Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

ЗАСТОСУВАННЯ БЛОКЧЕЙН-ТЕХНОЛОГІЇ ДЛЯ СТВОРЕННЯ ЕСКРОУ РАХУНКІВ

**Текстова частина до курсової роботи
за спеціальністю "Комп'ютерні науки і інформаційні технології" 6.050103**

Керівник курсової роботи
к.т.н., доц. Невмержицький Є. І.

_____

"____" _____ 2020 р.

Виконав студент Кушка М. О.
"____" _____ 2020 р.

Київ 2020

ЗМІСТ

1 ВСТУП

На сьогодні увесь процес купівлі-продажу нерухомості відбувається за допомогою відкриття ескроу-рахунку у банку. Спершу потенційний покупець перераховує необхідну суму грошей на такий ескроу рахунок, де він вже не має доступ до грошей, а процес контролюється посередником. Після успішного закріплення прав на нерухомість за покупцем посередник "вивільняє" кошти, що знаходяться на ескроу-рахунку та продавець їх отримує. Таким чином відбувається купівля нерухомості. Процес також зображений на рисунку Рисунок *1.1*.

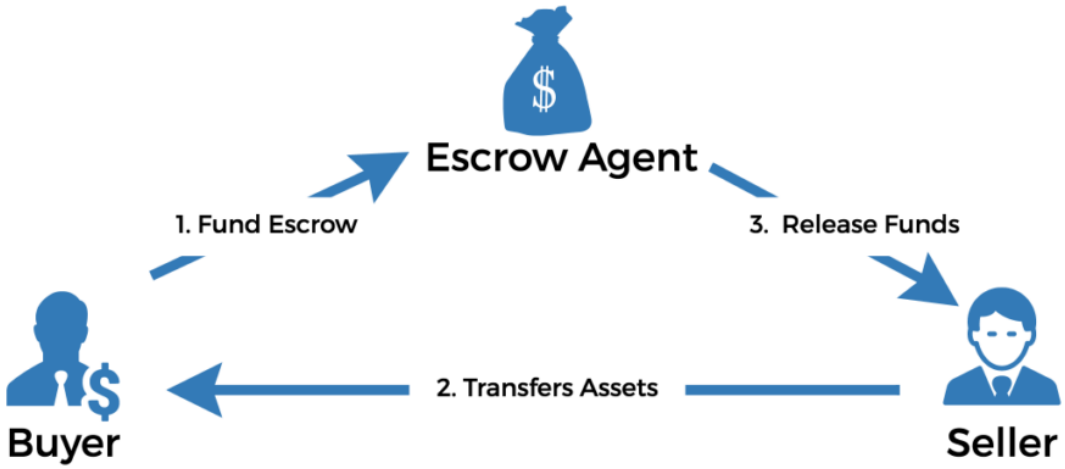

Рисунок 1.1 – Типове використання ескроу-рахунку

Ідеєю роботи є застосувати блокчейн до описаного вище процесу купівлі-продажу нерухомості, а саме створення та ведення ескроу-рахунку на блокчейні. Таким чином сторонам можна не хвилюватися щодо надійності та чесності банку, у якому відкритий такий рахунок, а також, що не менш важливо, спростити весь процес купівлі-продажу нерухомості.

Як результат застосування розробленого застосунку сторони матимуть можливість (за участю посередника) легко та надійно здійснити купівлю-продаж нерухомості у криптовалюті. Використання блокчейну також надасть можливість публічно переглядати факт продажу нерухомості однією стороною та купівлю іншою. Таким чином, за сприяння державних органів, можливо поступово частково або повністю перевести реєстр права власності на нерухоме майно до блокчейну.

2 БЛОКЧЕЙН

Блокчейн – це децентралізована база даних, або як його ще називають – розподілений реєстр. Технологію блокчейн запропоновала особа, або група осіб відома під псевдонімом Сатосі Накамото у 2009 році. Сатоші описав теоретично правила взаємодії між вузлами мережі, а також правила, за якими мусить досягатися консенсус [1].

Основна мета технології блокчейн полягає у відмови від єдиної точки обробки інформації, таким чином позбавляючись від єдиної точки відмови, або такої, яка може бути заблокована чи контрольована цензорами.

Кожен користувач системи може (за певний кошт) ініціювати транзакцію, що є подібною з транзакціями у банківській системі, або до транзакції у базі даних. Після створення транзакції її ініціатору необхідно зробити підтвердження того, що ця транзакція є його волевиявленням, а не кого-небуть іншого. Це досягається завдяки асиметричній криптографії. А саме: користувач генерує пару публічний-приватний ключ, з якої від використовує приватний ключ для підписання транзакції. Асиметрична криптографія працює таким чином, що будь-хто маючи публічний ключ користувача (а викладення публічного ключа у загальний доступ є гарною практикою), а також повідомлення, що підписане приватним ключем може перевірити, що користувач, що підписав повідомлення дійсно володіє приватним ключем від конкретного публічного ключа.

Таким чином після підписання транзакції приватним ключем, та зробивши публічний ключ загальнодоступним, користувач може поширювати підписану транзакцію. Він її, зазвичай, поширює між одним або декількома вузлами децентралізованої системи, що відповідають за обробку транзакцій. Після отримання таких транзакцій вузол (або вузли) розповсюджує дане повідомлення між іншими вузлами. Таким чином кожен вузол, що підключений до системи мусить отримати таке повідомлення.

На наступному етапі транзакція мусить бути додана у блок (тоді вона набуде статусу підтвердженої). Проте який саме вузол буде створювати наступний блок у системі диктується консенсусом [2]. Наприклад, у криптовалюті біткойн це Proof-of-Work (PoW), тому розглянемо подальші дії над транзакцією саме на прикладі цього консенсусу.

Згідно з консенсусом PoW кожен вузол, що бере участь у процесі майнингу [3] мусить сформувати блок, що не перевищує певний розмір.

Формується блок з будь-яких транзакцій мережі, що ще не були додані до блоків раніше (зазвичай майнер обирає такі транзакції, де користувачі йому платять найбільше). Після формування такого блоку кожним вузлом, власне, і починається робота. Майнер обирає довільне число, конкатенує його з хешем блоку, що ним щойно був сформований та бере хеш від отриманого значення. Процес повторюється доти, доки один з майнерів не знайде такий блок з випадковим числом, що хеш від цього значення буде починатися з N нулів, де N – автоналаштовуваний параметр системи.

Знайшовши таке число, майнер розповсюджує мережею цей блок з числом, щоб інші вузли могли впевнитися, що хеш дійсно відповідає правилам системи. Також слід зазначити наступне:

– кожен вузол не лише генерує блоки та перевіряє блоки інших, а й валідує транзакції. Таким чином будь-яка транзакція, що була зроблена проти правил системи не додається до блоку;

– коли майнер формує блок, хеш якого він буде підбирати, майнер додає першою транзакцією переказ з нуль-адреси (тобто адреса, що складається лише з нулів) на його адресу N біткойнів, де N – число, що автоматично формується системою. В такому випадку така транзакція теж вважатиметься валідною, якщо майнеру поталанить розв'язати задачу PoW.

Отже після додавання підписаної транзакції у блок така транзакція вважаються підтвердженою. Якщо ця транзакція була на перерахування коштів, то, відповідно, кошти можна вважати перерахованими. Є лише один нюанс: для більшої гарантії (у випадку потенційної атаки на блокчейн) може бути необхідним, щоб транзакцію додали не в один блок, а, скажімо, в шість.

