Міністерство блокчейну і криптовалюти України

Практичне завдання № 2
На Тему:
«Реалізація власного блокчейну»

Виконав:
L1MD5_171
L1MD5_285
L1MD5_150
L1MD5_100
L1MD5_95

Перевірив:
Distributedlab

2022 р.

Тема: Реалізація блокчейну.

Мета: Для реалізації власного блокчейну сформувати теоретичні відомості, визначити сферу застосування.

Хід роботи

Сфера застосування блокчейну

Перш ніж визначити сферу застосування блокчейну, пригадаємо його одну із найбільших переваг, а саме децентралізацію. Тобто це інформація, яка рівномірно розподілена по всіх вузлах мережі, дублюючись у кожному із них. При такому підході, доки хоча б один вузол працездатний, інформація не втратиться, як це могло би бути, наприклад, з інтернет мережею.

Враховуючи, таку вище наведену особливість блокчейну, децентралізацію можна застосувати у такій, на перший погляд незвичній сфері, як у мережі спортивних залів.

Плюси:

- Незалежність залів один від одного
- Незалежність від валюти країни, окрім USD.
- При розширені мережі спортивного залу за кордоном, клієнтові не доведеться розмінювати фіат, щоб заплатити за абонемент. Тобто, якщо клієнт відвідував спортивний заклад і купував його монети за UAH, то поїхавши згодом за кордон, де є спортивні заклади такої ж мережі, він має змогу користуватися монетами, придбаними за UAH.
- Можливість формування єдиної клієнтської бази даних.
- Можливість формування самообслуговування клієнтів.

Мінуси:

- Залежність спортивного закладу від електроенергії та інтернету
- Залежність клієнтів від інтернету та обов’язкової наявності смартфонів.
- Повільніша обробка транзакцій у порівнянні із платіжними системами Visa, Mastercard и PayPal.

Технічне завдання та реалізація

Метою технічного завдання є об’єднання мережі спортивних закладів із створенням спільної системи оплати та клієнтської бази даних.

Клієнти не мають змоги майнити дану монету спортивного закладу, а лише обмінюватись монетами, без права на їх продаж. Тобто накопичення монет задля їх подальшої реалізації по дорожчій ціні неможливе, а отже якщо клієнт придбав монету, то в очікуваному результаті буде відвідувати спортивний заклад, обмінюючи монету на послуги.

Під час реалізації найпростішого блокчейну для мережі спортивних закладів, було використано мову програмування C# із середовищем розробки Visual Studio Code 2022.

Функціонал розробленого блокчейну включає в себе:

- Створення 10 блоків з випадковою кі-стю монет
- Верифікація блоків

Після процесу розробки було проведено також тестування програмного коду:

- Тестування блокчейну без змін
- Тестування блокчейну зі зміною суми в одному з блоків
- Тестування блокчейну зі зміною суми в одному з блоків і регенерацією хеша

Структура блокчейну

Опис коду

```
public class Chain
{
    public double amount;
    public string hash;
    public string prevhash;
    public int index;

    static public int count = 0;
```

Клас Chain = один блок.

В собі цей клас має такі параметри:

- Кі-сть монет
- Хеш
- Хеш попереднього блоку

```
public void CreateBlock(double amount, string prevhash)
{
    this.amount = amount;
    this.prevhash = prevhash;
    index = count++;
    hash = GetHash();

}
```

Метод CreateBlock класу Chain створює новий блок.

Цей метод приймаю кі-сть монет та хеш попереднього блоку.

```
public string GetHash()
{
    MD5 md5Hash = MD5.Create();
    return GetMd5Hash(md5Hash, amount.ToString() + prevhash + index.ToString());

}
```

Для генерації хешу, використовується метод GetHash, який за допомогою методу GetMd5Hash (див. нижче) шифрує текст, який представляє з себе конкатенацію кі-сті монет, попереднього хешу та індексу в md5.

```
static string GetMd5Hash(MD5 md5Hash, string input)
{
    byte[] data = md5Hash.ComputeHash(Encoding.UTF8.GetBytes(input));
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < data.Length; i++)
    {
        sb.Append(data[i].ToString("x2"));
    }
    return sb.ToString();
}
```

Метод GetMd5Hash шифрує заданий текст в md5.

```
public static bool VerificationBC(List<Chain> blockchain)
{
    for (int i = 1; i < blockchain.Count; i++)
    {
        if (blockchain[i].hash != blockchain[i].GetHash()) return false;
        if (blockchain[i].prevhash != blockchain[i - 1].hash) return false;
    }
    return true;
}
```

В основному класі знаходиться метод VerificationBC, який приймає блоки.

Даний метод перевірає цілісність блокчейну.

Якщо хоча б в одному блоці буде порушене його хеш значення або хеш значення поперенього блоку, метод поверне false і в такому випадку, блокчейн не буде верифікованим.

Тестування

В данному проекті ми реалізували 3 тестування.

тестування.

```
public void VerificationOk()
{
    Random random = new Random();
    List<Chain> blockchain = new List<Chain>();

    //Create 10 blocks
    for (int i = 0; i < 10; i++)
    {
        Chain chain = new Chain();
        chain.CreateBlock(random.NextDouble(), blockchain.Count < 1 ? "" : blockchain.Last().hash);
        blockchain.Add(chain);
    }

    Assert.AreEqual(GachiBlockchain.Program.VerificationBC(blockchain), true);
}
```

Перше тестування - VerificationOk перевіряє чи проходить перевірку блокчейн, який не піддавався змінам після його створення.

Очікуваний результат: True

```
[TestMethod]
public void VerificationError1()
{
    Random random = new Random();
    List<Chain> blockchain = new List<Chain>();

    //Create 10 blocks
    for (int i = 0; i < 10; i++)
    {
        Chain chain = new Chain();
        chain.CreateBlock(random.NextDouble(), blockchain.Count < 1 ? "" : blockchain.Last().hash);
        blockchain.Add(chain);
    }

    // Change amount in 2-nd block
    blockchain[1].amount = 0.3;

    Assert.AreEqual(GachiBlockchain.Program.VerificationBC(blockchain), false);
}
```

Друге тестування - VerificationError1 перевіряє чи пройде перевірку блокчейн, який піддався зміні кі-сті монет у одному з блоків.

Очікуваний результат: False

```
[TestMethod]
public void VerificationError2()
{
    Random random = new Random();
    List<Chain> blockchain = new List<Chain>();

    //Create 10 blocks
    for (int i = 0; i < 10; i++)
    {
        Chain chain = new Chain();
        chain.CreateBlock(random.NextDouble(), blockchain.Count < 1 ? "" : blockchain.Last().hash);
        blockchain.Add(chain);
    }

    // Change amount in 2-nd block
    blockchain[1].amount = 0.3;
    // Regenerate new hash
    blockchain[1].hash = blockchain[1].GetHash();

    Assert.AreEqual(GachiBlockchain.Program.VerificationBC(blockchain), false);
}
```

Третє тестування - VerificationError2 перевіряє чи пройде перевірку блокчейн, який піддався зміні кі-сті монет у одному з блоків та в якому перегенерували хеш.

Очікуваний результат: False

Висновок

Під час реалізації даного практичного завдання, ми обрали та описали сферу під реалізацію блокчейну, який було реалізовано на мові програмування C#, із урахуванням функціоналу та вимог. Після реалізації було проведено тестування на працездатність програмного продукту.