Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри, канд. техн. наук, доцент
_____ Я. М. Крайник
« __ » _____ 2022 р.

КВАЛІФІКАЦІЙНА РОБОТА

Галузь знань: 12 Інформаційні технології
Спеціальність: 123 Комп'ютерна інженерія
Тема: **Система дослідження та моделювання роботи бібліотек стиснення**

Шифр: 123 – КР.1 – 405.21810528

Виконав:
студент 4 курсу, групи 405, спеціальності
123 Комп'ютерна інженерія
_____ О. В. Шевченко

Керівник:
Завідувач кафедри, канд. техн. наук, доцент
_____ Я. М. Крайник

Миколаїв 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри,
канд. техн. наук, доцент
_____ Я. М. Крайник
« __ » _____ 2022 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Видано студенту групи 405 факультету комп'ютерних наук

Шевченку Олександру Вікторовичу

(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи: Система дослідження та моделювання роботи бібліотек стиснення. _____

Затверджена наказом по ЧНУ від «__» _____ 2022 р. No _____

2. Строк представлення кваліфікаційної роботи «____» _____ 2022 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Застосунок має моделювати та наочно зображати різні види стисненнь, та надає змогу завантажи результати після обробки. _____

_____

_____

4. Перелік питань, що підлягають розробці

Виконати аналіз алгоритмів стиснення, які використовуються у системах передачі та збереження інформації та обрати найбільш популярні алгоритми, які будуть включені до складу реалізованої системи. Зробити аналіз та обрати програмні засоби, які необхідні для реалізації системи моделювання. Виконати моделювання розробки системи засобами UML.

Розробити систему моделювання бібліотек стиснення з використанням обраних програмних засобів та UML-моделювання.Продемонструвати роботу системи з різними типами даних та з різними алгоритмами.

5. Перелік графічних матеріалів

UML-діаграми,

блок-схеми алгоритмів,

гістограми порівняння результатів стиснення,

демонстрація інтерфейсів,

лістінги програмного коду

6. Завдання до спеціальної частини

Розглянути основні державні будівельні норми України, щодо вентиляції та кондиціонування, забирання зовнішнього та викид витяжного повітря, витрати припливного повітря та організація повітрообміну. Оволодіти навиками розрахунку систем повітрообміну при загально обмінній вентиляції виробничих приміщень.

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
ст. викладач А. О. Алексєєва	кафедра екології Медичного інституту ЧНУ ім. Петра Могили	спеціальна частина з охорони праці

Керівник роботи ____ канд. тех. наук, доцент, Я. М. Крайник ___ _____

(посада, прізвище, ім'я, по батькові) (підпис)

Завдання прийнято до виконання __Шевченко Олександр Вікторович_ _____

(прізвище, ім'я, по батькові студента) (підпис)

Дата видачі завдання «____» _____ 20_____ р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: Апаратно-програмний комплекс для людей з вадами дихальної системи

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КР	21.10.2021	02.11.2021	Виконано
2	Огляд літератури за темою роботи	03.11.2021	30.11.2021	Виконано
3	Складання календарного плану КР	01.12.2022	15.12.2021	Виконано
4	Аналіз предметної області	10.01.2022	23.01.1021	Виконано
5	Розробка проектних рішень	24.01.2022	10.03.2022	Виконано
6	Моделювання та конструювання додатку	10.03.2022	21.04.2022	Виконано
7	Перевірка працездатності, тестування розробленого додатку, аналіз результатів тестування	22.04.2022	22.05.2022	Виконано
8	Відгук керівника КР	23.05.2022	01.06.2022	Виконано
9	Оформлення КР та презентації	02.06.2022	09.06.2022	Виконано
10	Попередній захист	10.06.2022	11.06.2022	Виконано
11	Рецензування	14.06.2022	18.06.2022	Виконано
12	Завершення оформлення КР та презентації	18.06.2022	24.06.2022	Виконано
13	Захист кваліфікаційної роботи	27.06.2022	27.06.2022	Виконано

Розробив студент_____Шевченко Олександр Вікторович_____ _____

(прізвище, ім’я, по батькові) (підпис)

«__» _____ 2022 р.