Поясню детальніше що мається на увазі. Слід зазначити, що блок, окрім транзакцій користувачів, також містить хеш попереднього блоку, таким чином гарантуючи неможливість зміни попереднього блоку, який, у свою чергу, містить посилання на попередній блок і т. д. Таким чином і формується блокчейн (з англ. block chain, тобто ланцюг з блоків – див. рис. 1). Тобто якщо транзакція додана до шести блоків, то це означає, що вона додана до блоку, хеш якого міститься в наступному блоці, і т. д., щоб таких блоків було сумарно шість. Відповідно, у разі атаки на систему, атакуючому потрібно змінити шостий блок з кінця (вирішивши заново задачу PoW), після чого наступний блок теж доведеться перераховувати (бо хеш попереднього блоку було змінено) і т. д. шість разів. У випадку з біткойном, якщо транзакцію було

додано до шести блоків, то, незалежно від суми транзакції, рентабельність злому зводиться до нуля.

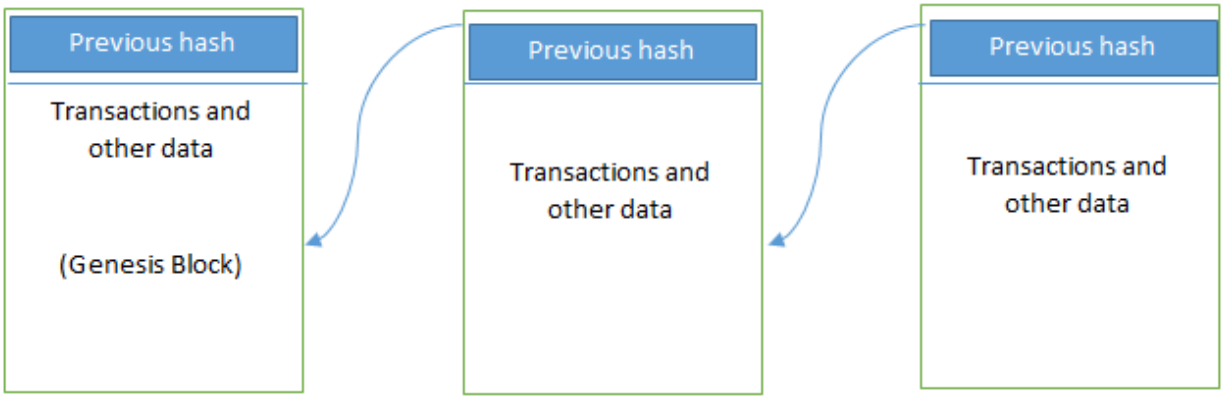

Рисунок 2.1 – Структура блокчейну [4]

3 СМАРТ-КОНТРАКТИ

Смарт-контрактом є програма, що виконується на блокчейні. Таким чином такі програми отримують додаткові переваги, що має технологія блокчейн. До переваг смарт-контрактів у порівнянні зі звичайними програмами належать:

- незмінність програмного коду (втім, зважаючи на нововведення щодо використання проксі-контрактів це не є зовсім вірно);
- надійність виконання (гарантується механізмом консенсусу відповідного блокчейну);
- передбачуване виконання;
- публічний доступ вихідного коду;
- упевненість у відповідності роботи смарт-контракту до його вихідного коду;
- можливість проводити аудит будь-якій особі чи організації.

Втім, разом з перевагами додаються і наступні недоліки:

- відсутність гарантій щодо можливості змінити програмний код смарт-контракту у разі зміні бізнес-вимог, чи виявленні вразливості у коді тощо;
- необхідність платити майнерам за кожну зміну стану смарт-контракту (за кожну функцію, що не повертає дані з смарт-контракту, а натомість оновлює певні змінні);
- необхідність чекати на додавання транзакції зі зміни стану смарт-контракту до блоку, що (на деяких блокчейнах та за певних умов) може затягнутися на багато годин, або навіть днів.

4 ПРАВОВІ АСПЕКТИ КРИПТОВАЛЮТ ТА ТЕХНОЛОГІЇ БЛОКЧЕЙН

Останнім часом в усьому світі стали надзвичайно поширеними різноманітні криптовалюти. Наразі згідно з даними криптовалютної біржі CoinMarketCap нараховується більше, ніж дві тисячі криптовалют [5]. У багатьох країнах світу вже є законодавство, що чітко визначає правовий статус та порядок обігу криптовалют. В Україні законодавче регулювання статусу криптовалют на даний момент, на жаль, відсутнє.

В Україні криптовалюти часто прирівнюють до нематеріальних активів. Наприклад, організація BRDO [6] у цьому зв'язку посилалася, зокрема, на підп. 14.1.120 п. 14.1 ст. 14 Податкового кодексу України [7]. Проте цей підпункт було виключено з Податкового кодексу України з 1 січня 2017 р. на підставі Закону України від 21.12.2016 р. № 1797-VIII. НБУ у «Спільній заяві фінансових регуляторів щодо статусу криптовалют в Україні» від 30.11.2017 р. стверджує, що «складна правова природа криптовалют не дозволяє визнати їх ані грошовими коштами, ані валютою і платіжним засобом іншої країни, ані валютною цінністю, ані електронними грошима, ані цінними паперами, ані грошовим сурогатом» [9]. Проте, згідно з визначенням електронних грошей на сайті НБУ: «електронні гроші (e-money) – одиниці вартості, які зберігаються на електронному пристрої, приймаються як засіб платежу іншими особами, ніж особа, яка їх випускає, і є грошовим зобов'язанням цієї особи, що виконується в готівковій або безготівковій формі» [10].

Оскільки Україна 27.06.2014 р. підписала Угоду про асоціацію з ЄС і вже почала поступово адаптувати своє законодавство відповідно до вимог законодавства ЄС, вважаємо доцільним розглянути відповідні рамкові документи ЄС з приводу нематеріальних активів. Так, згідно з § 1.3 «Рамкового документу щодо державної допомоги на наукові дослідження, технічний розвиток та провадження інноваційної діяльності» ЄС (2014/C 198/01) «нематеріальні активи означають такі активи, які не мають матеріального або фінансового втілення, такі як патенти, ліцензії, ноу-хау та інша інтелектуальна власність» [8]. Тобто згідно з даним документом криптовалюти не можна вважати нематеріальним активом, оскільки вони мають чітко виражене фінансове втілення.

В якості відомого прикладу розглянемо криптовалюту Bitcoin, оскільки вона має найбільшу ринкову капіталізацію згідно з даними CoinMarketCap [5].

Можна побачити, що дана криптовалюта має багато властивостей, які притаманні фіатним грошам (гроші, цінність яких забезпечується лише державою, а не власною вартістю).

По-перше, це чіткі правила емісії, що визначені правилами протоколу Bitcoin. Правила протоколу – це правила, за якими приймаються нові транзакції у блокчейні (розподілений реєстр даних, на якому базується кожна криптовалюта). Ними забезпечується передбачувана кількість криптовалюти, що доступна на конкретний момент часу. По-друге, у Bitcoin є визначений курс щодо гривні та інших валют. Так, найбільша в Україні криптовалютна біржа Kuna дозволяє здійснювати такий обмін за допомогою їхнього веб-сайту, а також своїх банкоматів, де можна безпосередньо отримати гривню готівкою за криптовалюту та купити криптовалюту за гривню.

