

## Лабораторна робота № 2

Виконали: Тивонюк Володимир та Виграновський Марко ФБ-41мн

### Тема:

Дослідження безпечної реалізації та експлуатації децентралізованих додатків.

### Мета роботи:

отримання навичок роботи із децентралізованими додатками та оцінка безпеки інформації при їх функціонуванні

**Для другого типу лабораторних робіт:** розробка децентралізованого додатку (наприклад, захисту інтелектуальної власності цифрового контенту) на обраній системі децентралізованих додатків.

Працюватиму з brownie-eth для контейнерів контрактів

```
s > 5 year > 2 > blockchain > lab3 >

Name                               Date modified      Type                Size
--
build                              5/1/2025 7:40 PM   File folder
contracts                          5/1/2025 7:40 PM   File folder
interfaces                         5/1/2025 7:40 PM   File folder
reports                            5/1/2025 7:40 PM   File folder
scripts                           5/1/2025 7:40 PM   File folder
tests                              5/1/2025 7:40 PM   File folder
.gitattributes                     5/1/2025 7:40 PM   Git Attributes Sour... 1 KB
.gitignore                        5/1/2025 7:40 PM   Git Ignore Source ... 1 KB

Windows PowerShell
PS C:\Users\Volodymyr\Desktop\Assignments\5 year\2\blockchain\lab3> brownie init
INFO: Could not find files for the given pattern(s).
Brownie v1.20.7 - Python development framework for Ethereum

SUCCESS: A new Brownie project has been initialized at C:\Users\Volodymyr\Desktop\Assignments\5 year\2\blockchain\lab3>
PS C:\Users\Volodymyr\Desktop\Assignments\5 year\2\blockchain\lab3>
```

```
! brownie-config.yaml
1  compiler:
2    | solc:
3    |   version: 0.8.19
4
5  networks:
6    | development:
7    |   gas_limit: "max"
```

Бачимо консоль робоча, середовище запускається:

```
PS C:\Users\Volodymyr\Desktop\Assignments\5 year\2\blockchain\lab3> brownie console
INFO: Could not find files for the given pattern(s).
Brownie v1.20.7 - Python development framework for Ethereum

Lab3Project is the active project.
This version of pWS is not compatible with your Node.js build:
Error: Cannot find module '../binaries/uws_win32_x64_127.node'
Require stack:
- C:\Users\Volodymyr\AppData\Roaming\npm\node_modules\ganache\node_modules\@trufflesuite\uws-js-unofficial\dist\cli.js
- C:\Users\Volodymyr\AppData\Roaming\npm\node_modules\ganache\dist\node\cli.js
Falling back to a NodeJS implementation; performance may be degraded.

Launching 'ganache-cli.cmd --chain.vmErrorsOnRPCResponse true --server.port 8545 --miner.blockGasLimit 10000000 --totalAccounts 10 --hardfork istanbul --wallet.mnemonic brownie'...
Brownie environment is ready.
>>>
```

## Створюю базовий контракт для завдання лабораторної роботи: захисту інтелектуальної власності цифрового контенту

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract ContentRegistry {

    struct ContentInfo {
        address owner;
        uint256 timestamp;
        bool exists;
    }

    mapping(bytes32 => ContentInfo) private contentRecords;

    event ContentRegistered(
        bytes32 indexed contentHash,
        address indexed owner,
        uint256 timestamp
    );

    function registerContent(bytes32 _contentHash) public {
        require(!contentRecords[_contentHash].exists, "Content hash already registered.");
        contentRecords[_contentHash] = ContentInfo({
            owner: msg.sender,
            timestamp: block.timestamp,
            exists: true
        });
        emit ContentRegistered(_contentHash, msg.sender, block.timestamp);
    }

    function verifyOwnership(bytes32 _contentHash) public view returns (address owner, uint256 timestamp) {
        require(contentRecords[_contentHash].exists, "Content hash not found.");
        ContentInfo storage info = contentRecords[_contentHash];
        return (info.owner, info.timestamp);
    }

