

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

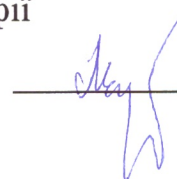
Пояснювальна записка

до випускної кваліфікаційної роботи магістра на тему:
“Мобільний iOS додаток для захищеного обміну зображеннями
за протоколом IPFS у децентралізованій мережі”

Виконав студент 2 курсу групи КІм-22-1
спеціальності 123 «комп'ютерна інженерія»

 Андрійчук Б. М.

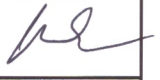
Керівник, завідувач кафедри комп'ютерних
наук, доктор філософії

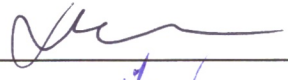
 Граф М. С.

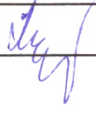
Рецензент, професор кафедри комп'ютерної
інженерії та кібербезпеки, доктор технічних
наук, доцент

 Воротніков В. В.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формування технічного завдання. Опрацювання літературних джерел.	4 листопада 2023 - 7 листопада 2023	
2	Проектування інфраструктури додатка	7 листопада 2023 - 13 листопада 2023	
3	Проектування Обробника	13 листопада 2023 - 15 листопада 2023	
4	Налаштування середовища та розробка Обробника	15 листопада 2023 - 18 листопада 2023	
5	Розробка Додатка	18 листопада 2023 - 19 грудня 2023	
6	Захист	28 грудня 2023	

Студент  Андрійчук Б.М.

Керівник  Граф М. С.

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Факультет інформаційно-комп'ютерних технологій

Кафедра комп'ютерної інженерії та кібербезпеки

Спеціальність 123 «комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри

комп'ютерної інженерії та
кібербезпеки

Єфіменко А.А.

13 жовтня 2023 року

ЗАВДАННЯ

На випускн у кваліфікаційну роботу

Студента: Андрійчука Богдана Михайловича

(прізвище, ім'я, по батькові)

Тема роботи Мобільний iOS додаток для захищеного обміну зображеннями за протоколом IPFS у децентралізованій мережі

Затверджені наказом університету від "13" жовтня 2023р. № 571/с

Термін здачі студентом закінченої роботи

Вихідні дані роботи (зазначається предмет і об'єкт дослідження) об'єктом дослідження є мобільний додаток iOS та Обробник IPFS запитів; предметом дослідження є обмін зображеннями за протоколами IPFS в децентралізованій системі IPFS

1. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Пірог О. В., доцент кафедри КІ та КБ	2.09.23	2.09.22
2	Пірог О. В., доцент кафедри КІ та КБ	6.10.23	6.10.23
3	Пірог О. В., доцент кафедри КІ та КБ	4.11.23	4.11.23

Студент Андрійчук Б.М.

Керівник Граф М. С.

Реферат

Випускна кваліфікаційна робота магістра складається з мобільного iOS додатку для захищеного обміну зображеннями за протоколом IPFS у децентралізованій мережі, та пояснювальної записки. Пояснювальна записка до випускної роботи містить 32 сторінки

Метою роботи є створення мобільного застосунку iOS для захищеного обміну зображеннями за протоколом IPFS у децентралізованій мережі.

В роботі визначено основні завдання на розробку додатків та інфраструктури, проаналізовано інформацію яка буде використана при розробці.

КЛЮЧОВІ СЛОВА: ДОДАТОК, ІНФОРМАЦІЙНА СИСТЕМА, БЛОКЧЕЙН, БЕЗПЕЧНИЙ ОБМІН, IPFS.

ABSTRACT

The master's thesis consists of an explanation for the development of a Handler for communication and data transfer with the IPFS network, an explanation for the development of an iOS mobile application, and the implementation of the IPFS Handler. It also includes an explanatory note. The explanatory note for the thesis is 32 pages long.

The aim of the thesis is to create an iOS mobile application for secure image exchange using the IPFS protocol in a decentralized network.

The main tasks for the development of the applications and infrastructure are defined in the thesis. Information to be used for the development is analyzed.

KEYWORDS: APPLICATION, INFORMATION SYSTEM, BLOCKCHAIN, SECURE EXCHANGE, IPFS.

Зміст

Вступ	7
Розділ 1. Дослідження методів передавання інформації у децентралізованих мережах Web3	9
1.1 Методи передачі даних в децентралізованих мережах	9
1.2 Загальна характеристика протоколу IPFS	10
1.3 Організація передавання зображень засобами протоколу IPFS	11
1.4 Опис вузла IPFS	13
1.5 Процедури завантаження та отримання зображень на IPFS	19
1.6 Застосування контейнеризації Docker в IPFS	23
Висновок до Розділу 1	25
Розділ 2. Проектування системи обміну інформацією та мобільного додатку	26
2.1 Проектування системи обміну інформацією на основі IPFS	26
2.2 Проектування мобільного застосунку на iOS	27
2.3 Інтерфейси та User-flow	28
Висновок до розділу 2	29
Розділ 3. Практична реалізація додатку для захищеного обміну зображеннями за протоколом IPFS	30
3.1 Обґрунтування вибору технології та реалізація додатку	30
3.2 Опис інтерфейсу та процедур застосування додатку	31
3.3 Тестування функціоналу додатку	38
Висновок до розділу 3	39
Висновки	40
Список використаної літератури	42
Додатки	44
Додаток А	45

Вступ

В сучасному цифровому світі децентралізація та безпека є надзвичайно важливими аспектами розробки мобільних додатків. Особливо це стосується обміну конфіденційною інформацією, такою як зображення.

Актуальність роботи полягає в тому, що застосування даного додатка може бути широким, від персонального обміну зображеннями між користувачами до професійного використання у будь якій сфері. Враховуючи актуальність проблеми обміну даними в децентралізованій мережі та потужні можливості протоколу IPFS, дана магістерська робота є важливим кроком у напрямку розвитку безпеки та захищеності обміну зображеннями на мобільних платформах.

Метою випускної магістерської роботи є розробка мобільного додатка для платформи iOS, який працюватиме на базі протоколу IPFS та забезпечуватиме захищений обмін зображеннями в децентралізованій мережі.

Досягнення мети роботи зумовлює виконання наступних завдань:

1. Проведення аналізу методів передачі даних в децентралізованій мережі.
2. Огляд та дослідження можливостей протоколу IPFS.
3. Створення механізму успішної передачі файлів через локальний вузол IPFS;
4. Створення проекту додатку для передачі зображень через IPFS вузол;
5. Створення Обробника IPFS, завдання якого полягає в опрацюванні запитів з мобільного додатку і забезпечення комунікації з локальним IPFS вузлом.
6. Створення iOS мобільного додатку.
7. Тестування функціональних можливостей створених інструментів для вдалої передачі зображення з телефону в

IPFS вузол та отримання інформації з нього.

Об'єктом дослідження роботи є процес проектування та розробки мобільного додатку iOS та Обробника IPFS запитів.

Предметом дослідження є методи, алгоритми та технології обміну зображенням, застосування протоколу IPFS в децентралізованій системі IPFS.

Розділ 1. Дослідження методів передавання інформації у децентралізованих мережах Web3

1.1 Методи передачі даних в децентралізованих мережах

Web3 (також відомий як Web 3.0 або децентралізований веб) - це поняття, яке відноситься до нової стадії розвитку Інтернету, де центральними принципами є децентралізація, блокчейн технології та контроль користувача над власною даними. Для передачі інформації в Web3 існує декілька методів, які включають:

1. Блокчейн: Блокчейн технології, такі як Ethereum, дозволяють використовувати “розумні контракти” для передачі даних та забезпечувати їх безпеку та незмінність. Цей метод дозволяє створювати децентралізовані додатки (DApps), де дані зберігаються на різних вузлах мережі.

2. IPFS: InterPlanetary File System (IPFS) дозволяє передавати файли в децентралізованому середовищі шляхом використання хеш-сум для ідентифікації та пошуку файлів. IPFS дозволяє передавати великі файли, забезпечуючи кешування та підтримку загального доступу.

3. Peer-to-Peer протоколи: Такі протоколи, як BitTorrent, дозволяють передавати дані без централізованого сервера. Вони спираються на обмін даними між користувачами, що забезпечує швидку та ефективну передачу великого обсягу інформації.

4. Децентралізовані мережі: Деякі нові протоколи, які розробляються в рамках Web3, пропонують децентралізовані мережі передачі даних, які базуються на блокчейн технологіях та протоколах довіри. Наприклад, MaidSafe дозволяє створювати мережі, де дані пересилаються шифровано і зашифровано без проміжного сервера.

Це лише кілька методів, які можна використовувати для передачі інформації в Web3. Оскільки ця галузь все ще активно розвивається, нові методи постійно з’являються та розробляються.

1.2 Загальна характеристика протоколу IPFS

IPFS (InterPlanetary File System) - це протокол і розподілена система зберігання файлів, яка дозволяє створювати перманентні та незмінні посилання на файли шляхом використання хеш-сум. Замість традиційної адресації файлів за допомогою URL, IPFS використовує Content Identifier (CID), який визначається хеш-сумою вмісту файли. Це означає, що файли, збережені в IPFS, доступні через мережу пір-до-піра, і можуть бути доступні незалежно від централізованих серверів. IPFS також забезпечує можливість верифікації цілісності файлу, автентифікацію та шифрування. Це робить IPFS корисним для збереження та розподілу розподілених додатків, контенту веб-сайтів та інших файлів.

Використання IPFS (InterPlanetary File System) для передачі даних в Web3 має багато переваг. Так як IPFS є децентралізованою системою збереження файлів, це означає, що файли розподіляються по вузлах мережі, а не зберігаються на централізованих серверах. Це дозволяє запобігти залежності від одного сервера та забезпечити розподілене зберігання даних. IPFS використовує хеш-суми (CID) для ідентифікації файлів. Кожен файл має унікальний CID, який визначається його вмістом. Це означає, що коли файл один раз додається до IPFS, його не можна змінити без зміни його CID. Таким чином, IPFS дозволяє створювати незмінні файли, які не можуть бути модифіковані або втрачені.

Також IPFS використовує механізми кешування та шарування, що дозволяє покращити швидкість передачі та отримання даних. Коли файл додається до IPFS, він автоматично кешується у вузлах мережі, що забезпечує швидкий доступ до нього. Крім того, IPFS дозволяє здійснювати шарування файлів, що означає, що файли можна отримувати частинами, забезпечуючи ефективне використання мережевого пропуску.

У IPFS файли шифруються та підписуються за допомогою алгоритмів шифрування та цифрового підпису. Це дозволяє забезпечити

конфіденційність даних та перевірити їх цілісність. IPFS має властивості шарованої мережі, що означає, що чим більше користувачів використовують IPFS, тим більше вузлів мережі стає доступними для збереження і передачі файлів. Це сприяє зростанню масштабу мережі без втрати ефективності.

