Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та охоронних систем
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА МЕНЕДЖЕРА ФАЙЛОВОЇ СИСТЕМИ З ВИКОРИСТАННЯМ ВЕБ-ІНТЕРФЕЙСУ НА БАЗІ RASPBERRY PI

DEVELOPMENT OF A FILE SYSTEM MANAGER WITH A WEB INTERFACE BASED ON RASPBERRY PI

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІ-42
Зень Руслан Андрійович

(підпис)

Керівник:
асистент
Кулакевич Олег Русланович

(підпис)

Кваліфікаційну роботу
допущено до захисту
«_____» червня 2026 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та безпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. Т. Терлецький
«__23__» ____12____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Зень Руслану Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Розробка менеджера файлової системи з використанням веб-інтерфейсу на базі Raspberry Pi*

Керівник роботи *асистент Кулакевич Олег Русланович*

затверджені наказом закладу вищої освіти від «20» грудня 2025 року № 536/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 28.05.2026 р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області, різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Аналіз інформації в сфері розробки файлових менеджерів
Вибір та обґрунтування апаратної та програмної складової
Реалізація менеджера файлової системи на базі raspberry pi
Висновки

5. Перелік графічного (ілюстративного) матеріалу:
Структура менеджера файлової системи
Інтерфейс вебзастосунку
Процес налаштування Raspberry Pi
Реалізація основних функцій файлового менеджера
Організація віддаленого доступу до системи

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз проблеми за темою роботи та постановка завдань дослідження	*Кулакевич О. Р., асистент*		
Вибір програмної та апаратної складової для реалізації завдання	*Кулакевич О. Р., асистент*		
Практична реалізація менеджера файлової системи	*Кулакевич О. Р., асистент*		
Нормоконтроль	*Багнюк Н. В., доцент*		
Гарант ОП	*Лавренчук С. В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О. І., ст. викладач*		

7. Дата видачі завдання 23.12.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми, аналіз предметної області та наявних рішень*	до 10.02.2026 р.	
2.	*Аналіз проблеми за темою роботи та постановка завдань дослідження*	до 02.03.2026 р.	
3.	*Вибір програмної та апаратної складової для реалізації завдання*	до 02.04.2026 р.	
4.			
5.	*Практична реалізація менеджера файлової системи та формування додатків*	до 10.04.2026 р.	
6.	*Представлення остаточного варіанту кваліфікаційної роботи керівникові*	до 01.05.2026 р.	
7.	*Нормоконтроль*	до 23.05.2026 р.	
8.	*Інструментальна перевірка на академічний плагіат*	до 26.05.2026 р.	
9.	*Здача кваліфікаційної роботи та всіх супровідних документів на кафедру*	до 28.05.2026 р.	

Здобувач вищої освіти _____ Руслан ЗЕНЬ
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Олег КУЛАКЕВИЧ
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Зень Р. А. Розробка менеджера файлової системи з використанням веб-інтерфейсу на базі Raspberry pi. Рукопис.

Кваліфікаційна робота бакалавра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2026.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел, додатку.

Перший розділ присвячено першому предметній області та обґрунтовано актуальність розробки менеджера файлової системи з використанням вебінтерфейсу. Проведено аналіз існуючих рішень для роботи з файлами через браузер, а також розглянуто особливості використання Raspberry Pi у якості локального серверного середовища.

В другому розділі здійснено вибір та обґрунтування програмних і апаратних засобів розробки. Розлянуто особливості мови програмування PHP та її фреймворку Laravel, обрано бібліотеку для побудови інтерфейсу та вебсервер. Також описано характеристики різних версій Raspberry Pi.

Третій розділ присвячено практичній реалізації системи. Описано процес налаштування Raspberry Pi, встановлення та конфігурацію програмного середовища, розгортання вебзастосунку та реалізацію основних його функцій, продеменстровано організацію віддаленого доступу та наведено можливі напрями подальшого вдосконалення системи.

Ключові слова: файлова система, менеджер файлової системи, Raspberry Pi, Laravel, файлове сховище.

ANNOTATION

Zen R. Development of a file system manager with a web interface based on Raspberry pi. Manuscript.

Qualifying work of a bachelor of EP «Computer Engineering» specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2026.

Qualification work consists of an introduction, three sections, conclusions, a references, and an appendix.

The first section is devoted to the subject area and substantiates the relevance of developing a file system manager using a web interface. An analysis of existing browser-based file management solutions was carried out, and the features of using Raspberry Pi as a local server environment were considered.

The second section presents the selection and justification of software and hardware development tools. The features of the PHP programming language and its Laravel framework are described, a library for building the user interface and a web server were selected. The characteristics of different Raspberry Pi versions are also discussed.

The third section is devoted to the practical implementation of the system. It describes the process of configuring Raspberry Pi, installing and configuring the software environment, deploying the web application, and implementing its main functions. The organization of remote access is demonstrated, and possible directions for further improvement of the system are presented.

Keywords: file system, file system manager, Raspberry Pi, Laravel, file storage.

ЗМІСТ

ВСТУП

У сучасних умовах активного розвитку інформаційних технологій спостерігається постійне зростання обсягів цифрових даних, що зумовлює необхідність їх впорядкованого зберігання та ефективного керування. Особливого значення набувають програмні засоби, які забезпечують зручну взаємодію користувача з файловою системою, дозволяючи виконувати базові операції над файлами та каталогами. Традиційні підходи до керування файлами передбачають використання локальних програм або віддаленого доступу через спеціалізовані протоколи, що не завжди є зручним та універсальним рішенням.

Застосування веб-технологій відкриває можливість створення кросплатформених інструментів, доступ до яких здійснюється за допомогою звичайного браузера без потреби встановлення додаткового програмного забезпечення. У поєднанні з використанням одноплатних комп’ютерів, зокрема Raspberry Pi, це дозволяє створювати компактні, енергоефективні та доступні системи керування файлами. Такий підхід забезпечує можливість організації доступу до файлової системи пристрою та спрощує процес адміністрування даних. Отже, розробка менеджера файлової системи на базі Raspberry Pi із використанням веб-інтерфейсу є актуальним завданням, що поєднує сучасні підходи до веб-розробки та використання вбудованих систем.

Метою роботи є розробка менеджера файлової системи на базі одноплатного комп’ютера Raspberry Pi із використанням веб-інтерфейсу, який забезпечує керування файлами та каталогами в межах веб-застосунку, розгорнутого на даній платформі.

Об’єкт дослідження – процеси організації та функціонування файлових систем, а також принципи взаємодії користувача з ними за допомогою програмних засобів у середовищі обмежених апаратних ресурсів. У межах дослідження розглядаються підходи до реалізації доступу до файлової структури та виконання операцій над її об’єктами із застосуванням серверних технологій.

Предмет дослідження – програмна реалізація менеджера файлової системи, що функціонує на платформі Raspberry Pi та забезпечує керування файлами через веб-інтерфейс. Особлива увага приділяється архітектурі застосунку, способам обробки запитів користувача, взаємодії з файловою системою операційної системи та формуванню зручного інтерфейсу доступу.

Для досягнення поставленої мети у роботі передбачено виконання таких завдань:

– реалізувати механізми виконання основних операцій над файлами та каталогами засобами веб-застосунку;

– розробити користувацький веб-інтерфейс для взаємодії з файловою системою пристрою;

– дослідити особливості використання Raspberry Pi як платформи для розгортання програмних рішень;

– візуалізувати структуру файлової системи та процеси взаємодії користувача з нею;

– спроектувати архітектуру програмного забезпечення з урахуванням ефективності використання ресурсів;

РОЗДІЛ 1
АНАЛІЗ ІНФОРМАЦІЇ В СФЕРІ РОЗРОБКИ ФАЙЛОВИХ МЕНЕДЖЕРІВ

1.1 Поняття файлової системи та файлового менеджера

«Файлова система – порядок, що визначає спосіб організації, зберігання та найменювання даних на носіях інформації в комп’ютерах, а також в іншому електронному обладнанні: цифрових фотоапаратах, мобільних телефонах і т.п. Файлова система визначає формат вмісту та фізичного зберігання інформації, яку прийнято групувати у вигляді файлів. Конкретна файлова система визначає розмір імені файлу (папки), максимальний можливий розмір файлу і розділу, набір атрибутів файлу. Деякі файлові системи надають сервісні можливості, наприклад, розмежування доступу або шифрування файлів» [1].

Структурною основою файлової системи є ієрархічна організація каталогів, що дозволяє впорядковувати файли за функціональними або логічними ознаками. Кожен файл супроводжується метаданими, які містять інформацію про його розмір, тип, час створення та модифікації, а також права доступу. Для відстеження фізичного розташування даних на носії використовуються спеціальні структури керування, такі як таблиці розміщення файлів або inode-таблиці, що забезпечують швидкий доступ до блоків пам’яті та оптимізацію використання дискового простору.

Сучасні файлові системи реалізують механізми підвищення надійності та продуктивності, зокрема журналювання операцій, кешування даних у оперативній пам’яті, перевірку цілісності та відновлення після збоїв. Важливою складовою є система керування правами доступу, яка дозволяє розмежовувати доступ користувачів і процесів до файлів і каталогів, забезпечуючи інформаційну безпеку та захист від несанкціонованих дій.

Залежно від середовища використання, застосовуються різні типи файлових систем (таблиця 1.1). Для персональних комп’ютерів та серверів поширеними є NTFS, ext4, XFS та Btrfs, для змінних носіїв використовуються

FAT32 та exFAT, а для мережевого доступу до даних застосовуються мережеві файлові системи, такі як NFS або SMB. У вбудованих системах і мікрокомп'ютерах, зокрема на платформі Raspberry Pi, найчастіше використовується файлова система ext4, яка поєднує високу продуктивність, підтримку журналювання та стабільність роботи на флеш-носіях.

Таблиця 1.1 – Види файлових систем та сфера їх застосування [2]

Файлова система	Використовується	Найкраще підходить для	Обмеження
FAT32	USB-накопичувачі, застарілі версії Windows	Забезпечення сумісності між різними пристроями	Максимальний розмір файлу – 4 ГБ, відсутня підтримка журналювання
NTFS	Операційні системи Windows	Роботи з великими файлами та розширеної системи прав доступу	Відсутня повноцінна нативна підтримка запису в macOS
EXT4	Операційні системи сімейства Linux	Висока швидкодія, надійність та підтримка журналювання	Обмежена підтримка в середовищі Windows
exFAT	USB-накопичувачі, SD-карти	Передавання великих файлів між різними операційними системами	Менш надійна порівняно з NTFS та EXT4

Файлова система є ключовим елементом інформаційної інфраструктури будь-якої обчислювальної системи, оскільки саме вона забезпечує структуроване зберігання даних, ефективний доступ до них і захист інформації, що є необхідною передумовою для реалізації програмних застосунків, зокрема веб-орієнтованих менеджерів файлових систем.

«Файловий менеджер – комп'ютерна програма, що надає інтерфейс користувача для роботи з файловою системою та файлами. Дозволяє виконувати найчастіші операції з файлами: створення, відкриття, програвання, запуск, перегляд, редагування, переміщення, перейменування, копіювання, вилучення,

зміну атрибутів та властивостей, пошук файлів та призначення прав. Існує два види файлових менеджерів –навігаційні та ортодоксальні. Основна їх відмінність одне від одного – у ортодоксальних є дві панелі, реалізовано відповідну модель роботи» [3].

Залежно від середовища виконання та функціонального призначення, файлові менеджери можуть мати різні форми реалізації. Класичні десктопні файлові менеджери інтегровані в операційні системи та працюють як локальні клієнтські застосунки, забезпечуючи доступ до файлової системи конкретного пристрою. Поряд із цим існують веб-орієнтовані файлові менеджери, які реалізуються у вигляді серверних застосунків і надають доступ до файлової системи через браузер, що особливо актуально для адміністрування серверів, хмарних середовищ і вбудованих платформ.

Архітектурно файлові менеджери можуть будуватися за клієнт-серверною моделлю (рисунок 1.1), де клієнтська частина відповідає за візуалізацію інтерфейсу та взаємодію з користувачем, а серверна частина виконує операції з файловою системою, обробляє запити та забезпечує контроль доступу. У веб-застосунках така взаємодія реалізується через HTTP-протокол із використанням серверних фреймворків, що дозволяє створювати масштабовані та модульні системи управління файлами.

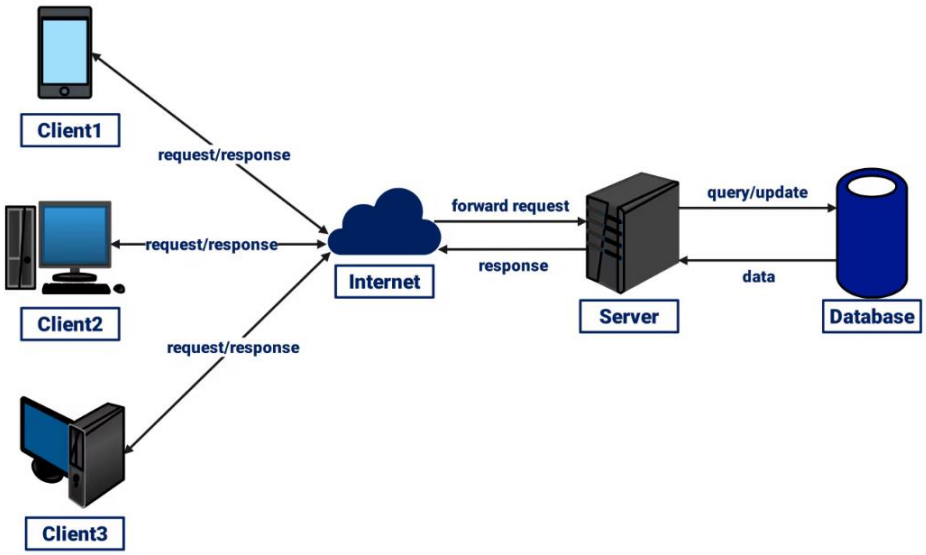

Рисунок 1.1 – принцип роботи клієнт клієнт-серверної моделі [4]

Функціональні можливості сучасних файлових менеджерів значно розширилися порівняно з базовими операціями. До них належать пошук і фільтрація файлів, робота з архівами, перегляд і редагування текстових файлів, керування правами доступу, синхронізація даних, інтеграція з мережевими сховищами та хмарними сервісами. У веб-реалізаціях додатково застосовуються механізми автентифікації та авторизації користувачів, що дозволяє розмежувати доступ до ресурсів і забезпечити безпеку даних. Сфери застосування файлових менеджерів охоплюють широкий спектр інформаційних систем і обумовлені потребою впорядкованого зберігання, обробки та доступу до даних.