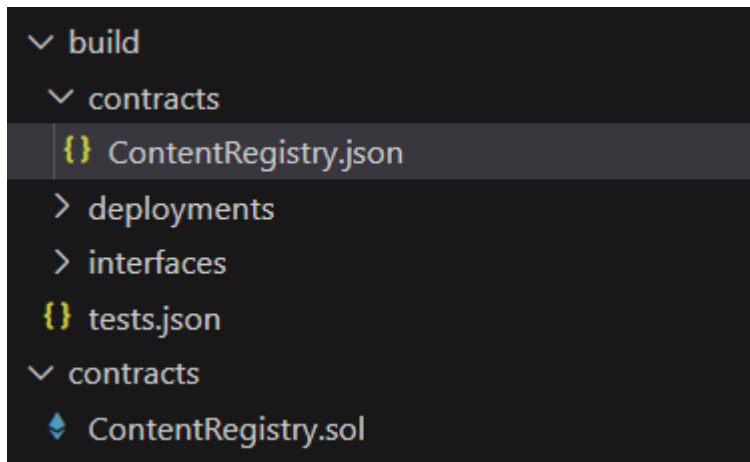
    function isRegistered(bytes32 _contentHash) public view returns (bool) {
        return contentRecords[_contentHash].exists;
    }
}
```

Смарт-контракт **ContentRegistry** робить так що користувач незмінно реєструє унікальний хеш (bytes32) їхнього цифрового контенту на блокчейні, записуючи адресу реєстранта (msg.sender) та точний час реєстрації (block.timestamp).

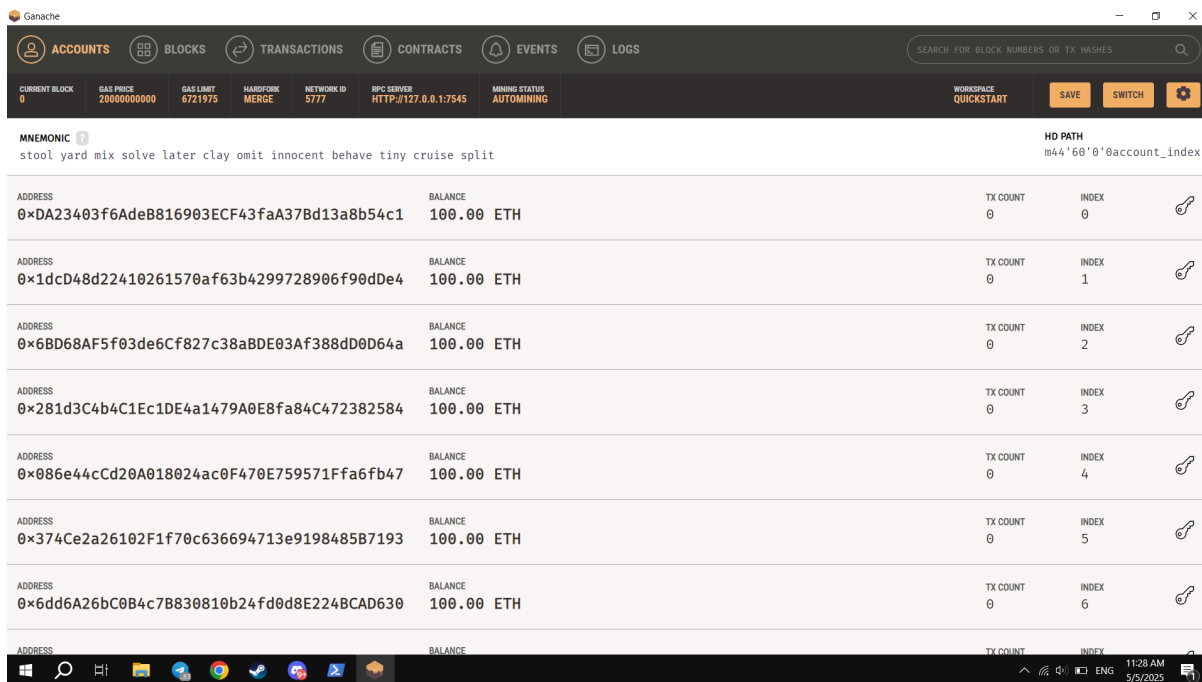
Гарантує унікальність кожного запису за допомогою перевірки require, надає публічні функції для перевірки факту реєстрації хешу (isRegistered) та отримання даних про власника й час (verifyOwnership), а також випромінює подію ContentRegistered для сповіщення зовнішніх систем про успішну реєстрацію.