1.3 Організація передавання зображень засобами протоколу IPFS

Передача даних у IPFS базується на протоколі передачі даних TCP/IP. Кожен файл, який зберігається в IPFS, ідентифікується за допомогою його хешу, який є унікальною послідовністю байтів. Цей хеш генерується на основі вмісту файлу, що гарантує його унікальність.

Коли користувач хоче передати файл через IPFS, його клієнт генерує ідентифікатор хешу файлу. Якщо файл уже присутній у мережі IPFS, то клієнт може просто посилатися на існуючий хеш, дозволяючи зменшити загальний обсяг переданих даних.

Якщо файл або його фрагменти не знайдені в мережі IPFS, клієнт може скористатися протоколом для пошуку файлу у світовій мережі. Інші вузли IPFS, які мають доступ до цього файлу, можуть повернути файл або його фрагменти по запиту користувача. Цей підхід дозволяє зменшити залежність від централізованих серверів і робить IPFS більш стійким до відмов.

Однією з рис IPFS є кешування даних. Коли користувач отримує файли через IPFS, його клієнт також зберігає ці файли локально. Це дозволяє підвищити швидкість доступу до файлів у подальшому, оскільки клієнт може отримати файли безпосередньо з локального кешу, не залежно від доступності інших вузлів IPFS.

Отже, передача даних через протокол IPFS відрізняється від традиційних методів передачі файлів через Інтернет. Вона базується на

хеш-ідентифікаторах файлів і можливості пошуку та обміну файлами між різними вузлами мережі. Це робить IPFS ефективним і розподіленим рішенням для зберігання та передачі даних.

Використання протоколу IPFS передбачає встановлення його складових за допомогою відповідних пакунків.

Додавання зображень до IPFS здійснюється за допомогою команди `ipfs add`. Ця команда додає зображення до IPFS і повертає хеш, який ідентифікує збережену копію. IPFS бере ваше зображення і обробляє його. IPFS розбиває зображення на невеликі частини, які називаються блоками даних. Кожен блок отримує свій унікальний хеш, який ідентифікує цей блок. IPFS створює спрямований ациклічний граф (DAG) за допомогою цих блоків даних. DAG - це структура даних, в якій вузли вказують на інші вузли за допомогою їх хешів. В DAG кожен блок даних вказує на своїх батьківських блоків, утворюючи ланцюжок зв'язків.

Наступним кроком IPFS обчислює унікальний хеш для всього зображення шляхом гешування кореневого блоку DAG. Цей хеш стає ідентифікатором зображення в IPFS. Після обчислення хешу IPFS перевіряє, чи є вже такий блок даних в IPFS мережі. Якщо цей блок вже існує, IPFS просто створює посилання на нього, а не зберігає дублікат. Якщо блок ще не існує в IPFS мережі, IPFS ретранслює блок, розповсюджуючи його на інші вузли в мережі. Коли блок дочасно досягне деякої ваги, він буде збережений довічно в мережі. Після того, як зображення успішно додано до IPFS мережі, отримуємо хеш, який ідентифікує його. Цей хеш можна використовувати для звернення до зображення в майбутньому і передачі його іншим користувачам.

Щоб отримати зображення через IPFS, потрібно хеш зображення. Використовуємо хеш та команду `ipfs get <хеш>`

IPFS перевіряє свою локальну базу даних, щоб з'ясувати, чи має він вже копію цього блоку даних. Якщо так, IPFS одразу повертає блок зображення.

Якщо блок даних не знайдено в локальній базі IPFS, IPFS використовує хеш для пошуку блоку у всій IPFS мережі. IPFS взаємодіє з іншими вузлами мережі, запитуючи власників блоків даних про наявність копій зображення. Вузли, які мають копію блока, надсилають цю інформацію назад до вашого вузла. IPFS вибирає найближчий до нас вузол, що має копію блока даних, із яким можна встановити пряме з'єднання. Вузол надсилає блок даних зображення через це з'єднання.

Після отримання блока даних зображення, IPFS вузол перевіряє цілісність даних за допомогою хеш-перевірки. Це робиться шляхом порівняння отриманого блока даних з хешем. Якщо цілісність підтверджена, IPFS зберігає отриманий блок даних у своїй локальній базі. Відтепер можна знову отримати це зображення, використовуючи той самий хеш, без необхідності звертатись до мережі IPFS.

1.4 Опис вузла IPFS

Кожний вузол має в собі підсистеми, які виконують чітко врегульованні функції. Три ключові відповідальності підсистем IPFS:

1. Представлення та адресація даних.
2. Маршрутизація даних.
3. Передача даних.

Хоча ці три є основними відповідальностями, функціональність IPFS виходить за рамки цих трьох.

В IPFS дані адресуються за їхнім змістом (адресація змістом), а не місцем, таким як IP-адреса (адресація місцем).

Таблиця 2.1

Мета та підсистема, яка виконує її

Мета	Підсистема
------	------------

Представлення та організація даних	CIDs, IPLD, UnixFS, MFS, DAG-CBOR, DAG-JSON, файли CAR
Маршрутизація контенту Зв'язок між CID та IP-адресами	Kademlia DHT, Делегована маршрутизація через HTTP, Bitswap, mDNS
Передача даних	Bitswap, HTTP Gateways, Sneakernet, Graphsync, more in development
Адресація для даних та пірів	Multiformats
Місткість між IPFS та HTTP	IPFS-шлюзи, Специфікація API для закріплення
Змінність та динамічне ім'я	IPNS (Міжпланетна система імен), DNSLink
Взаємодія між пірами	libp2p (TCP, QUIC, WebRTC, WebTransport)

Розглянемо кожну підсистему більш детально. Ідентифікатор контенту (CID) У IPFS дані розбиваються на блоки, кожен з яких отримує унікальний ідентифікатор, називаний ідентифікатором контенту (CID). Загалом CID обчислюється шляхом поєднання хешу даних із їхнім кодом. Код генерується за допомогою Multiformats (дані можуть себе описувати, використовуючи розширювані протоколи).

IPFS використовує файли Content Addressable aRchive (CAR) для зберігання та передачі серіалізованого архіву IPLD-даних, які адресуються за вмістом. Файли CAR подібні до TAR-файлів, оскільки призначені для зберігання колекцій даних, які адресуються за вмістом.

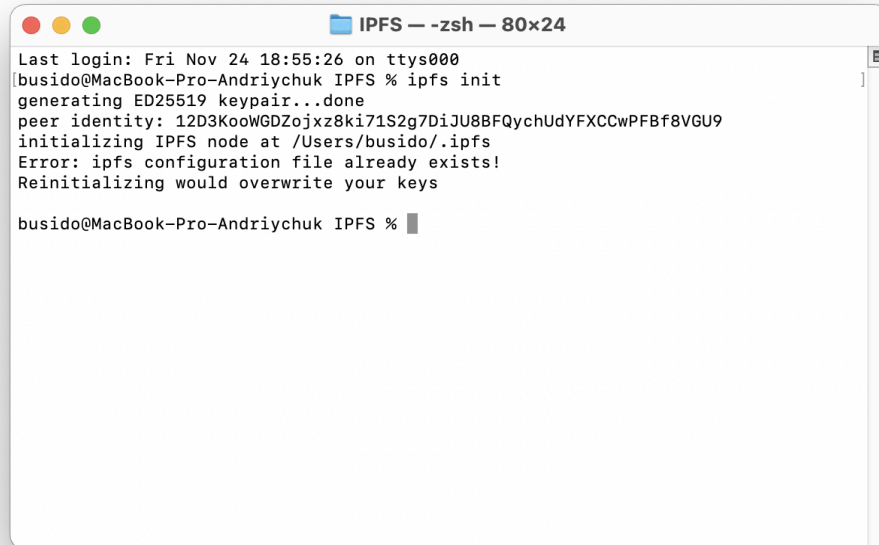
IPFS використовує InterPlanetary Linked Data (IPLD), щоб працювати з CIDs та даними, які адресуються за вмістом. IPFS використовує IPLD для представлення відносин між даними, які адресуються за вмістом, такими як каталоги файлів та інші ієрархічні структури, за допомогою напрямленого ациклічного графа (DAG), який називається графом Мерклі. Використовуючи IPLD для загальної функціональності, IPFS може забезпечити більш адаптований, конкретний механізм для представлення та адресації файлів, каталогів та їхніх символічних посилань, який називається UnixFS. З UnixFS IPFS може розбивати та посилати дані, що надто великі для зберігання в одному блоку, використовуючи подільне представлення для зберігання та управління даними.

IPFS має багато способів маршрутизовувати контент. IPFS використовує Kademlia, розподілений хеш-таблицю (DHT), призначену для децентралізованих пір-до-пір мереж. Kademlia допомагає знаходити піри в мережі IPFS, які зберігають дані, які ви шукаєте. Кадемлія DHT може бути уявлена як велика таблиця, розподілена по численних вузлах, яка зберігає інформацію про те, які піри (IP-адреси) мають які дані (CID). Kademlia забезпечує високоефективну, самоорганізовану систему, яка витримує зміни вузлів. Kademlia використовує libp2p для встановлення зв'язку. Також розглянемо спосіб, який проектувався в подальшій магістерській роботі. Делегована маршрутизація контенту через HTTP - це механізм, який використовують реалізації IPFS для перекладу маршрутизації контенту на інший процес/сервер за допомогою HTTP API. Наприклад, якщо вузол IPFS не реалізує розподілену хеш-таблицю (DHT), делегований роутер може виконувати пошук у DHT від його імені. Основна перевага делегованої маршрутизації полягає в тому, що вузли не зобов'язані реалізовувати функціональність маршрутизації самостійно, якщо у них недостатньо обчислювальних ресурсів для цього або вони хочуть побудувати систему IPFS з власним бекендом для маршрутизації. Таким

чином, делегована маршрутизація через HTTPS надає вузлам IPFS стандартний інтерфейс, який забезпечує більшу гнучкість у відношенні до того, як працює маршрутизація контенту.

Крім маршрутизації даних вузли в мережі IPFS повинні ефективно розповсюджувати та доставляти дані, які адресуються за вмістом, з урахуванням того, що деякі вузли в мережі вже мають копію даних, а інші вузли не мають копії, але хочуть отримати її. Для обробки передачі даних IPFS використовує наступні підсистеми: Bitswap, IPFS HTTP Gateway, Sneakernet. Вузли IPFS використовують Bitswap, протокол мережі, орієнтований на повідомлення та пір-до-піра, як для маршрутизації контенту, так і для передачі даних. За допомогою Bitswap будь-які піри, з якими з'єднаний вузол IPFS, можуть передавати запитані блоки безпосередньо цьому вузлу, не потрібно проходити через DHT. Піри також зберігають списки бажаних блоків, так що, якщо пір отримує запитані дані пізніше, він може передати їх вузлу, який спочатку подавав запит. Gateway IPFS діє як міст між традиційними HTTP-клієнтами та IPFS. За допомогою шлюзу користувачі можуть завантажувати файли, каталоги та інші дані IPLD, які зберігаються в IPFS, так, ніби вони були збережені на традиційному веб-сервері.