У сфері персональних обчислювальних систем файлові менеджери використовуються для повсякденної роботи з даними, зокрема організації документів, мультимедійних файлів і програмних ресурсів. Вони спрощують виконання типових операцій, таких як копіювання, переміщення, перейменування та видалення файлів, що є необхідним елементом ефективної роботи користувача з операційною системою.

В корпоративному середовищі файлові менеджери відіграють важливу роль у керуванні спільними сховищами даних. Вони застосовуються для централізованого доступу до файлів, розмежування прав користувачів, резервного копіювання та архівації інформації.

У сфері веб-розробки та адміністрування серверів файлові менеджери використовуються як інструмент керування структурою веб-ресурсів, конфігураційними файлами та даними користувачів. Вони забезпечують швидкий доступ до файлової системи сервера, спрощують процес розгортання веб-застосунків, оновлення контенту та обслуговування хостингових платформ.

Освітня та наукова діяльність також є важливою сферою застосування файлових менеджерів. У навчальних проєктах вони використовуються для демонстрації принципів роботи файлових систем, організації даних та реалізації клієнт-серверної взаємодії. У наукових дослідженнях файлові менеджери полегшують зберігання та структурування експериментальних даних, що сприяє відтворюваності та систематизації результатів.

Окреме місце займає використання файлових менеджерів у вбудованих та спеціалізованих системах, зокрема на одноплатних комп’ютерах. У таких умовах файлові менеджери виконують функцію інтерфейсу доступу до локальних або мережевих сховищ, що дозволяє реалізовувати файлові сервери, медіацентри та навчальні стенди з обмеженими апаратними ресурсами. Це робить файлові менеджери універсальним компонентом сучасних інформаційних систем, придатним як для прикладних, так і для навчально-дослідницьких завдань.

1.2 Огляд існуючих рішень в сфері менеджерів файлової системи

У сучасних обчислювальних системах файлові менеджери є невід’ємним інструментом для організації, перегляду та керування даними. Існує значна кількість програмних рішень, які відрізняються за функціональністю, архітектурою та сферою застосування, що зумовлено різноманітністю платформ і вимог користувачів.

Аналіз існуючих рішень у сфері менеджерів файлової системи свідчить про значну різноманітність підходів до організації доступу та управління даними. Десктопні файлові менеджери характеризуються високою швидкодією та глибокою інтеграцією з операційною системою, що забезпечує повний контроль над файловими ресурсами. Однак такі рішення прив’язані до конкретного пристрою і не забезпечують зручного віддаленого доступу, що обмежує їх застосування в умовах розподілених систем та вбудованих платформ.

Традиційні файлові менеджери для настільних операційних систем реалізуються у вигляді десктопних застосунків і інтегруються з графічним середовищем операційної системи. До таких рішень належать провідники файлів, що постачаються разом з операційними системами Windows, Linux та macOS, а також спеціалізовані менеджери з розширеним функціоналом. Вони забезпечують ієрархічну навігацію по каталогах, підтримку операцій копіювання, переміщення, перейменування та видалення файлів, а також інтеграцію з системними службами безпеки та керування правами доступу.

З розвитком веб-технологій набули поширення веб-орієнтовані файлові менеджери, що працюють через браузер і використовуються для адміністрування серверів, хостинг-платформ і корпоративних інформаційних систем. Подібні системи функціонують за клієнт-серверною моделлю, де користувацький інтерфейс реалізовано у вигляді веб-інтерфейсу, а операції з файлами виконуються серверною частиною через API файлової системи. Веб-файлові менеджери забезпечують віддалений доступ до даних, багатокористувацьку роботу, а також інтеграцію з механізмами автентифікації та авторизації. Водночас більшість таких рішень орієнтовані на потужні серверні платформи та потребують значних обчислювальних ресурсів, що робить їх менш придатними для використання на малопотужних пристроях, таких як Raspberry Pi. Крім того, існуючі веб-файлові менеджери часто мають складну архітектуру, використовують розширені залежності та бази даних, що ускладнює розгортання та супровід системи.

Серед найбільш відомих рішень на сучасному ринку програмного забезпечення можна виділити Total Commander та Windows File Explorer для платформи Windows, а також Dolphin і Nautilus, що використовуються в середовищі Linux.

Окрему групу становлять веб-орієнтовані файлові менеджери, зокрема elFinder, Tiny File Manager та FileRun, які забезпечують віддалений доступ до файлової системи через браузер.

Проведений аналіз показує (таблиця 1.2), що сучасні файлові менеджери відрізняються між собою за принципом роботи, набором функцій та способом організації доступу до даних. Частина рішень орієнтована на локальне використання, тоді як веборієнтовані системи забезпечують керування файлами через браузер без необхідності встановлення додаткового програмного забезпечення на клієнтському пристрої.

Таблция 1.2 – порівняння найпопулярніших менеджерів файлової системи

Назва файлового менеджера	Тип рішення	Платформа	Основні можливості	Підтримка веб-доступу
Total Commander	Десктопний	Windows	Двопанельний інтерфейс, архівація, плагіни, FTP	Ні
Windows File Explorer	Десктопний	Windows	Базові операції з файлами, інтеграція з ОС	Ні
Dolphin	Десктопний	Linux	Вкладки, панелі, робота з мережевими ресурсами	Ні
Nautilus	Десктопний	Linux	Інтеграція з GNOME, базові операції з файлами	Ні
elFinder	Веб	Кросплатформний	Багатокористувацький доступ, архівація, перегляд медіа	Так
Tiny File Manager	Веб	Кросплатформний	Прості операції з файлами через браузер	Так

Крім того, у корпоративному сегменті поширеними є системи класу NAS, такі як Synology DSM і TrueNAS, що поєднують функції зберігання даних і керування файлами через веб-інтерфейс.

«NAS – це централізована система зберігання даних, що містить один або кілька накопичувачів, і підключена до локальної мережі через Ethernet або, частіше, Wi-Fi. Як частина домашньої або офісної мережі, NAS забезпечує доступ до даних із різних пристроїв, таких, як ноутбуки, телефони, планшети, Smart-телевізори та навіть ігрові консолі» [5].

Системи мережевого зберігання даних типу NAS надають розвинені засоби управління файловими ресурсами та користувачами, але такі системи орієнтовані на спеціалізоване обладнання та мають складну архітектуру, що ускладнює їх адаптацію для невеликих компаній або експериментальних проєктів на базі Raspberry Pi.

Таким чином, існуючі програмні засоби не повністю задовольняють вимоги до модульного та розширюваного менеджера файлової системи, призначеного для роботи на вбудованій платформі з веб-інтерфейсом. Це зумовлює необхідність розробки власного програмного рішення, яке поєднуватиме простоту розгортання, мінімальні апаратні вимоги, підтримку віддаленого доступу та сучасні механізми безпеки. Розробка менеджера файлової системи на базі Raspberry Pi з використанням веб-технологій дозволяє вирішити зазначені проблеми та створити гнучку платформу для управління файловими ресурсами в умовах обмежених ресурсів пристрою.

1.3 Використання Raspberry Pi як платформи для файлового менеджера

У межах кваліфікаційної роботи передбачається використання апаратної платформи Raspberry Pi як основи для розгортання файлового менеджера. Використання одноплатного комп’ютера зумовлено необхідністю створення компактної, енергоефективної та економічно доцільної системи керування файлами, яка може функціонувати як локальний сервер або мережеве сховище даних.

«Raspberry Pi – одноплатний комп’ютер, розроблений британським фондом Raspberry Pi Foundation. Головне призначення – сприяти вивченню базових комп’ютерних навичок школярами. У жовтні 2013 розробники оголосили про продаж більше двох мільйонів цих плат, що закріпило за Raspberry Pi звання найпопулярнішої платформи для ентузіастів» [6].

Архітектурно Raspberry Pi є повноцінною обчислювальною системою, що інтегрує центральний процесор, графічний процесор, оперативну пам’ять та контролери периферійних інтерфейсів на одній друкованій платі. Такий підхід забезпечує компактність, низьку вартість та енергоефективність пристрою, що є важливими характеристиками для побудови розподілених і вбудованих систем. Одноплатні комп’ютери Raspberry Pi підтримують широкий спектр інтерфейсів,

включаючи USB, HDMI, Ethernet, Wi-Fi, Bluetooth та GPIO (рисунок 1.2), що дозволяє інтегрувати їх у різноманітні апаратні середовища та взаємодіяти з зовнішніми пристроями, датчиками та модулями зберігання даних.

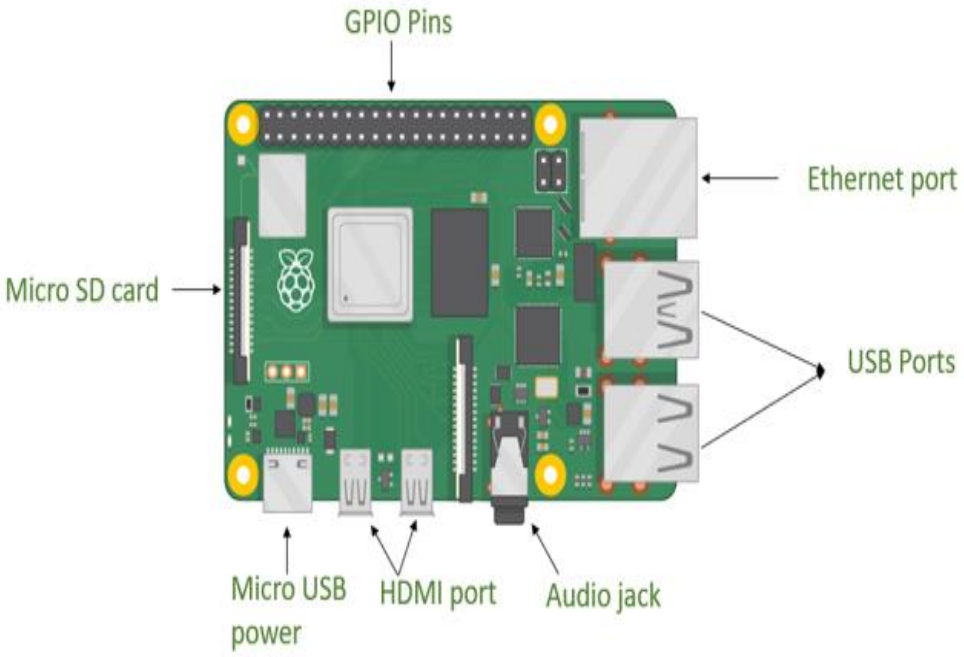

Рисунок 1.2 – Схема архітектури raspberry Pi [7]

Програмне забезпечення Raspberry Pi базується переважно на операційних системах сімейства Linux, серед яких найбільш поширеною є Raspberry Pi OS. Відкрита архітектура та підтримка великої кількості програмних бібліотек і фреймворків забезпечують можливість використання даної платформи для розробки серверних додатків, веб-сервісів, мережевих сховищ даних, систем моніторингу та керування ресурсами. Завдяки наявності повноцінного стеку мережевих протоколів Raspberry Pi може функціонувати як мережевий вузол, сервер або клієнт у локальних і глобальних мережах.

Лінійка одноплатних комп’ютерів Raspberry Pi представлена значною кількістю моделей, які відрізняються технічними характеристиками, функціональними можливостями та сферою застосування. З моменту появи першої версії платформи Raspberry Pi у 2012 році відбувся суттєвий розвиток апаратної частини, що супроводжувався зростанням продуктивності процесорів,

обсягу оперативної пам’яті, швидкості мережевих інтерфейсів та підтримки сучасних стандартів зберігання даних.

Еволюція Raspberry Pi демонструє поступовий перехід від простих навчальних мікрокомп’ютерів до повноцінних компактних обчислювальних систем, здатних виконувати функції серверів, мережевих сховищ і контролерів розподілених систем. Сучасні моделі Raspberry Pi підтримують багатоядерні процесори, гігабітні мережеві інтерфейси, швидкісні порти USB та можливість підключення високопродуктивних накопичувачів, що робить їх придатними для реалізації прикладних задач у сфері веб-розробки та керування файловими ресурсами.

Різноманіття моделей дозволяє обрати оптимальну конфігурацію залежно від вимог до обчислювальної потужності, енергоспоживання та вартості (рисунок 1.3).

Feature	Raspberry Pi 3	Raspberry Pi 4	Raspberry Pi 5	Raspberry Pi Zero (W/2 W)
Processor	Quad-core Cortex-A53	Quad-core Cortex-A72	Quad-core Cortex-A76	Single-core (Zero/W) or Quad-core (Zero 2 W) Cortex-A53
RAM	1 GB (Model B)	2 GB, 4 GB, or 8 GB	4 GB or 8 GB	512 MB
Wi-Fi	802.11n (2.4 GHz)	802.11ac (dual-band)	802.11ac (dual-band)	802.11n (2.4 GHz)
Bluetooth	4.1	5.0	5.0	4.1 (Zero W) or 4.2 (Zero 2 W)
Ethernet	10/100 Mbps	Gigabit Ethernet	Gigabit Ethernet	None
USB Ports	4 x USB 2.0	2 x USB 3.0, 2 x USB 2.0	2 x USB 3.0, 2 x USB 2.0	1 x micro-USB (data)
HDMI	1 x HDMI	2 x micro-HDMI	2 x micro-HDMI	1 x mini-HDMI
GPIO	40-pin	40-pin	40-pin	40-pin (unpopulated)
Power	5V/2.5A (microUSB)	5V/3A (USB-C)	5V/5A (USB-C)	5V/1.2A (microUSB)
Size	Standard (85.6 x 56.5 mm)	Standard (85.6 x 56.5 mm)	Standard (85.6 x 56.5 mm)	Compact (65 x 30 mm)
Price	$35 (Model B)	35(2GB),35(2GB),55 (4 GB), $75 (8 GB)	60(4GB),60(4GB),80 (8 GB)	5(PiZero),5(PiZero),10 (Pi Zero W), $15 (Pi Zero 2 W)

Рисунок 1.3 – характеристики різних моделей Raspberry Pi [8]

Важливою особливістю Raspberry Pi є його застосування у сфері зберігання та обробки даних. Платформа може використовуватися як основа для створення файлових серверів, персональних мережевих сховищ та систем резервного копіювання. Поєднання підтримки зовнішніх накопичувачів, мережевих протоколів доступу до файлів та можливості налаштування файлових систем

робить Raspberry Pi перспективним апаратним компонентом для реалізації менеджерів файлових систем та систем керування даними.