```
PS C:\Users\Volodymyr\Desktop\Assignments\5 year\2\blockchain\lab3> brownie compile
INFO: Could not find files for the given pattern(s).
Brownie v1.20.7 - Python development framework for Ethereum

Project has been compiled. Build artifacts saved at C:\Users\Volodymyr\Desktop\Assignments\5 year\2\blockchain\lab3\build\contracts
PS C:\Users\Volodymyr\Desktop\Assignments\5 year\2\blockchain\lab3>
```



Для хосту тестової мережі та отримання валюти для тестування застосовую Ganache



Далі потрібен простенький скрипт щоб задеплоїти контракт, він саме приконектився до ганаче і можна було виконувати транзакції в мережі.

```
import json
import os
from web3 import Web3
from dotenv import load_dotenv
from pathlib import Path

load_dotenv()
GANACHE_URL = os.getenv("GANACHE_URL", "http://127.0.0.1:7545")
BUILD_DIR = Path("build")
ABI_FILE = BUILD_DIR / "contracts" / "ContentRegistry.json"
DEPLOYER_ACCOUNT_INDEX = 0

def main():
```

```
print(f"Підключення до Ganache: {GANACHE_URL}")
w3 = Web3(Web3.HTTPProvider(GANACHE_URL))
if not w3.is_connected():
    print("ПОМИЛКА: Не вдалося підключитися до Ganache.")
    return

try:
    with open(ABI_FILE, 'r') as f:
        contract_json = json.load(f)
        abi = contract_json['abi']
        bytecode = contract_json['bytecode']
except FileNotFoundError:
    print(f"ПОМИЛКА: Не знайдено файл {ABI_FILE}. Запустіть 'brownie compile'.")
    return
except KeyError:
    print(f"ПОМИЛКА: Не вдалося знайти 'abi' або 'bytecode' у файлі {ABI_FILE}.")
    return
except Exception as e:
    print(f"ПОМИЛКА завантаження ABI/байткоду: {e}")
    return

try:
    deployer_account = w3.eth.accounts[DEPLOYER_ACCOUNT_INDEX]
    print(f"Акаунт: {deployer_account}")
    print(f"Баланс: {w3.from_wei(w3.eth.get_balance(deployer_account), 'ether')} ETH")
except IndexError:
    print(f"ПОМИЛКА: Акаунт з індексом {DEPLOYER_ACCOUNT_INDEX} не знайдено в Ganache.")
    return
except Exception as e:
    print(f"ПОМИЛКА отримання акаунту: {e}")
    return

print("Розгортання контракту ContentRegistry...")
Contract = w3.eth.contract(abi=abi, bytecode=bytecode)
try:
    tx_hash = Contract.constructor().transact({'from': deployer_account, 'gas': 1500000})
    print(f"Транзакцію надіслано: {tx_hash.hex()}")
    print("Очікування підтвердження...")
    tx_receipt = w3.eth.wait_for_transaction_receipt(tx_hash, timeout=120)
```

```

    if tx_receipt.status == 1:
        contract_address = tx_receipt.contractAddress
        print("-" * 50)
        print(f"КОНТРАКТ УСПІШНО РОЗГОРНУТО в Ganache UI!")
        print(f"Адреса: {contract_address}")
        print(f"Блок: {tx_receipt.blockNumber}")
        print("-" * 50)
    else:
        print("ПОМИЛКА: Розгортання не вдалося (транзакція revert).")

if __name__ == "__main__":
    main()

```

```

PS C:\Users\Volodymyr\Desktop\lab3> python scripts/deploy_web3.py
Підключення до Ganache: http://127.0.0.1:7545
Акаунт: 0x621A47e032DaDc55924b7c290D405EdceF320f32
Баланс: 99.999614660329497109 ETH
Розгортання контракту ContentRegistry...
Транзакцію надіслано: 0xc952c27bddb374d87497b5620e6e2d9fee0125ea45a3a9d965978728d906cd8a
Очікування підтвердження...
-----
КОНТРАКТ УСПІШНО РОЗГОРНУТО в Ganache UI!
Адреса: 0xDDC8f97aFDF5F0b112dc925ED4Fc35a5F15322Be
Блок: 5
-----
PS C:\Users\Volodymyr\Desktop\lab3> 

```

Далі саме розробка застосунку для апки, формування базового інтерфейсу - важливо записати адресу контракту для використання в апці.