Для початку потрібно ініціалізувати вузол IPFS на комп'ютері та створити його конфігураційний файл. Якщо він вже існує, то такий вивід буде отримано результати, наведений на рис. 1.1.

A terminal window titled "IPFS — -zsh — 80x24" showing the output of the 'ipfs init' command. The output includes the last login time, keypair generation status, peer identity, and an error message indicating that the configuration file already exists.

```
Last login: Fri Nov 24 18:55:26 on ttys000
busido@MacBook-Pro-Andriychuk IPFS % ipfs init
generating ED25519 keypair...done
peer identity: 12D3KooWGDZojxz8ki71S2g7DiJU8BFQychUdYFXCCwPFBf8VGU9
initializing IPFS node at /Users/busido/.ipfs
Error: ipfs configuration file already exists!
Reinitializing would overwrite your keys

busido@MacBook-Pro-Andriychuk IPFS %
```

Рисунок 1.1 - Ініціалізація IPFS вузла

Для того, щоб мати змогу доступу до мережі, необхідно запустити відповідний демон. Результат запуску наведено на рис. 1.2.



```
IPFS — ipfs daemon — 80x40
Swarm listening on /ip4/127.0.0.1/tcp/4001
Swarm listening on /ip4/127.0.0.1/udp/4001/quic-v1
Swarm listening on /ip4/127.0.0.1/udp/4001/quic-v1/webtransport/certhash/uEiD1magzNM5Hg5JPKn9rYnroYD4sX73dD_--4tJZSYL7iA/certhash/uEiAvJLXzAcLYemvMqMRmR51KpS8259uYRSLzU0TDY7_IA
Swarm listening on /ip4/192.168.0.101/tcp/4001
Swarm listening on /ip4/192.168.0.101/udp/4001/quic-v1
Swarm listening on /ip4/192.168.0.101/udp/4001/quic-v1/webtransport/certhash/uEiD1magzNM5Hg5JPKn9rYnroYD4sX73dD_--4tJZSYL7iA/certhash/uEiAvJLXzAcLYemvMqMRmR51KpS8259uYRSLzU0TDY7_IA
Swarm listening on /ip6:::1/tcp/4001
Swarm listening on /ip6:::1/udp/4001/quic-v1
Swarm listening on /ip6:::1/udp/4001/quic-v1/webtransport/certhash/uEiD1magzNM5Hg5JPKn9rYnroYD4sX73dD_--4tJZSYL7iA/certhash/uEiAvJLXzAcLYemvMqMRmR51KpS8259uYRSLzU0TDY7_IA
Swarm listening on /p2p-circuit
Swarm announcing /ip4/127.0.0.1/tcp/4001
Swarm announcing /ip4/127.0.0.1/udp/4001/quic-v1
Swarm announcing /ip4/127.0.0.1/udp/4001/quic-v1/webtransport/certhash/uEiD1magzNM5Hg5JPKn9rYnroYD4sX73dD_--4tJZSYL7iA/certhash/uEiAvJLXzAcLYemvMqMRmR51KpS8259uYRSLzU0TDY7_IA
Swarm announcing /ip4/188.163.72.225/udp/64011/quic-v1
Swarm announcing /ip4/188.163.72.225/udp/64011/quic-v1/webtransport/certhash/uEiD1magzNM5Hg5JPKn9rYnroYD4sX73dD_--4tJZSYL7iA/certhash/uEiAvJLXzAcLYemvMqMRmR51KpS8259uYRSLzU0TDY7_IA
Swarm announcing /ip4/192.168.0.101/tcp/4001
Swarm announcing /ip4/192.168.0.101/udp/4001/quic-v1
Swarm announcing /ip4/192.168.0.101/udp/4001/quic-v1/webtransport/certhash/uEiD1magzNM5Hg5JPKn9rYnroYD4sX73dD_--4tJZSYL7iA/certhash/uEiAvJLXzAcLYemvMqMRmR51KpS8259uYRSLzU0TDY7_IA
Swarm announcing /ip6:::1/tcp/4001
Swarm announcing /ip6:::1/udp/4001/quic-v1
Swarm announcing /ip6:::1/udp/4001/quic-v1/webtransport/certhash/uEiD1magzNM5Hg5JPKn9rYnroYD4sX73dD_--4tJZSYL7iA/certhash/uEiAvJLXzAcLYemvMqMRmR51KpS8259uYRSLzU0TDY7_IA
RPC API server listening on /ip4/127.0.0.1/tcp/5001
WebUI: http://127.0.0.1:5001/webui
Gateway server listening on /ip4/127.0.0.1/tcp/8080
Daemon is ready
```

Рисунок 1.2 - Запуск IPFS

На рис. 1.2 наведені журнальні дані, що підтверджують успішний запуск демона IPFS. Тепер можемо використовувати його для управління файлами, генерації хешів та обміну даними з іншими користувачами IPFS.

1.5 Процедури завантаження та отримання зображень на IPFS

Процедура завантаження зображення зображення на IPFS передбачає кілька кроків, а саме:

1. Створення зображення.
2. Завантаження зображення.
3. Отримання хешу.

Демонстрація цієї процедури наведена на рис. 1.3–1.4.

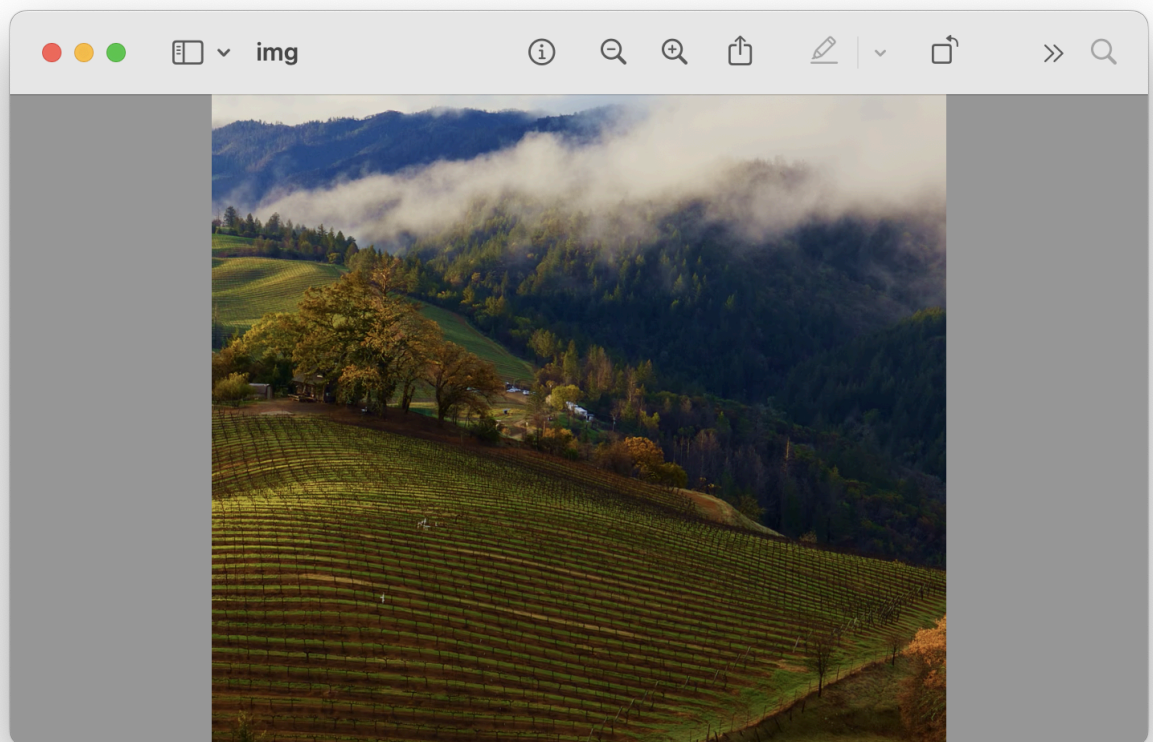
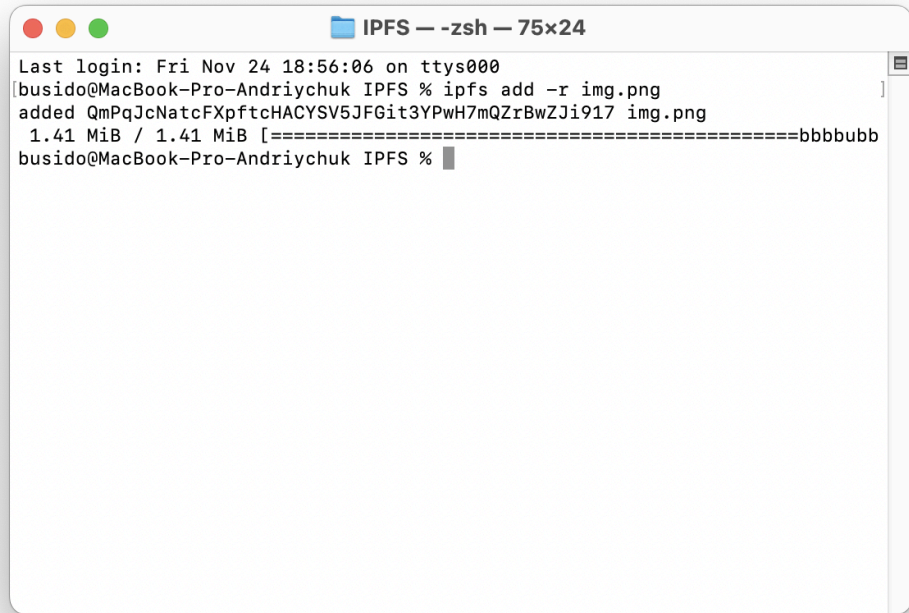


Рисунок 1.3 - Зображення

A terminal window titled "IPFS — -zsh — 75x24" with standard macOS window controls (red, yellow, green buttons). The terminal text shows a successful IPFS add command for a file named "img.png".

```
Last login: Fri Nov 24 18:56:06 on ttys000
[busido@MacBook-Pro-Andriychuk IPFS % ipfs add -r img.png
added QmPqJcNatcFXpftcHACYSV5JFGit3YPwH7mQZrBwZJi917 img.png
 1.41 MiB / 1.41 MiB [=====bbbbbb]
busido@MacBook-Pro-Andriychuk IPFS %
```

Рисунок 1.4 - Завантаження зображення і отримання хеша

Використовуючи раніше збережений кеш, можна отримати зображення назад.