Завдяки низькій вартості, енергетичній ефективності та гнучкості налаштування Raspberry Pi широко використовується в освітніх установах, наукових дослідженнях та комерційних проєктах. Платформа дозволяє швидко створювати прототипи програмно-апаратних рішень, тестувати архітектурні підходи та досліджувати ефективність алгоритмів обробки та зберігання даних. Таким чином, Raspberry Pi є універсальною апаратною платформою, що може бути використана як основа для реалізації програмних систем керування файловими ресурсами та дослідження їх функціональних можливостей.

РОЗДІЛ 2
ВИБІР ТА ОБҐРУНТУВАННЯ АПАРАТНОЇ ТА ПРОГРАМНОЇ СКЛАДОВОЇ

2.1 Вибір апаратної платформи для реалізації менеджера

Для реалізації програмного забезпечення файлового менеджера використано апаратну платформу Raspberry Pi 3. Вибір цієї моделі є результатом комплексної оцінки її технічних характеристик, вартості та практичних вимог проєкту, а не випадковою альтернативою. Хоча на ринку одноплатних комп’ютерів існують новіші моделі, зокрема Raspberry Pi 4 та Raspberry Pi 5, які характеризуються підвищеною обчислювальною потужністю, збільшеним обсягом оперативної пам’яті та удосконаленими мережевими інтерфейсами, аналіз показав, що для поставлених задач локального файлового сервера і веб-інтерфейсу не завжди потрібні значно більші апаратні ресурси.

Raspberry Pi 3 є одноплатним комп’ютером, оснащеним чотириядерним процесором ARM Cortex-A53 з тактовою частотою 1,2 ГГц та 1 ГБ оперативної пам’яті, що забезпечує достатню продуктивність для розгортання веб-сервера та обробки запитів файлового менеджера (рисунок 2.1).

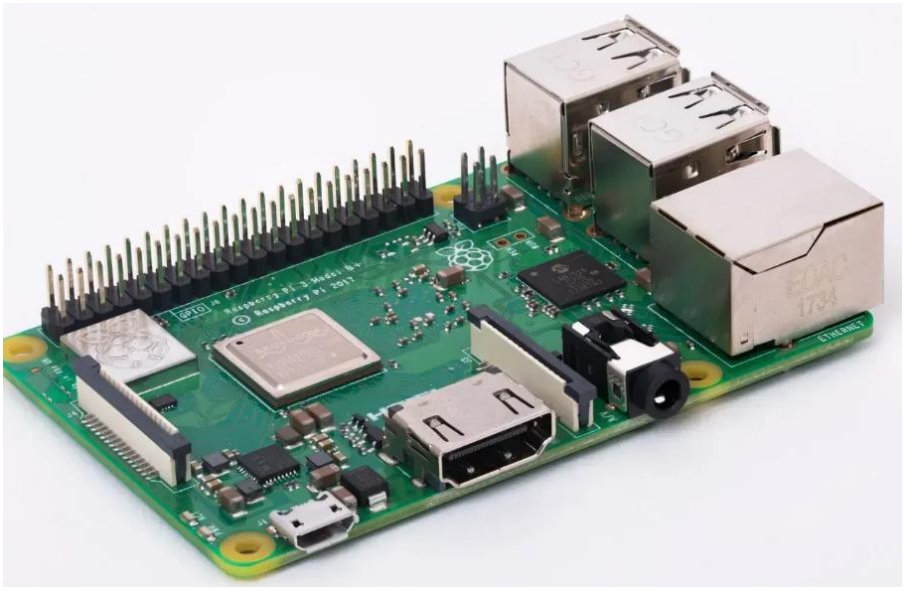

Рисунок 2.1 – Обрана для використання модель Raspberry Pi [9]

Платформа має вбудований Ethernet-інтерфейс зі швидкістю до 100 Мбіт/с, а також підтримує бездротові з’єднання Wi-Fi і Bluetooth, що дозволяє інтегрувати пристрій у локальну мережу та організувати віддалений доступ до системи. Наявність портів USB 2.0 забезпечує можливість підключення зовнішніх накопичувачів для розширення обсягу файлового сховища [10].

Raspberry Pi 4 і Raspberry Pi 5 мають переваги у вигляді більш швидких процесорів та підтримки великих обсягів оперативної пам’яті, що забезпечує кращу продуктивність у випадках високого навантаження або при обробці великих обсягів даних. Однак в контексті розробки файлового менеджера, який обробляє переважно операції читання, запису та навігації в межах файлової системи, продуктивності Raspberry Pi 3 є достатньо для забезпечення стабільної та ефективної роботи застосунку. Важливим фактором є також те, що більшість задач обробляються серверною частиною застосунку на рівні операційної системи та фреймворку, а не за рахунок інтенсивних обчислень, що робить додаткові апаратні ресурси новіших моделей менш критичними. З технічної точки зору, можливість підключення зовнішніх накопичувачів через USB-інтерфейси дозволяє розширити обсяг доступного сховища даних, що є важливою умовою для реалізації функцій файлового менеджера [11].

Окремої уваги заслуговує енергоефективність пристрою. У порівнянні з традиційними персональними комп’ютерами або серверними рішеннями, Raspberry Pi 3 споживає значно менше електроенергії, що є важливим фактором у випадку тривалої безперервної роботи системи. Компактні габарити та відсутність необхідності у складній системі охолодження спрощують розміщення пристрою та знижують експлуатаційні витрати.

Економічний аспект також відіграє ключову роль у виборі плати. Raspberry Pi 3 має суттєво нижчу вартість порівняно з новішими моделями, що робить її привабливим вибором для проєктів із обмеженим бюджетом або навчальних лабораторних робіт. Враховуючи, що розроблювана система не передбачає використання ресурсомістких сервісів або складних алгоритмів обробки даних,

використання більш дорогих моделей не забезпечує пропорційного підвищення функціональних можливостей і не виправдовує додаткових витрат.

Таким чином, обрана апаратна платформа забезпечує необхідний баланс між вартістю, продуктивністю та експлуатаційною ефективністю. Raspberry Pi 3 є достатньою для реалізації поставлених функціональних вимог, включаючи підтримку веб-сервера, обробку HTTP-запитів, взаємодію з файловою системою, а також інтеграцію з мережею. Це дозволяє зекономити ресурси проєкту без істотного погіршення продуктивності, що обґрунтовує вибір цієї моделі серед доступних варіантів апаратних платформ.

2.2 Обґрунтування вибору програмної складової

Програмна складова є невід’ємною частиною розроблюваної системи, оскільки саме вона визначає функціональні можливості, продуктивність та зручність взаємодії користувача з файловим менеджером. Вибір програмних засобів здійснювався з урахуванням вимог до надійності, масштабованості, безпеки та сумісності з обраною апаратною платформою.

У якості програмної основи для функціонування системи обрано операційну систему на базі Linux, що обумовлено її високою стабільністю, відкритістю та широкими можливостями налаштування. Операційні системи сімейства Linux є стандартом для серверних рішень, оскільки забезпечують ефективне управління ресурсами, підтримку багатозадачності та надійні механізми безпеки. Крім того, Linux має розвинену екосистему програмного забезпечення, що включає веб-сервери, інтерпретатори мов програмування та засоби роботи з файловими системами, що є необхідним для реалізації веб-орієнтованого файлового менеджера.

З урахуванням обраної апаратної платформи, доцільним є використання спеціалізованої операційної системи Raspberry Pi OS, яка оптимізована для роботи на пристроях Raspberry Pi. Дана система базується на дистрибутиві Debian і адаптована до апаратних особливостей одноплатного комп’ютера, що

забезпечує стабільну та ефективну роботу навіть за обмежених ресурсів. Raspberry Pi OS підтримує широкий спектр драйверів, мережевих протоколів і серверного програмного забезпечення, що спрощує розгортання веб-застосунків та організацію доступу до файлової системи [12].

Важливою перевагою використання Raspberry Pi OS є її сумісність із більшістю сучасних веб-технологій, зокрема стеком PHP та відповідними фреймворками. Це дозволяє безперешкодно реалізувати серверну частину файлового менеджера та забезпечити взаємодію з клієнтським інтерфейсом через мережу. Таким чином, поєднання операційної системи Linux та її спеціалізованого дистрибутиву Raspberry Pi OS створює надійне програмне середовище для реалізації розроблюваної системи [13].

З огляду на використання операційної системи на базі Linux та її дистрибутиву Raspberry Pi OS для реалізації веб-орієнтованого файлового менеджера доцільним є застосування класичного веб-орієнтованого серверного стеку, що поєднує веб-сервер, інтерпретатор мови програмування та допоміжні програмні компоненти.

У якості веб-сервера використано Apache HTTP Server, який забезпечує обробку HTTP-запитів, маршрутизацію звернень клієнта та передачу даних між користувацьким інтерфейсом і серверною частиною застосунку. Даний сервер характеризується високою стабільністю, гнучкістю налаштування та широкою підтримкою в середовищі Linux, що робить його одним із найбільш поширених рішень для розгортання веб-застосунків [14].

Для організації зберігання та обробки даних у розробленому застосунку використовувалися реляційні бази даних, що базуються на мові структурованих запитів SQL. Дана мова забезпечує виконання основних операцій із даними, зокрема створення, модифікацію, вибірку та видалення записів у таблицях. У межах застосунку SQL використовується через механізми фреймворку Laravel, що дозволяє абстрагувати роботу з базою даних за допомогою ORM-засобів та спрощує взаємодію з нею на рівні програмного коду.

Для спрощення адміністрування бази даних застосовувався веб-інтерфейс phpMyAdmin, який надає зручні інструменти для керування структурою бази даних та її вмістом.

«phpMyAdmin – це безкоштовний програмний інструмент, написаний на PHP , призначений для адміністрування MySQL та MariaDB через веб інтрефейс. Він підтримує широкий спектр операцій (керування базами даних, таблицями, стовпцями, зв'язками, індексами, користувачами тощо) через зручний інтерфейс, а також дозволяє виконувати команди SQL безпосередньо» [15].

Використання phpMyAdmin дозволяє виконувати більшість операцій без необхідності роботи з командним рядком, що значно спрощує процес налаштування та супроводу бази даних у середовищі Raspberry Pi.

Таким чином, використаний набір технологій забезпечує повноцінне функціонування веб-застосунку на базі одноплатного комп’ютера, а також спрощує процес його розгортання, супроводу та подальшого використання.

2.3 Вибір мови програмування для реалізації застосунку

Для реалізації логіки використовується мова програмування PHP, яка виконується на стороні сервера та забезпечує обробку запитів, взаємодію з файловою системою та формування динамічного вмісту веб-сторінок.

PHP (Hypertext Preprocessor) – це мова сценаріїв загального призначення, що особливо широко використовується для веб-розробки. Він вбудований в HTML і виконується на стороні сервера, що дозволяє створювати динамічні веб-сторінки та програми. PHP є одним із основних інструментів класичного стеку LAM [16].

Взаємодія між веб-сервером та інтерпретатором PHP забезпечує повноцінне середовище виконання, у якому реалізується клієнт-серверна архітектура системи. Такий підхід дозволяє ефективно обробляти запити, забезпечувати доступ до ресурсів файлової системи та реалізовувати необхідну логіку застосунку, що є необхідною умовою для функціонування інтерактивного

файлового менеджера. Взаємодія між веб-сервером та інтерпретатором PHP забезпечує повноцінне середовище виконання, у якому реалізується клієнт-серверна архітектура системи. Такий підхід дозволяє ефективно обробляти запити, забезпечувати доступ до ресурсів файлової системи та реалізовувати необхідну бізнес-логіку застосунку.

Для реалізації веб-застосунку застосовано фреймворк Laravel, який є одним із найбільш популярних інструментів у сучасній веб-розробці. Його використання обумовлене високим рівнем абстракції, зручністю побудови архітектури застосунків та наявністю широкого набору вбудованих засобів для реалізації типових задач (рисунок 2.2).

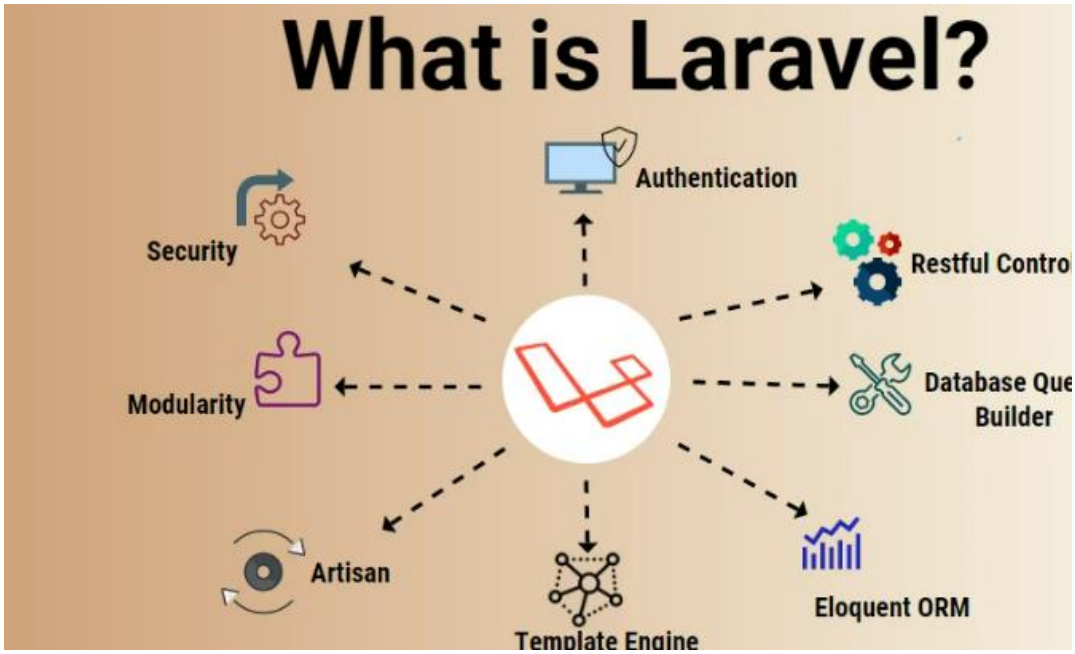

Рисунок 2.2 – Структура фреймворку Laravel [17]

Laravel реалізує архітектурний підхід MVC (Model-View-Controller), що дозволяє логічно розділити обробку даних, бізнес-логіку та представлення. Такий підхід сприяє підвищенню структурованості коду, спрощує його підтримку та подальше розширення функціональності системи. Завдяки цьому розроблюваний файловий менеджер може бути організований у вигляді чітко визначених модулів, що відповідають за окремі аспекти роботи із файловою системою.