Керівник роботи ___канд. тех. наук, доцент, Я. М. Крайник____ _____

(посада, прізвище, ім’я, по батькові) (підпис)

«__» _____ 2022 р.

АНОТАЦІЯ

до кваліфікаційної роботи

«Система дослідження та моделювання роботи бібліотек стиснення»

Студент: Шевченко Олександр Вікторович

Керівник: канд. тех. наук, доцент Крайник Я. М.

Кваліфікаційна робота присвячена розробці системи дослідження та моделювання роботи бібліотек стиснення. Актуальність цієї роботи обумовлена малою кількістю на ринку веб систем для порівняння та дослідження бібліотек стиснення файлів. Метою кваліфікаційної роботи є створення системи дослідження та моделювання роботи бібліотек стиснення.

Практичним значенням результатів дослідження та розробки БР є можливість створеним сервісом отримувати точні дані по бібліотекам стиснення для конкретних файлів та отримання найкращого результату для подальшого збереження даних.

Пояснювальна записка кваліфікаційної роботи складається зі вступу, трьох розділів, висновків та чотирьох додатків.

У вступі визначається актуальність теми, сформульовані мета, об'єкт, предмет та завдання роботи.

У першому розділі аналізуються наявні наукові та комерційні розробки, види бібліотек стиснення. Визначено основні характеристики та обрано основні компоненти для створення системи моделювання.

В другому розділі виконується вибір технологій та представлення алгоритму роботи модулю для стиснення.

В третьому розділі відбувається реалізація системи дослідження та моделювання роботи бібліотек стиснення. Також проводиться тестування та демонстрація робочого процесу. Розроблюється клієнтське та серверне програмне забезпечення для представлення та обробки зібраних даних.

В додатку А наводиться код для серверної частини роутеру шляхів.

В додатку Б наводиться код для серверного застосунку класу стиснення файлів.

В додатку В наводиться код для клієнтського застосунку.

В додатку Г наводиться результат роботи додатку.

Кваліфікаційна робота складається з 95 сторінок, 34 рисунки, 4 додатків та 40 використаних джерел посилання.

Ключові слова: стиснення, бібліотеки, JavaScript, фреймворк, Express, Vue.js

ABSTRACT

of the Bachelor's Thesis

«The system for investigation and modeling of compression libraries»

Student: Shevchenko Alexander Viktorovich

Scientific Advisor: Ph.D. in Computer Systems and Components, Associate Professor Y. M. K

Qualification work is devoted to the development of a system of research and modeling of compression libraries. The relevance of this work is due to the small number of web systems on the market to compare and study file compression libraries. The purpose of the qualification work is to create a system of research and modeling of compression libraries.

The practical significance of the results of research and development of BR is the ability of the created service to obtain accurate data on compression libraries for specific files and get the best result for further storage of data.

The explanatory note of the qualification work consists of an introduction, three sections, conclusions and four appendices.

The introduction determines the relevance of the topic, formulates the purpose, object, subject and objectives of the work.

The first section analyzes the existing scientific and commercial developments, types of compression libraries. The main characteristics are determined and the main components for creating a modeling system are selected.

In the second section the choice of technologies and representation of algorithm of work of the module for compression is carried out.

In the third section, the system of research and modeling of compression libraries is implemented. Workflow testing and demonstration is also performed. Client and server software for presentation and processing of the collected data is developed.

Appendix A provides the code for the server side of the path router.

Appendix B provides the code for the server application of the file compression class.

Appendix B provides the code for the client application.

Appendix D provides the result of the application.

The qualification work consists of 95 pages, 34 figures, 4 appendices and 40 used reference sources.