```

└─ static
  │ # style.css
└─ templates
  │ <> index.html

```

Дані файли нічим не цікаві, базові темплейти для візуалу

### Веб-інтерфейс (Flask + Web3.py):

У бекенді реалізував логіку обчислення SHA-256 хешу для файлів, отримання списку доступних акаунтів та взаємодії зі смарт-контрактом, викликаючи його функції (isRegistered, verifyOwnership, registerContent) для перевірки та реєстрації контенту від імені обраного користувача, обробивши результати та помилки для відображення користувачу на веб-сторінці.

```

import os
import hashlib
import time
import json

```

```

from pathlib import Path
from flask import Flask, render_template, request, redirect, url_for, flash
from web3 import Web3, HTTPProvider
from web3.exceptions import ContractLogicError
from dotenv import load_dotenv

# --- Configuration ---
load_dotenv()
GANACHE_URL = os.getenv("GANACHE_URL", "http://127.0.0.1:7545")
CONTRACT_ADDRESS = os.getenv("CONTRACT_ADDRESS",
"0xcd65DF3Ba5aA8002968e1034482b67C5e431b747")
ABI_PATH = Path("build") / "contracts" / "ContentRegistry.json"

app = Flask(__name__)
app.secret_key = os.urandom(24)

try:
    w3 = Web3(HTTPProvider(GANACHE_URL))
    if not w3.is_connected():
        raise ConnectionError(f"Failed to connect to Ganache at {GANACHE_URL}")
    print(f"Connected to Ganache: {GANACHE_URL}")
except Exception as e:
    print(f"CRITICAL: Ganache connection error: {e}")
    exit()

try:
    with open(ABI_PATH, 'r') as f:
        contract_json = json.load(f)
        contract_abi = contract_json['abi']
except Exception as e:
    print(f"CRITICAL: Failed to load ABI from {ABI_PATH}. Run 'brownie compile'.
Error: {e}")
    exit()

try:
    checksum_address = w3.to_checksum_address(CONTRACT_ADDRESS)
    contract = w3.eth.contract(address=checksum_address, abi=contract_abi)
    print(f"Contract instance loaded: {contract.address}")
except Exception as e:
    print(f"CRITICAL: Failed to create contract instance for {CONTRACT_ADDRESS}.
Error: {e}")
    exit()

# --- Helper Functions ---

```

```

def calculate_file_hash(file_stream):
    hash_obj = hashlib.sha256()
    for chunk in iter(lambda: file_stream.read(4096), b''):
        hash_obj.update(chunk)
    file_stream.seek(0)
    return hash_obj.digest() # Return raw bytes for web3.py

def get_ganache_accounts():
    try:
        return w3.eth.accounts
    except Exception as e:
        print(f"Error getting Ganache accounts: {e}")
        return []

# --- Flask Routes ---
@app.route('/')
def index():
    ganache_accounts = get_ganache_accounts()
    return render_template('index.html', accounts=ganache_accounts,
verification_result=None)

@app.route('/register', methods=['POST'])
def register():
    ganache_accounts = get_ganache_accounts()
    if 'contentFile' not in request.files:
        flash('File not submitted.', 'error')
        return redirect(url_for('index'))

    file = request.files['contentFile']
    selected_account = request.form.get('account')

    if file.filename == '':
        flash('No file selected.', 'error')
        return redirect(url_for('index'))
    if not selected_account or selected_account not in ganache_accounts:
        flash('Invalid account selected.', 'error')
        return redirect(url_for('index'))

    if file:
        try:
            content_hash_bytes = calculate_file_hash(file.stream)
            content_hash_hex = "0x" + content_hash_bytes.hex()
            print(f"Calculated hash for registration: {content_hash_hex}")

```

```

        is_registered =
contract.functions.isRegistered(content_hash_bytes).call()
        if is_registered:
            owner, _ =
contract.functions.verifyOwnership(content_hash_bytes).call()
            flash(f'Content (hash: {content_hash_hex[:10]}...) already
registered by {owner}!', 'warning')
            return redirect(url_for('index'))

        print(f"Sending registerContent transaction from
{selected_account}...")
        tx_hash =
contract.functions.registerContent(content_hash_bytes).transact({
            'from': selected_account,
            'gas': 500000
        })

        print(f"Waiting for transaction receipt: {tx_hash.hex()}")
        receipt = w3.eth.wait_for_transaction_receipt(tx_hash, timeout=120)