Однією з ключових переваг Laravel є наявність вбудованого механізму роботи з файловими сховищами, зокрема підсистеми Storage, яка забезпечує зручну абстракцію доступу до файлової системи. Це дозволяє виконувати операції над файлами та директоріями без необхідності прямої взаємодії з низькорівневими функціями операційної системи, що підвищує безпеку та переносимість застосунку.

Завдяки своїй модульній структурі та підтримці розширень, Laravel створює сприятливе середовище для інтеграції додаткових інструментів, що спрощують розробку інтерфейсу користувача. Зокрема, використання спеціалізованих бібліотек дозволяє значно скоротити час створення адміністративної частини застосунку та підвищити її функціональність без необхідності розробки складної клієнтської логіки.

У цьому контексті доцільним є застосування інструменту Filament, який органічно доповнює можливості Laravel і забезпечує швидке створення сучасного та зручного веб-інтерфейсу для взаємодії користувача з файловою системою.

Важливою особливістю Filament є підтримка реактивної взаємодії з інтерфейсом без необхідності використання складних клієнтських фреймворків. Це досягається завдяки інтеграції з технологіями, що забезпечують динамічне оновлення даних, що особливо важливо для відображення змін у файловій системі в режимі реального часу. Такий підхід значно спрощує розробку та підвищує зручність використання системи [18].

Крім того, Filament забезпечує високий рівень гнучкості та розширюваності, що дозволяє адаптувати інтерфейс під специфічні вимоги проєкту. Наявність вбудованих механізмів керування доступом, валідації даних та обробки форм сприяє підвищенню безпеки та надійності веб-застосунку.

Додатково слід зазначити, що у межах розробки інтерфейсу застосунку використано шаблонізатор Blade, який є складовою фреймворку Laravel. Blade забезпечує зручний механізм формування HTML-сторінок із можливістю вбудовування серверної логіки без порушення структури розмітки [19].

Його застосування дозволяє ефективно організувати повторне використання елементів інтерфейсу, таких як шаблони сторінок, компоненти та окремі фрагменти відображення даних. Завдяки цьому спрощується підтримка коду, зменшується його дублювання та підвищується узгодженість зовнішнього вигляду веб-застосунку.

Таким чином, обрана програмна складова забезпечує необхідні умови для реалізації функціонального, стабільного та зручного у використанні менеджеру. Використані інструменти дозволяють ефективно організувати серверну логіку, взаємодію з файловою системою та формування інтерфейсу користувача, що відповідає вимогам до сучасних інформаційних систем.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ МЕНЕДЖЕРА ФАЙЛОВОЇ СИСТЕМИ НА БАЗІ RASPBERRY PI

3.1 Встановлення операційної системи на Raspberry Pi

Для функціонування веб-застосунку необхідно підготувати програмне середовище, що передбачає встановлення операційної системи.

Перед записом образу операційної системи важливо підготувати носій інформації, зокрема форматування SD-карти у відповідний формат. Форматування SD-карти передбачає очищення її вмісту та створення файлової системи, яка підтримується середовищем встановлення операційної системи.

Очищення SD-карти проведено від попередніх даних та відформатовано із використанням файлової системи FAT32, оскільки вона забезпечує сумісність із завантажувачем Raspberry Pi та дозволяє коректно ініціалізувати процес встановлення.

Форматування виконується за допомогою програмного забезпечення Raspberry Pi Imager.

«Raspberry Pi Imager – це офіційна утиліта, розроблена для спрощення процесу створення завантажувальних носіїв інформації для пристроїв Raspberry Pi. Вона надає зручний графічний інтерфейс, який дозволяє користувачам вибирати операційну систему, завантажувати її за потреби та записувати безпосередньо на SD-карту або USB-накопичувач. Інструмент підтримує Windows, macOS та Linux, що робить його доступним для широкого кола користувачів, від початківців до досвідчених розробників. Він також включає параметри налаштування, такі як попереднє налаштування облікових даних Wi-Fi, імен хостів та SSH-доступу перед першим завантаженням» [20].

Після виконання форматування SD-карти та її підготовки до використання виконується процес встановлення операційної системи. Для цього, як і в випадку з форматуванням використано Raspberry Pi Imager в якому можна обрати необхідний образ системи, налаштувати параметри встановлення та виконати

запис на носій інформації. Застосування даного програмного засобу забезпечує зручність виконання процедури встановлення, мінімізує ймовірність помилок та дозволяє швидко підготувати пристрій до подальшого налаштування і використання.

При встановленні операційної системи зроблено вибір типу пристрою. У запропонованому переліку апаратних платформ обрано модель Raspberry Pi 3 (рисунок 3.1). Вона відповідає використовуваному пристрою та визначає подальшу конфігурацію параметрів під час процесу встановлення.

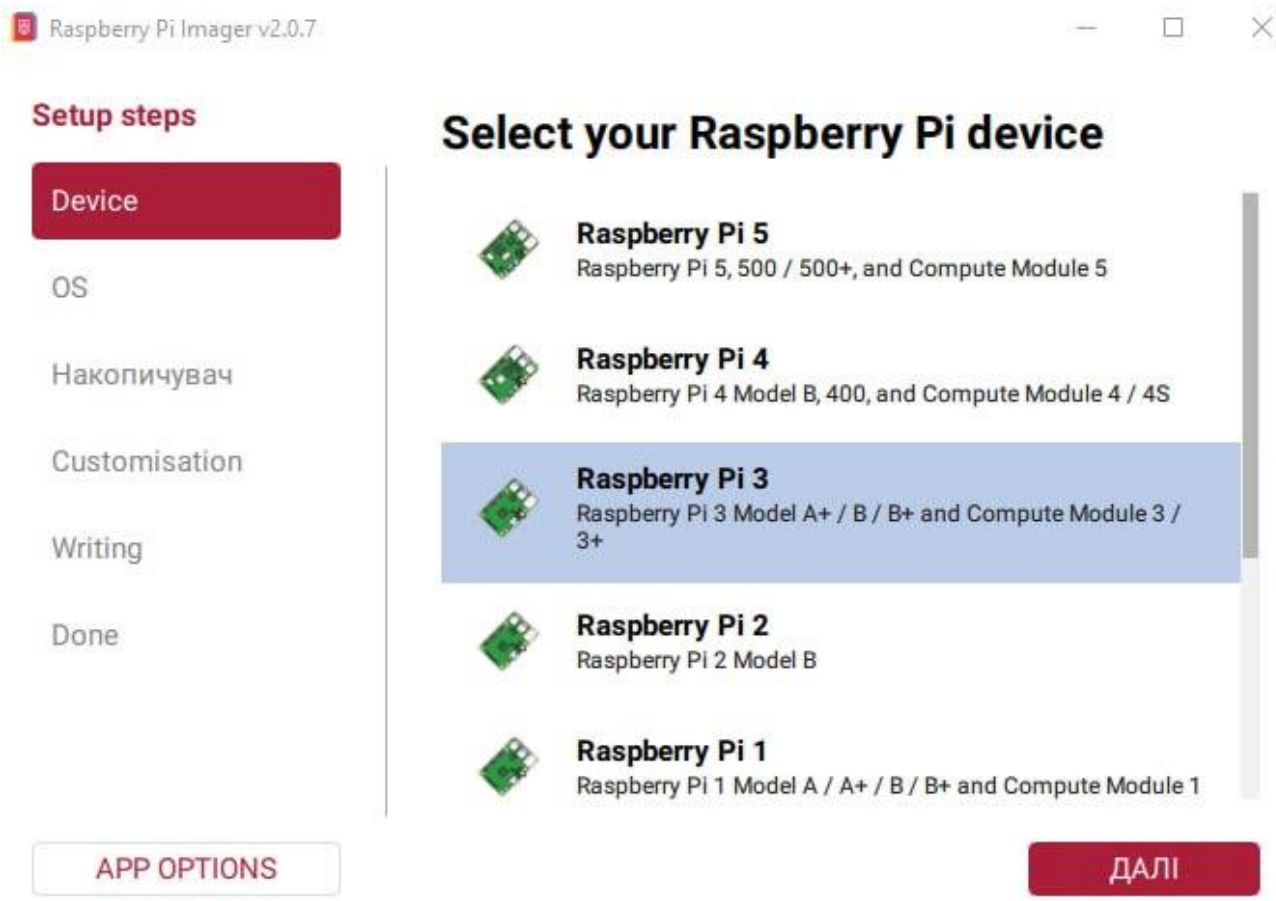

Рисунок 3.1 – Вибір моделі пристрою у Raspberry Pi Imager

Далі здійснений вибір операційної системи, яка буде записана на підготовлений носій. У переліку доступних варіантів обрано Raspberry Pi OS (64-bit). Це актуальна версієя системи, також вона підтримує сучасні програмні засоби (рисунок 3.2). Цей варіант забезпечує необхідну

сумісність із сучасним програмним забезпеченням та підтримує роботу веб-застосунків.

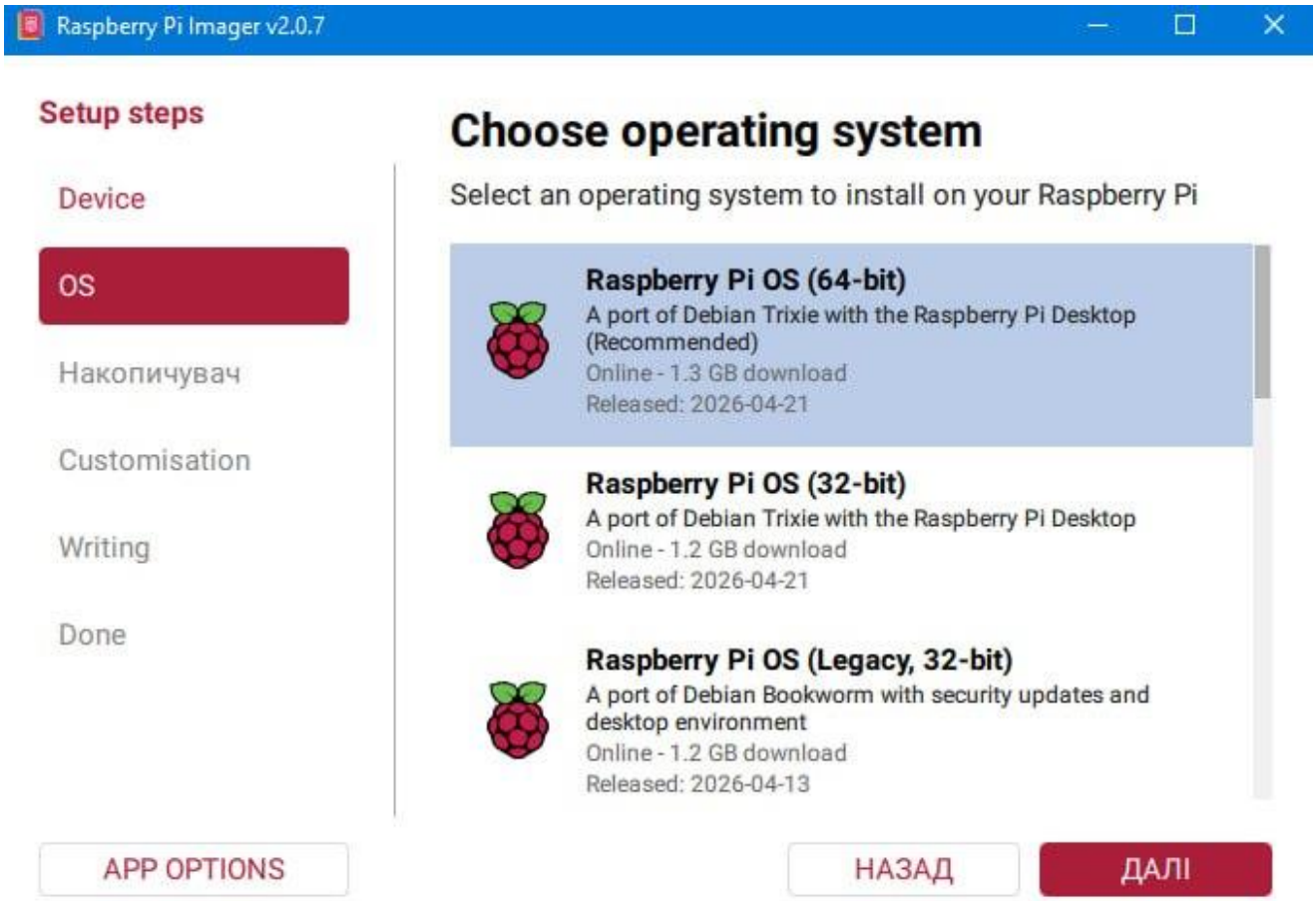

Рисунок 3.2 – Вибір операційної системи у Raspberry Pi Imager

Наступним кроком здійснюється вибір носія інформації, на який буде записано образ операційної системи. У даному випадку обрано SD-карту обсягом 58 ГБ, що використовується як основний накопичувач для Raspberry Pi, оскільки саме з неї відбувається завантаження та подальша робота системи (рисунок 3.3).

Великий обсяг накопичувача потрібний для стабільної роботи пристрою. При маленькому обсязі SD-карти можливі затримки в роботі системи і довгий запуск пристрою.