Keywords: compression, libraries, JavaScript, framework, Express, Vue.js

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API	–	Application Programming Interface
DOM	–	Document Object Model
FLAC	–	Free Lossless Audio Codec
GPL	–	General Public License
I/O	–	input/output
JS	–	JavaScript
LZ77	–	Lempel-Ziv 77
LZMA	–	Lempel-Ziv-Markov chain-Algorithm
LZRW	–	Lempel–Ziv Ross Williams
MIT	–	Massachusetts Institute of Technology
PNG	–	Portable Network Graphics
SEO	–	Search Engine Optimization
SPA	–	Single-page application
UML	–	Unified Modeling Language
USB	–	Universal Serial Bus
UX	–	User Experience

ВСТУП

У сучасному світі об’єм даних зростає з великою швидкістю. Забезпечення ефективного збереження даних, дає змогу використовувати та сортувати великі обсяги файлів продуктивніше та дає можливість отримати більшу кількість файлів у той самій кількості пам’яті. Саме тому методи стиснення файлів є невідкладною стороною для досягнення результативності в роботі з даними.

В сучасному світі Інтернет-технології стали невід’ємною частиною майже всіх видів діяльності. Майже кожне підприємство, організація, навчальний заклад та інші мають свої корпоративні вебзастосуноки. Постійний трафік данних дає велике навантаження на сервери, особливо коли завантажуеться великі файли. Тому багато програм використовують попереднє стиснення даних, а після відправки виконують розархівування файлів.

Мета: розроблення системи моделювання роботи бібліотек стиснення для визначення кращих алгоритмів стиснення.

Об’єкт: процеси стиснення інформації у комп'ютерних системах, засоби для реалізації стиснення**.**

Предмет: процеси стиснення інформації у комп'ютерних системах, засоби для реалізації стиснення

Для досягнення поставленої мети необхідно вирішити такі **завдання**:

1. Виконати аналіз алгоритмів стиснення, які використовуються у системах передачі та збереження інформації та обрати найбільш популярні алгоритми, які будуть включені до складу реалізованої системи.
2. Виконати аналіз та обрати програмні засоби, які необхідні для реалізації системи моделювання.
3. Виконати моделювання розробки системи засобами UML.

4. Розробити систему моделювання бібліотек стиснення з використанням обраних програмних засобів та UML-моделювання.
5. Продемонструвати роботу системи з різними типами даних та з різними алгоритмами.

Практичне значення результатів проходження ПП: розроблена система наддасть можливість визначати, який алгоритм стиснення краще працює для конкретного набору даних та на основі цього надасть можливість прийняти рішення щодо того, яким чином такі дані можуть бути стиснені для передачі та збереження, а також для інших варіантів використання у комп'ютерних системах.

Апробація: результати дипломної роботи були представлені у форматі тез на Ольвійському форумі, який проводився з 23.06.2022 по 26.06.2022

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД ЛІТЕРАТУРИ ТА ПАТЕНТНОЇ ІНФОРМАЦІЇ

Стиснення файлу – це зменшення його розміру за збереження вихідних даних. У цьому випадку файл займає менше місця на пристрої, що полегшує його зберігання та передачу через інтернет або іншим способом. Важливо відзначити, що стиск не безмежний і зазвичай ділиться на два основні типи: з втратами і без втрат.

1.1 Переваги стиснення

Починаючи з 1970-х років комп’ютерні вчені використовували математичні алгоритми для пошуку в комп’ютерному коді, щоб знайти способи зменшити розмір файлу. Відтоді постійно зростала потреба, викликана розвитком Інтернету, у створенні кращих схем стиснення та максимальному зменшенні розміру будь-якого файлу.[1]

1.1.1 Зберігання

Стиснення файлів зменшує обсяг місця, необхідного для зберігання даних. Використання стиснутих файлів може звільнити цінний простір на жорсткому диску або веб-сервері. Деякі файли, як-от файли Word, можна стиснути до 90 відсотків від початкового розміру. Інші файли, такі як файли JPEG або MP3, не можуть бути стиснуті далі, оскільки ці типи файлів уже стиснуті.

