ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет радіоелектроніки
Кафедра ЕОМ

Модель децентралізованого зберігання даних для децентралізованого застосунку і смарт-контракту у мережі Ethereum

Кваліфікаційна робота
другий (магістерський) рівень

Виконав:
студент групи СПм-23-2
Селіванов Іван Олексійович

Керівник:
проф. Фесенко Т.Г

Харків
2025

Мета роботи та завдання

Об'єктом дослідження є децентралізована система зберігання даних, заснована на технології блокчейн.

Предметом дослідження є особливості функціонування смарт-контрактів та їх використання для створення моделі децентралізованого зберігання даних.

Метою роботи є створення моделі децентралізованого зберігання даних для децентралізованого застосунку, яка забезпечує прозорість, безпеку та підтвердження автентичності цифрових активів.

Завдання:

- дослідити актуальні проблеми інформаційної безпеки;
- окреслити наявні методи забезпечення безпеки даних;
- проаналізувати можливості та принципи роботи мережі Ethereum;
- розробити смарт-контракт за стандартом ERC-721;
- розробити децентралізований застосунок для зберігання даних та перевірити його роботу на практиці.

Існуючі варіанти забезпечення інформаційної безпеки: 3

- нормативно-правові акти;
- шифрування;
- фаєрволи;
- моніторинг та аудити безпеки;
- антивірусне ПЗ.

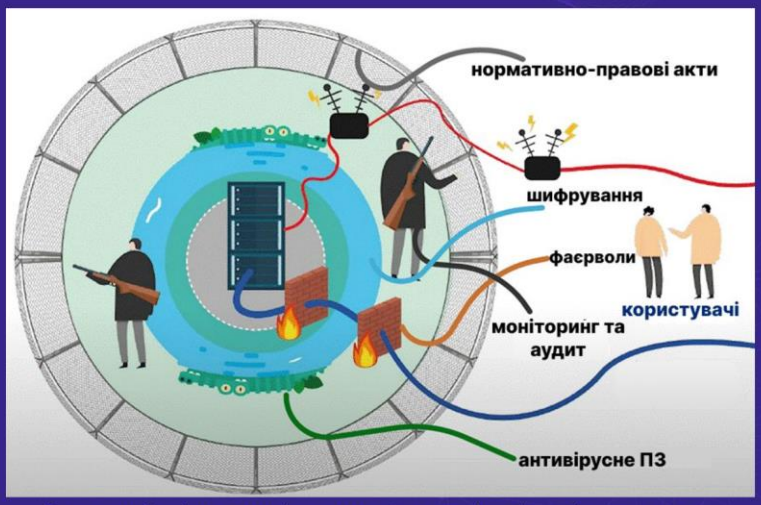


Рисунок 1 – Ілюстрація комплексу варіантів забезпечення інформаційної безпеки

Аналіз існуючих варіантів забезпечення інформаційної безпеки 4

Зазначені методи безпеки не завжди повноцінно виконують свої функції.

Це продемонструвала остання масштабна кібератака на реєстри Міністерства юстиції України. Фактично було паралізовано значну частину господарської діяльності в країні на декілька тижнів.

Результат атаки: після потрапляння хакерів в інфраструктуру міністерства відбулось викрадення та видалення понад 1 мільярда рядків даних.

Рисунок 2 – Графік кількості кібератак на державні структури за даними Держспецзв'язку

Основні властивості даних в блокчейн-системах:

5

- зашифрованість;
- незмінність;
- прозорість;
- токенізація;
- розподіленість;
- децентралізованість.

Рисунок 3 – Основні властивості даних в блокчейні

Модель та алгоритм роботи децентралізованого застосунку

6

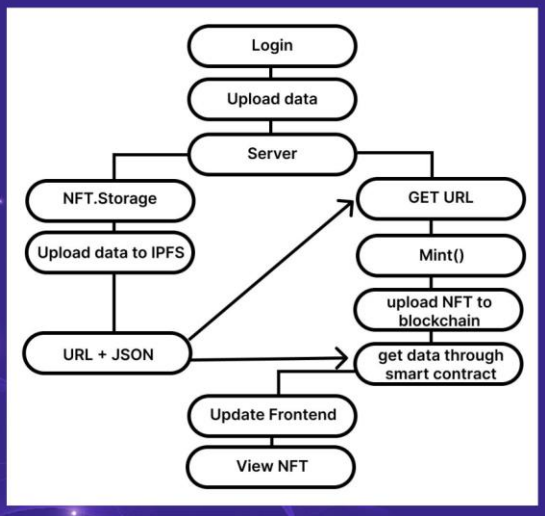


Рисунок 4 – Модель децентралізованого застосунку

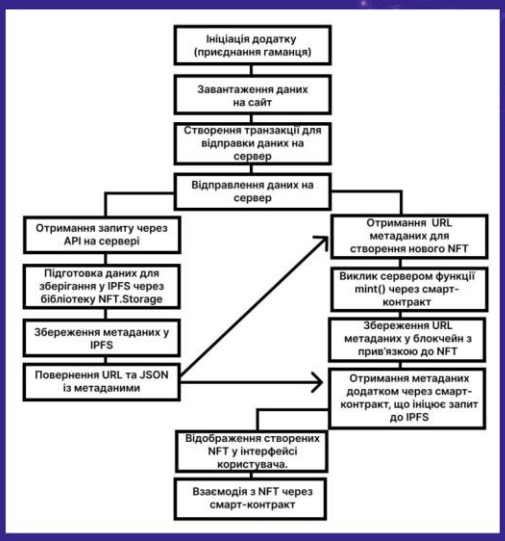


Рисунок 5 – Алгоритм роботи застосунку

Вибір технологій для реалізації та функціональні можливості застосунку

7

Технології:

- в якості IDE було обрано Visual Studio Code;
- смарт-контракт — Solidity;
- фронтенд та сервер — TypeScript та React;
- стандарти ERC — OpenZeppelin;
- збереження даних — NFT.Storage;
- проведення транзакцій —Metamask;
- розгортання та тестування — Etherscan.