Криптовалюта Bitcoin працює за алгоритмом консенсусу (процес, що використовується для досягнення згоди з приводу певної одиниці даних у розподіленій мережі) PoW (Proof-Of-Work), ідея якого полягає у тому, що для можливості підтвердити транзакції в блокчейні і таким чином отримати винагороду, необхідно перебирати випадкові числа з метою знайти потрібне. Імовірність знаходження цього числа корелює лише з потужністю комп'ютера чи комп'ютерів, що працюють над цією задачею. Таким чином, особа або група осіб (майнер), що знайш*ов(ла)* це число, не зацікавлен*ий(ні)* у тому, щоб підтверджувати транзакцію, що була здійснена проти цих правил, оскільки в такому випадку він(вони) не отрима*є(ють)* винагороду. Ця особливість Bitcoin забезпечує надзвичайно високу чесність усієї системи. Підтвердженням цього може слугувати той факт, що для цієї криптовалюти не було жодного разу успішно здійснено атаку подвійного використання коштів, що є суттєвою перевагою Bitcoin порівняно з іншими криптовалютами. Це дозволяє протягом більш ніж десять років успішно використовувати Bitcoin як засіб оплати або конвертації в фіатні гроші та відкриває великий потенціал щодо ще ширшого використання його у майбутньому.

Розглянемо також досвід регулювання криптовалют деякими відомими державами, що є лідерами світової банківської системи – Швейцарію та Німеччину. Так, згідно з пп. 3.1 п. 3 документу FINMA (Органу нагляду за фінансовими ринками Швейцарії) щодо регуляції ICO «платіжні токени (або криптовалюти) – це токени, що призначені для використання зараз чи у майбутньому як засіб платежу для придбання товарів та послуг, або як

грошова одиниця, або як обмін цінностей» [11]. Тобто, згідно з швейцарським законодавством, криптовалюти повністю прирівняні до фіатних грошей. Більш того, у Швейцарії (наприклад у муніципалітеті Кьяссо) Bitcoin приймається як засіб сплати податків [12], що свідчить про їх впевненість у цій криптовалюті. У Законі про кредитну систему (KWG) Німеччини криптовалюти визначені як одиниці обліку, а отже, є фінансовими інструментами [13].

Проте, різні криптовалюти використовують різні алгоритми консенсусу, серед яких є як більш, так і менш надійні. Одним з факторів, що впливають на надійність криптовалюти, є кількість вузлів, які підтверджують транзакції. Чим більше вузлів має блокчейн, тим надійнішою є криптовалюта. Так, Bitcoin має на сьогодні більше одного мільйона незалежних майнерів з усього світу, що виключає можливість домовленості між ними з метою вчинення будь-яких протиправних дій, і це, разом з іншими факторами, забезпечує його надійність. А, наприклад, така криптовалюта як EOS має лише 21 валідатор, що суттєво погіршує її надійність. Кількість валідаторів – лише один з факторів надійності криптовалют. Але цей критерій не є притаманним ані фіатним, ані електронним грошам.[14]

5 ОПИС ПРОГРАМНОГО ПРОДУКТУ

Програмний продукт складається зі смарт-контрактів, що написані для Ефіріум блокчейну та веб-застосунку для взаємодії з ними.

5.1 Смарт-контракти

Код смарт-контрактів написаний мовою програмування Solidity, що є основною мовою програмування у Ефіріум блокчейні. Смарт-контракт на Solidity за конструкцією схожий на клас у звичайних мовах програмування. Усього було створено два контракти: Escrow та Registry.

Контракт Escrow призначений для створення та ведення ескроу-рахунків, а також містять функціонал достатній для проведення процесу купівлі-продажу нерухомості.

Контракт Registry має на меті вести облік усіх контрактів Escrow. Таким чином за допомогою цього контракту можна знайти різні пропозиції купівлі-продажу нерухомості, що дозволяє побудувати відповідний веб-інтерфейс.

Опис функцій наведено смарт-контрактів наведено у таблицях Таблиця *5.1* та .

Таблиця 5.1 – Опис функцій смарт-контракту Escrow

Номер	Назва	Чи є публічною?	Опис
1	initialize	так	Ініціалізація основних параметрів контракту: – ефіріум-адреса продавця; – ефіріум-адреса посередника; – ефіріум-адреса покупця – адреса будинку; – ціна будинку.
2	becomeBuyer	так	Функція для того, щоб стати покупцем, якщо у будинку ще відсутній покупець. Покупцем стає той, хто викликає функцію (за умови виконання вимог).

3	removeBuyer	так	Видалення покупця за умови завершення часу на купівлю будинку (3 дні).
4	transferFundsToSeller	ні	Функція, що доступна лише посереднику. Посередник мусить перевірити (за межами блокчейну, що право власності було успішно передано від продавця до покупця), після чого викликати цю функцію, що перерахує гроші з ескроу-рахунку до продавця.
5	- (отримання коштів)	так	Функція для отримання Ефірів смарт-контрактом.
6	_payBackToBuyer	ні	Внутрішня функція. Перераховує кошти назад покупцеві (використовується функцією removeBuyer).

Таблиця 5.2 - Опис функцій смарт-контракту Registry

Номер	Назва	Чи є публічною?	Опис
1	getNumOfEscrow	так	Отримання кількості ескроу-контрактів, про які знає реєстр.
2	escrowAddrs	так	Отримання адреси ескроу-контракту за індексом у масиві. Використовується для перебору усіх адрес, а функція getNumOfEscrow() допомагає не вийти за межі масиву.
3	addEscrow	так	Додавання адреси ескроу-контракту до списку контрактів.

Смарт-контракт Escrow.sol було розроблено як оновлюваний, що означає, що його код у майбутньому можна буде оновити за необхідності (проте, з деякими обмеженнями).

Також для розроблення якісного смарт-контракту гарною практикою є стовідсоткове покриття його тестами. Це робиться через критичність знаходження багів на етапі роботи смарт-контракту. Опис та результат проходження тестів див. на рисунку Рисунок *5.1*.

```
Contract: Escrow
  Initialize
    ✓ Should be initialized with correct values (197ms)
  Become a buyer
    ✓ Cannot become a buyer if there is already a buyer (130ms)
    ✓ Cannot become a buyer if the house was already sold (228ms)
    ✓ Should become a buyer successfully (103ms)
  Remove a buyer
    ✓ Cannot remove a buyer if it still has time to pay (144ms)
    ✓ Cannot remove a buyer if the house is already sold (213ms)
    ✓ Should remove a buyer if there is no buyer (96ms)
    ✓ Should remove a buyer if time to pay has passed (291ms)
  Receive funds
    ✓ Non-buyer cannot send Ethers to the contract (63ms)
    ✓ Only buyer can send Ethers to the contract (101ms)
  Transfer funds to the seller
    ✓ Cannot transfer funds if the buyer has not payed enough (133ms)
    ✓ Cannot transfer funds if there is no buyer (54ms)
    ✓ Should transfer all funds to the seller (378ms)


  13 passing (3s)

✨  Done in 12.86s.
```

Рисунок 5.1 – Результати проходження тестів до смарт-контракту Escrow.sol

5.2 Веб-застосунок

У доповнення до системи контрактів було розроблено графічний інтерфейс, що дозволяє користувачам переглядати пропозиції з купівлі нерухомості та ставати учасниками відповідних контактів. Графічний інтерфейс відображає усі зміни у контрактах, а тому є актуальним та зручним.

На Рисунок *5.2* та Рисунок *5.3* можна побачити вигляд головної сторінки веб-сайту та сторінку з описом конкретного будинку відповідно.