Витрати на зберігання ваших даних зменшуються за рахунок стиснення файлів для зберігання, оскільки можна зберігати більше файлів у доступному просторі, коли вони стискаються. Не потрібно буде купувати другий диск на 250 МБ, якщо у вас є 500 МБ нестиснутих даних і жорсткий диск об’ємом 250 МБ, на якому їх можна зберігати. Не потрібно буде купувати додатковий жорсткий диск, якщо стиснути файли даних до 50 відсотків їх розміру без

стиснення. Цю економію можна застосувати до витрат на підтримку інтернет-з’єднання.[2]

1.1.2 Швидкості передачі

Час, який потрібен для надсилання чогось через Інтернет, залежить від розміру переданого файлу. Стиснення файлів перед відправленням через Інтернет може значно скоротити час, який це потрібно. Для передачі стисненого текстового файлу знадобиться одна десята часу, як і для передачі того самого нестисненого файлу. Стиснення файлів також зменшує фінансові витрати на роботу мережі, оскільки для передачі даних потрібно менше обладнання та пропускної здатності.

1.1.3 Архівування та резервне копіювання

Коли зберігаються великі групи подібних файлів, наприклад група послідовних відео, їх можна об’єднати в архівні файли. Архівні файли не обов’язково стискати, але часто вони мають розмір кілька гігабайт, тому зазвичай їх стискають, щоб заощадити місце для зберігання та час передачі. Більшість програм стиснення також мають можливість архівувати файли в один великий файл. Більшість операційних систем стискають свої файли резервних копій, які є дуже великими архівними файлами, оскільки ці файли просто займають місце, поки не виникне потреба в них.

Резервне копіювання є надлишковим типом робочого навантаження, оскільки процес часто захоплює ті самі файли. Організація, яка виконує повне резервне копіювання, часто матиме приблизно однакові дані від резервної копії до резервної копії. [4]

Стиснення часто використовується для даних, доступ до яких обмежений, оскільки процес може бути інтенсивним і сповільнювати роботу системи. Однак адміністратори можуть без проблем інтегрувати стиснення у свої системи резервного копіювання.

- Стиснення даних перед створенням резервної копії має основні переваги:
- Дані займають менше місця, оскільки коефіцієнт стиснення може досягати 100:1, але часто зустрічається від 2:1 до 5:1.
- Якщо стиснення виконується на сервері перед передачею, час, необхідний для передачі даних, і загальна пропускна здатність мережі різко скорочуються.
- На стрічці стиснутий, менший образ файлової системи можна сканувати швидше, щоб досягти певного файлу, зменшуючи затримку відновлення .

1.1.4 Цифрове телебачення

Цифрове телебачення завдячує своїм поширенням технології стиснення файлів. Сигнал цифрового телебачення займає набагато меншу смугу пропускання, ніж аналоговий сигнал, оскільки цифрові файли стискаються перед передачею [5]. На додаток до звичайного стиснення файлів, це стиснення відео передає лише частини наступного кадру, що змінюється, що може різко зменшити розмір файлу.

1.2 Недоліки стиснення

Стиснення є математично інтенсивним процесом. Тому це може зайняти багато часу, коли у нас є велика кількість файлів для стиснення. Крім того, рівень стиснення змінюється відповідно до використовуваного алгоритму, як наслідок, час стиснення також змінюється.

Алгоритми стиснення даних є системним процесом і забирають цінні ресурси під час процесу стиснення даних[4]. Іноді з’являється помилка «недолік пам’яті» в деяких системах, які мають менше ресурсів, ніж вимагає алгоритм стиснення даних.

Щоб розпакувати завантажений стиснутий файл, потрібна відповідна програма, щоб розпакувати його. Часто користувач не знаходить таку програму в системі чи Інтернеті. Окрім того для розпакування зазвичай потрібно додаткове місце.

Також якщо помістити зашифровані файли в уже відформатований архів, є велика ймовірність, що вони стануть незашифрованими, коли їх розпакують. Цей недолік кодування може призвести до ненавмисного розкриття особистої інформації та конфіденційних даних.