Функціональні можливості:

- інтеграція гаманця;
- створення NFT;
- взаємодія з смарт-контрактом;
- перегляд балансу гаманця;
- децентралізоване зберігання;
- моніторинг транзакцій;
- розгортання контракту;

8

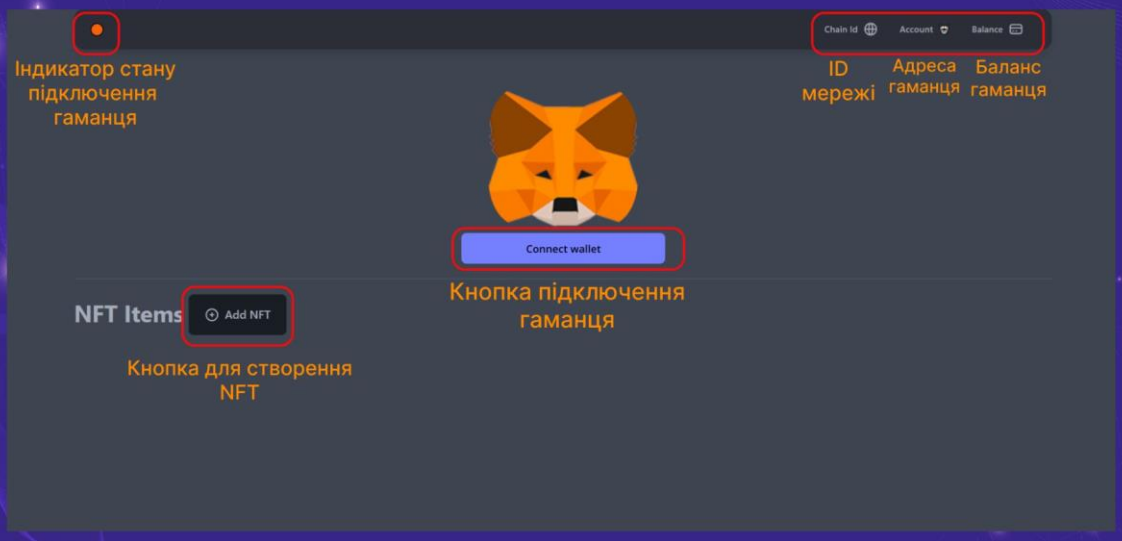


Рисунок 6 – Інтерфейс головної сторінки сайту

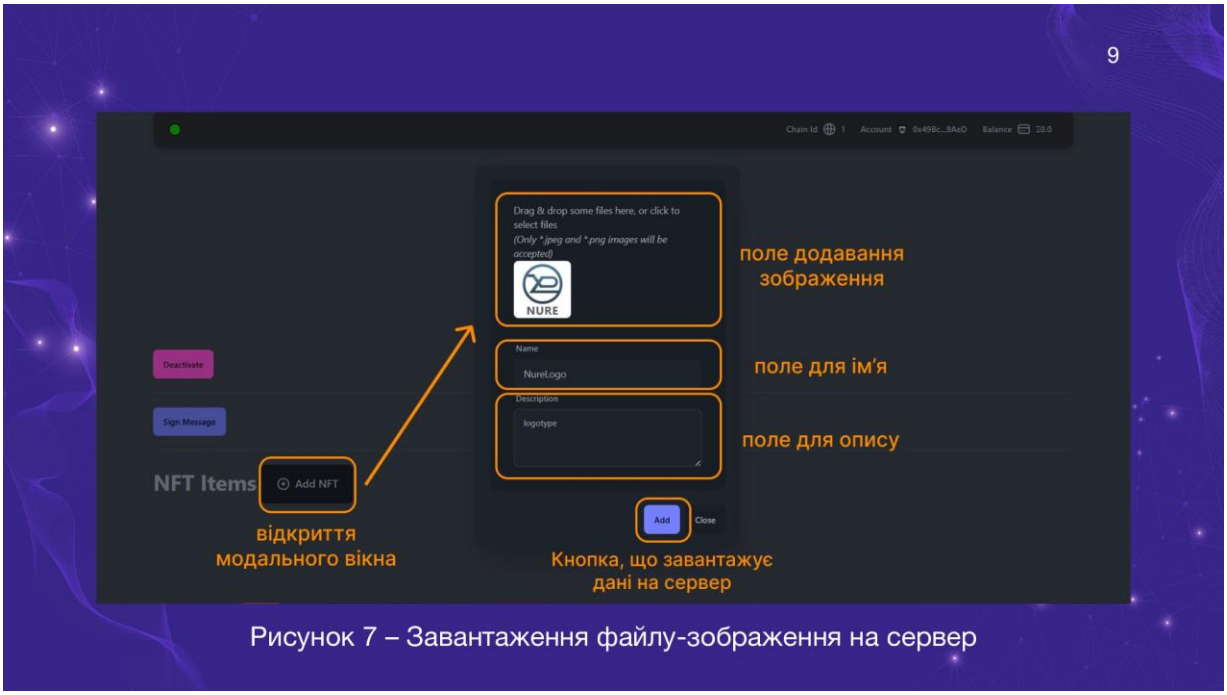


Рисунок 7 – Завантаження файлу-зображення на сервер

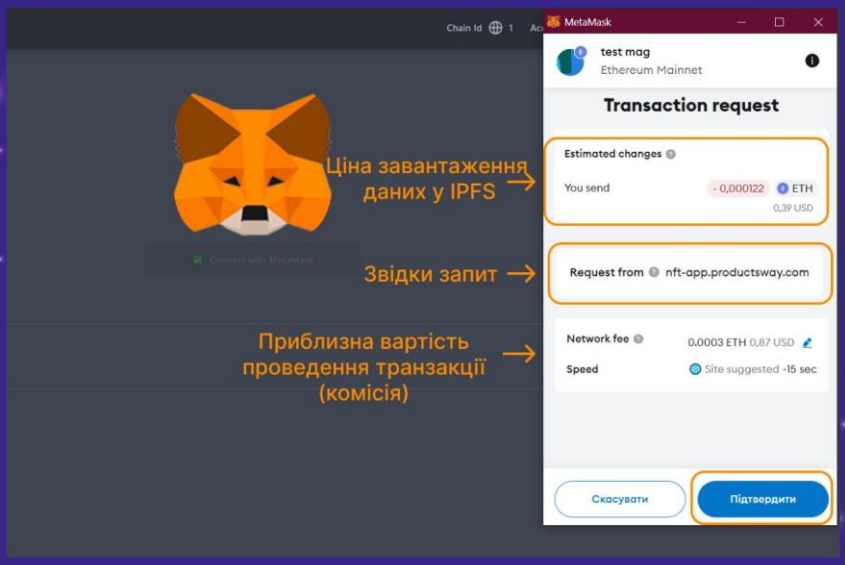


Рисунок 8 – Підписання транзакції, що завантажує дані у IPFS через бібліотеку NFT.Storage