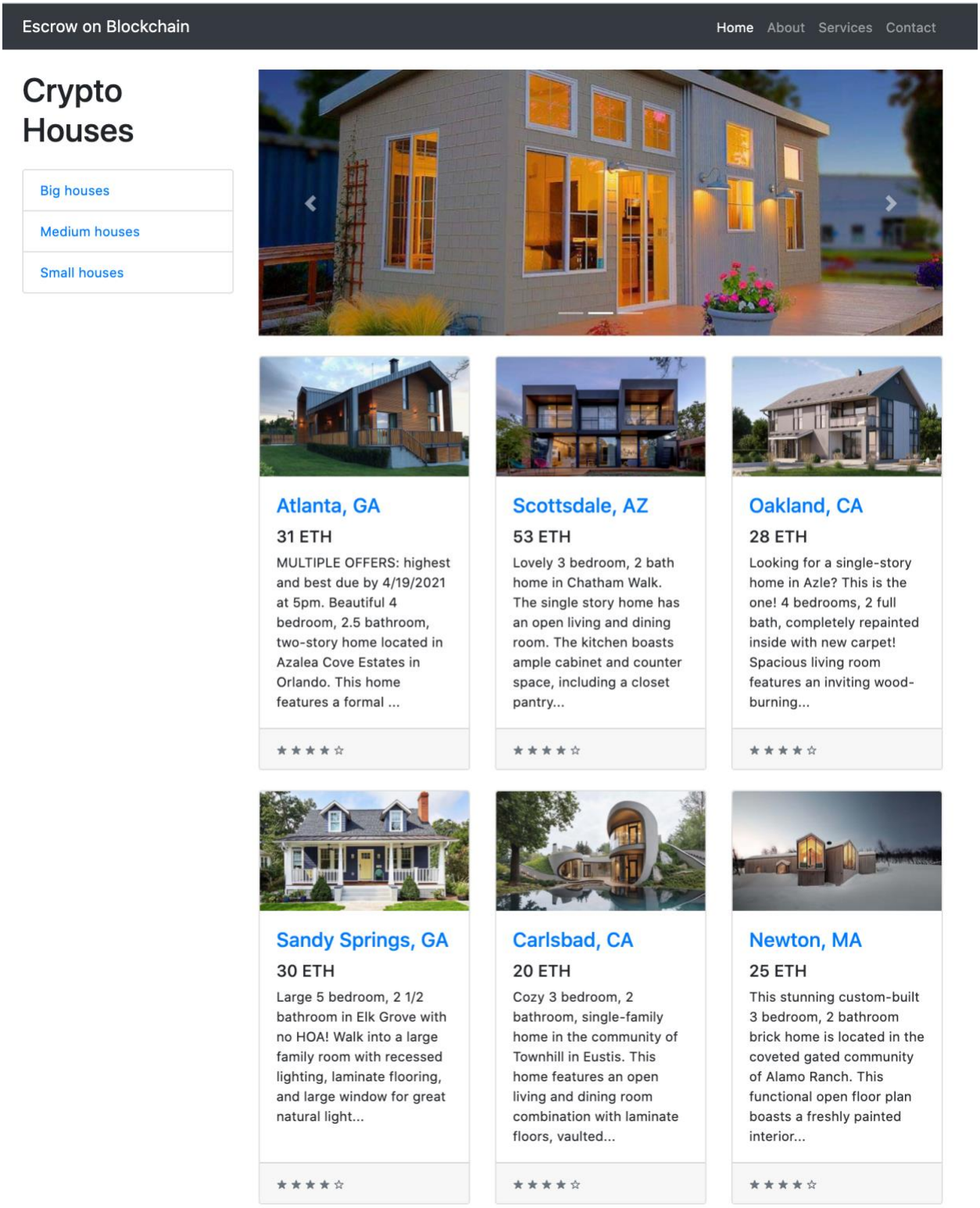

Рисунок 5.2 - Головна сторінка для пошуку будинку

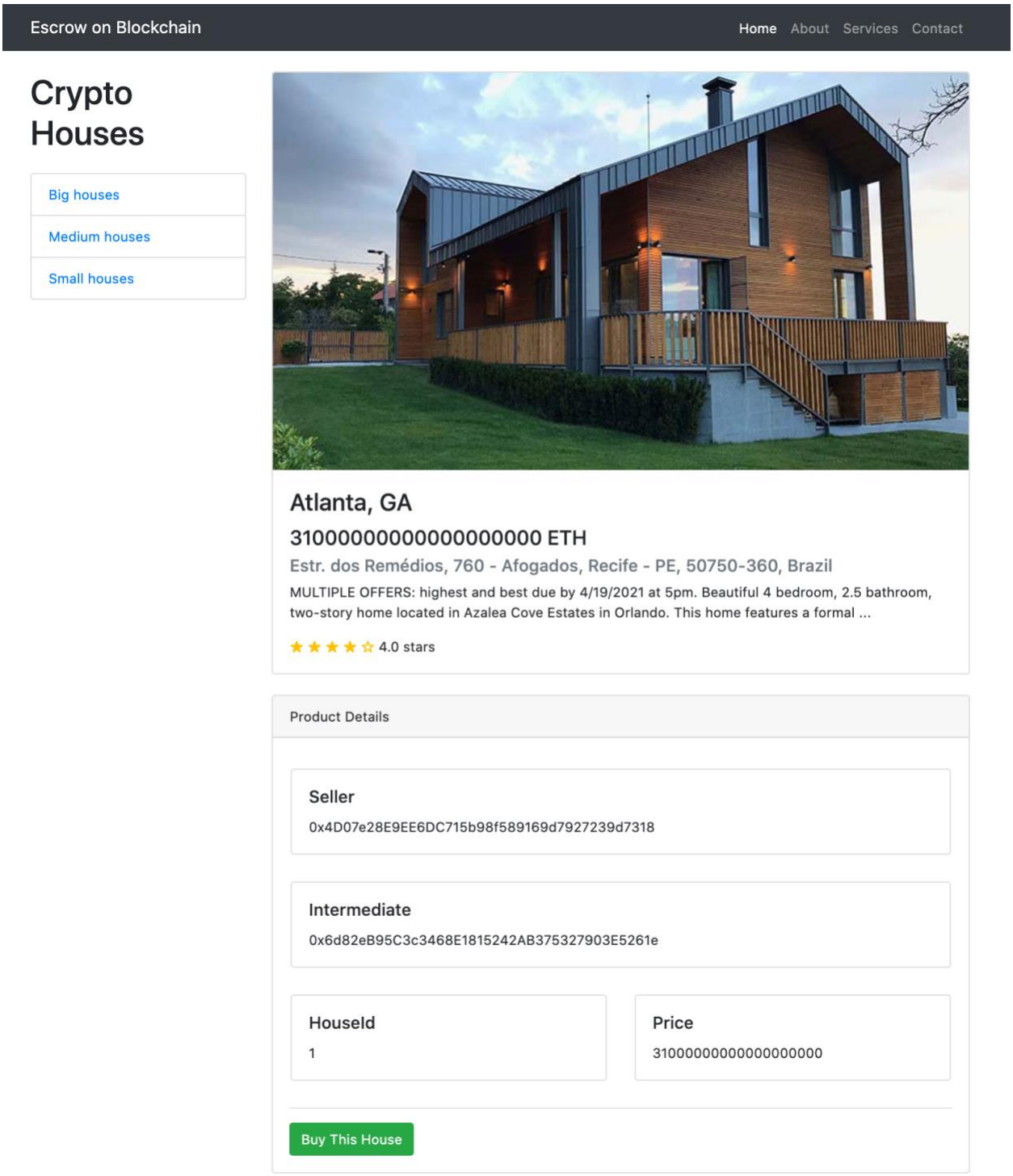

Рисунок 5.3 - Сторінка з детальною інформацією по будинку

6 ВИКОРИСТАНІ ТЕХНОЛОГІЇ

6.1 Розробка смарт-контрактів

Як мова розробки смарт-контрактів була обрана Solidity, що є основною мовою програмування у Ефіріум блокчейні.

