Надіслати рецепт» в додатку «Сторінка пацієнта».

Очікуваний результат: Рецепти додалися в перелік рецептів в додатку «Сторінка аптеки».

Тест пройдено? Так.

У ході тестування були протестовані всі функціональні можливості програми, також був створено і переданий аптеці рецепт.

# ВИСНОВКи

У рамках даної роботи було розроблено веб-додаток для видачі рецептурних препаратів з використанням технології блокчейн.

Для досягнення поставленою цілі були виконані всі поставлені завдання, а саме.

1. Був проведено огляд наукової літератури на тему створення смартконтрактів і огляд аналогів.
2. Був обраний блокчейн для створення смарт-контракт.
3. Був розроблений і розміщено в блокчейне смарт-контракт, здійснює випуск NFT.
4. Було розроблено веб-додаток, здійснююче видачу рецептурних препаратів з використанням технології блокчейн.
5. Було проведено тестування розробленого смарт-контракту і веб-системи.

У майбутньому планується нарощувати функціонал веб-додатків та смарт-контракту, шляхом додавання нових функцій, таких як множинний вибір рецептів, додавання додаткового смарт-контракту та сторінки веб-додатку для отримання можливості змінювати перелік сертифікованих лікарів і розробку сторінки веб-додатки для контролюючих органів, що дозволяє переглянути перелік ліків, відпущених аптекою.

ЛІТЕРАТУРА

1. Лоран Лелу. Блокчейн від А до Я. Все про технологію десятиліття. – М.: Ексмо, 2018 року. – 256 с.

2. Варнавський, А.В. Токен або криптовалюта: технологічний зміст і економічна сутність / Фінанси: теорія і практика, Т. 22, № 5, 2018 року. З. 138-140

3. Команов П.А., Ревазов Х.Ю., Тавас Д.А. Дослідження безпеки смарт-контрактів Ethereum // МНІЖ. 2021. №1-1 (103)

4. Карпичов, В.Ю. Функціональна модель смарт-контракту на платформі Ethereum // Праці НДТУ ім. Р. Є. Алексєєва. 2019. №2 (125). URL: https://cyberleninka.ru/article/n/funktsionalnaya-model-smart-kontrakta na-platforme-ethereum (Дата звернення: 13.04.2022 р.).

5. Карпичов, В.Ю. Функціональне моделювання (IDEF0) як метод дослідження блокчейн технології/В.Ю. Карпичев // Праці НДТУ. 2018 року. – № 4. – З. 22-32.

6. Алієв, І. А. Уразливості смарт-контрактів блокчейн-платформи Ethereum // Наукові записки молодих дослідників. 2019. №3. URL: https://cyberleninka.ru/article/n/uyazvimosti-smart-kontraktov-blokcheynplatformy-ethereum (дата звернення: 26.02.2022 р.).

7. J. Liu та Z. Liu, «A Survey on Security Verification of Blockchain Smart Contracts, in IEEE Access, vol. 7, pp. 77894-77904, 2019, doi: 10.1109/ACCESS.2019.2921624.

8. S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han і F. Wang, «Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends,»в IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 49, no. 11, pp. 2266-2277, Nov. 2019, doi: 10.1109/TSMC.2019.2895123.

9. Haraka Hewa, Mika Ylianttila, Madhusanka Liyanage, Survey on blockchain засновані на smart contracts: Applications, opportunities and challenges, Journal of Network and Комп'ютер Applications, Volume 177, 2021.

10. Herian R. та ін. NFT-Legal Token Classification. – EU Blockchain Observatory & Forum, 2021

11. The Best Web Application Development Languages in 2022 [Електронний ресурс]. URL: https://inveritasoft.com/blog/the-best-web-application-development-languages (дата звернення: 02.04.2022 р).

12. Saks E. JavaScript Frameworks: Angular vs React vs Vue. – 2019.

13. React. JavaScript-бібліотека для створення інтерфейсів користувача. [Електронний ресурс]. URL: https://ua.reactjs.org (дата звернення: 03.04.2022 р).

14. web3.js Ethereum JavaScript API [Електронний ресурс]. URL: https://web3js.readthedocs.io/en/v1.7.3/ (дата звернення: 03.04.2022 р).