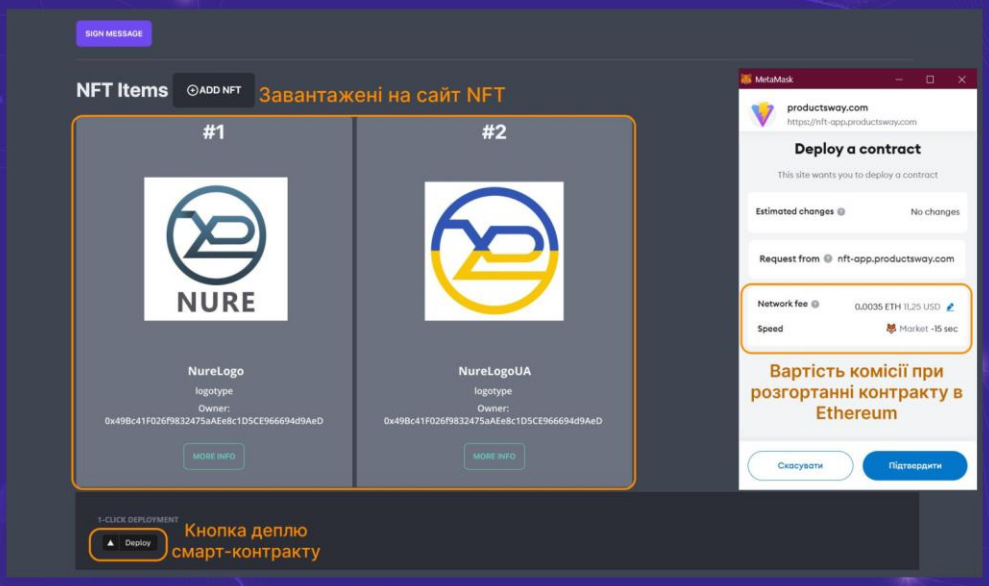


Рисунок 9 – Розгортання смарт-контракту

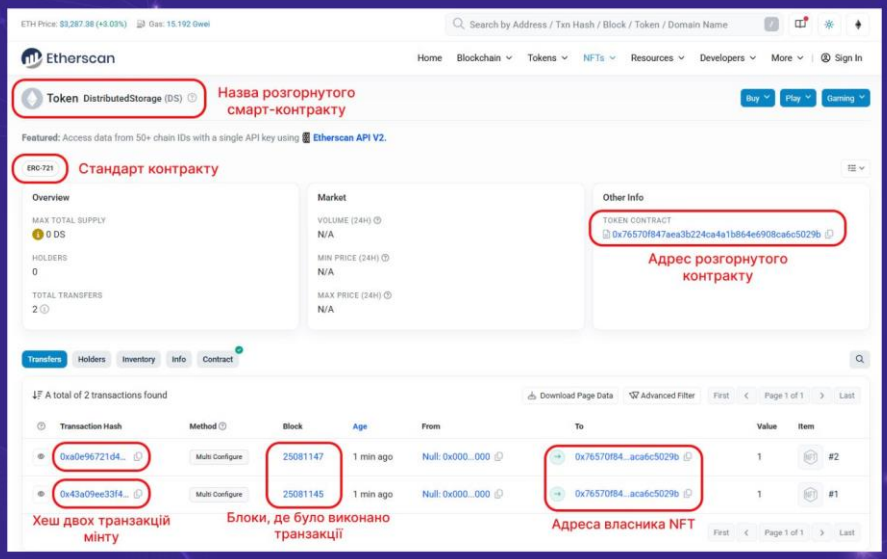


Рисунок 10 – Сайт Etherscan та інформація про розгорнутий смарт-контракт

13

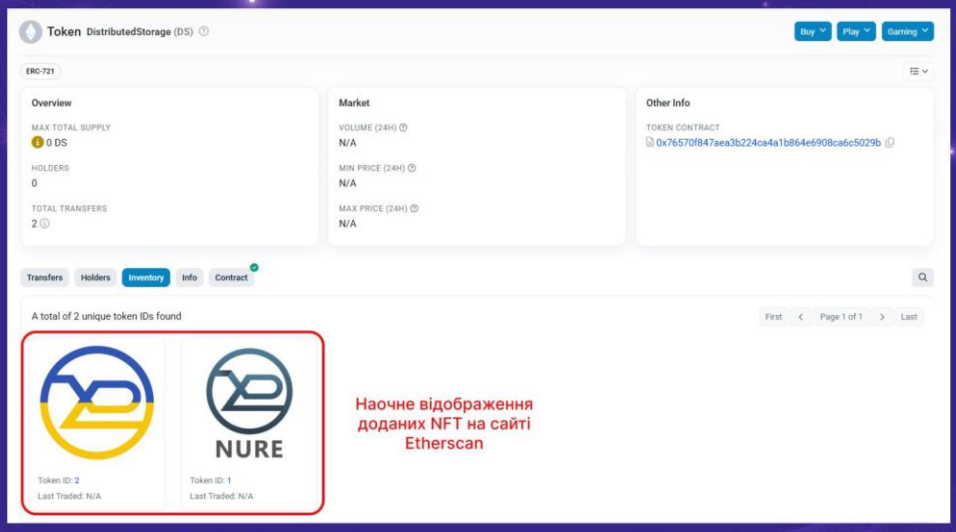


Рисунок 11 – Відображення створених NFT на сторінці контракту токена через Etherscan

Висновки

14

Досліджено сучасні загрози інформаційній безпеці в цифровому середовищі, що включають ризики крадіжки даних, залежність від централізованих систем і можливість втрати інформації через зловмисні дії чи технічні несправності. На базі цього обґрунтовано необхідність використання децентралізованих технологій, як альтернативного варіанту забезпечення прозорості та безпеки даних.

Виявлено ключові переваги блокчейну Ethereum: незмінність, масштабованість і стабільність, що робить його оптимальним середовищем для розробки децентралізованих застосунків.

Розроблено модель децентралізованого зберігання даних, що відповідає вимогам прозорості, безпеки та ефективності роботи з цифровими активами.

На базі моделі розроблено децентралізований застосунок у мережі Ethereum для взаємодії зі смарт-контрактами та з інтеграцію окремих бібліотек для зберігання даних.

Було протестовано створену систему і підтверджено її працездатність. Тестування показало доступність взаємодії між користувачем, смарт-контрактом та блокчейн-мережею.

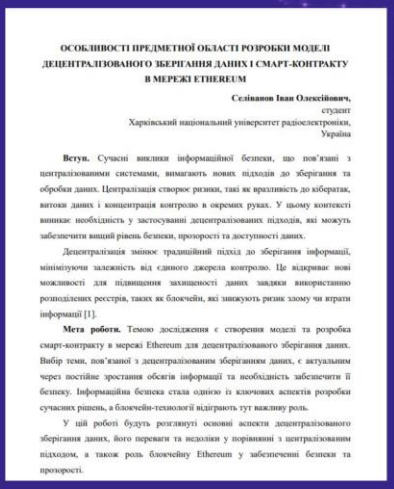
Апробація результатів кваліфікаційної роботи

15

EUROPEAN CONGRESS OF SCIENTIFIC DISCOVERY

Proceedings of I International Scientific and Practical Conference
Madrid, Spain
29-31 December 2024

Madrid, Spain
2024
2



ОСОБЛИВОСТІ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗРОБКИ МОДЕЛІ ДЕЦЕНТРАЛІЗОВАНОГО ЗБЕРІГАННЯ ДАНИХ І СМАРТ-КОНТРАКТУ В МЕРЕЖІ ETHEREUM

Селіванов Іван Олексійович,
студент
Харківський національний університет радіоелектроніки,
Україна

Вступ. Сучасні виклики інформаційної безпеки, що пов'язані з централізованими системами, вимагають нових підходів до зберігання та обробки даних. Централізація створює ризики, такі як вразливість до кібератак, витоки даних і концентрація контролю в окремих руках. У цьому контексті виникає необхідність у застосуванні децентралізованих підходів, які можуть забезпечити вищий рівень безпеки, прозорості та доступності даних.