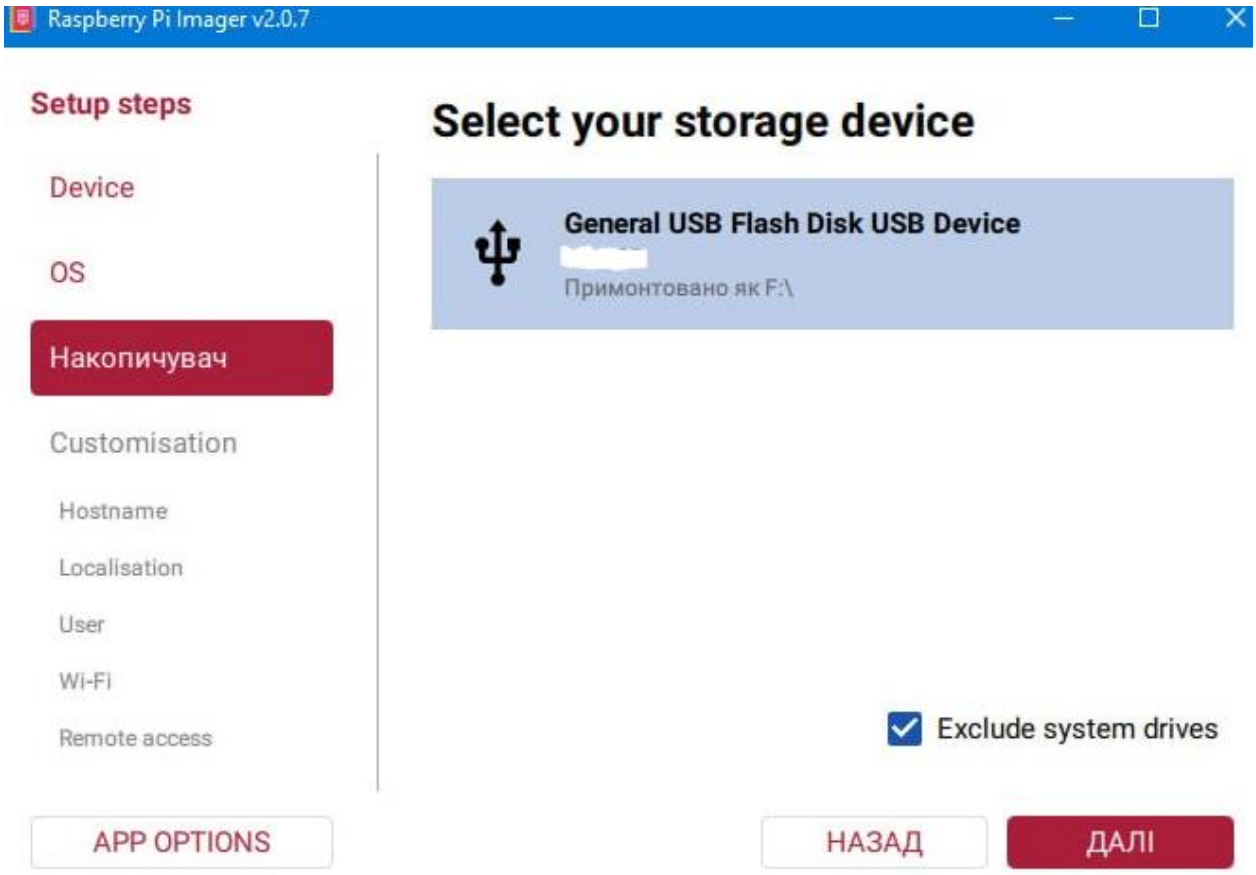

Рисунок 3.3 – Вибір носія інформації у Raspberry Pi Imager

При налаштуванні додаткових параметрів системи зазначено параметри облікового запису користувача, зокрема ім’я користувача та пароль (рисунок 3.4), які використовуються для автентифікації та подальшого адміністрування системи. Зазначені облікові дані забезпечують доступ до операційної системи після її встановлення та дозволяють виконувати налаштування і запуск програмного забезпечення.

Також, у процесі конфігурації виконано підключення до бездротової мережі Wi-Fi для забезпечення мережевої взаємодії, зокрема для встановлення програмних компонентів, оновлення системи та доступу до веб-застосунку. Активовано службу SSH, яка дозволяє здійснювати віддалене підключення до Raspberry Pi за допомогою терміналу, що значно спрощує процес керування пристроєм без необхідності використання периферійних пристроїв, таких як монітор, клавіатура та миша.

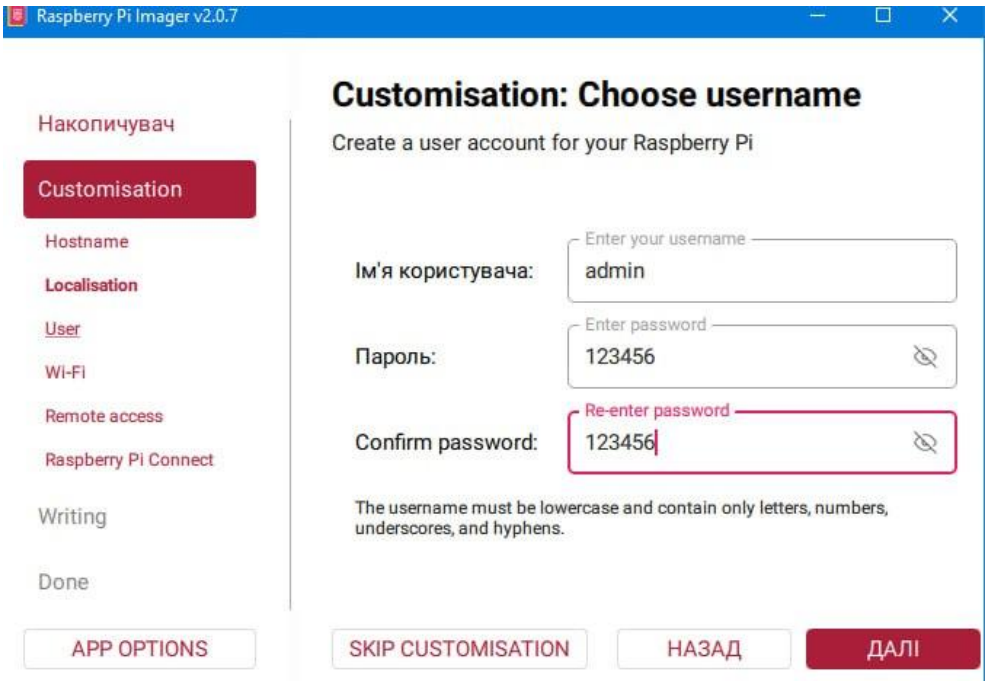

Рисунок 3.4 – Налаштування облікового запису та параметрів доступу

Після налаштування всіх необхідних параметрів почався запис образу операційної системи на носій (рисунок 3.5). Виконується автоматичне перенесення системних файлів на SD-карту, а також її підготовка до подальшого використання як завантажувального носія.

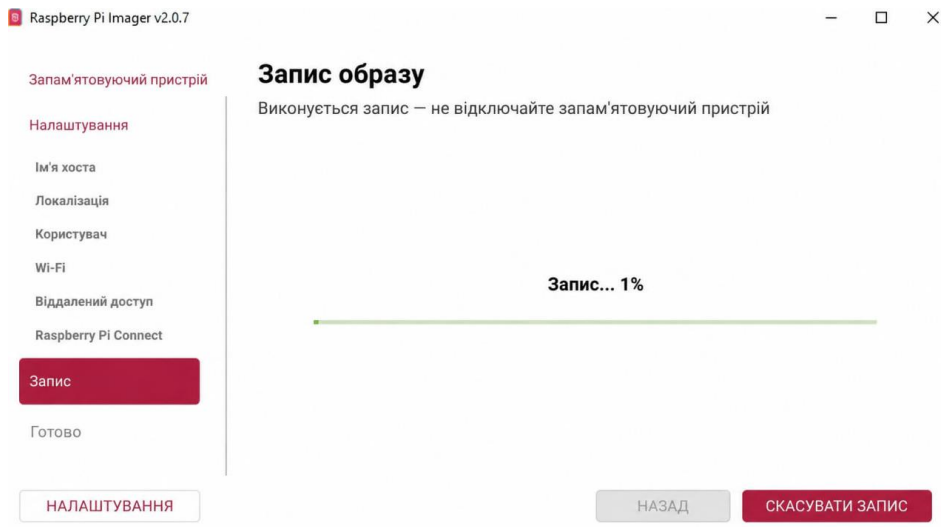

Рисунок 3.5 – Процес запису образу операційної системи на SD-карту

По завершенню процесу запису операційної системи Raspberry Pi успішно завантажився із підготовленого носія. Під час завантаження ініціалізовано

системне середовище, після чого відображається графічний інтерфейс робочого столу (рисунок 3.6).

Рисунок 3.6 – Успішне завантаження операційної системи Raspberry Pi OS

Після встановлення операційної системи пристрій готовий до подальшого використання та розгортання веб-застосунку.

3.2 Налаштування програмного середовища та встановлення залежностей

Після встановлення операційної системи та виконання базових налаштувань апаратної платформи потрібно розгорнути програмне середовище, яке необхідно для функціонування веб-застосунку. Здійснюється встановлення та конфігурація програмних компонентів, які забезпечують обробку запитів, виконання серверної логіки, взаємодію з файловою системою та підтримку роботи користувацького інтерфейсу.

3.2.1 Підготовка системи до встановлення програмного забезпечення

Перед встановленням основних програмних компонентів виконується підготовка операційної системи. Здійснено оновлення системних пакетів, встановлення базових утиліт та перевірку мережевого підключення.

На початку виконано оновлення списку доступних пакетів за допомогою команди «sudo apt update», після чого здійснено оновлення вже встановленого програмного забезпечення із використанням команди «sudo apt upgrade». Це дозволило отримати актуальні версії пакетів та усунути можливі вразливості системи.

Для зручного адміністрування системи налаштовано віддалений доступ за допомогою SSH.

Підключення до пристрою здійснюється через мережу з використанням відповідного клієнта шляхом виконання команди «ssh username@ip_address», де вказується ім’я користувача та IP-адреса пристрою (рисунок 3.7). Такий спосіб доступу типовий для виконання команд на Raspberry Pi, встановлення програмного забезпечення та налаштування системи у віддаленому режимі.

Рисунок 3.7 – Підключення до Raspberry Pi через SSH

Застосування SSH значно спрощує процес розробки та розгортання веб-застосунку, оскільки всі необхідні дії можуть здійснюватись дистанційно через термінал.

3.2.2 Встановлення та налаштування LAMP-середовища

Для функціонування веб-застосунку потрібно розгорнути серверне середовище, яке забезпечує обробку HTTP-запитів, виконання серверної логіки та взаємодію з базою даних. Для цього використано LAMP-середовище.

«LAMP – це комбінація операційної системи та відкритого програмного забезпечення, призначених для організації роботи динамічних веб-сайтів та веб-додатків. Акронім LAMP є похідним від перших літер компонентів, що входять до його складу: ОС Linux, веб-сервер Apache, бази даних MySQL та мова програмування PHP/Perl/Python» [21].

Спочатку виконується встановлення веб-сервера Apache за допомогою команди «sudo apt install apache2». Після завершення встановлення здійснено запуск сервера та перевірку його працездатності шляхом відкриття локальної адреси пристрою в браузері. У результаті відображається стандартна стартова сторінка Apache (рисунок 3.8), що підтверджує коректність встановлення та готовність до обробки HTTP-запитів.

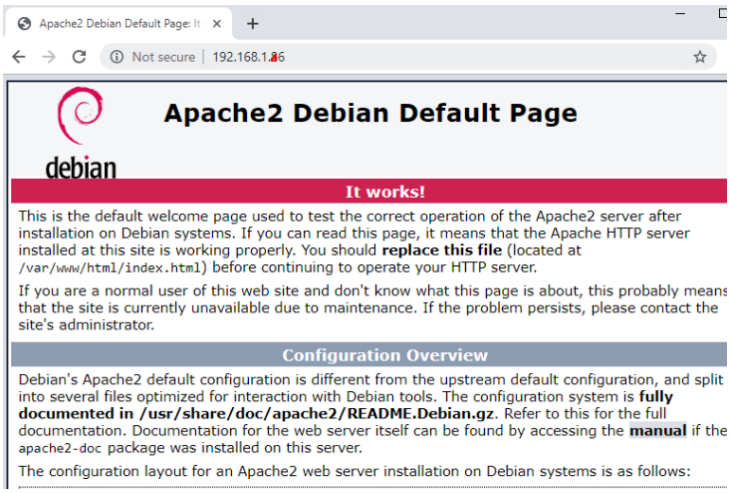

Рисунок 3.8 – Стартова сторінка Apache

Далі встановлена система керування базами даних MySQL із використанням команди «sudo apt install mysql-server». Після виконання базового налаштування системи керування базами даних додатково створено окремого користувача для роботи із застосунком. За допомогою команди

«sudo mysql --user=root --password» виконано підключення до бази даних, після чого створено нового користувача з використанням SQL-запиту «CREATE USER 'admin'@'localhost' IDENTIFIED BY 'пароль'» (рисунок 3.9). Далі надано повні привілеї доступу до баз даних, після чого зміни застосовано командою «FLUSH PRIVILEGES».

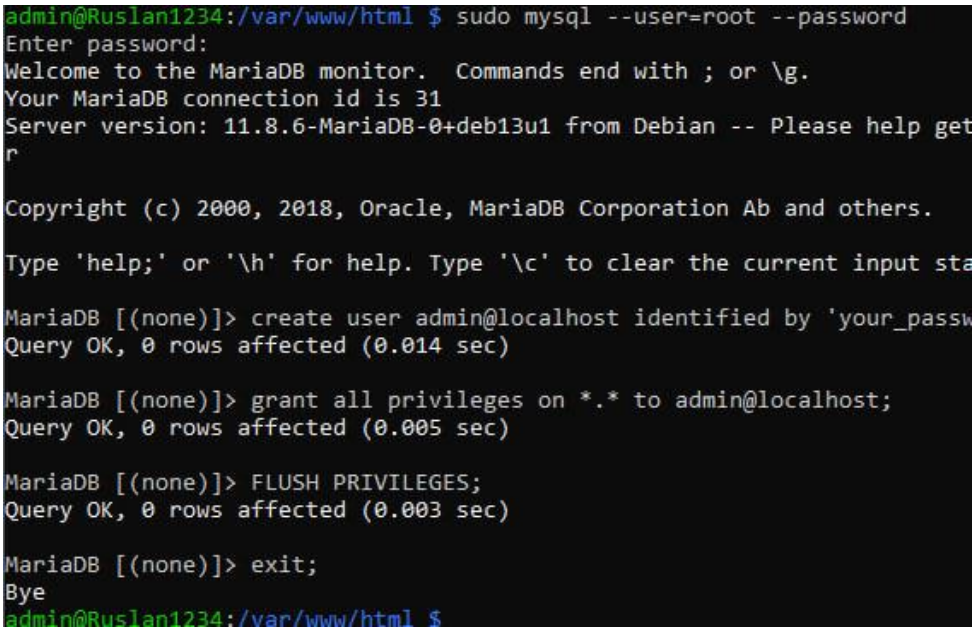

Рисунок 3.9 – Створення користувача та налаштування MySQL

Для забезпечення виконання серверної логіки встановлено інтерпретатор мови програмування PHP разом із необхідними розширеннями. Встановлення здійснено за допомогою команди «sudo apt install php libapache2-mod-php phpmysql». Додаткові модулі забезпечують коректну інтеграцію PHP із веб-сервером та системою керування базами даних.