        if receipt.status == 1:
            flash(f'Content registered successfully! Hash:
{content_hash_hex}. Tx: {receipt.transactionHash.hex()}', 'success')
            print(f"Transaction successful: {receipt.transactionHash.hex()}")
        else:
            flash('Registration failed: Transaction reverted (hash might
exist?).', 'error')
            print(f"Transaction failed (status=0):
{receipt.transactionHash.hex()}")

    except ContractLogicError as e:
        flash(f'Contract error: {e}', 'error')
        print(f"Contract logic error: {e}")
    except Exception as e:
        flash(f'Unexpected server error during registration: {e}', 'error')
        print(f"General registration error: {e}")

    return redirect(url_for('index'))

return redirect(url_for('index'))

@app.route('/verify', methods=['POST'])
def verify():
    ganache_accounts = get_ganache_accounts()

```



```

content_hash_bytes = None
content_hash_hex = None
file = request.files.get('verifyFile')
hash_input = request.form.get('verifyHash')
verification_result = {}

if file and file.filename != '':
    try:
        content_hash_bytes = calculate_file_hash(file.stream)
        content_hash_hex = "0x" + content_hash_bytes.hex()
        print(f"Calculated hash for verification (from file): {content_hash_hex}")
    except Exception as e:
        flash(f'Error processing file: {e}', 'error')
        return render_template('index.html', accounts=ganache_accounts, verification_result=None)
    elif hash_input:
        if not (hash_input.startswith('0x') and len(hash_input) == 66):
            flash('Invalid hash format (must be 0x followed by 64 hex chars).', 'error')
            return render_template('index.html', accounts=ganache_accounts, verification_result=None)
        try:
            content_hash_bytes = bytes.fromhex(hash_input[2:]) # Remove '0x' before converting
            content_hash_hex = hash_input
            print(f"Hash for verification (input): {content_hash_hex}")
        except ValueError:
            flash('Invalid hash format (cannot convert to bytes).', 'error')
            return render_template('index.html', accounts=ganache_accounts, verification_result=None)
        else:
            flash('No file selected or hash entered for verification.', 'error')
            return render_template('index.html', accounts=ganache_accounts, verification_result=None)

    if content_hash_bytes and content_hash_hex:
        try:
            is_registered = contract.functions.isRegistered(content_hash_bytes).call()

            if is_registered:
                owner, timestamp = contract.functions.verifyOwnership(content_hash_bytes).call()

```

```

        verification_result = {
            'hash': content_hash_hex,
            'owner': owner,
            'timestamp': timestamp,
            'time_str': time.strftime('%Y-%m-%d %H:%M:%S UTC',
time.gmtime(timestamp)),
            'error': None
        }
        print(f"Verification result: Owner={owner},
Timestamp={timestamp}")
    else:
        verification_result = {
            'hash': content_hash_hex,
            'error': f"Content with hash {content_hash_hex[:10]}... not
found in registry."
        }
        print(f"Verification result: Hash {content_hash_hex} not
found.")

    except ContractLogicError as e:
        print(f"Contract logic error during verification: {e}")
        verification_result = {
            'hash': content_hash_hex,
            'error': f"Contract error verifying hash
{content_hash_hex[:10]}... ({e})"
        }
    except Exception as e:
        flash(f'Server error during verification: {e}', 'error')
        print(f"Error calling contract function: {e}")
        verification_result = {'error': f"Blockchain interaction error: {e}"}

    return render_template('index.html', accounts=ganache_accounts,
verification_result=verification_result)

# --- Server Start ---
if __name__ == '__main__':
    print(f"Starting Flask server on http://127.0.0.1:5000")
    print(f"Connected to Ganache: {GANACHE_URL}")
    print(f"Using contract: {CONTRACT_ADDRESS}")
    app.run(host='0.0.0.0', port=5000, debug=True)

```

Ключовими функціями будуть раути в Flask - на створення підпису хешем та перевірки хешу. Саме там виконується звернення до контракту, його серверу і блокчейн адрес.

Бачимо логінг процесу, сталий стан веб застосунку.