Децентралізація змінює традиційний підхід до зберігання інформації, мінімізуючи залежність від єдиного джерела контролю. Це відкриває нові можливості для підвищення захищеності даних завдяки використанню розподілених реєстрів, таких як блокчейн, які знижують ризик злому чи втрати інформації [1].

Мета роботи. Темою дослідження є створення моделі та розробка смарт-контракту в мережі Ethereum для децентралізованого зберігання даних. Вибір теми, пов'язаної з децентралізованим зберіганням даних, є актуальним через постійне зростання обсягів інформації та необхідність забезпечити її безпеку. Інформаційна безпека стала однією із ключових аспектів розробки сучасних рішень, а блокчейн-технології відіграють тут важливу роль.

У цій роботі будуть розглянуті основні аспекти децентралізованого зберігання даних, його переваги та недоліки у порівнянні з централізованим підходом, а також роль блокчейну Ethereum у забезпеченні безпеки та прозорості.

Посилання на публікацію:
https://sci-conf.com.ua/wp-content/uploads/2024/12/EUROPEAN-CONGRESS-OF-SCIENTIFIC-DISCOVERY-29-31.12.2024.pdf

ДОДАТОК Б
ПРОГРАМНИЙ КОД ЗАСТОСУНКУ

Б.1 Файли застосунку, що розташовані у папці contracts

Лістинг Б.1 – Код з файлу за шляхом contracts\ DistributedStorage.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.2;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import
"@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorag
e.sol";
import
"@openzeppelin/contracts/token/ERC721/extensions/ERC721Burnable.
sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/Counters.sol";


contract DistributedStorage is ERC721, ERC721URIStorage,
ERC721Burnable, Ownable {
    using Counters for Counters.Counter;
    Counters.Counter private _tokenIdCounter;
    constructor() ERC721("DistributedStorage", "DS") {}
    function safeMint(address to, string memory nftTokenURI)
public onlyOwner {
        _safeMint(to, _tokenIdCounter.current());
        _setTokenURI(_tokenIdCounter.current(), nftTokenURI);
        _tokenIdCounter.increment();
    }

    function _burn(
        uint256 tokenId
    ) internal override(ERC721, ERC721URIStorage) {
        super._burn(tokenId);
    }

    function tokenURI(
        uint256 tokenId
    ) public view override(ERC721, ERC721URIStorage) returns
(string memory) {
        return super.tokenURI(tokenId);
    }
```

```
    function supportsInterface(
        bytes4 interfaceId
    ) public view override(ERC721, ERC721URIStorage) returns
(bool) {
        return super.supportsInterface(interfaceId);
    }

    function currentCounter() public view returns (uint256) {
        return _tokenIdCounter.current();
    }

    function freeMint(address to, string memory nftTokenURI)
public {
        _safeMint(to, _tokenIdCounter.current());
        _setTokenURI(_tokenIdCounter.current(), nftTokenURI);
        _tokenIdCounter.increment();
    }
}
```

Б.2 Файли застосунку, що розташовані у папці scripts

Лістинг Б.2 – Код з файлу за шляхом scripts\deploy.ts

```
import "@nomiclabs/hardhat-ethers";
import hre from "hardhat";

async function main() {
  const DistributedStorage = await
hre.ethers.getContractFactory("DistributedStorage");
  const logo = await DistributedStorage.deploy();

  await logo.deployed();
  console.log("DistributedStorage deployed to:", logo.address);
  console.log("owner", await logo.owner());
}

main()
  .then(() => process.exit(0))
  .catch((error) => {
    console.error(error);
    process.exit(1);
  });
```

Лістинг Б.3 – Код з файлу за шляхом scripts\mint.ts

```
import "@nomiclabs/hardhat-ethers";
import { config } from "dotenv";
```

```
import { readdirSync, readFileSync } from "fs";
import hre from "hardhat";
import { NFTStorage, File } from "nft.storage";
import PQueue from "p-queue";
import { join } from "path";

import { DistributedStorage } from "../src/types";

config();

const apiKey = process.env.NFT_STORAGE_KEY || "";
const client = new NFTStorage({ token: apiKey });

async function uploadFile({ file, name, description }: { file:
File; name: string; description: string }) {
  console.log("Uploading file to nft storage", { file, name,
description });
  const metadata = await client.store({
    name,
    description,
    image: file,
  });
  return metadata;
}

const CONTRACT_ADDRESS = process.env.VITE_NFT_DEPLOYED_ADDRESS
|| "";
const OWNER = process.env.OWNER_ADDRESS || "";

async function main() {
  const DistributedStorage = await
hre.ethers.getContractFactory("DistributedStorage");
  const logo = (await
DistributedStorage.attach(CONTRACT_ADDRESS)) as
DistributedStorage;

  console.log("DistributedStorage deployed to:", logo.address);
  console.log("Name", await logo.name());
  console.log("Symbol", await logo.symbol());

  const queue = new PQueue({ concurrency: 1 });
  const folders = readdirSync(join(__dirname, "assets"));
  for (const folder of folders) {
    const files = readdirSync(join(__dirname, "assets",
folder));
    console.log("Folder", folder);
    console.log("Files", files);
    for (const file of files) {
      const filePath = join(__dirname, "assets", folder, file);
      await queue.add(() =>
        mintNft({
          logo,
          filePath,
```

```
          name: file,
          description: join(folder, file),
        })
      );
    }
  }
}

async function mintNft({
  logo,
  filePath,
  name = "",
  description = "",
}: {
  logo: DistributedStorage;
  filePath: string;
  name?: string;
  description?: string;
}) {
  console.log("Minting NFT", { filePath, name, description });
  const file = readFileSync(filePath);

  const metaData = await uploadFile({
    file: new File([file.buffer], name, {
      type: "image/png", // image/png
    }),
    name,
    description,
  });

  console.log("Uploaded file to nft storage", metaData);
  const mintTx = await logo.safeMint(OWNER, metaData?.url);
  const tx = await mintTx.wait();
  console.log("Minted NFT", tx.blockHash);
}

main()
  .then(() => process.exit(0))
  .catch((error) => {
    console.error(error);
    process.exit(1);
  });
```

Б.3 Файли застосунку, що розташовані у папці src

Лістинг Б.4 – Код з файлу за шляхом src\App.tsx

```
/* eslint-disable @typescript-eslint/no-unsafe-argument */
import { useWeb3React, Web3ReactProvider } from "@web3-
```