6.2 Розгортання у блокчейні

Для успішного розгортання смарт-контрактів був використаний NodeJS-пакет "truffle". Він дозволяє достатньо просто компілювати та розгортати смарт-контракти, що розробляються.

6.3 Оновлювані смарт-контракти

Також був використаний пакет "@openzeppelin/truffle-unpgrades" для можливості розгортання через "truffle" одразу оновлюваних смарт-контрактів.

6.4 Тестування

Для тестування була використана мова програмування NodeJS разом з двома пакетами для тестування: "mocha" та "chai".

6.5 Веб-сайт

Для розробки графічної частини проєкту були використані наступні технології:

- NodeJS (для розробки серверної частини застосунку);
- HTML (для розробки сторінок веб-сайту);
- CSS (для стилізації сторінок веб-сайту);
- JavaScript (для логіки роботи веб-сайту);
- web3.js (для взаємодії зі смарт-контрактами у Ефіріум блокчейні).

7 КОД ПРОГРАМНОГО ПРОДУКТУ

Нижче наведено код програмного продукту, але його також можна подивитися на моєму GitHub [15].

7.1 Смарт-контракти

contracts/Escrow.sol

```
// SPDX-License-Identifier: UNLICENSED

pragma solidity ^0.6.12;


import "@openzeppelin/contracts-ethereum-package/contracts/Initializable.sol";


contract Escrow is Initializable {
    address payable public seller;
    address public intermediate;
    address payable public buyer;
    uint256 public price;


    bool public isSold;
    uint256 public houseId;
    string public houseAddress;
    uint256 public payTimeLimit;


    modifier onlyIntermediate() {
        require(msg.sender == intermediate, "Only intermediate can call the function");
        _;
    }


    event NewBuyer(address buyer);
    event RemoveBuyer(address buyer);
    event TransferFundsToSeller(address seller, uint256 value);
    event Payment(address payer, uint256 value);
```

```
function initialize(
    address payable seller_,
    address intermediate_,
    uint256 houseId_,
    string memory houseAddress_,
    uint256 price_
) public initializer {
    isSold = false;
    seller = seller_;
    intermediate = intermediate_;
    houseId = houseId_;
    houseAddress = houseAddress_;
    price = price_;
}

function becomeBuyer() public {
    require(buyer == address(0), "There is alredy a buyer");
    require(isSold == false, "The house is already sold");
    payTimeLimit = block.timestamp + 3 days;
    buyer = msg.sender;

    emit NewBuyer(buyer);
}

function removeBuyer() public {
    require(block.timestamp >= payTimeLimit, "The buyer still has time to pay");
    require(isSold == false, "The house is already sold");

    address payable oldBuyer = buyer;
    buyer = address(0);
    _payBackToBuyer(oldBuyer);
```

```
        emit RemoveBuyer(oldBuyer);
    }

    function transferFundsToSeller() onlyIntermediate public {
        require(address(this).balance >= price, "The buyer has not payed enough");
        isSold = true;
        emit TransferFundsToSeller(seller, address(this).balance);
        seller.transfer(address(this).balance);
    }

    receive() external payable {
        require(msg.sender == buyer, "Only buyer can send Ether");
        emit Payment(msg.sender, msg.value);
    }

    function _payBackToBuyer(address payable oldBuyer) internal {
        oldBuyer.transfer(address(this).balance);
    }
}
```

contracts/Registry.sol

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.6.12;

contract Registry {
    address[] public escrowAddrs;

    function getNumOfEscrow() public view returns(uint256) {
        return escrowAddrs.length;
    }

    function addEscrow(address _escrow) public {
        require(_escrow != address(0), "Escrow should be non-zero address");
```

```
        escrowAddrs.push(_escrow);
    }
}
```

contracts/Migrations.sol

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.8.0;

contract Migrations {
  address public owner = msg.sender;
  uint public last_completed_migration;

  modifier restricted() {
    require(
      msg.sender == owner,
      "This function is restricted to the contract's owner"
    );
    _;
  }

  function setCompleted(uint completed) public restricted {
    last_completed_migration = completed;
  }
}
```

migrations/1_initial_migration.js

```
const Migrations = artifacts.require('Migrations')

module.exports = (deployer) => {
  deployer.deploy(Migrations)
}
```

migrations/2_deploy_escrow.js

```
const { deployProxy } = require('@openzeppelin/truffle-upgrades')

const Escrow = artifacts.require('Escrow')
```

```
module.exports = async function (deployer) {
  const arguments = [
    '0x4D07e28E9EE6DC715b98f589169d7927239d7318',
    '0x4D07e28E9EE6DC715b98f589169d7927239d7318',
    '123',
    '401 Seventh Avenue, New York, NY',
    '1000000000000000000',
  ]
  const instance = await deployProxy(Escrow, arguments, { deployer })
  console.log('Deployed', instance.address)
}
```