Після встановлення основних компонентів виконано перевірку їх взаємодії. Для цього створено тестовий файл у директорії веб-сервера за допомогою команди «sudo nano /var/www/html/index.html», у якому розміщено простий PHP-код із текстом «Hello World». Відкриття даного файлу у браузері дозволяє переконатися у коректній роботі веб-сервера та обробці HTTP-запитів (рисунок 3.10).

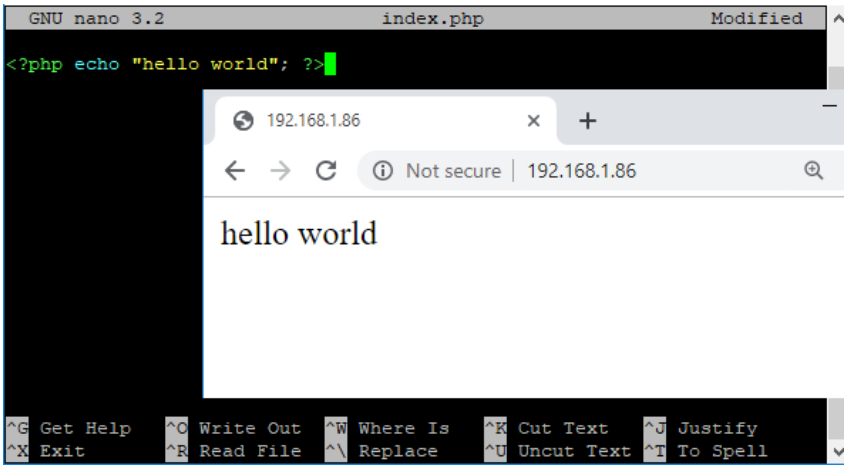

Рисунок 3.10 – Результат виконання PHP-скрипта у веб-браузері

Для зручного адміністрування баз даних додатково встановлено веб-інтерфейс керування СКБД – phpMyAdmin.

Встановлення phpMyAdmin здійснено за допомогою команди «sudo apt install phpmyadmin». У процесі інсталяції обрано веб-сервер Apache для автоматичної інтеграції, а також виконано налаштування доступу до системи керування базами даних.

Після завершення встановлення доступ до phpMyAdmin здійснюється через веб-браузер за адресою «http://localhost/phpmyadmin» або за IP-адресою пристрою (рисунок 3.11). Це дозволяє виконувати адміністрування баз даних у зручному графічному середовищі без необхідності використання командного рядка.

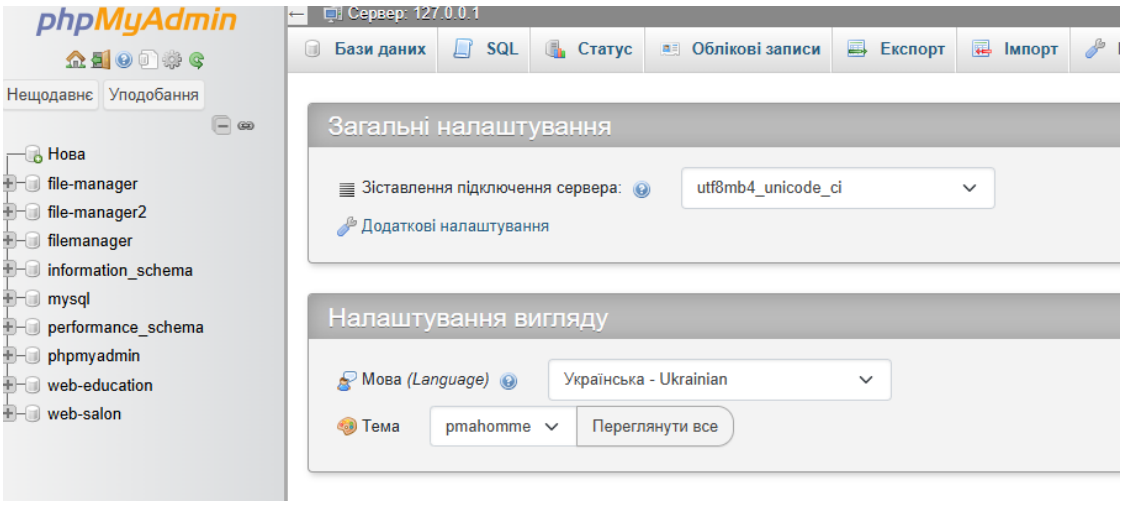

Рисунок 3.11 – Інтерфейс phpMyAdmin для керування базами даних

Використання phpMyAdmin значно спрощує процес роботи з базами даних та підвищує ефективність керуванням веб-застосунком.

3.2.3 Встановлення Laravel та Composer

Після підготовки серверного середовища встановлений Laravel, який використовується як основа для розробки веб-застосунку.

Для встановлення Laravel попередньо налаштовано менеджер залежностей Composer, який використовується для керування бібліотеками та компонентами проєкту. Завантаження інсталяційного скрипта виконано за допомогою команди «php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"», після чого здійснено перевірку цілісності файлу та запуск встановлення командою «php composer-setup.php» (рисунок 3.12).

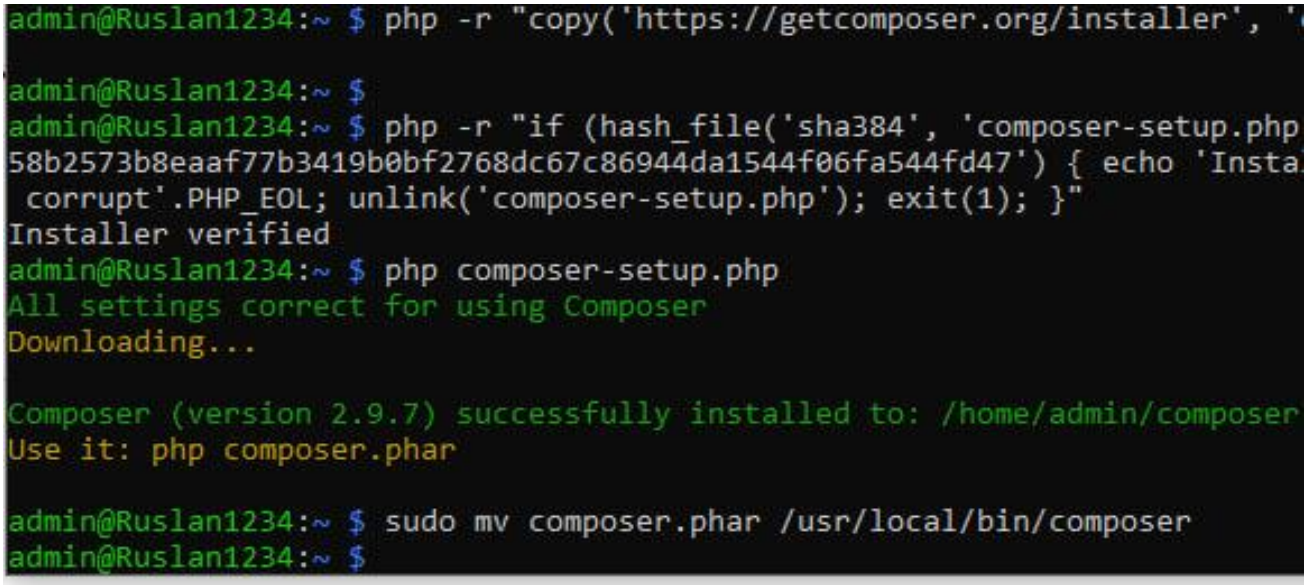

Рисунок 3.12 – Процес встановлення та налаштування менеджера залежностей Composer

Отриманий файл переміщено у системну директорію за допомогою команди «sudo mv composer.phar /usr/local/bin/composer», це дає можливість використання Composer як глобального інструмента .

3.2.4 Встановлення та налаштування Filament

Встановлення Filament виконано за допомогою менеджера залежностей Composer із використанням команди «composer require filament/filament». У процесі інсталяції автоматично завантажуються необхідні бібліотеки та інтегруються у структуру Laravel-проєкту.

Після встановлення пакета виконано його початкове налаштування за допомогою команди «php artisan filament:install», що дозволяє створити базову конфігурацію та підготувати систему до використання адміністративної панелі. У результаті виконання цієї команди до проєкту додаються необхідні компоненти, маршрути та службові файли.

Для отримання доступу до адміністративної панелі створено обліковий запис користувача із використанням команди «php artisan make:filament-user». У процесі виконання команди задаються облікові дані, які надалі використовуються для авторизації в системі.

Після завершення налаштування доступ до інтерфейсу Filament здійснюється через веб-браузер за відповідною адресою застосунку. Тепер можливо керувати даними та функціональними можливостями системи у зручному графічному середовищі.

3.3 Розробка та тестування менеджера файлової системи

Розробка програмної частини застосунку здійснювалась на окремому робочому середовищі, після чого виконано розгортання проєкту на одноплатному комп’ютері Raspberry Pi. На рисунку 3.13 зображено структуру проєкту менеджера файлової системи, організовану відповідно до архітектури фреймворку Laravel.

Каталог app містить основну серверну логіку застосунку, зокрема контролери, моделі та службові класи. У директоріях resources і public розміщуються елементи користувацького інтерфейсу, стилі, шаблони сторінок та публічні файли, доступні через браузер (додаток А).

Для роботи з маршрутами та конфігурацією системи використовуються каталоги routes і config, тоді як директорія storage призначена для зберігання службових файлів, журналів та кешованих даних. Окремо також присутні файли керування залежностями composer.json і package.json, які використовуються для підключення необхідних бібліотек та модулів.

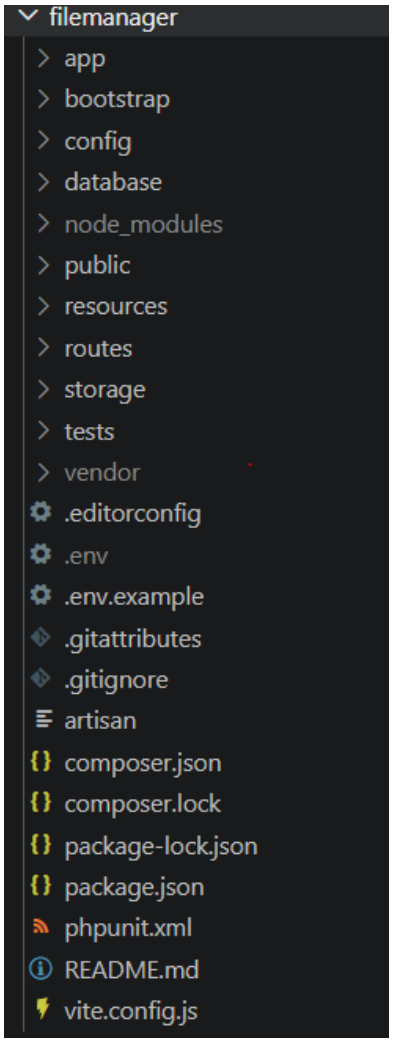

Рисунок 3.13 – Структура менеджера файлової системи

Така структура відображає розподіл функціональних компонентів застосунку та їх взаємодію між собою.

3.3.1 Реалізація функціональних можливостей

Реалізація менеджера файлової системи зосереджена на виконанні основних операцій над файлами та каталогами в межах веб-застосунку. Логіка роботи побудована із використанням можливостей фреймворку Laravel, зокрема

підсистеми Storage, яка забезпечує єдиний інтерфейс для взаємодії з файловою системою.

Функціональні можливості реалізовано у вигляді окремих методів, кожен з яких відповідає за конкретну операцію. Перегляд вмісту директорій здійснюється шляхом отримання списку файлів і папок за заданим шляхом із подальшим передаванням цих даних у представлення. Це дозволяє формувати актуальну структуру файлової системи при кожній взаємодії користувача.

Для створення нових елементів файлової системи реалізовано окремі методи для роботи з файлами та каталогами.

Метод createFolder() працює з каталогами. На початку назва папки очищується від зайвих пробілів і перевіряється на порожнє значення. Далі формується шлях до нової директорії з використанням Str::slug(), що дозволяє привести назву до безпечного для файлової системи формату. Якщо папка з таким шляхом ще не існує, вона створюється за допомогою методу makeDirectory() (лістинг 3.1).

Лістинг 3.1 – Реалізація функції створення папки

```
public function createFolder()
    {
        $name = trim($this->newItemName);
        if ($name === '') {
            $this->addError('newItemName', 'Назва папки не може бути
порожньою.');
            return;
        }

        $folderPath = $this->currentFolder . '/' . Str::slug($name);

        if (!Storage::disk('data')->exists($folderPath)) {
            Storage::disk('data')->makeDirectory($folderPath);
        }

        $this->newItemName = '';
        $this->loadPerPage($this->currentFolder);
        $this->closeCreateModal();
    }
```

кінець лістингу 3.1

Метод createFile() відповідає за створення нового файлу в межах поточної директорії (лістинг 3.2).