```
react/core";
import { BigNumber, ethers, getDefaultProvider } from "ethers";
import React, { useEffect, useState } from "react";
import toast, { Toaster } from "react-hot-toast";
import { NftProvider } from "use-nft";

import DistributedStorageArtifacts from
"./artifacts/contracts/DistributedStorage.sol/DistributedStorage
.json";
import AddItemModal from "./components/AddItemModal";
import Demo from "./components/Demo";
import { getLibrary } from "./components/Demo";
import { Nft } from "./components/Nft";
import { Pagination } from "./components/Pagination";
import logger from "./logger";
import { networkName, CHAIN_ID } from "./networkName";
import { type DistributedStorage } from "./types";

export const CONTRACT_DEPLOYED_ADDRESS =
import.meta.env.VITE_NFT_DEPLOYED_ADDRESS;

declare global {
  interface Window {
    ethereum: ethers.providers.ExternalProvider;
  }
}

async function requestAccount() {
  if (window.ethereum?.request) return window.ethereum.request({
method: "eth_requestAccounts" });

  throw new Error("Missing install Metamask. Please access
https://metamask.io/ to install extension on your browser");
}

const API_URL = import.meta.env.VITE_API_URL;

function NFTApp() {
  const { library, account } = useWeb3React();
  const [total, setTotal] = useState(0);
  const [page, setPage] = useState(1);
  const [isOpen, setIsOpen] = useState(false);
  const limit = 12;

  const fetchTotal = () => {
    logger.warn("fetchTotal");
    const provider = library || new
ethers.providers.Web3Provider(window.ethereum);
    const contract = new ethers.Contract(
      CONTRACT_DEPLOYED_ADDRESS,
      DistributedStorageArtifacts.abi,
      provider,
    ) as DistributedStorage;
```

```
    contract
      .currentCounter()
      .then((result) => {
        setTotal(BigNumber.from(result).toNumber());
      })
      .catch(logger.error);
  };

  useEffect(() => {
    try {
      fetchTotal();
    } catch (error) {
      logger.error(error);
    }
  }, [library]);

  const onMintNftTokeN = async (tokenUrl: string) => {
    const provider = library || new
ethers.providers.Web3Provider(window.ethereum);
    const signer = provider.getSigner();
    const contract = new ethers.Contract(
      CONTRACT_DEPLOYED_ADDRESS,
      DistributedStorageArtifacts.abi,
      signer,
    ) as DistributedStorage;
    try {
      if (!account) {
        await requestAccount();
        return;
      }

      const transaction = await contract.freeMint(account,
tokenUrl);
      toast
        .promise(transaction.wait(), {
          loading: `Transaction submitted. Wait for
confirmation...`,
          success: <b>Transaction confirmed!</b>,
          error: <b>Transaction failed!.</b>,
        })
        .catch(logger.error);

      transaction
        .wait()
        .then(() => {
          fetchTotal();
        })
        .catch(logger.error);
    } catch (error) {
      logger.error(error);
    }
  };
```

```
  const onUpload = async ({ name, description, file }: { name:
string; description: string; file: File }) => {
    toast("Uploading... Please wait for a moment!");
    const formdata = new FormData();
    formdata.append("name", name);
    formdata.append("description", description);
    if (file) formdata.append("file", file);

    setIsOpen(false);
    const response = await fetch(API_URL ? API_URL +
"/nft/upload" : "/api/nft/upload", {
      method: "POST",
      body: formdata,
    });
    const result = await response.json();
    if (result.error) {
      toast.error(result.message);
    }

    return result;
  };

  return (
    <div>
      <h2 className="my-4 text-4xl font-bold">
        NFT Items
        <button
          type="button"
          className="ml-2 hover:text-green-200 btn btn-lg
hover:btn-active"
          onClick={() => {
            setIsOpen(true);
          }}
        >
          <svg
            xmlns="http://www.w3.org/2000/svg"
            className="w-6 h-6"
            fill="none"
            viewBox="0 0 24 24"
            stroke="currentColor"
          >
            <path
              strokeLinecap="round"
              strokeLinejoin="round"
              strokeWidth={2}
              d="M12 9v3m0 0v3m0-3h3m-3 0H9m12 0a9 9 0 11-18 0 9
9 0 0118 0z"
            />
          </svg>
          Add NFT
        </button>
      </h2>
```

```
      <AddItemModal
        isOpen={isOpen}
        onAdd={(formData, files) => {
          onUpload({
            name: formData.name,
            description: formData.description,
            file: files[0],
          })
            .then(async (result) => {
              if (result.url) {
                toast.success(`Uploaded ${result.url}`);
                return onMintNftTokeN(result.url);
              }
            })
            .catch(logger.error);
        }}
        onClose={() => {
          setIsOpen(false);
        }}
      />

      <div className="container grid gap-2 sm:grid-cols-1
md:grid-cols-2 xl:grid-cols-3">
        {Array.from(Array(limit).keys())
          .filter((i) => i + 1 + (page - 1) * limit < total)
          .map((i) => (
            <Nft key={i} tokenId={String(i + 1 + (page - 1) *
limit)} />
          ))}
      </div>
      <Pagination currentPage={page} totalPage={Math.ceil(total
/ limit)} onChange={setPage} />
    </div>
  );
}

const ethersConfig = {
  provider: getDefaultProvider(networkName[Number(CHAIN_ID)] ||
"homestead"),
};

function App() {
  return (
    <Web3ReactProvider getLibrary={getLibrary}>
      <Toaster position="top-right" />
      <NftProvider fetcher={["ethers", ethersConfig]}>
        <div className="container mx-auto">
          <Demo />
          <NFTApp />
          <footer className="p-10 footer bg-base-200 text-base-
content">
            <div>
              <span className="footer-title">1-click
```

```
Deployment</span>
              <a
                className="pl-2"

href="https://vercel.com/new/git/external?repository-
url=https://github.com/jellydn/nft-app/"
              >
                <img src="https://vercel.com/button"
alt="Deploy" />
              </a>
            </div>
          </footer>
        </div>
      </NftProvider>
    </Web3ReactProvider>
  );
}

export default App;
```

Лістинг Б.5 – Код з файлу за шляхом src\main.ts

```
import React from "react";
import ReactDOM from "react-dom/client";

import App from "./App";
import "./index.css";

ReactDOM.createRoot(document.getElementById("root")!).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Лістинг Б.6 – Код з файлу за шляхом src\main.ts

```
import React from "react";
import ReactDOM from "react-dom/client";

import App from "./App";
import "./index.css";

ReactDOM.createRoot(document.getElementById("root")!).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Лістинг Б.7 – Код з файлу за шляхом src\types\contracts\DistributedStorage.ts