test/Escrow.test.js

```
const { expect } = require('chai')
const { BN, expectEvent, expectRevert } = require('@openzeppelin/test-helpers')
const { ZERO_ADDRESS } = require('@openzeppelin/test-helpers/src/constants')
const timeMachine = require('ganache-time-traveler')
const web3 = require('../utils/test')

const Escrow = artifacts.require('Escrow')

const SECONDS_IN_DAY = 86400

contract('Escrow', ([seller, intermediate, buyer, other]) => {
  before(async () => {
    this.houseId = '123'
    this.houseAddress = '401 Seventh Avenue, New York, NY'
    this.price = new BN(web3.utils.toWei('100', 'ether'))
    this.escrow = await Escrow.new()
    await this.escrow.initialize(seller, intermediate, this.houseId, this.houseAddress, this.price)
  })

  beforeEach(async () => {
    const snapshot = await timeMachine.takeSnapshot()
    this.snapshotId = snapshot.result
  })
```

```
afterEach(async () => {
  await timeMachine.revertToSnapshot(this.snapshotId)
})

describe('Initialize', async () => {
  it('Should be initialized with correct values', async () => {
    expect(await this.escrow.seller()).to.equal(seller)
    expect(await this.escrow.intermediate()).to.equal(intermediate)
    expect(await this.escrow.buyer()).to.equal(ZERO_ADDRESS)
    expect(await this.escrow.houseId()).to.be.bignumber.equal(new BN(this.houseId))
    expect(await this.escrow.houseAddress()).to.equal(this.houseAddress)
    expect(await this.escrow.price()).to.be.bignumber.equal(this.price)
  })
})

describe('Become a buyer', async () => {
  it('Cannot become a buyer if there is already a buyer', async () => {
    await this.escrow.becomeBuyer({ from: buyer })
    await expectRevert(this.escrow.becomeBuyer({ from: other }), 'There is alredy a buyer')
  })

  it('Cannot become a buyer if the house was already sold', async () => {
    // Selling a house
    await this.escrow.becomeBuyer({ from: buyer })
    await this.escrow.send(this.price, { from: buyer })
    await this.escrow.transferFundsToSeller({ from: intermediate })

    await expectRevert(this.escrow.becomeBuyer({ from: other }), 'There is alredy a buyer')
  })

  it('Should become a buyer successfully', async () => {
    await expectEvent(await this.escrow.becomeBuyer({ from: buyer }), 'NewBuyer', { buyer })
    expect(await this.escrow.buyer()).to.equal(buyer)
  })
})

describe('Remove a buyer', async () => {
```

```
it('Cannot remove a buyer if it still has time to pay', async () => {
  await this.escrow.becomeBuyer({ from: buyer })
  await expectRevert(this.escrow.removeBuyer(), 'The buyer still has time to pay')
})

it('Cannot remove a buyer if the house is already sold', async () => {
  // Selling a house
  await this.escrow.becomeBuyer({ from: buyer })
  await this.escrow.send(this.price, { from: buyer })
  await this.escrow.transferFundsToSeller({ from: intermediate })

  await timeMachine.advanceTimeAndBlock(3 * SECONDS_IN_DAY)

  await expectRevert(this.escrow.removeBuyer(), 'The house is already sold')
})

it('Should remove a buyer if there is no buyer', async () => {
  const before = { buyer: await this.escrow.buyer() }

  await expectEvent(await this.escrow.removeBuyer(), 'RemoveBuyer', { buyer: ZERO_ADDRESS })

  const after = { buyer: await this.escrow.buyer() }

  expect(after.buyer).to.equal(ZERO_ADDRESS)
  expect(after.buyer).to.equal(before.buyer)
})

it('Should remove a buyer if time to pay has passed', async () => {
  const value = new BN(web3.utils.toWei('1', 'ether'))

  await this.escrow.becomeBuyer({ from: buyer })
  await this.escrow.send(value, { from: buyer })

  const before = {
    buyer: await this.escrow.buyer(),
    buyerBalance: new BN(await web3.eth.getBalance(buyer)),
    contractBalance: new BN(await web3.eth.getBalance(this.escrow.address)),
```

```
    }

    await timeMachine.advanceTimeAndBlock(3 * SECONDS_IN_DAY)

    await expectEvent(await this.escrow.removeBuyer(), 'RemoveBuyer', { buyer })
    await this.escrow.becomeBuyer({ from: other })

    const after = {
      buyer: await this.escrow.buyer(),
      buyerBalance: new BN(await web3.eth.getBalance(buyer)),
      contractBalance: new BN(await web3.eth.getBalance(this.escrow.address)),
    }

    expect(before.buyer).to.equal(buyer)
    expect(after.buyer).to.equal(other)
    expect(after.buyerBalance.sub(before.buyerBalance)).to.be.bignumber.equal(value)
    expect(before.contractBalance.sub(after.contractBalance)).to.be.bignumber.equal(value)
  })
})

describe('Receive funds', async () => {
  it('Non-buyer cannot send Ethers to the contract', async () => {
    const before = {
      contract: await web3.eth.getBalance(this.escrow.address),
    }
    await expectRevert(this.escrow.send(web3.utils.toWei('1', 'ether'), { from: other }), 'Only buyer can send Ether')
    const after = {
      contract: await web3.eth.getBalance(this.escrow.address),
    }

    expect(after.contract, 'Contract balance should not change').to.equal(before.contract)
  })

  it('Only buyer can send Ethers to the contract', async () => {
    const beforeContractBalance = new BN(await web3.eth.getBalance(this.escrow.address))
    const value = new BN(web3.utils.toWei('1', 'ether'))
```

```
    await this.escrow.becomeBuyer({ from: buyer })
    await expectEvent(await this.escrow.send(value, { from: buyer }), 'Payment', { payer: buyer, value })

    const afterContractBalance = new BN(await web3.eth.getBalance(this.escrow.address))

    expect(afterContractBalance.sub(beforeContractBalance)).to.be.bignumber.equal(value)
  })
})

describe('Transfer funds to the seller', async () => {
  it('Cannot transfer funds if the buyer has not payed enough', async () => {
    const lessThanPrice = this.price.sub(new BN(web3.utils.toWei('1', 'ether')))
    await this.escrow.becomeBuyer({ from: buyer })
    await this.escrow.send(lessThanPrice, { from: buyer })
    await expectRevert(this.escrow.transferFundsToSeller({ from: intermediate }), 'The buyer has not payed enough')
  })

  it('Cannot transfer funds if there is no buyer', async () => {
    await expectRevert(this.escrow.transferFundsToSeller({ from: intermediate }), 'The buyer has not payed enough')
  })

  it('Should transfer all funds to the seller', async () => {
    const beforeSellerBalance = new BN(await web3.eth.getBalance(seller))
    const moreThanPrice = this.price.add(new BN(web3.utils.toWei('1', 'ether')))

    await this.escrow.becomeBuyer({ from: buyer })
    await this.escrow.send(moreThanPrice, { from: buyer })
    await expectEvent(
      await this.escrow.transferFundsToSeller({ from: intermediate }),
      'TransferFundsToSeller',
      { seller, value: moreThanPrice }
    )

    const afterSellerBalance = new BN(await web3.eth.getBalance(seller))
    const isSold = await this.escrow.isSold()

    expect(isSold).to.equal(true)
```

```
    expect(afterSellerBalance.sub(beforeSellerBalance)).to.be.bignumber.equal(moreThanPrice)
   })
  })
})
```

7.1 Веб-сайт

server.js

```
const express = require('express')
const path = require('path')
const bodyParser = require('body-parser')

const app = express()
const port = 3000

app.use('/', express.static(path.join(path.resolve(), 'app', 'public')))
app.use(bodyParser.urlencoded({ extended: true }))
app.use(bodyParser.json())

app.post('/deployContract', (req, res) => {
  console.log(req.body)
  res.send({ status: 'success' })
})

app.listen(port, () => console.info(`The app is listening on port ${port}`))
```

app/public/javascript/main.js

```
const web3 = new window.Web3('https://rinkeby.infura.io/v3/b2d48917ee7f4f5dbe048259c3a8a554')

async function getEscrowInstanceByAddress(address) {
  return new web3.eth.Contract(window.escrowAbi, address)
}

async function getAHouse(escrowAddr) {
  const escrow = await getEscrowInstanceByAddress(escrowAddr)

  const seller = await escrow.methods.seller().call()
```

```
  const intermediate = await escrow.methods.intermediate().call()
  const houseId = await escrow.methods.houseId().call()
  const houseAddress = await escrow.methods.houseAddress().call()
  const price = await escrow.methods.price().call()

  return {
    seller,
    intermediate,
    houseId,
    houseAddress,
    price,
  }
}

async function getHouses() {
  const registryAddr = '0x56024F89Ce457a653aa775626670B0A3aE5A44E8'
  const registry = new web3.eth.Contract(window.registryAbi, registryAddr)
  const numOfEscrow = parseInt(await registry.methods.getNumOfEscrow().call(), 10)
  console.log({ numOfEscrow })

  const ids = new Array(numOfEscrow).fill().map((_, i) => i)
  const escrowAddrs = await Promise.all(ids.map((i) => registry.methods.escrowAddrs(i).call()))

  return Promise.all(
    escrowAddrs.map(async (addr) => {
      const escrow = await getEscrowInstanceByAddress(addr)

      const seller = await escrow.methods.seller().call()
      const intermediate = await escrow.methods.intermediate().call()
      const houseId = await escrow.methods.houseId().call()
      const houseAddress = await escrow.methods.houseAddress().call()
      const price = await escrow.methods.price().call()

      return {
        seller,
        intermediate,
        houseId,
```

```
        houseAddress,
        escrowAddr: addr,
        price,
      }
    }),
  )
}

async function initialize(provider) {
  console.log(provider.isConnected())
  const accounts = await provider.request({ method: 'eth_accounts' })
  const balance = await provider.request({ method: 'eth_getBalance', params: [accounts[0]] })
  console.log({ balance })
}
```