Лістинг 3.2 – Реалізація функції створення файлу

```
public function createFile()
    {
        $this->validate([
            'newItemName' => 'required|string|max:255',
        ]);

        $path = rtrim($this->currentFolder, '/') . '/' . $this-
>newItemName;

        Storage::disk('data')->put($path, '');

        $this->newItemName = '';
        $this->loadPerPage($this->currentFolder);
        $this->closeCreateModal();
    }
```

кінець лістингу 3.2

Спочатку виконується валідація введених даних, зокрема перевіряється, щоб назва файлу була задана, мала текстовий формат і не перевищувала допустиму довжину. Це дозволяє уникнути створення некоректних або порожніх імен.

Після цього формується повний шлях до файлу. Для цього використовується поточна директорія $this->currentFolder, до якої додається введена користувачем назва. Додатково застосовується функція rtrim(), що усуває можливий зайвий символ «/» у кінці шляху, щоб уникнути помилок при його формуванні.

Створення файлу здійснюється за допомогою методу put() підсистеми Storage, який записує дані у вказаний шлях. У даному випадку створюється порожній файл, оскільки передається порожній рядок як вміст. Такий підхід дозволяє швидко ініціалізувати новий файл без додаткових операцій.

Після створення файлу виконується оновлення стану інтерфейсу. Поле введення очищується, а список елементів у поточній директорії повторно

завантажується за допомогою методу loadPerPage(). Це забезпечує відображення нового файлу без необхідності перезавантаження сторінки. Завершальним етапом є закриття модального вікна створення, що сигналізує про завершення операції.

Реалізовано механізм перейменування файлів і директорій. Для цього передбачено відкриття модального вікна, у якому користувач задає нове ім’я об’єкта. Поточний шлях зберігається у змінній $this->renamePath, що дозволяє виконати подальшу обробку. Формування нового шляху здійснюється на основі функції dirname(), яка визначає батьківську директорію, та методу getNewName(), що повертає нове ім’я об’єкта (рисунок 3.14). Такий підхід дозволяє змінювати лише назву без впливу на розташування файлу.

```php
public function copyItem($path)
{
    $this->action = 'copy';
    $this->sourcePath = $path;
    $this->loadDirectories('/');
    $this->showFolderSelector = true;
}

public function moveItem($path)
{
    $this->action = 'move';
    $this->sourcePath = $path;
    $this->loadDirectories('/');
    $this->showFolderSelector = true;
}

public function getNewName(): string
{...
}

public function openRenameModal($path)
{...
}

public function renameItem()
{
    $path = $this->renamePath;
    $newPath = dirname($path) . '/' . $this->getNewName();

    if (Storage::disk('data')->exists($path)) {
        Storage::disk('data')->move($path, $newPath);
        session()->flash('success', 'Файл успішно перейменовано!');
    } else {
        session()->flash('error', 'Файл не знайдений!');
    }

    $this->showRenameModal = false;
    $this->loadPerPage($this->currentFolder);
```

Рисунок 3.14 – Реалізація функції перейменування файлу

Метод renameItem() перевіряє існування об’єкта за заданим шляхом за допомогою Storage::disk('data')->exists(). У випадку успішної перевірки виконується операція переміщення із використанням методу move(), що фактично змінює ім’я файлу або директорії.

У разі відсутності об’єкта виводиться повідомлення про помилку, що дозволяє уникнути некоректних операцій. Після завершення дії модальне вікно закривається, а список файлів оновлюється, що забезпечує відображення актуального стану файлової системи.

Операції копіювання та переміщення реалізовано через збереження шляху до об’єкта та вибір цільової директорії. Тип операції визначається змінною $this->action, що дозволяє використовувати спільну логіку для обробки обох сценаріїв.

Також реалізовано механізм видалення файлів і директорій. Дана операція враховує тип об’єкта та передбачає різні сценарії обробки для файлів і папок (лістинг 3.3).

Лістинг 3.3 – Реалізація функції видалення файлів і папок

```
public function deleteItem($path)
    {
        // Перевірка, чи існує файл або папка
        if (Storage::disk('data')->exists($path)) {
            // Якщо це файл
            if (in_array($path, Storage::disk('data')-
>files(dirname($path)))) {
                Storage::disk('data')->delete($path);
                session()->flash('success', 'Файл успішно видалено!');
            }

            // Якщо це папка
            elseif (in_array($path, Storage::disk('data')-
>directories(dirname($path)))) {
                Storage::disk('data')->deleteDirectory($path);
                session()->flash('success', 'Папка успішно видалена!');
            }

        } else {
            session()->flash('error', 'Файл або папка не знайдені!');
        }
        $this->loadPerPage($this->currentFolder);
    }
```

кінець лістингу 3.3

Метод deleteItem() починається з перевірки існування об’єкта за заданим шляхом із використанням Storage::disk('data')->exists(). Це дозволяє уникнути помилок у випадку, якщо файл або папка вже були видалені або шлях задано некоректно.

Після цього визначається тип об’єкта. Якщо шлях відповідає файлу, що перевіряється через список файлів у батьківській директорії, виконується його видалення за допомогою методу delete(). У випадку, якщо об’єкт є директорією, використовується метод deleteDirectory(), який видаляє папку разом із її вмістом.

У разі успішного виконання операції формується повідомлення про результат за допомогою механізму session()->flash(), що дозволяє відобразити відповідне повідомлення користувачу. Якщо об’єкт не знайдено, генерується повідомлення про помилку.

Після завершення операції виконується оновлення вмісту поточної директорії через виклик методу loadPerPage(), що забезпечує відображення актуального стану файлової системи.

Реалізовані функціональні можливості охоплюють основні операції роботи з файловою системою та забезпечують повний цикл взаємодії користувача з файлами і директоріями. Використання засобів Laravel дозволило уніфікувати доступ до файлової системи та забезпечити коректну обробку всіх операцій у межах застосунку.

3.3.2 Розгортання проєкту на Raspberry Pi

Після завершення розробки програмної частини застосунку проводиться його розгортання на Raspberry Pi.

Отримання проєкту на пристрої виконано за допомогою системи керування версіями Git із використанням команди «git clone», що дозволяє завантажити актуальну версію вихідного коду з віддаленого репозиторію (рисунок 3.15). Після клонування проєкту виконано перехід до робочої директорії застосунку.

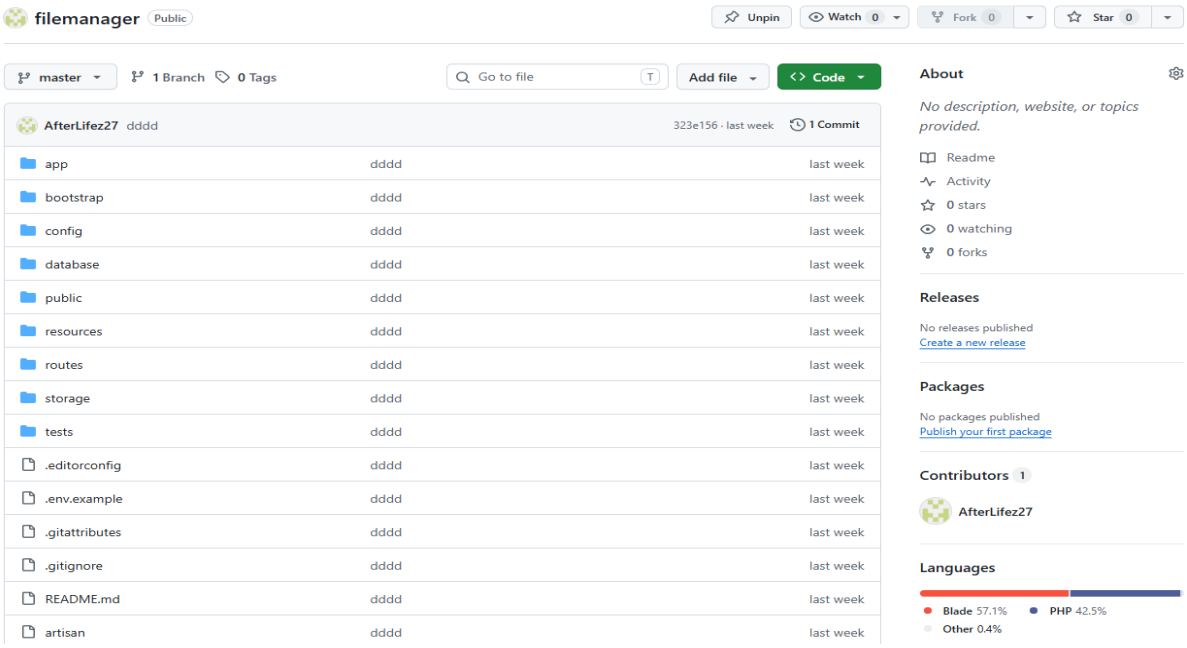

Рисунок 3.15 – Система керування версіями Git

Необхідні залежності встановлені за допомогою Composer. Команда «composer install» забезпечує завантаження всіх бібліотек, визначених у файлі composer.json, що дозволяє відновити повноцінне середовище виконання Laravel-застосунку на пристрої.

Виконано налаштування конфігураційного файлу середовища. Для цього створено файл .env на основі шаблону командою «cp .env.example .env», після чого згенеровано унікальний ключ застосунку командою «php artisan key:generate». Це необхідно для коректної роботи механізмів шифрування та безпеки.

Після налаштування середовища виконано міграції бази даних із використанням команди «php artisan migrate», що дозволило створити необхідні таблиці відповідно до структури застосунку. Запуск застосунку здійснено у середовищі веб-сервера, що забезпечує обробку HTTP-запитів та взаємодію з користувачем через браузер.

У результаті виконаних дій веб-застосунок став доступним за IP-адресою пристрою в локальній мережі (рисунок 3.16).

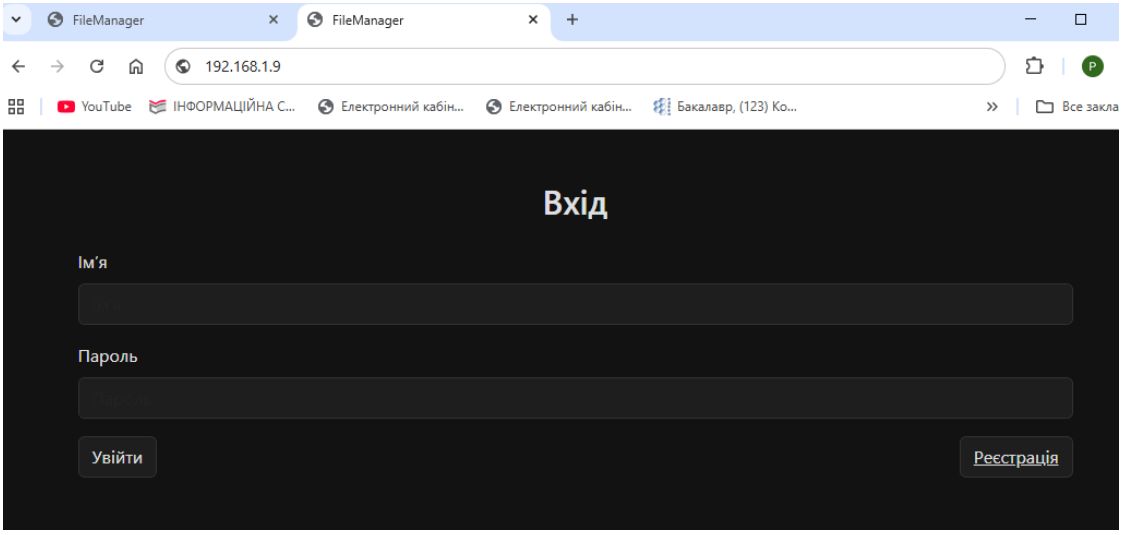

Рисунок 3.16 – Сторінка авторизації веб-застосунку

Перед використанням застосунка потрібно пройти реєстрацію, а після цього здійснити вхід.

3.3.3 Реалізація користувацького веб-інтерфейсу

Основним елементом інтерфейсу є таблиця, у якій відображається вміст поточної директорії. Для кожного елемента виводиться його назва, тип та доступні дії. Дані для відображення формуються на сервері та передаються у представлення після обробки запиту користувача (рисунок 3.17).

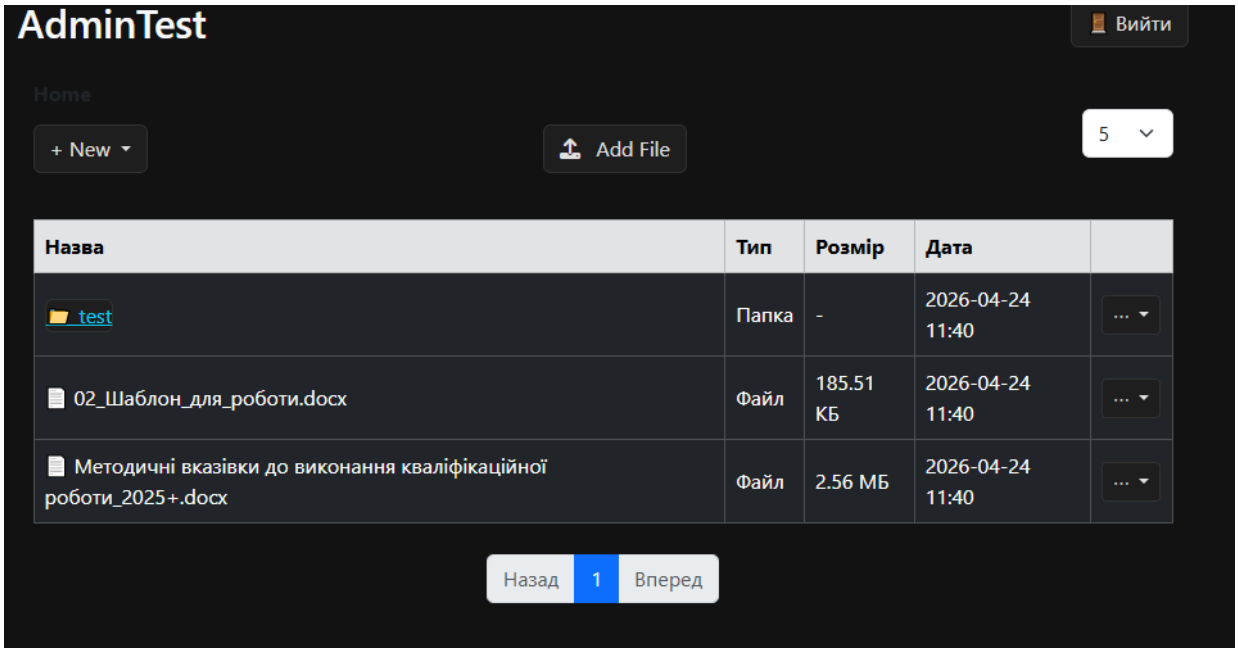

Рисунок 3.17 – Інтерфейс менеджера файлової системи

Інтерфейс реалізований із використанням пакета Filament, який інтегрується у фреймворк Laravel та надає готові компоненти для побудови адміністративних панелей. Таблицю побудовано на основі серверного рендерингу з використанням Livewire, це потрібно для динамічного оновлення даних без перезавантаження сторінки.