```
import type {
  BaseContract,
  BigNumber,
  BigNumberish,
  BytesLike,
  CallOverrides,
  ContractTransaction,
  Overrides,
  PopulatedTransaction,
  Signer,
  utils,
} from "ethers";
import type {
  FunctionFragment,
  Result,
  EventFragment,
} from "@ethersproject/abi";
import type { Listener, Provider } from
"@ethersproject/providers";
import type {
  TypedEventFilter,
  TypedEvent,
  TypedListener,
  OnEvent,
} from "../common";

export interface DistributedStorageInterface extends
utils.Interface {
  functions: {
    "approve(address,uint256)": FunctionFragment;
    "balanceOf(address)": FunctionFragment;
    "burn(uint256)": FunctionFragment;
    "currentCounter()": FunctionFragment;
```

```
        "freeMint(address,string)": FunctionFragment;
        "getApproved(uint256)": FunctionFragment;
        "isApprovedForAll(address,address)": FunctionFragment;
        "name()": FunctionFragment;
        "owner()": FunctionFragment;
        "ownerOf(uint256)": FunctionFragment;
        "renounceOwnership()": FunctionFragment;
        "safeMint(address,string)": FunctionFragment;
                    "safeTransferFrom(address,address,uint256)":
FunctionFragment;
               "safeTransferFrom(address,address,uint256,bytes)":
FunctionFragment;
        "setApprovalForAll(address,bool)": FunctionFragment;
        "supportsInterface(bytes4)": FunctionFragment;
        "symbol()": FunctionFragment;
        "tokenURI(uint256)": FunctionFragment;
                        "transferFrom(address,address,uint256)":
FunctionFragment;
        "transferOwnership(address)": FunctionFragment;
      };

      getFunction(
        nameOrSignatureOrTopic:
          | "approve"
          | "balanceOf"
          | "burn"
          | "currentCounter"
          | "freeMint"
          | "getApproved"
          | "isApprovedForAll"
          | "name"
          | "owner"
          | "ownerOf"
          | "renounceOwnership"
          | "safeMint"
          | "safeTransferFrom(address,address,uint256)"
```

```
            | "safeTransferFrom(address,address,uint256,bytes)"
            | "setApprovalForAll"
            | "supportsInterface"
            | "symbol"
            | "tokenURI"
            | "transferFrom"
            | "transferOwnership"
        ): FunctionFragment;

        encodeFunctionData(
          functionFragment: "approve",
          values: [string, BigNumberish]
        ): string;
        encodeFunctionData(functionFragment: "balanceOf", values:
[string]): string;
          encodeFunctionData(functionFragment:   "burn",   values:
[BigNumberish]): string;
        encodeFunctionData(
          functionFragment: "currentCounter",
          values?: undefined
        ): string;
        encodeFunctionData(
          functionFragment: "freeMint",
          values: [string, string]
        ): string;
        encodeFunctionData(
          functionFragment: "getApproved",
          values: [BigNumberish]
        ): string;
        encodeFunctionData(
          functionFragment: "isApprovedForAll",
          values: [string, string]
        ): string;
          encodeFunctionData(functionFragment:   "name",   values?:
undefined): string;
          encodeFunctionData(functionFragment:   "owner",   values?:
```

```
undefined): string;
       encodeFunctionData(
         functionFragment: "ownerOf",
         values: [BigNumberish]
       ): string;
       encodeFunctionData(
         functionFragment: "renounceOwnership",
         values?: undefined
       ): string;
       encodeFunctionData(
         functionFragment: "safeMint",
         values: [string, string]
       ): string;
       encodeFunctionData(
                                                  functionFragment:
"safeTransferFrom(address,address,uint256)",
         values: [string, string, BigNumberish]
       ): string;
       encodeFunctionData(
                                                  functionFragment:
"safeTransferFrom(address,address,uint256,bytes)",
         values: [string, string, BigNumberish, BytesLike]
       ): string;
       encodeFunctionData(
         functionFragment: "setApprovalForAll",
         values: [string, boolean]
       ): string;
       encodeFunctionData(
         functionFragment: "supportsInterface",
         values: [BytesLike]
       ): string;
        encodeFunctionData(functionFragment:  "symbol",  values?:
undefined): string;
       encodeFunctionData(
         functionFragment: "tokenURI",
         values: [BigNumberish]
```

```
    ): string;
    encodeFunctionData(
      functionFragment: "transferFrom",
      values: [string, string, BigNumberish]
    ): string;
    encodeFunctionData(
      functionFragment: "transferOwnership",
      values: [string]
    ): string;

     decodeFunctionResult(functionFragment: "approve", data:
BytesLike): Result;
    decodeFunctionResult(functionFragment: "balanceOf", data:
BytesLike): Result;
      decodeFunctionResult(functionFragment: "burn", data:
BytesLike): Result;
    decodeFunctionResult(
      functionFragment: "currentCounter",
      data: BytesLike
    ): Result;
     decodeFunctionResult(functionFragment: "freeMint", data:
BytesLike): Result;
    decodeFunctionResult(
      functionFragment: "getApproved",
      data: BytesLike
    ): Result;
    decodeFunctionResult(
      functionFragment: "isApprovedForAll",
      data: BytesLike
    ): Result;
      decodeFunctionResult(functionFragment: "name", data:
BytesLike): Result;
     decodeFunctionResult(functionFragment: "owner", data:
BytesLike): Result;
     decodeFunctionResult(functionFragment: "ownerOf", data:
BytesLike): Result;
```

```
        decodeFunctionResult(
          functionFragment: "renounceOwnership",
          data: BytesLike
        ): Result;
         decodeFunctionResult(functionFragment:  "safeMint",  data:
BytesLike): Result;
        decodeFunctionResult(
                                                  functionFragment:
"safeTransferFrom(address,address,uint256)",
          data: BytesLike
        ): Result;
        decodeFunctionResult(
                                                  functionFragment:
"safeTransferFrom(address,address,uint256,bytes)",
          data: BytesLike
        ): Result;
        decodeFunctionResult(
          functionFragment: "setApprovalForAll",
          data: BytesLike
        ): Result;
        decodeFunctionResult(
          functionFragment: "supportsInterface",
          data: BytesLike
        ): Result;
         decodeFunctionResult(functionFragment:  "symbol",  data:
BytesLike): Result;
         decodeFunctionResult(functionFragment:  "tokenURI",  data:
BytesLike): Result;
        decodeFunctionResult(
          functionFragment: "transferFrom",
          data: BytesLike
        ): Result;
        decodeFunctionResult(
          functionFragment: "transferOwnership",
          data: BytesLike
        ): Result;
```