При завантажені стислих файлів через Інтернет або у вкладеннях електронної пошти, комп’ютер не скануватиме їх.[6] Вони можуть переносити шкідливі віруси, так як антивіруси не можуть провести їх аналіз. Тому стисненні формати можуть зашкодити вашому комп’ютеру після розпакування файлів.

1.3 Стиснення з втратам

Стиснення з втратами зменшує розмір файлу, видаляючи непотрібні біти інформації. Найчастіше зустрічається у форматах зображень, відео та аудіо, де немає потреби в ідеальному представленні вихідного медіа. MP3 і JPEG - два популярні приклади. Але стиск із втратами не зовсім підходить для файлів, де важлива вся інформація [8]. Наприклад, у текстовому файлі або електронній таблиці воно призведе до викривленого висновку.

MP3 містить не всю аудіоінформацію з оригінального запису. Цей формат відкидає деякі звуки, які люди не чують. Можна помітити, що вони зникли тільки на професійному обладнанні з дуже високою якістю звуку, тому для звичайного використання видалення цієї інформації дозволить зменшити розмір файлу практично без недоліків.

При збереженні у форматі із втратами, часто можна встановити рівень якості. Наприклад, багато графічних редакторів мають повзунок для вибору якості JPEG від 0 до 100. Економія на рівні 90 або 80 відсотків призводить до

невеликого зменшення розміру файлу з незначною візуальною різницею. Але збереження в поганій якості або повторне збереження одного файлу у форматі з втратами погіршить його.

Рисунок 1.1 – Зображення стиснене на 10% та 90%

Стиск із втратами відмінно підходить для більшості медіафайлів. Це дуже важливо для таких компаній як Spotify та Netflix, які постійно транслюють великі обсяги інформації. Максимальне зменшення розміру файлу при збереженні якості робить їхню роботу більш ефективною.

1.4 Стиснення без втратами

Стиснення без втрат дозволяє зменшити розмір файлу так, щоб можна було відновити початкову якість. На відміну від стиснення із втратами, цей спосіб не видаляє жодної інформації. Абстрактний приклад, на рис. 2.2.1 10 цеглин: дві сині, п'ять жовтих і три червоні.[6]

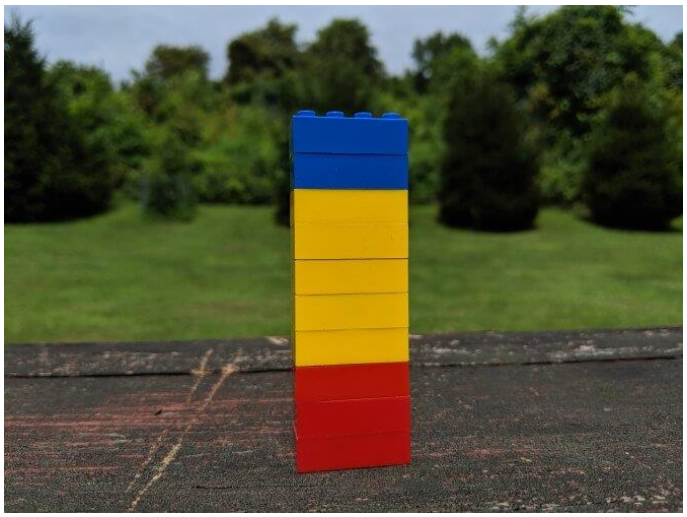

Рисунок 1.2 – Приклад файлу до стиснення

Замість показувати всі 10 блоків, ми можемо видалити всі цеглини одного кольору, крім одного. Використовуючи цифри, щоб показати, скільки цегли кожного кольору було, ми представляємо ті ж дані, використовуючи набагато менше цегли — три замість десяти.