Для виконання операцій над файлами використовується контекстне меню, яке містить реалізовані дії, зокрема копіювання, переміщення, перейменування, архівацію та видалення. Вибір дії ініціює відповідний метод на сервері, після чого інтерфейс оновлюється відповідно до результату виконання (рисунок 3.18). Взаємодія з сервером відбувається через Livewire-компоненти, які передають дані безпосередньо у відповідні метод. Інтерфейс також містить елементи керування для створення нових об’єктів та навігації між сторінками списку.

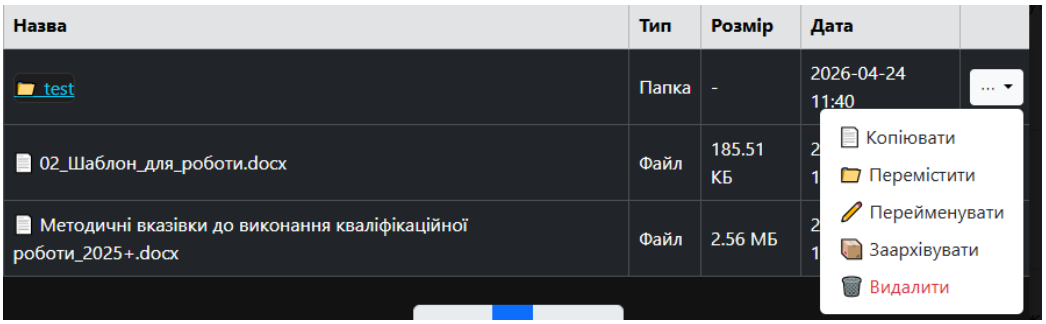

Рисунок 3.18 – Виконання операцій над файлами через веб-інтерфейс

Інтерфейс також відображає повідомлення про результати виконання операцій. У разі успішного виконання дії або виникнення помилки користувачу виводиться відповідне повідомлення, що формується на основі даних із сесії.

3.3.4 Забезпечення віддаленого доступу та тестування застосунку

Після розгортання веб-застосунку його робота обмежується пристроями в одній локальній мережі. Для вирішення цієї задачі використано сервіс ngrok, який надає публічний доступ до локального сервера через мережу Інтернет.

Встановлення ngrok виконано шляхом завантаження архіву з офіційного сайту та його розпакування на пристрої. Після цього виконано автентифікацію за допомогою персонального токена. Для запуску тунелю використано команду

«ngrok http 80», яка перенаправляє зовнішні HTTP-запити на локальний веб-сервер, що працює на Raspberry Pi (рисунок 3.19).

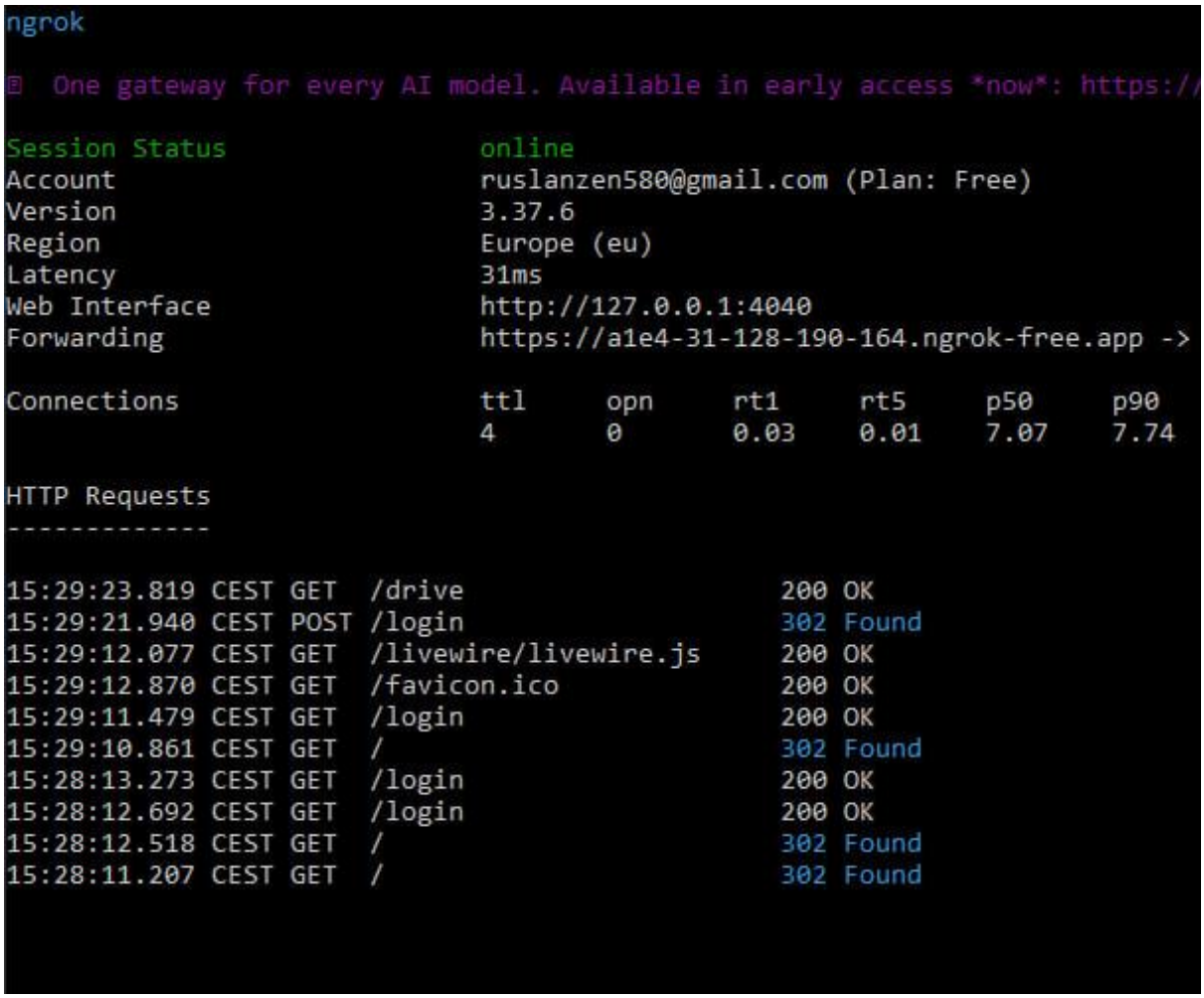

Рисунок 3.19 – Робота ngrok та перенаправлення запитів до локального сервера

Після запуску ngrok формується унікальна публічна адреса, за якою веб-застосунок стає доступним з інших пристроїв. Завдяки цьому можливо виконувати тестування інтерфейсу та функціональних можливостей системи поза межами локальної мережі, зокрема з комп’ютерів та мобільних пристроїв. Також наведено інформацію про активний тунель, включаючи публічну адресу доступу та журнал HTTP-запитів, що дозволяє відстежити взаємодію користувача із застосунком.

На рисунку 3.20 представлено плату Raspberry Pi з підключеними інтерфейсами живлення та передачі даних.

Рисунок 3.20 – Апаратна реалізація системи на базі Raspberry Pi

Всі запити, що надходять через створений тунель, обробляються на пристрої Raspberry Pi, на якому розгорнуто веб-застосунок. Пристрій виконує функції серверної частини системи та забезпечує обробку HTTP-запитів.

У майбутньому систему можна адаптувати для роботи в умовах постійного мережевого доступу шляхом використання статичної IP-адреси та окремого серверного розгортання. Це дозволить відмовитися від використання тунелів ngrok та забезпечити стабільне підключення до вебзастосунку через мережу.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено та впроваджено веб-орієнтований менеджер файлової системи, який функціонує на базі одноплатного комп’ютера Raspberry Pi. Створене програмне рішення демонструє можливість використання даної платформи як середовища для розгортання прикладних веб-застосунків та підтверджує практичну доцільність обраного підходу.

Здійснено реалізацію механізмів виконання основних операцій над файлами та каталогами в межах веб-застосунку. Забезпечено підтримку створення, видалення, перейменування, копіювання та переміщення файлів із використанням серверної логіки на базі мови програмування PHP. Організовано коректну взаємодію із файловою системою операційної системи, що забезпечує стабільність та надійність виконання операцій.

Розроблено користувацький веб-інтерфейс для взаємодії з файловою системою. Інтерфейс реалізовано із застосуванням фреймворку Laravel та інструментів Filament, що дозволило створити структуроване, зручне та інтуїтивно зрозуміле середовище керування. На основі отриманих результатів можна зробити висновок, що використання сучасних веб-технологій значно спрощує організацію взаємодії користувача із системою без потреби у встановленні додаткового програмного забезпечення.

Досліджено особливості використання одноплатного комп’ютера Raspberry Pi як платформи для розгортання веб-застосунків. У ході роботи проаналізовано його функціональні можливості та обмеження, що дозволило підтвердити придатність даного пристрою для реалізації подібних програмних рішень у навчальних та прикладних цілях.

Візуалізовано структуру файлової системи та процеси взаємодії користувача з нею. Реалізовано відображення ієрархії каталогів із можливістю навігації та виконання дій у межах веб-інтерфейсу, що забезпечує зручність роботи з файловими ресурсами.

Спроектовано архітектуру програмного забезпечення з урахуванням принципів модульності та ефективного використання ресурсів. Виконано розподіл функціональних компонентів системи, що забезпечує зрозумілу структуру застосунку та можливість його подальшого розвитку.

Таким чином, розроблена система забезпечує ефективне керування файловими ресурсами в межах веб-застосунку, розгорнутого на базі Raspberry Pi. Результати роботи підтверджують доцільність використання одноплатних комп’ютерів як серверного середовища для реалізації подібних програмних рішень.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Поняття файлової системи. Збереження даних – Інформатика. Інформатика – Електронний підручник з інформатики. URL: https://informatics.dp.ua/faylova-systema-zberezhennya-danykh/ (дата звернення: 05.02.2026).

2. What is a File System? NTFS, EXT4, FAT32 Explained. Manrahul\'s Blog. URL: https://www.manrahul.in/what-is-a-file-system-ntfs-ext4-fat32 (date of access: 06.02.2026).

3. Учасники проєктів Вікімедіа. Файловий менеджер – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Файловий_менеджер (дата звернення: 06.02.2026).

4. Network diagram for client server. EdrawMax. URL: https://edrawmax.wondershare.com/templates/network-diagram-for-client-server.html (date of access: 09.02.2026).

5. Мережеве сховище. Мережеве сховище для кожного. Kingston. URL: https://www.kingston.com/ua/blog/personal-storage/home-nas-is-for-everyone (дата звернення: 10.02.2026).

6. What guide is a Raspberry Pi?. Opensource.com. URL: https://opensource.com/resources/raspberry-pi (date of access: 20.02.2026).

7. GeeksforGeeks. Architecture of Raspberry Pi - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/computer-organization-architecture/architecture-of-raspberry-pi (date of access: 21.02.2026).

8. Evans C. What are the differences between various Raspberry Pi models (e.g., Pi 3, Pi 4, Pi 5, Pi)?. AMPHEO PTY LTD. URL: https://surl.li/nimxis (date of access: 24.02.2026).

9. Raspberry pi 3. Evo. URL: https://evo.net.ua/ru/raspberry-pi-3-model-b/?srsltid=AfmBOop65N5zUKdl1D-bn9gqDih7ivb3T6jb1FLHBtdtxf_cbDE2ECNu (date of access: 26.02.2026).

10. Raspberry Pi comparison chart - MONRASPBERRY. MyRaspberry. URL: https://monraspberry.com/en/raspberry-pi-comparison-chart/ (date of access: 27.02.2026).

11. RaspberryPI models comparison | Comparison tables - SocialCompare. Free Collaborative Comparison engine : Create Comparison tables & Compare everything on SocialCompare. URL: https://socialcompare.com/en/comparison/raspberrypi-models-comparison (date of access: 10.03.2026).

12. Raspberry pi OS installation. Raspberry pi. URL: https://www.raspberrypi.com/documentation/computers/os.html (date of access: 14.03.2026).

13. Grime C. Turn a Raspberry Pi Into a Web Server. Chrisgrime.medium.URL: https://chrisgrime.medium.com/turn-a-raspberry-pi-into-a-web-server-ea7f395e6fe8 (date of access: 17.03.2026).

14. Welcome! - The Apache HTTP Server Project. Welcome! - The Apache HTTP Server Project. URL: https://httpd.apache.org (date of access: 20.03.2026).

15. PhpMyAdmin. phpMyAdmin. URL: https://www.phpmyadmin.net/ (date of access: 20.03.2026).

16. PHP: Вступ до розробки на PHP - Manual. PHP. URL: https://www.php.net/manual/uk/introduction.php (дата звернення: 25.03.2026).

17. What is Laravel?. EDUCBA. URL: https://www.educba.com/what-is-laravel/ (date of access: 27.03.2026).

18. Awam D. Filament v4 Stable: Faster, Safer, and Packed with New Features. Medium. URL: https://medium.com/@developerawam/filament-v4-stable-faster-safer-and-packed-with-new-features-f826d415ee5a (date of access: 01.04.2026).

19. Blade Templates | Laravel 12.x - The clean stack for Artisans and agents.Laravel. URL: https://laravel.com/docs/12.x/blade (date of access: 01.04.2026).

20. Raspberry Pi Imager. Source forge. URL: https://sourceforge.net/projects/raspberry-pi-imager.mirror/ (date of access: 05.04.2026).

21. Raspberry Pi: Install Apache + MySQL + PHP (LAMP Server) | Random Nerd Tutorials. Random Nerd Tutorials. URL: https://randomnerdtutorials.com/raspberry-pi-apache-mysql-php-lamp-server/ (date of access: 09.04.2026