```
      events: {
        "Approval(address,address,uint256)": EventFragment;
        "ApprovalForAll(address,address,bool)": EventFragment;
        "BatchMetadataUpdate(uint256,uint256)": EventFragment;
        "MetadataUpdate(uint256)": EventFragment;
        "OwnershipTransferred(address,address)": EventFragment;
        "Transfer(address,address,uint256)": EventFragment;
      };

            getEvent(nameOrSignatureOrTopic:       "Approval"):
EventFragment;
          getEvent(nameOrSignatureOrTopic:    "ApprovalForAll"):
EventFragment;
        getEvent(nameOrSignatureOrTopic:  "BatchMetadataUpdate"):
EventFragment;
          getEvent(nameOrSignatureOrTopic:    "MetadataUpdate"):
EventFragment;
       getEvent(nameOrSignatureOrTopic: "OwnershipTransferred"):
EventFragment;
            getEvent(nameOrSignatureOrTopic:       "Transfer"):
EventFragment;
    }
```

Лістинг Б.8 – Код з файлу за шляхом src\types\factories\contracts\DistributedStorage__factory.ts

```
import { Signer, utils, Contract, ContractFactory, Overrides }
from "ethers";
import type { Provider, TransactionRequest } from
"@ethersproject/providers";
import type {
  DistributedStorage,
  DistributedStorageInterface,
} from "../../contracts/DistributedStorage";
```

```
const _abi = [
  {
    inputs: [],
    stateMutability: "nonpayable",
    type: "constructor",
  },
  {
    anonymous: false,
    inputs: [
      {
        indexed: true,
        internalType: "address",
        name: "owner",
        type: "address",
      },
      {
        indexed: true,
        internalType: "address",
        name: "approved",
        type: "address",
      },
      {
        indexed: true,
        internalType: "uint256",
        name: "tokenId",
        type: "uint256",
      },
    ],
    name: "Approval",
    type: "event",
  },
...
{
    inputs: [
      {
```

```
        internalType: "address",
        name: "newOwner",
        type: "address",
      },
    ],
    name: "transferOwnership",
    outputs: [],
    stateMutability: "nonpayable",
    type: "function",
  },
] as const;
const _bytecode =
  "0x60806040...033";

type DistributedStorageConstructorParams =
  | [signer?: Signer]
  | ConstructorParameters<typeof ContractFactory>;

const isSuperArgs = (
  xs: DistributedStorageConstructorParams
): xs is ConstructorParameters<typeof ContractFactory> =>
xs.length > 1;

export class DistributedStorage__factory extends ContractFactory
{
  constructor(...args: DistributedStorageConstructorParams) {
    if (isSuperArgs(args)) {
      super(...args);
    } else {
      super(_abi, _bytecode, args[0]);
    }
  }

  override deploy(
    overrides?: Overrides & { from?: string }
  ): Promise<DistributedStorage> {
```

```
            return     super.deploy(overrides     ||     {})     as
Promise<DistributedStorage>;
  }
  override getDeployTransaction(
    overrides?: Overrides & { from?: string }
  ): TransactionRequest {
    return super.getDeployTransaction(overrides || {});
  }
  override attach(address: string): DistributedStorage {
    return super.attach(address) as DistributedStorage;
  }
   override  connect(signer:  Signer):  DistributedStorage__factory
{
    return super.connect(signer) as DistributedStorage__factory;
  }

  static readonly bytecode = _bytecode;
  static readonly abi = _abi;
  static createInterface(): DistributedStorageInterface {
              return      new      utils.Interface(_abi)      as
DistributedStorageInterface;
  }
  static connect(
    address: string,
    signerOrProvider: Signer | Provider
  ): DistributedStorage {
      return  new  Contract(address,  _abi,  signerOrProvider)  as
DistributedStorage;
  }
}
```

Лістинг Б.9– Код з файлу за шляхом src\main.ts

```
import React from "react";
import ReactDOM from "react-dom/client";
```

```
import App from "./App";
import "./index.css";

ReactDOM.createRoot(document.getElementById("root")!).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Лістинг Б.10 – Код з файлу за шляхом src\@openzeppelin\contracts\token\ERC721\extensions\ERC721URIStorage.sol

```
pragma solidity ^0.8.0;

import "../ERC721.sol";
import "../../../interfaces/IERC4906.sol";

abstract contract ERC721URIStorage is IERC4906, ERC721 {
    using Strings for uint256;

    // Optional mapping for token URIs
    mapping(uint256 => string) private _tokenURIs;

    function supportsInterface(bytes4 interfaceId) public view
virtual override(ERC721, IERC165) returns (bool) {
        return interfaceId == bytes4(0x49064906) ||
super.supportsInterface(interfaceId);
    }

    function tokenURI(uint256 tokenId) public view virtual
override returns (string memory) {
        _requireMinted(tokenId);

        string memory _tokenURI = _tokenURIs[tokenId];
        string memory base = _baseURI();

        // If there is no base URI, return the token URI.
        if (bytes(base).length == 0) {
            return _tokenURI;
        }
        // If both are set, concatenate the baseURI and tokenURI
(via abi.encodePacked).
        if (bytes(_tokenURI).length > 0) {
            return string(abi.encodePacked(base, _tokenURI));
        }

        return super.tokenURI(tokenId);
    }

    function _setTokenURI(uint256 tokenId, string memory
```

```
_tokenURI) internal virtual {
        require(_exists(tokenId), "ERC721URIStorage: URI set of
nonexistent token");
        _tokenURIs[tokenId] = _tokenURI;

        emit MetadataUpdate(tokenId);
    }

    function _burn(uint256 tokenId) internal virtual override {
        super._burn(tokenId);

        if (bytes(_tokenURIs[tokenId]).length != 0) {
            delete _tokenURIs[tokenId];
        }
    }
}
```