Для того щоб отримати зображення, надсилаємо запит на мережу IPFS з CID. У мережі використовується Kademlia DHT та інші механізми для визначення, які піри (вузли) мають цей CID і можуть надати вміст. Після отримання інформації про піри, які мають вміст за вказаним CID, вибирається одного чи кілька пірів для отримання даних. Вибрані піри використовують протокол Bitswap для обміну блоками даних. Замість завантаження цілого файлу вузол може запитувати лише конкретні блоки, які він потребує. Коли вузол отримує блоки даних від інших пірів, він збирає їх разом для створення повного файлу за допомогою ідентифікатора CID. Отриманий файл може бути збережений на локальному пристрої клієнта для подальшого використання.

Виконуємо команду для отримання файлу і його збереження:

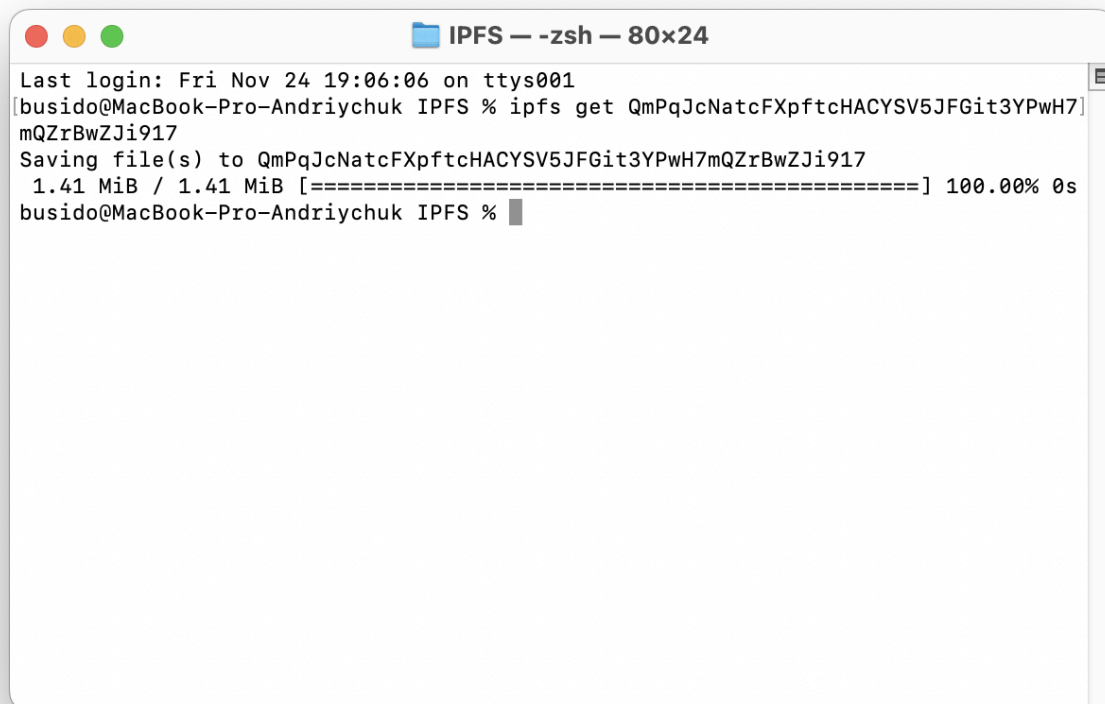


Рисунок 1.5 - Завантаження зображення з IPFS

Ми отримали файл з мережі, для того щоб операційна система правильно задекодувала інформацію з нього вкажемо більш читаєму назву і розширення:

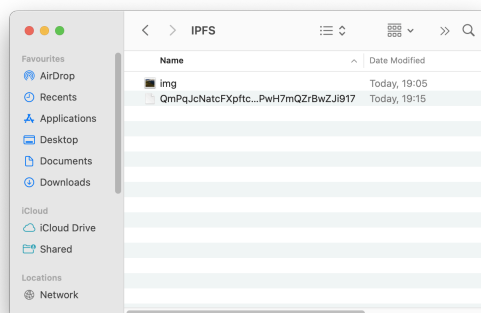


Рисунок 1.7 - Оригінально отриманий файл

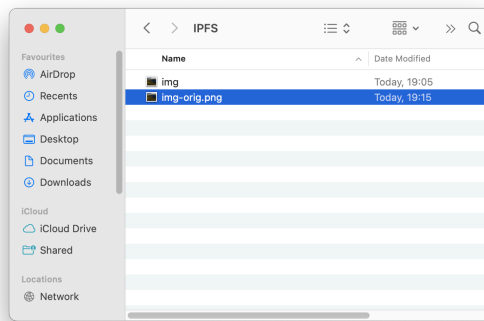


Рисунок 1.7 - Його коректно задекодована версія

Також можна використовувати розгорнуті локально HTTP канали для отримання зображення:

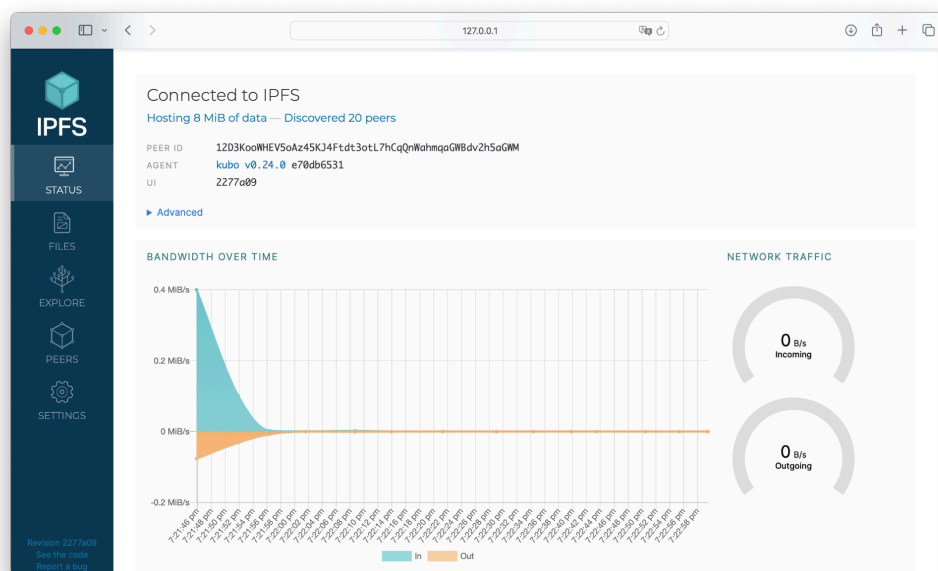


Рисунок 1.8 - локально розгорнути дашборд з інформацією

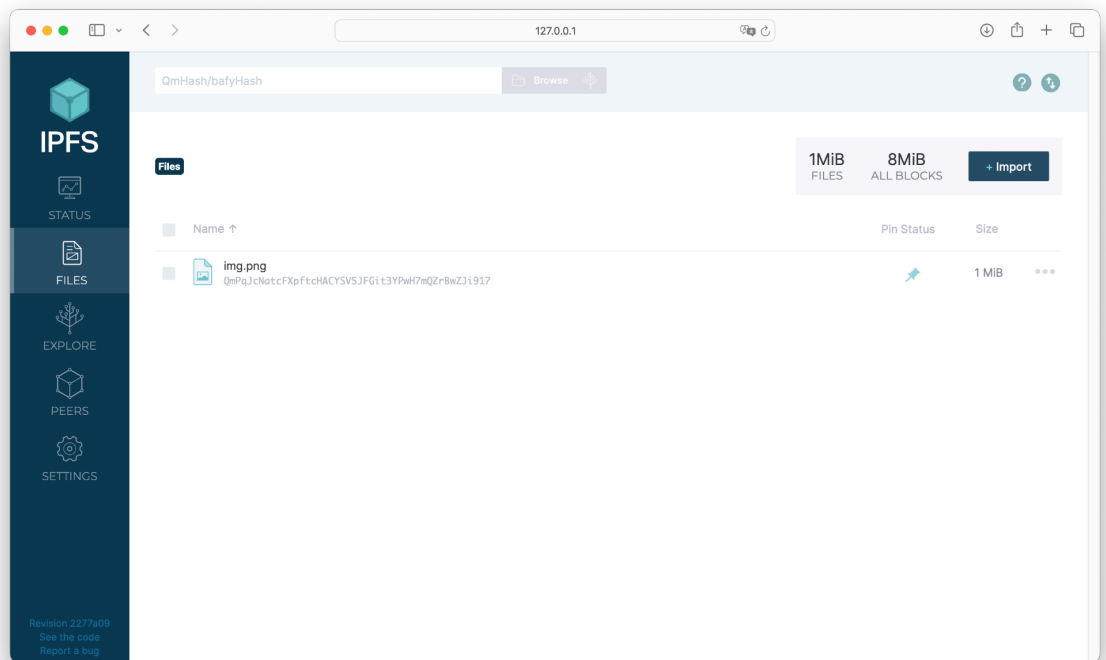


Рисунок 1.9 - Файл завантажений з комп'ютера

1.6 Застосування контейнеризації Docker в IPFS

Для того щоб дослідити кластеризацію і створення додаткового шляху контернізації вузлів IPFS, скористаємося Docker.