```
PS C:\Users\Volodymyr\Desktop\lab3> python app.py
Connected to Ganache: http://127.0.0.1:7545
Contract instance loaded: 0xcd65DF3Ba5aA8002968e1034482b67C5e431b747
Starting Flask server on http://127.0.0.1:5000
Connected to Ganache: http://127.0.0.1:7545
Using contract: 0xcd65DF3Ba5aA8002968e1034482b67C5e431b747
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://192.168.0.169:5000
Press CTRL+C to quit
* Restarting with stat
Connected to Ganache: http://127.0.0.1:7545
Contract instance loaded: 0xcd65DF3Ba5aA8002968e1034482b67C5e431b747
Starting Flask server on http://127.0.0.1:5000
Connected to Ganache: http://127.0.0.1:7545
Using contract: 0xcd65DF3Ba5aA8002968e1034482b67C5e431b747
* Debugger is active!
* Debugger PIN: 141-409-462
127.0.0.1 - - [05/May/2025 13:35:58] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [05/May/2025 13:35:59] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [05/May/2025 13:35:59] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [05/May/2025 13:36:40] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [05/May/2025 13:36:40] "GET /static/style.css HTTP/1.1" 304 -
█
```

В ганаче під час деплойменту бачимо створення контракту:

TX HASH				CONTRACT CREATION
0xc952c27bddb374d87497b5620e6e2d9fee0125ea45a3a9d965978728d906cd8a				
FROM ADDRESS	CREATED CONTRACT ADDRESS	GAS USED	VALUE	
0x621A47e032DaDc55924b7c290D405EdceF320f32	0xDDC8f97aFDF5F0b112dc925ED4Fc35a5F15322Be	198133	0	

Застосунок:

## Захист Інтелектуальної Власності

Децентралізована реєстрація цифрового контенту

### Реєстрація Нового Контенту

1. Оберіть файл для реєстрації:

Вибрати файл

Файл не вибрано

2. Оберіть ваш акаунт (з Ganache):

-- Виберіть акаунт --

Зареєструвати Хеш Контенту

## Перевірка Власності Контенту

1. Оберіть файл для перевірки:

Вибрати файл

Файл не вибрано

АБО

2. Введіть хеш контенту (формат 0х...):

0х...

Перевірити Реєстрацію

Функціонал хешування файлу:

1. Оберіть файл для реєстрації:

Вибрати файл

app.py

2. Оберіть ваш акаунт (з Ganache):

0xda07F064A96B88f7A1B263A4E65048738132c0e4

Зареєструвати Хеш Контенту

Content registered successfully! Hash:

0x12c7fa63e62da508da1e13abd762d91fc853ff356b2644e7e9092975b880acae.

Тх:

0x7192f092f2411d5c5946ee902c66c59cdec672765ed745a5594d8122e0af5d08

Функціонал перевірки хешу або файлу:

### Результат Перевірки

Статус: **Зареєстровано**

Хеш контенту:


0x12c7fa63e62da508da1e13abd762d91fc853ff356b2644e7e9092975b880acae

Адреса власника: 0xda07F064A96B88f7A1B263A4E65048738132c0e4

Час реєстрації (Unix): 1746441744

Час реєстрації (UTC): 2025-05-05 10:42:24 UTC

В Ganache бачимо транзакції і логи:

← BACK		BLOCK 6			
GAS USED	GAS LIMIT	MINED ON	BLOCK HASH		
90399	6721975	2025-05-05 13:42:24	0x4ad7e9ae80386891b25aac9abd2f20e465d0ed4a91df2d3e331a4b4271e3fa70		
TX HASH					CONTRACT CALL
0x7192f092f2411d5c5946ee902c66c59cdec672765ed745a5594d8122e0af5d08					
FROM ADDRESS	TO CONTRACT ADDRESS		GAS USED	VALUE	
0xda07F064A96B88f7A1B263A4E65048738132c0e4	0xcd65DF3Ba5aA8002968e1034482b67C5e431b747		90399	0	
ADDRESS	BALANCE	TX COUNT		INDEX	
0xda07F064A96B88f7A1B263A4E65048738132c0e4	100.00 ETH	1		7	

Приклади помилок: Файл не зареєстрований

### Результат Перевірки

Content with hash 0x20624e8b... not found in registry.

Перевірений хеш:

0x20624e8b9258553b4120b2682760da21499bb0d5a02ce29b720c903ad882021b

Спроба реєстрації на іншого юзера того ж файлу:

## Захист Інтелектуальної Власності

Децентралізована реєстрація цифрового контенту

Content (hash: 0x12c7fa63...) already registered by 0xda07F064A96B88f7A1B263A4E65048738132c0e4!

Логи в консолі апки підтвержують коректну роботу:

```

127.0.0.1 - - [05/May/2025 13:38:40] "GET /static/style.css HTTP/1.1" 304 -
Calculated hash for registration: 0x12c7fa63e62da508da1e13abd762d91fc853ff356b2644e7e9092975b880acae
Sending registerContent transaction from 0xda07F064A96B88f7A1B263A4E65048738132c0e4...
Waiting for transaction receipt: 0x7192f092f2411d5c5946ee902c66c59cdec672765ed745a5594d8122e0af5d08
Transaction successful: 0x7192f092f2411d5c5946ee902c66c59cdec672765ed745a5594d8122e0af5d08
127.0.0.1 - - [05/May/2025 13:42:24] "POST /register HTTP/1.1" 302 -
127.0.0.1 - - [05/May/2025 13:42:24] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [05/May/2025 13:42:24] "GET /static/style.css HTTP/1.1" 304 -
Calculated hash for verification (from file): 0x12c7fa63e62da508da1e13abd762d91fc853ff356b2644e7e9092975b880acae
Verification result: Owner=0xda07F064A96B88f7A1B263A4E65048738132c0e4, Timestamp=1746441744
127.0.0.1 - - [05/May/2025 13:43:30] "POST /verify HTTP/1.1" 200 -
127.0.0.1 - - [05/May/2025 13:43:30] "GET /static/style.css HTTP/1.1" 304 -
Calculated hash for verification (from file): 0x20624e8b9258553b4120b2682760da21499bb0d5a02ce29b720c903ad882021b
Verification result: Hash 0x20624e8b9258553b4120b2682760da21499bb0d5a02ce29b720c903ad882021b not found.
127.0.0.1 - - [05/May/2025 13:45:02] "POST /verify HTTP/1.1" 200 -
127.0.0.1 - - [05/May/2025 13:45:02] "GET /static/style.css HTTP/1.1" 304 -
Calculated hash for registration: 0x12c7fa63e62da508da1e13abd762d91fc853ff356b2644e7e9092975b880acae
127.0.0.1 - - [05/May/2025 13:46:46] "POST /register HTTP/1.1" 302 -
127.0.0.1 - - [05/May/2025 13:46:46] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [05/May/2025 13:46:46] "GET /static/style.css HTTP/1.1" 304 -
Calculated hash for registration: 0x12c7fa63e62da508da1e13abd762d91fc853ff356b2644e7e9092975b880acae
127.0.0.1 - - [05/May/2025 13:49:13] "POST /register HTTP/1.1" 302 -
127.0.0.1 - - [05/May/2025 13:49:13] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [05/May/2025 13:49:13] "GET /static/style.css HTTP/1.1" 304 -
Calculated hash for registration: 0x3b62a901ecedfb4b654a2e8bfd01b3bf21439e96e8a44f60d15a248d0086ce9f
Sending registerContent transaction from 0x621A47e032DaDc55924b7c290D405EdceF320f32...
Waiting for transaction receipt: 0xb55453a23e74f97fb19f08f602a8da828eacb1a26ea1b53a8dba05ef2f44dea3
Transaction successful: 0xb55453a23e74f97fb19f08f602a8da828eacb1a26ea1b53a8dba05ef2f44dea3
127.0.0.1 - - [05/May/2025 13:49:31] "POST /register HTTP/1.1" 302 -

```

## Висновки:

У ході виконання лабораторної роботи було успішно розроблено та досліджено децентралізований додаток для захисту інтелектуальної власності цифрового контенту на базі блокчейну Ethereum.

Було створено смарт-контракт на Solidity, який дозволяє незмінно реєструвати хеші контенту разом з адресою власника та часом реєстрації. Для розробки, компіляції, тестування та розгортання контракту використовувався фреймворк Brownie та локальний блокчейн Ganache. Було реалізовано взаємодію з контрактом за допомогою Python-скриптів та створено веб-інтерфейс на Flask з використанням бібліотеки web3.py для надання користувацького доступу до функціоналу DApp.

В процесі роботи було отримано практичні навички розробки, розгортання, тестування та взаємодії з DApps, а також виявлено важливість правильного налаштування середовища та розуміння взаємодії між компонентами системи (смарт-контракт, блокчейн, бекенд).