app/public/javascript/houseInfo.js

```
window.houseInfo = {
  1: {
    title: 'Atlanta, GA',
    description:
      'MULTIPLE OFFERS: highest and best due by 4/19/2021 at 5pm. Beautiful 4 bedroom, 2.5 bathroom, two-story home located in Azalea Cove Estates in Orlando. This home features a formal ...',
  },
  2: {
    title: 'Scottsdale, AZ',
    description:
      'Lovely 3 bedroom, 2 bath home in Chatham Walk.  The single story home has an open living and dining room. The kitchen boasts ample cabinet and counter space, including a closet pantry...',
  },
  3: {
    title: 'Oakland, CA',
    description:
      'Looking for a single-story home in Azle? This is the one! 4 bedrooms, 2 full bath, completely repainted inside with new carpet! Spacious living room features an inviting wood-burning...',
  },
  4: {
```

```
    title: 'Sandy Springs, GA',
    description:
      'Large 5 bedroom, 2 1/2 bathroom in Elk Grove with no HOA! Walk into a large family room with recessed lighting, laminate flooring, and large window for great natural light...',
  },
  5: {
    title: 'Carlsbad, CA',
    description:
      'Cozy 3 bedroom, 2 bathroom, single-family home in the community of Townhill in Eustis. This home features an open living and dining room combination with laminate floors, vaulted...',
  },
  6: {
    title: 'Newton, MA',
    description:
      'This stunning custom-built 3 bedroom, 2 bathroom brick home is located in the coveted gated community of Alamo Ranch. This functional open floor plan boasts a freshly painted interior...',
  },
}
```

app/public/item.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />
    <meta name="description" content="" />
    <meta name="author" content="" />

    <title>The house</title>

    <!-- Bootstrap core CSS -->
    <link href="vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet" />

    <!-- Custom styles for this template -->
    <link href="css/shop.css" rel="stylesheet" />
  </head>
```

```
<body>
  <!-- Navigation -->
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
    <div class="container">
      <a class="navbar-brand" href="/">Escrow on Blockchain</a>
      <button
        class="navbar-toggler"
        type="button"
        data-toggle="collapse"
        data-target="#navbarResponsive"
        aria-controls="navbarResponsive"
        aria-expanded="false"
        aria-label="Toggle navigation"
      >
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarResponsive">
        <ul class="navbar-nav ml-auto">
          <li class="nav-item active">
            <a class="nav-link" href="#"
              >Home
              <span class="sr-only">(current)</span>
            </a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">About</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">Services</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">Contact</a>
          </li>
        </ul>
      </div>
    </div>
```

```
</nav>

<!-- Page Content -->
<div class="container">
  <div class="row">
    <div class="col-lg-3">
      <h1 class="my-4">Crypto Houses</h1>
      <div class="list-group">
        <a href="#" class="list-group-item">Big houses</a>
        <a href="#" class="list-group-item">Medium houses</a>
        <a href="#" class="list-group-item">Small houses</a>
      </div>
    </div>
    <!-- /.col-lg-3 -->

    <div class="col-lg-9">
      <div class="card mt-4">
        <img class="card-img-top img-fluid" id="house-image" src="#" alt="" />
        <div class="card-body">
          <h3 class="card-title" id="house-name"></h3>
          <h4 id="house-price"></h4>
          <h5 class="text-muted" id="house-address"></h5>
          <p class="card-text" id="house-description"></p>
          <span class="text-warning">&#9733; &#9733; &#9733; &#9733; &#9734;</span>
          4.0 stars
        </div>
      </div>
      <!-- /.card -->

      <div class="card card-outline-secondary my-4">
        <div class="card-header">Product Details</div>
        <div class="card-body">
          <!-- House details -->
          <div class="row" id="house-details"></div>
          <!-- /.details-->
          <hr />
          <a href="#" class="btn btn-success">Buy This House</a>
```

```
        </div>
      </div>
      <!-- /.card -->
    </div>
    <!-- /.col-lg-9 -->
  </div>
</div>
<!-- /.container -->

<!-- Footer -->
<footer class="py-5 bg-dark">
  <div class="container">
    <p class="m-0 text-center text-white">Copyright &copy; Crypto houses 2021</p>
  </div>
  <!-- /.container -->
</footer>

<!-- Bootstrap core JavaScript -->
<script src="vendor/jquery/jquery.min.js"></script>
<script src="vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/web3@latest/dist/web3.min.js"></script>
<script src="javascript/abi.js"></script>
<script src="javascript/houseInfo.js"></script>
<script src="javascript/main.js"></script>
<script>
  const capitalizeFirstLetter = (str) => str.charAt(0).toUpperCase() + str.slice(1)

  window.onload = async () => {
    const escrowAddr = document.location.href.split('#')[1]
    const house = await getAHouse(escrowAddr)
    console.log({ house })

    const houseId = house.houseId
    const details = {
      seller: house.seller,
      intermediate: house.intermediate,
      houseId,
```

```
        price: house.price,
      }

      $('title').text(name)
      $('#house-image').attr('src', `img/house-${houseId}.jpg`)
      $('#house-name').text(window.houseInfo[houseId].title)
      $('#house-price').text(`${house.price} ETH`)
      $('#house-description').text(window.houseInfo[houseId].description)
      $('#house-address').text(house.houseAddress)

      for (const [name, value] of Object.entries(details)) {
        const width = `${value}`.match(/0x/g) ? 12 : 6