Лістинг Б.11 – Код з файлу за шляхом src\components\AddItemModal.tsx

```
import React, { useEffect, useState } from "react";
import { useDropzone } from "react-dropzone";
import { useForm } from "react-hook-form";

type FormValues = {
  name: string;
  description: string;
};

type AddItemModalProps = {
  readonly isOpen: boolean;
  readonly onAdd: (formData: FormValues, files: File[]) => void;
  readonly onClose: () => void;
};

function AddItemModal({ isOpen, onAdd, onClose }:
AddItemModalProps) {
  const {
    register,
    handleSubmit,
    control,
    formState: { errors },
    reset,
  } = useForm<FormValues>();

  const [files, setFiles] = useState<
    Array<
      File & {
        preview: string;
      }
    >
  >([]);
```

```
  const { getRootProps, getInputProps } = useDropzone({
    accept: {
      "image/*": [".jpeg", ".png"],
    },
    onDrop(acceptedFiles) {
      setFiles(
        acceptedFiles.map((file) =>
          Object.assign(file, {
            preview: URL.createObjectURL(file),
          }),
        ),
      );
    },
  });

  const thumbs = files.map((file) => (
    <div key={file.name} className="avatar">
      <div className="mb-8 w-auto h-24 rounded-btn">
        <img src={file.preview} alt={file.name} />
      </div>
    </div>
  ));

  useEffect(
    () => () => {
      files.forEach((file: any) => {
        URL.revokeObjectURL(file.preview);
      });
    },
    [files],
  );

  const onSubmit = handleSubmit((data) => {
    onAdd(data, files);
    reset();
    setFiles([]);
  });

  return (
    <div className={isOpen ? "modal modal-open" : "modal"}>
      <form onSubmit={onSubmit}>
        <div className="modal-box">
          <div className="p-10 card bg-base-200">
            <section className="form-control">
              <div {...getRootProps({ className: "container"
})}>
                <input
                  {...getInputProps({
                    className: "input",
                  })}
                />
                <p>Drag & drop some files here, or click to
select files</p>
```

```
                <em>(Only *.jpeg and *.png images will be
accepted)</em>
              </div>
              <aside>{thumbs}</aside>
            </section>
            <div className="form-control">
              <label className="label">
                <span className="label-text">Name</span>
              </label>
              <input
                type="text"
                placeholder="name"
                className={errors.name ? "input input-error" :
"input"}
                {...register("name", {
                  required: true,
                })}
              />
              {errors.name && (
                <label className="label">
                  <span className="label-text-
alt">{errors.name.message ?? "Required"}</span>
                </label>
              )}
            </div>
            <div className="form-control">
              <label className="label">
                <span className="label-text">Description</span>
              </label>
              <textarea
                {...register("description", {
                  required: true,
                })}
                className={
                  errors.description
                    ? "h-24 textarea textarea-bordered textarea-
error"
                    : "h-24 textarea textarea-bordered"
                }
                placeholder="Description"
              />
              {errors.description && (
                <label className="label">
                  <span className="label-text-
alt">{errors.description.message ?? "Required"}</span>
                </label>
              )}
            </div>
          </div>

          <div className="modal-action">
            <button type="submit" className="btn btn-primary">
              Add
```

```
            </button>
            <button type="button" className="btn"
onClick={onClose}>
              Close
            </button>
          </div>
        </div>
      </form>
    </div>
  );
}

export default AddItemModal;
```

Лістинг Б.12 – Код з файлу за шляхом src\components\Nft.tsx

```
import React from "react";
import { useNft } from "use-nft";

import Image from "./Image";

export const CONTRACT_DEPLOYED_ADDRESS =
import.meta.env.VITE_NFT_DEPLOYED_ADDRESS;

export function Nft({ tokenId }: { readonly tokenId: string }) {
  const { loading, error, nft } =
useNft(CONTRACT_DEPLOYED_ADDRESS, tokenId);

  if (loading)
    return (
      <div className="text-center text-gray-200 bg-gray-700
rounded shadow card">
        <h2 className="my-2 text-4xl font-bold card-
title">#{tokenId}</h2>
        <div className="justify-center items-center card-body">
          <div className="text-center btn btn-ghost btn-sm btn-
circle loading">Loading…</div>
        </div>
      </div>
    );

  if (error ?? !nft)
    return (
      <div className="alert alert-error">
        <div className="flex-1">
          <label>{error?.message}</label>
        </div>
      </div>
    );

  return (
```

```
    <div className="text-center text-gray-50 bg-gray-500
rounded-md shadow-md card">
      <h2 className="my-2 text-4xl font-bold card-
title">#{tokenId}</h2>
      <figure className="px-4 pt-4">
        <Image src={nft.image} alt={nft.name} className="mask
mask-sircleu" />
      </figure>
      <div className="card-body">
        <h2 className="card-title">{nft.name}</h2>
        <p>{nft.description}</p>
        <p>Owner: {nft.owner}</p>
        <div className="justify-center card-actions">
          <a target="_blank" href={nft.metadataUrl}
className="btn btn-outline btn-accent" rel="noreferrer">
            More info
          </a>
        </div>
      </div>
    </div>
  );
}
```

Б.4 Файли застосунку, що розташовані у папці server

Лістинг Б.13 – Код з файлу за шляхом server\src\app.ts

```
import AutoLoad, { AutoloadPluginOptions } from
"@fastify/autoload";
import { FastifyPluginAsync } from "fastify";
import { join } from "path";

export type AppOptions = {} & Partial<AutoloadPluginOptions>;

const app: FastifyPluginAsync<AppOptions> = async (fastify,
opts): Promise<void> => {
  void fastify.register(AutoLoad, {
    dir: join(__dirname, "plugins"),
    options: opts,
  });

  void fastify.register(AutoLoad, {
    dir: join(__dirname, "routes"),
    options: opts,
  });
};

export default app;
export { app };
```

Лістинг Б.14 – Код з файлу за шляхом server\src\server.ts

```
import closeWithGrace from "close-with-grace";
import * as dotenv from "dotenv";
import Fastify from "fastify";

dotenv.config();

const app = Fastify({
  logger: true,
});

app.register(import("./app"));

const closeListeners = closeWithGrace({ delay: 500 }, async
function (opts: any) {
  if (opts.err) {
    app.log.error(opts.err);
  }
  await app.close();
});

app.addHook("onClose", async (instance, done) => {
  closeListeners.uninstall();
  done();
});

app.listen({ port: Number(process.env.PORT || 3000), host:
"0.0.0.0" }, (err: any) => {
  if (err) {
    app.log.error(err);
    process.exit(1);
  }
});
```

Лістинг Б.15 – Код з файлу за шляхом server\package.json

```
{
  "name": "nft-api",
  "version": "1.0.0",
  "description": "NFT API",
  "keywords": [],
  "license": "MIT",
  "author": "selivanov",
  "main": "app.ts",
  "directories": {
    "test": "test"
```

```
  },
  "scripts": {
    "build": "tsup",
    "build:ts": "tsup --env.NODE_ENV production",
    "dev": "tsx watch src/server.ts",
    "start": "tsx src/server.ts",
    "test": "npm run build:ts && tsc -p test/tsconfig.test.json
&& cross-env TS_NODE_FILES=true tap --ts test/**/*.test.ts"
  },
  "dependencies": {
    "@fastify/autoload": "5.10.0",
    "@fastify/cors": "9.0.1",
    "@fastify/multipart": "8.3.0",
    "@fastify/sensible": "5.6.0",
    "@sinclair/typebox": "0.33.17",
    "close-with-grace": "2.1.0",
    "fastify": "4.28.1",
    "fastify-cli": "6.3.0",
    "fastify-plugin": "4.5.1",
    "fastify-tsconfig": "2.0.0",
    "nft.storage": "7.2.0",
    "typescript": "5.6.3"
  },
  "devDependencies": {
    "@types/busboy": "1.5.4",
    "@types/node": "20.16.15",
    "@types/tap": "15.0.12",
    "cross-env": "7.0.3",
    "fast-json-stringify": "6.0.0",
    "tap": "21.0.1",
    "tsup": "8.3.0",
    "tsx": "4.19.1"
  },
  "engines": {
    "node": ">=18.12.0"
  },
  "tsup": {
    "entry": [
      "src/**/*.ts"
    ],
    "splitting": false,
    "sourcemap": true,
    "clean": true,
    "format": [
      "cjs"
    ],
    "target": "es2018"
  }
}
```