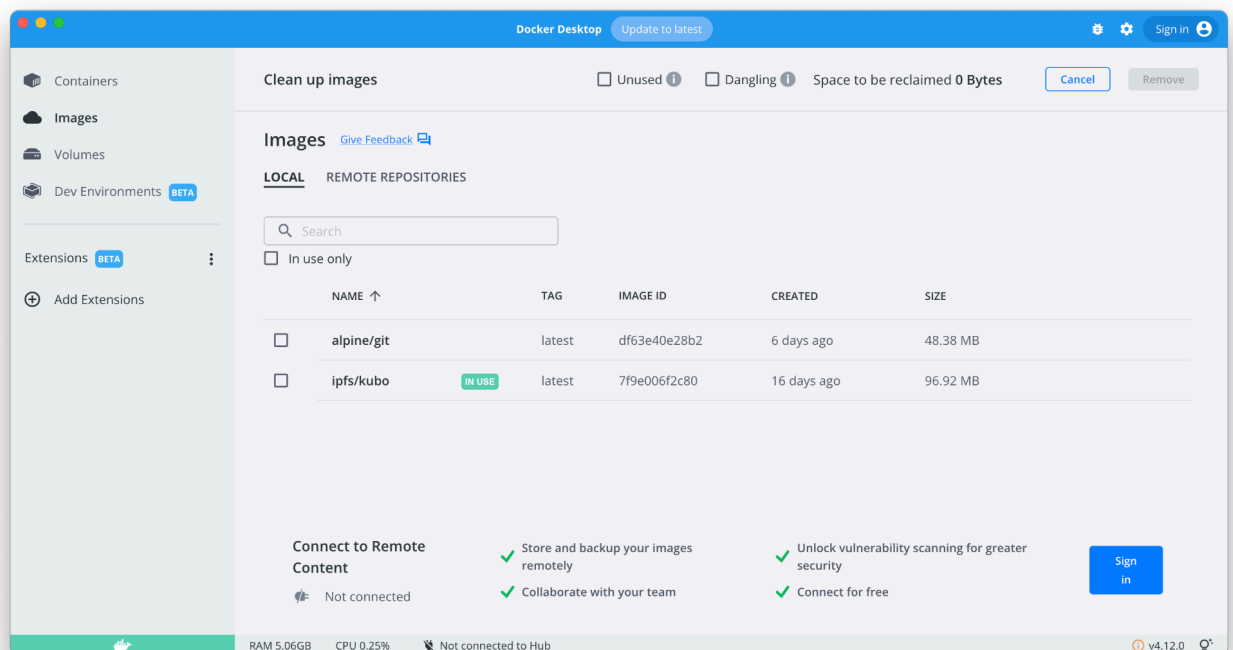
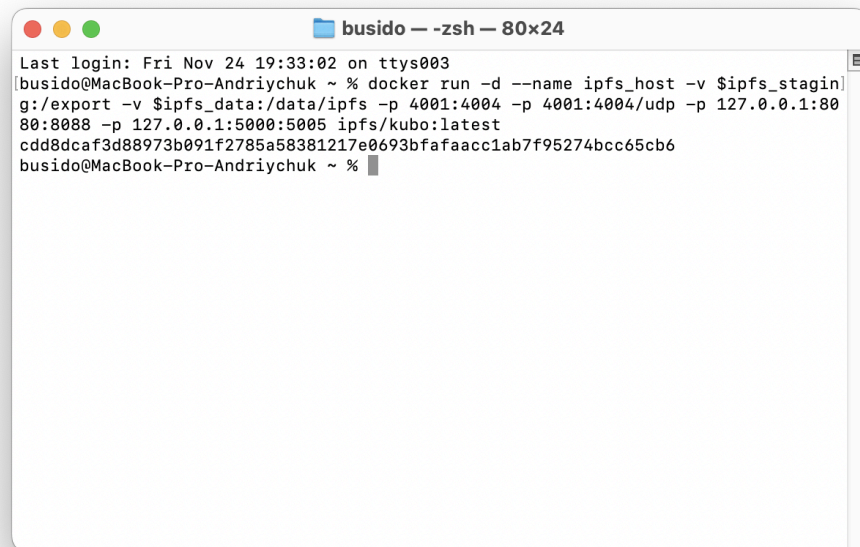


Рисунок 1.10 - Завантежимо Зображення IPFS

Створимо Контейнер для IPFS вузла:

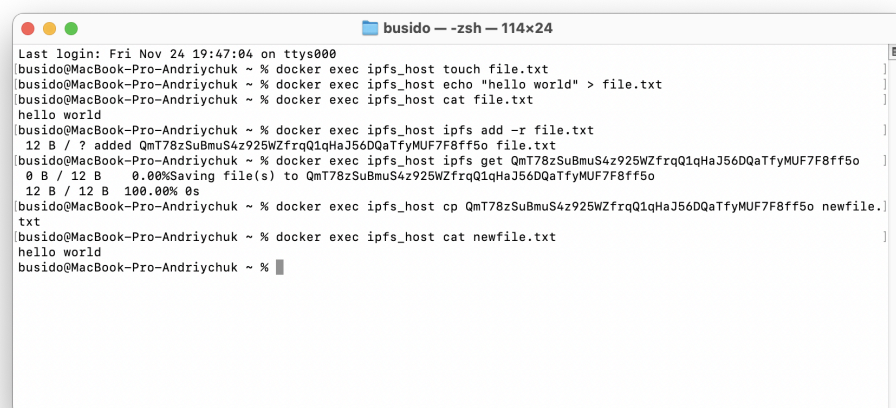


```
busido — -zsh — 80x24
Last login: Fri Nov 24 19:33:02 on ttys003
busido@MacBook-Pro-Andriychuk ~ % docker run -d --name ipfs_host -v $ipfs_stagin|
g:/export -v $ipfs_data:/data/ipfs -p 4001:4004 -p 4001:4004/udp -p 127.0.0.1:80
80:8088 -p 127.0.0.1:5000:5005 ipfs/kubo:latest
cdd8dc3d88973b091f2785a58381217e0693bfafaacc1ab7f95274bcc65cb6
busido@MacBook-Pro-Andriychuk ~ %
```

Рисунок 1.11 - створимо контейнер з назвою ipfs_host

Тепер створимо файл **file.txt** та додамо в нього текст. Наступним кроком завантажимо файл на IPFS і потім вивантежемо.

Щоб коректно декодувати інформацію з даних зкопіюємо вміст файлу в новий з назвою **newfile.txt** та виведемо на екран. [мал 12]



```
busido — -zsh — 114x24
Last login: Fri Nov 24 19:47:04 on ttys000
busido@MacBook-Pro-Andriychuk ~ % docker exec ipfs_host touch file.txt
busido@MacBook-Pro-Andriychuk ~ % docker exec ipfs_host echo "hello world" > file.txt
busido@MacBook-Pro-Andriychuk ~ % docker exec ipfs_host cat file.txt
hello world
busido@MacBook-Pro-Andriychuk ~ % docker exec ipfs_host ipfs add -r file.txt
12 B / ? added QmT78zSuBmuS4z925WZfrqQ1qHaJ56DQaTfyMUF7F8ff5o file.txt
busido@MacBook-Pro-Andriychuk ~ % docker exec ipfs_host ipfs get QmT78zSuBmuS4z925WZfrqQ1qHaJ56DQaTfyMUF7F8ff5o
0 B / 12 B 0.00%Saving file(s) to QmT78zSuBmuS4z925WZfrqQ1qHaJ56DQaTfyMUF7F8ff5o
12 B / 12 B 100.00% 0s
busido@MacBook-Pro-Andriychuk ~ % docker exec ipfs_host cp QmT78zSuBmuS4z925WZfrqQ1qHaJ56DQaTfyMUF7F8ff5o newfile.
txt
busido@MacBook-Pro-Andriychuk ~ % docker exec ipfs_host cat newfile.txt
hello world
busido@MacBook-Pro-Andriychuk ~ %
```

Рисунок 1.12 - Результат отриманого файлу

Висновок до Розділу 1

В даному розділі я описав основні методи передавання інформації в децентралізованих мережах, та приділив увагу характеристикам протоколу IPFS. Дійшов до висновку, що передача даних через протокол IPFS відрізняється від традиційних методів передачі файлів через Інтернет. Вона базується на хеш-ідентифікаторах файлів і можливості пошуку та обміну файлами між різними вузлами мережі. Це робить IPFS ефективним і розподіленим рішенням для зберігання та передачі даних.

Також в даному розділі наведені описи та процеси створення вузлу IPFS для розуміння процесів, які відбуваються при передачі файлів в децентралізованій мережі.

Розібрали процеси отримання та завантаження файлів, шляхом збереження хешів зображень та використання підпроцесів вузла IPFS для перенаправлення, адресування та роутингу на шлюзи IPFS.

Розділ 2. Проектування системи обміну інформацією та мобільного додатку

2.1 Проектування системи обміну інформацією на основі IPFS

В процесі проектування я реалізував архітектуру залежності наступним чином:

1. додаток, як інтерфейс користувача;
2. обробник IPFS запитів з задекларованими методами роботи з вузлом IPFS;
3. вузол IPFS.

Додаток реалізує механізми і інтерфейси, через які користувач може комунікувати з зображеннями та робити операції над ними. В даній роботі моєю метою було створити застосунок, який зможе зберігати і отримувати раніше збережене зображення в вузол IPFS, однак не перевантажувати сам додаток інструкціями, як саме це робити.

При проектуванні Обробника, я покладався на простоту реалізації його як залежності основного додатку. Він реалізовує інтерфейси, за якими мобільний застосунок в цій магістерській роботі, або будь який інший застосунок, має змогу виконувати операції з вузлом, шляхом виклику відповідних методів. Обробник запитів використовує вбудований механізм URLSession у мові програмування Swift та реалізує методи спілкування з IPFS вузлом.

URLSession - це високорівневий інтерфейс, який надає API для виконання різних мережових операцій. В конкретному випадку, використання `dataTask` методу URLSession ініціює асинхронний HTTP GET-запит.

Метод `dataTask` створює об'єкт завдання, яке може завантажувати дані з Інтернету. Це асинхронна операція, і вона використовує замикання (`completion handler`), щоб сповістити вашу програму, коли завдання буде

завершено. Після того, як запит виконаний, отримані дані можна обробити в цьому замиканні.

Щодо передачі файлів, в даному випадку метод призначений для виконання GET-запитів, які не вимагають відправки файлів. Якщо потрібно відправити файли, можна використовувати метод `dataTask`. Для цього, необхідно створити запит типу POST або інший, що підтримує відправку файлів, і викликати `dataTask` з цим запитом та вказати шлях до файлу або дані, які потрібно відправити. В даному випадку ми кодуємо дані зображення і відправляємо саме інформацію що це файл.

Цей обробник запитів також імплементований як Swift Package, що є зручним засобом для організації та розповсюдження коду між проектами. Swift Package дозволяє пакетам Swift мати структуру, в тому числі визначені залежності, та легко інтегрується в інші проекти.

Посилання на репозиторій наведено у додатку.

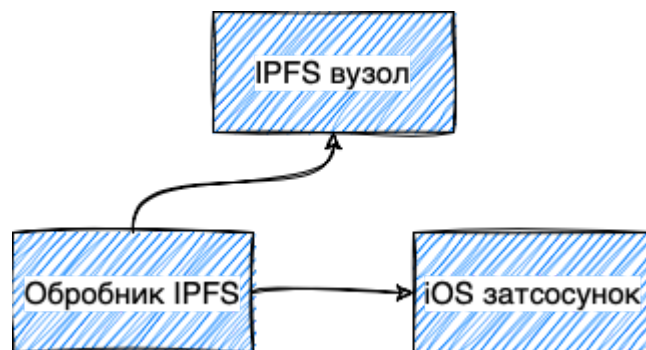


Рисунок 2.13 - схема залежності Обробника та застосунку з IPFS вузлом

2.2 Проєктування мобільного застосунку на iOS

В рамках магістерської роботи, як студент я створив мобільний застосунок для iOS з використанням технології SwiftUI. Застосунок має три основні екрани:

Початковий екран: Початковий екран містить дві кнопки: "Завантажити зображення" та "Зберегти зображення". При натисканні на

кнопку "Зберегти зображення" користувач переходить на екран для вибору зображення з галереї та можливість зберегти його на локальний вузол IPFS.

Екран вибору зображення для збереження: На цьому екрані користувач може вибрати зображення зі своєї галереї за допомогою кнопки "Вибрати зображення". Обране зображення можна зберегти на локальний вузол IPFS, натиснувши відповідну кнопку.