        $('#house-details').append(`
        <div class="p-3 col-sm-${width}">
          <div class="card">
            <div class="card-body">
              <h5 class="card-title">${capitalizeFirstLetter(name)}</h5>
              <p class="card-text">${value}</p>
            </div>
          </div>
        </div>
      `)
      }
    }
  </script>
 </body>
</html>
```

app/public/index.html

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />
  <meta name="description" content="" />
  <meta name="author" content="" />
```

```
  <title>Crypto Houses</title>

  <!-- Bootstrap core CSS -->
  <link href="vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet" />

  <!-- Custom styles for this template -->
  <link href="css/shop.css" rel="stylesheet" />
</head>

<body>
  <!-- Navigation -->
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
    <div class="container">
      <a class="navbar-brand" href="/">Escrow on Blockchain</a>
      <button
        class="navbar-toggler"
        type="button"
        data-toggle="collapse"
        data-target="#navbarResponsive"
        aria-controls="navbarResponsive"
        aria-expanded="false"
        aria-label="Toggle navigation"
      >
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarResponsive">
        <ul class="navbar-nav ml-auto">
          <li class="nav-item active">
            <a class="nav-link" href="#"
              >Home
              <span class="sr-only">(current)</span>
            </a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">About</a>
          </li>
```

```
        <li class="nav-item">
          <a class="nav-link" href="#">Services</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Contact</a>
        </li>
      </ul>
    </div>
  </div>
</nav>

<!-- Page Content -->
<div class="container">
  <div class="row">
    <div class="col-lg-3">
      <h1 class="my-4">Crypto Houses</h1>
      <div class="list-group">
        <a href="#" class="list-group-item">Big houses</a>
        <a href="#" class="list-group-item">Medium houses</a>
        <a href="#" class="list-group-item">Small houses</a>
      </div>
    </div>
    <!-- /.col-lg-3 -->

    <div class="col-lg-9">
      <div id="carouselExampleIndicators" class="carousel slide my-4" data-ride="carousel">
        <ol class="carousel-indicators">
          <li data-target="#carouselExampleIndicators" data-slide-to="0" class="active"></li>
          <li data-target="#carouselExampleIndicators" data-slide-to="1"></li>
          <li data-target="#carouselExampleIndicators" data-slide-to="2"></li>
        </ol>
        <div class="carousel-inner" role="listbox">
          <div class="carousel-item active">
            <img class="d-block img-fluid" src="img/carousel-1.jpg" alt="First slide" />
          </div>
          <div class="carousel-item">
            <img class="d-block img-fluid" src="img/carousel-2.jpg" alt="Second slide" />
```

```
    </div>
    <div class="carousel-item">
      <img class="d-block img-fluid" src="img/carousel-3.jpg" alt="Third slide" />
    </div>
  </div>
  <a class="carousel-control-prev" href="#carouselExampleIndicators" role="button" data-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="sr-only">Previous</span>
  </a>
  <a class="carousel-control-next" href="#carouselExampleIndicators" role="button" data-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>
</div>

<div class="row" id="available-houses">
  <!-- <div class="col-lg-4 col-md-6 mb-4">
    <div class="card h-100">
      <a href="#"><img class="card-img-top" src="http://placehold.it/700x400" alt="" /></a>
      <div class="card-body">
        <h4 class="card-title">
          <a href="#">Item One</a>
        </h4>
        <h5>$24.99</h5>
        <p class="card-text">
          Lorem ipsum dolor sit amet, consectetur adipisicing elit. Amet numquam aspernatur!
        </p>
      </div>
      <div class="card-footer">
        <small class="text-muted">&#9733; &#9733; &#9733; &#9733; &#9734;</small>
      </div>
    </div>
  </div> -->
</div>
<!-- /.row -->
</div>
<!-- /.col-lg-9 -->
```

```
  </div>
  <!-- /.row -->
</div>
<!-- /.container -->

<!-- Footer -->
<footer class="py-5 bg-dark">
  <div class="container">
    <p class="m-0 text-center text-white">Copyright &copy; Crypto houses 2021</p>
  </div>
  <!-- /.container -->
</footer>

<!-- Bootstrap core JavaScript -->
<script src="vendor/jquery/jquery.min.js"></script>
<script src="vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/web3@latest/dist/web3.min.js"></script>
<script src="javascript/abi.js"></script>
<script src="javascript/houseInfo.js"></script>
<script src="javascript/main.js"></script>
<script>
  window.onload = async () => {
    const houses = await getHouses()
    console.log({ houses })

    for (let i = 0; i < houses.length; i++) {
      // const title = `House ${houses[i].houseId}`
      const { houseId } = houses[i]
      const image = `img/house-${houseId}.jpg`
      const link = `item.html#${houses[i].escrowAddr}`
      const price = `${houses[i].price / 1e18}`

      console.log({ houseId })
      console.log(window.houseInfo[houseId])

      $('#available-houses').append(
        `\
```

```
          <div class="col-lg-4 col-md-6 mb-4">\
                <div class="card h-100">\
                  <a href="${link}"><img class="card-img-top" src="${image}" alt="" /></a>\
                  <div class="card-body">\
                    <h4 class="card-title">\
                      <a href="${link}">${window.houseInfo[houseId].title}</a>\
                    </h4>\
                    <h5>${price} ETH</h5>\
                    <p class="card-text">${window.houseInfo[houseId].description}</p>\
                  </div>\
                  <div class="card-footer">\
                    <small class="text-muted">&#9733; &#9733; &#9733; &#9733; &#9734;</small>\
                  </div>\
                </div>\
              </div>\
            `,
        )
      }
    }
  </script>
 </body>
</html>
```

8 ВИСНОВКИ

В ході даної курсової роботи мною було розроблено смарт-контракти для створення ескроу-рахунків, а також проведення процесу купівлі-продажу нерухомості за допомогою смарт-контрактів. Також було розроблено веб-інтерфейс для полегшення взаємодії користувачам з контрактами, а також візуального представлення та пошуку нерухомості з можливістю її придбання з використанням технології блокчейн. Дана розробка уможливлює проводити процес продажу-купівлі нерухомості з меншим ризиком втрати коштів через недобросовісні дії сторін за допомогою використання блокчейну. Таким чином використання розробленої системи в Україні може збільшити рівень довіри між учасниками купівлі-продажу нерухомості, що може позитивно позначитися на кількості та сумі угод.

9 ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Сатосі Накамото [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A1%D0%B0%D1%82%D0%BE%D1%81%D1%96_%D0%9D%D0%B0%D0%BA%D0%B0%D0%BC%D0%BE%D1%82%D0%BE.

[2] Consensus Mechanism (Cryptocurrency) [Електронний ресурс] – Режим доступу до ресурсу: https://www.investopedia.com/terms/c/consensus-mechanism-cryptocurrency.asp.

[3] Майнінг [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D0%B9%D0%BD%D1%96%D0%BD%D0%B3.

[4] Imran Bashir. "Mastering Blockchain, Second Edition." // Packt – 2018.

[5] Криптовалютна біржа CoinMarketCap [Електронний ресурс] – Режим доступу до ресурсу: https://coinmarketcap.com.

[6] Самоходський І., Шелест О. Зелена книга регулювання ринку криптовалют. [Б.м.] : Офіс ефективного регулювання. 2018. травень. 69 с. 18 [Електронний ресурс] – Режим доступу до ресурсу: https://cdn.regulation.gov.ua/fe/5b/20/42/regulation.gov.ua_Зелена-Книга.-Ринок-Криптовалют.pdf.

[7] Податковий кодекс України від 2 грудня 2010 р. №2755-VI. ВВР України, 2011 р., №13-14, 15-16, 17, ст.112.

[8] Framework for State aid for research and development and innovation (2014/C 198/01). Official Journal of the European Union vol. 57, 27.06.2014.

[9] Спільна заява фінансових регуляторів щодо статусу криптовалют в Україні. [Електронний ресурс] – Режим доступу до ресурсу: https://bank.gov.ua/control/uk/publish/printable_article;jsessionid=175FC525F862C3D896D522CFC9CC47BB?art_id=59735329&showTitle=true.

[10] Глосарій банківської термінології. [Електронний ресурс] – Режим доступу до ресурсу:

https://bank.gov.ua/control/uk/publish/article?art_id=123302&cat_id=123211.

[11] FINMA, Guidelines for enquiries regarding the regulatory framework for initial coin offerings (ICOs) (16.02.2018). [Електронний ресурс] – Режим доступу до ресурсу: https://www.finma.ch/en/~/media/finma/dokumente/dokumentencenter/myfinma/1bewilligung/fintech/wegleitung-ico.pdf?la=en.

[12] Муніципалітет Кьяссо. [Електронний ресурс] – Режим доступу до ресурсу: https://www.chiasso.ch/possibilita-pagamento-imposte-bitcoin.

[13] Gesetz über das Kreditwesen (Kreditwesengesetz – KWG). [Електронний ресурс] – Режим доступу до ресурсу: http://www.gesetze-im-internet.de/kredwg/KWG.pdf .

[14] Кушка М. О. Правовий статус криптовалют за кордоном та в Україні. / М. О. Кушка, В. В. Іщенко // Роль юридичної науки в забезпеченні правоохоронної діяльності: матеріали підсумкової науково-практичної конференції (Київ, 25 квіт. 2019 р.) / – Київ, Україна, 2019. – С. 229– 232.

[15] GitHub, Kushka M., Escrow On Blockchain [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/kushkamisha/Escrow-on-blockchain.