Екран завантажених зображень: При натисканні кнопки "Завантажити зображення" на початковому екрані користувач переходить на екран, де відображається список раніше завантажених зображень. Кожен елемент списку включає відомості про хеш зображення. Користувач може вибрати конкретне зображення для перегляду.

Цей застосунок реалізується з використанням SwiftUI для побудови інтерфейсу користувача, а обробка завдань, таких як завантаження та збереження зображень, виконується за допомогою відповідних API. Робота з локальним вузлом IPFS реалізується з використанням бібліотек реалізованої і описаної вище та присутня взаємодії з файловим системним простором на iOS.

2.3 Інтерфейси та User-flow

Розглянемо взаємодію користувача і застосунку.

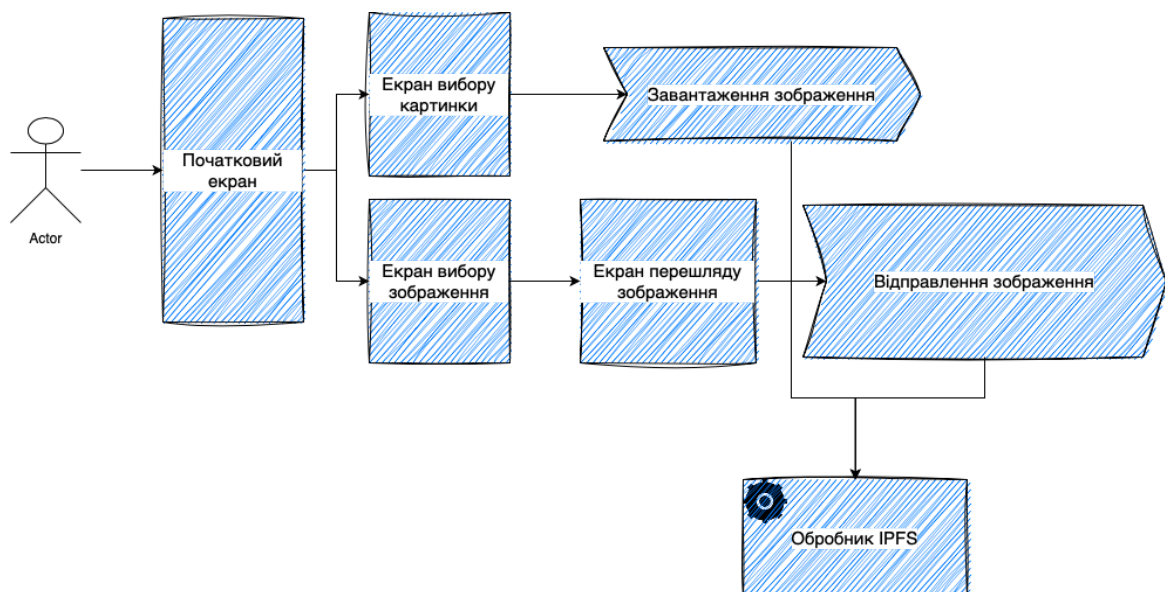


Рисунок 2.15 - Опис порядку взаємодії користувача і додатку

Застосунок в свою чергу спілкується з Обробником IPFS, який передає інформацію на вузол, розгорнутий локально.

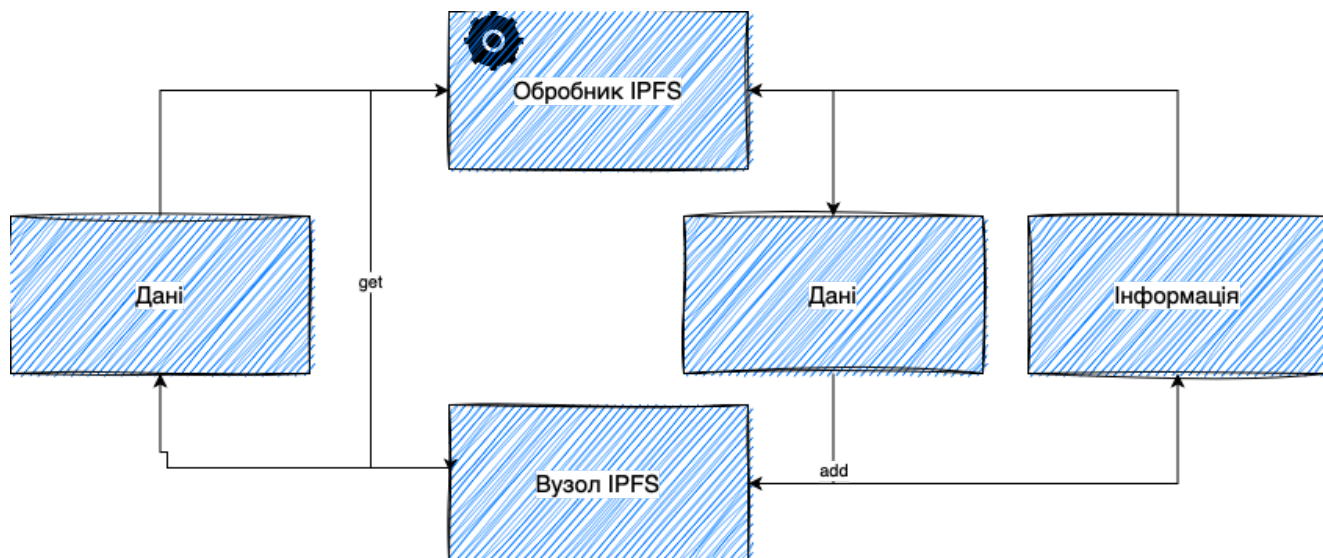


Рисунок 2.16 - Опис порядку взаємодії Обробника та вузла IPFS

Даний проект дозволяє гнучко використовувати будь які елементи системи. Команда людей може працювати окремо над реалізаціями функціоналу, незалежно один від одного, створюючи швидку та якісну розробку програмного забезпечення.

Висновок до розділу 2

В даному розділі я спроектував систему обміну зображенням використовуючи протокол IPFS з вузлом та мобільний застосунок.

Вдалося реалізувати проект мобільного застосунку зі всіма імплементаціями інтерфейсних частин. Дані інтерфейсні частини в повній мірі відповідають завданням, які поставлені застосунку і перед користувачем.

Продемонстрував порядок взаємодії користувача з системою. Показав як саме система працює після відправки запиту на збереження чи отримання зображення.

Розділ 3. Практична реалізація додатку для захищеного обміну зображеннями за протоколом IPFS

3.1 Обґрунтування вибору технології та реалізація додатку

При розробці мобільного iOS додатку для захищеного обміну зображеннями за протоколом IPFS у децентралізованій мережі, використання SwiftUI може бути обґрунтованим з кількох причин.

Спочатку, декларативний підхід SwiftUI дозволяє розробникам описувати бажаний вигляд інтерфейсу, а не писати велику кількість імперативного коду для створення його. Це спрощує процес розробки і робить код більш зрозумілим.

SwiftUI гармонійно інтегрується з іншими технологіями в екосистемі Apple, такими як Combine, Core Data та інші, що спрощує використання реактивного програмування та роботу з базою даних.

Також, розробка для різних платформ Apple стає більш зручною завдяки SwiftUI, оскільки він дозволяє створювати додатки для iOS, macOS, watchOS та tvOS, використовуючи спільний код та інтерфейс.

Спільнота розробників активно вкладає зусилля в розвиток SwiftUI, надаючи різноманітні ресурси та підтримку, що може значно полегшити вивчення та використання цього фреймворку.

На підсумок, використання SwiftUI в даному проекті може сприяти швидкій, зрозумілій та ефективній розробці мобільного додатку для обміну зображеннями в децентралізованій мережі на платформі iOS.

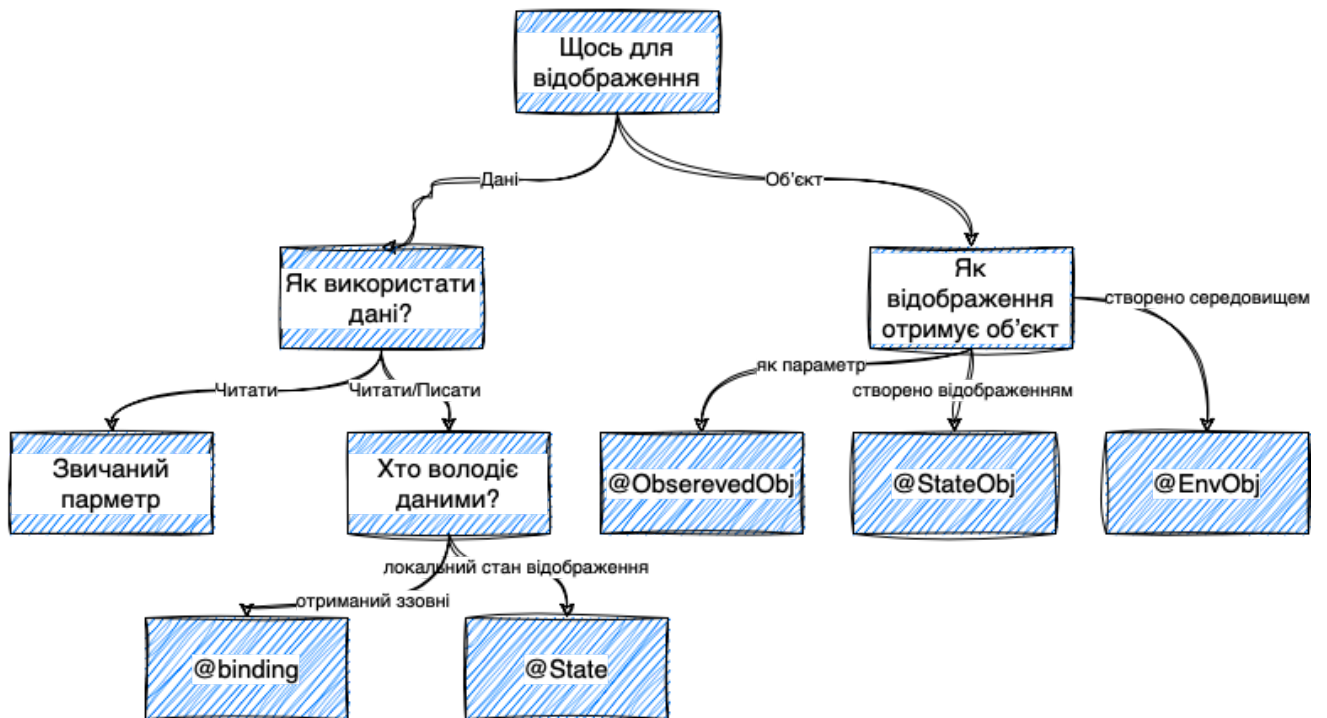


Рисунок 2.14 - Опис інформації і її обробки в стандартному застосунку SwiftUI

3.2 Опис інтерфейсу та процедур застосування додатку

В даній роботі я створив iOS застосунок під назвою AppIPFS.
Початковий екран:

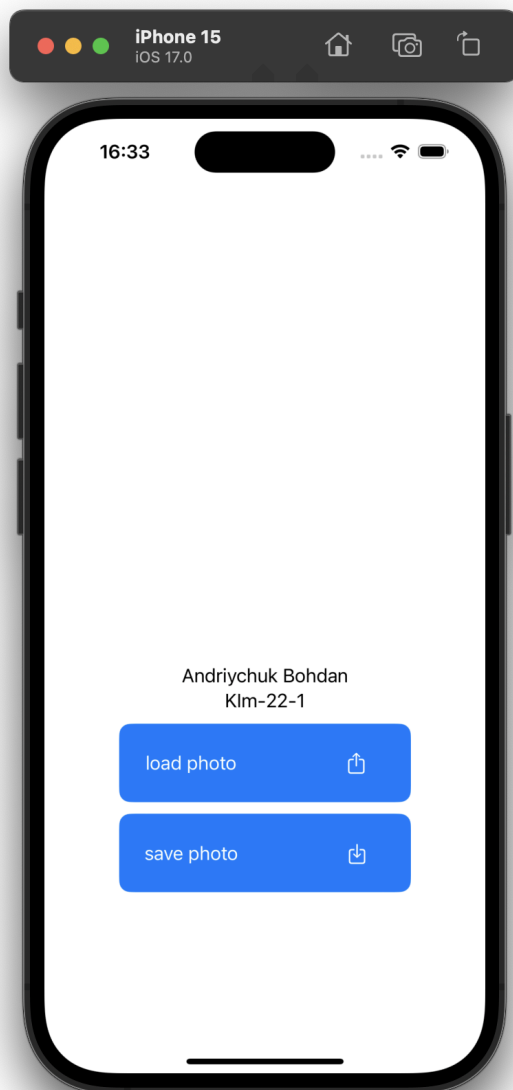


Рисунок 3.17 - Початковий екран

Користувач натискає на кнопку зберегти зображення.

Наступний екран:

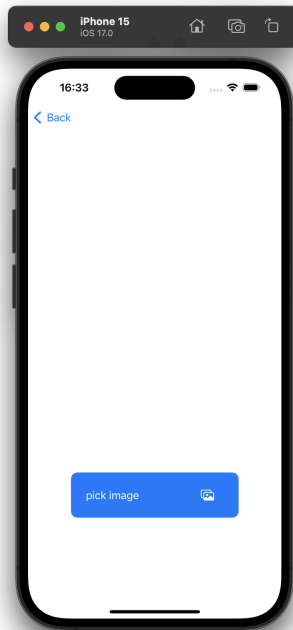


Рисунок 3.18 - Экран выбора изображения для сохранения

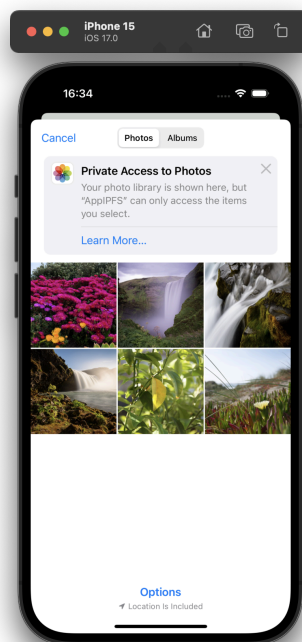


Рисунок 3.19 - Вибір зображення для збереження

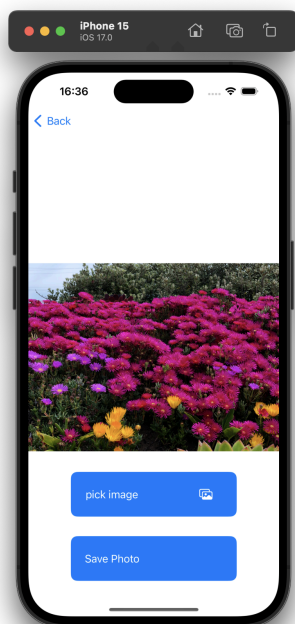


Рисунок 3.20 - Екран вибору зображення для збереження

Після того як користувач вибрав зображення, ми можемо зберегти його. Натиснувши на кнопку зберегти ми завантажуюмо його на IPFS вузол. Додаток отримує інформацію про зображення та передає її в Обробник. Інформацію зображення попередню кодуємо в Base64, записуємо закодовану інформацію в файл і відправляємо його в вузол IPFS через Обробник. Обробник використовує URL запит в який закодує інформацію в файл і форматі Multifom. Таким чином ми реалізуємо задекларований процес Збереження в IPFS. Локальний вузол IPFS отримує файл з зображенням та проводить процес збереження зображення, описаний в попередніх розділах. Цей процес займає деякий час, в залежності від мережевого з'єднання, тому всі ці процеси відбуваються в фонових режимах для того, щоб користувач мав умови користування кісного додатку та без підвисання.

Поки завантажуються зображення, покажемо іконку завантаження, як показано на Рис. 3.21.

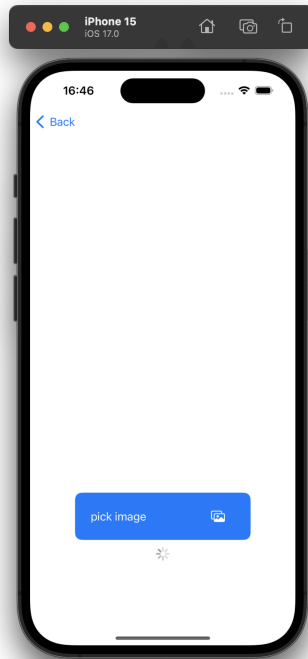


Рисунок 3.21 - Завантаження фото

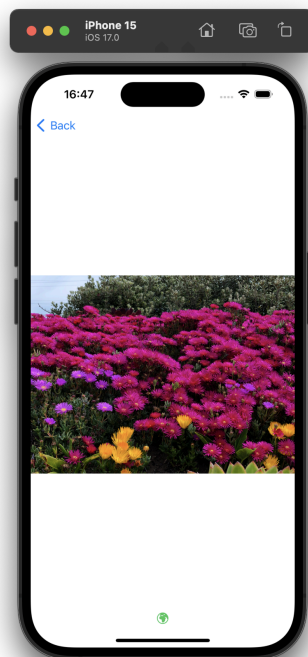


Рисунок 3.22 - Результат завантаження фото

Після того як завантажили, ми можемо побачити значок успішного завантаження, та отримати в логах інформацію з вузла IPFS:

```
Response:
Optional("
{\"Name\": \"filenamenoext\"
, \"Hash\": \"QmV9mr3gkk1j77DfJEtNW5Tzg2phZTp1ZjNkzEuPTuUphv\"
, \"Size\": \"5504066\"} \\n")
```

Рисунок 3.23 - Інформація з логів Обробника

Ця інформація нам дуже корисна, вона містить розмір файлу, завантаженого на вузол IPFS, назву файлу, та його хеш.

Зберігаємо хеш зображення та використання для отримання його, повернувшись на Початковий екран.

Натиснувши Back (назад), ми перейдемо до початкового екрану і тепер дістанемо таблицю з завантаженими нами файлами:

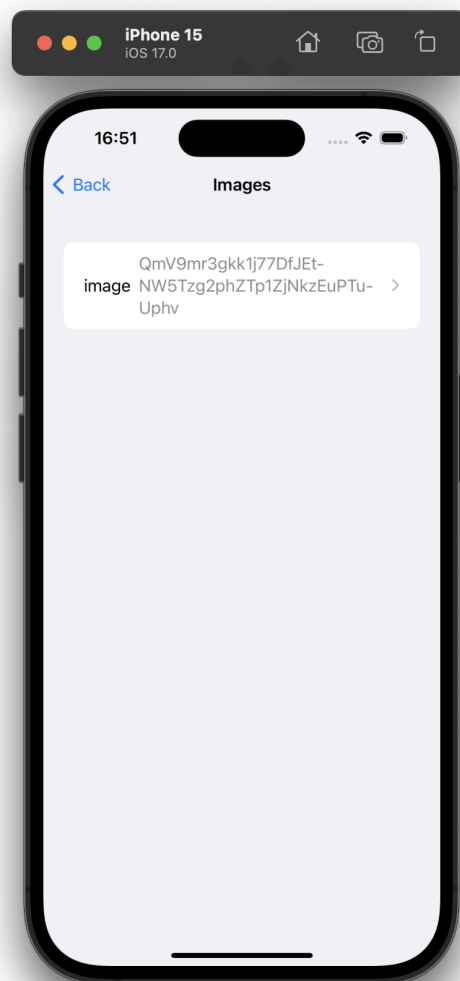


Рисунок 3.24 - Список завантажених зображень

Тепер вибравши відповідну комірку ми можемо завантажити фото:

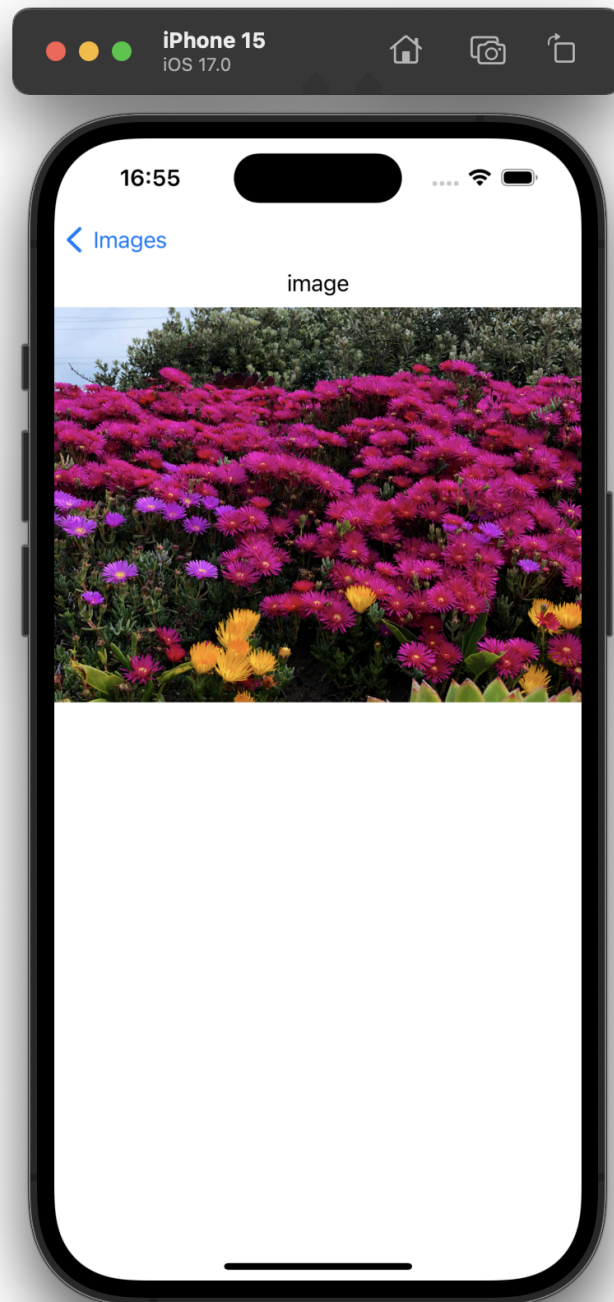


Рисунок 3.25 - Отримане зображення

Завантаження зображення відбувається за розглянутими нами методами в попередніх розділах. Ми отримуємо дані з файлу, вичитуємо їх. Так як ми записували закодовані в Base64 інформацію зображення, ми

можемо декодувати її і проініціалізувати з ними зображення. Результат ви можете бачити на Рис. 3.25.

3.3 Тестування функціоналу додатку

Для того щоб переконатися що наше зображення було завантажено в вузол, можемо отримати його з самого вузла безпосередньо з консолі. Для цього використаємо той самий хеш що і в попередній раз (в додатку):



```
busido@macbook IPFS % ipfs get QmV9mr3gkk1j77DfJEtNW5Tzg2phZTp1ZjNkzEuPTuUphv
Saving file(s) to QmV9mr3gkk1j77DfJEtNW5Tzg2phZTp1ZjNkzEuPTuUphv
5.25 MiB / 5.25 MiB [=====] 100.00% 0s
busido@macbook IPFS % cp QmV9mr3gkk1j77DfJEtNW5Tzg2phZTp1ZjNkzEuPTuUphv data.txt
busido@macbook IPFS %
```

Рисунок 3.26 - Запит на отримання інформації з вузла IPFS

Після того як викачали вміст файлу з IPFS, копіюємо його вміст в файл з назвою data.txt. Відкриємо цей файл:

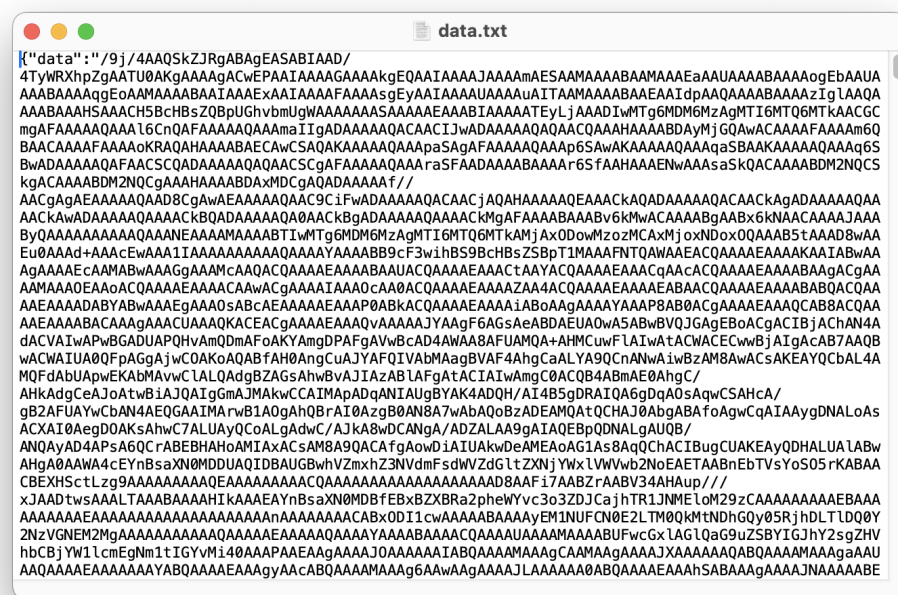


Рисунок 3.27 - Закодована інформації картки з Застосунку

Так як наша інформація передається в закодованому вигляді ми можемо бачити, що дані записані в файл закодовані.

Висновок до розділу 3

В цьому розділі ми реалізували процес створення та проектування мобільного застосунку, вузла IPFS та Обробника IPFS. Створили систему в, якій всі ці елементи мають місце і працюють для збереження та отримання зображення.

Було виконано і реалізований застосунок з інтерфейсами та механізмами відображення зображення.

Було реалізовано Обробник IPFS запитів, завдання якого є спілкування з локальним IPFS вузлом.

Протестували роботу додатку, напряму спілкуючись з вузлом IPFS.

Висновки

У цій роботі висвітлено основні методи передавання інформації в децентралізованих мережах через протокол IPFS, який відрізняється своєю ефективністю та розподіленістю. Описано процеси створення та взаємодії вузлів IPFS, а також механізми отримання та завантаження файлів.

Мені вдалося спроектувати систему обміну зображеннями з використанням IPFS, реалізувати мобільний застосунок для iOS. В роботі зазначено успішне втілення інтерфейсних частин та взаємодію користувача з системою. Також описано процес створення та проектування мобільного застосунку, вузла IPFS та Обробника IPFS, які працюють разом для збереження та отримання зображень. Виконано реалізацію інтерфейсів та механізмів відображення зображень, а також протестовано додаток взаємодії з вузлом IPFS.

Варто зазначити, що виконано успішний проект з імплементацією та тестуванням системи обміну зображеннями на базі протоколу IPFS: мобільний застосунок, вузол та обробник IPFS запитів. Система відповідає вимогам, забезпечуючи ефективну обробку та збереження даних в централізованому середовищі. В роботі описане глибоке розуміння методів передавання даних через IPFS, його характеристик та процесів роботи з вузлами мережі.

Проект мобільного застосунку і вузла IPFS відзначається успішною реалізацією інтерфейсів, взаємодії з користувачем та співпраці з локальним вузлом IPFS. Важливо відзначити розробку оптимізованих та надійних механізмів обробки та збереження зображень у децентралізованій мережі.

Узагальнюючи, система визнана як високоефективна та перспективна, забезпечуючи необхідні функціональність та безпеку для задоволення потреб користувачів в децентралізованому обміні

зображеннями. Одним з елементів оптимізації - це кластеризація вузлів. В попередніх розділах я розглянув можливість створення кластерів IPFS вузлів в Docker, щоб забезпечити високу доступність та надійність. також географічний розподіл важливий елемент удосконалення системи, так як розгортання IPFS вузлів у різних географічних областях дає можливість для забезпечення оптимальної швидкості передачі даних. Можливість створення централізованого сервісу для реєстрації та управління мобільними застосунками з використанням IPNS (InterPlanetary Name System) для обміну метаданими між мобільними застосунками та забезпечення актуальності даних допоможе створити стійку і якісну мережу. Масштабування мобільних iOS застосунків є наступним кроком. Використання методів оптимізації зображень перед їхнім завантаженням або передачею для зменшення обсягу даних та прискорення процесу передачі. Також можливість кешування та локального збереження зображень на пристроях користувачів дає якісні результати та дає змогу для зменшення необхідності завантаження даних знову.

Подальший розвиток системи може включати в себе реалізацію запропонованих заходів, таких як географічний розподіл вузлів, масштабування мобільних застосунків, розширення функціоналу та можливості монетизації. Захист конфіденційності, аутентифікація та авторизація дозволять створити безпечне середовище для користувачів.

Список використаної літератури

1. "Розробка мобільних додатків для iOS: Програмування на Swift" / Пол Маккартні. – К.: Видавництво "Діалектика", 2020.
2. "Розробка мобільних додатків для Android: Програмування на Kotlin" / Бретт Маклафлін. – К.: Видавництво "Діалектика", 2021.
3. "Розробка мобільних додатків з React Native" / Алекс Шарман. – К.: Видавництво "Діалектика", 2019.
4. "Професійна розробка мобільних додатків для Android з використанням Java" / Білл Феллер, Гансон Чарльз. – К.: Видавництво "Кодекс", 2018.
5. "Розробка мобільних додатків з використанням React Native: Кроссплатформенні рішення для iOS та Android" / Етіенне Бурлі. – К.: Видавництво "Кодекс", 2020.
6. "Розробка Android-додатків за допомогою Android Studio: Повний посібник" / Філіпс, Білл. – К.: Видавництво "Кодекс", 2019.
7. "Swift: Розробка додатків для iOS та macOS" / Поль Хадсон. – К.: Видавництво "Діалектика", 2021.
8. "Android-програмування: Високий ступінь важкості" / Ірина Гуманіцька. – К.: Видавництво "Діалектика", 2019.
9. "React Native в дії" / Нейджел Борден, Габріель Пілес. – К.: Видавництво "Діалектика", 2018.
10. "Розробка мобільних додатків на Kotlin: Реальні приклади з Google Play" / Люк Дженкінс. – К.: Видавництво "Діалектика", 2020.
11. "Програмування для Android: Повний посібник від Big Nerd Ranch" / Білл Філіпс, Кріс Стюарт, Брайан Гарднер. – К.: Видавництво "Кодекс", 2017.
12. "Програмування для iOS: Повний посібник від Big Nerd Ranch" / Джо Консоле, Аарон Хільгесс, Кріс Ландау. – К.: Видавництво "Кодекс", 2019.
13. "Володіння React Native" / Ерік Масіелло, Джейкоб Фрідманн.

– К.: Видавництво "Діалектика", 2018.

14. "Розробка мобільних додатків з Ionic 2: Кросплатформенні додатки з Ionic 2, Angular 2 і Cordova" / Кріс Гріффіт. – К.: Видавництво "Діалектика", 2017.

15. "Flutter для початківців: Вступ до створення кросплатформених мобільних застосунків з використанням Flutter і Dart 2" / Алессандро Біессек. – К.: Видавництво "Діалектика", 2020.

16. "Програмування на Swift UI для iOS: Вступ до Swift з використанням Xcode для початківців" / Крейг Клейтон. – К.: Видавництво "Діалектика", 2019.

17. "Внутрішня безпека Android: Глибокий посібник до архітектури безпеки Android" / Ніколяй Еленков. – К.: Видавництво "Кодекс", 2015.

18. "Програмування на iOS 13 для початківців: Початок роботи з побудови iOS-додатків за допомогою Swift 5.1 і Xcode 11, 4-е видання" / Крейг Клейтон. – К.: Видавництво "Діалектика", 2019.

19. "Професійний Android, 4-е видання" / Рето Мейєр. – К.: Видавництво "Кодекс", 2018.

20. "Swift в глибину" / Тірд ін 'т Він. – К.: Видавництво "Діалектика", 2015.

Додатки

Посилання на репозиторій додатку

1. Репозиторій Обробника IPFS-handler:
 - <https://github.com/dananddy/IPFS-handler.git>
2. Репозиторій додатку AppIPFS:
 - <https://github.com/dananddy/AppIPFS.git>