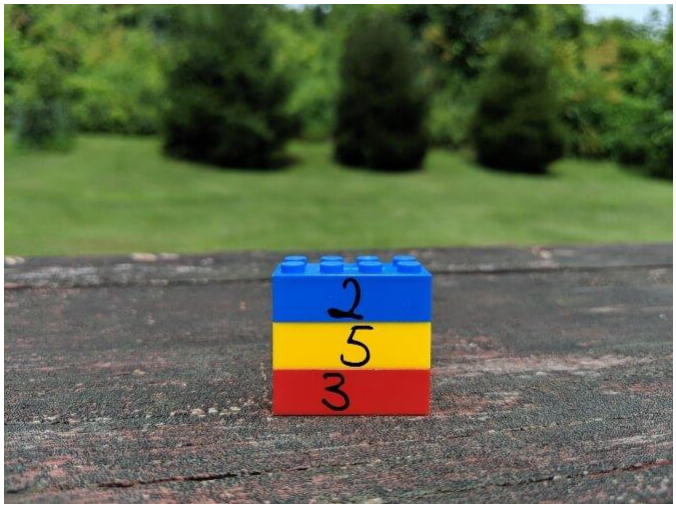

Рисунок 1.3 – Приклад файлу після стиснення

Це проста ілюстрація того, як здійснити стиск без втрат. Така сама інформація зберігається більш ефективним способом. Розглянемо реальний файл: mmmmmuuuuuuuuoooooooooooo. Його можна стиснути до більш короткої форми: m5u7o12. Це дозволяє використовувати 7 символів замість 24 для представлення тих самих даних.

ZIP-файли – популярний приклад стиснення без втрат. Зберігати інформацію у вигляді ZIP-файлів більш ефективно, при цьому коли розпаковується архів, там є вся оригінальна інформація. Це актуально для файлів, тому що після стиснення з втратами розпакована версія буде пошкоджена і непридатна для використання.[9]

Інші розповсюджені формати без втрат — PNG для зображень та FLAC для аудіо. Формати відео без втрат трапляються рідко, тому що вони займають багато місця.

1.5 Введення в формати і бібліотеки стиснення

Для огляду та заміру швидкодії було використано наступні характеристики ПК:

Список розглянутих бібліотек:

- LZ4
- QuickLZ
- Snappy
- Compress
- Zlib
- Gzip
- Bzip2
- Rar
- 7z
- Xz
- LZJB

1.5.1 LZ4 і FastLZ

LZ4 - алгоритм стиснення даних без втрат, орієнтований на високу швидкість стиснення та розпакування. Також може добре підтримувати

багатопотокове середовище і отримувати вищу швидкість стиснення та розпакування

Compression / Thread	Ratio	Speed	Decoding
-1 / 4 threads	2.195	880 MB/s	3.80 GB/s
-1 / 1 thread	2.195	333 MB/s	1.76 GB/s
-9 / 4 threads	2.716	58.0 MB/s	4.38 GB/s
-9 / 1 thread	2.716	23.1 MB/s	2.13 GB/s

Рисунок 1.4 – Швидкодія бібліотеки LZ4

LZ4 використовує лише етап узгодження словника (LZ77), і на відміну від інших поширених алгоритмів стиснення не поєднує його з етапом ентропійного кодування (наприклад , кодування Хаффмана в DEFLATE)

Алгоритм LZ4 представляє дані у вигляді серії послідовностей. Кожна послідовність починається з однобайтового маркера, який розбивається на два 4-бітові поля. Перше поле представляє кількість літеральних байтів, які потрібно скопіювати у вихідні дані. Друге поле представляє кількість байтів для копіювання з уже декодованого вихідного буфера (при цьому 0 представляє мінімальну довжину відповідності 4 байти). Значення 15 в будь-якому з бітових полів вказує на те, що довжина більша і є додатковий байт даних, який потрібно додати до довжини. Значення 255 у цих додаткових байтах вказує на те, що потрібно додати ще один байт.[12]

Отже, довільні довжини представлені серією додаткових байтів, що містять значення 255. Рядок літералів йде після маркера та будь-яких додаткових байтів, необхідних для вказівки довжини рядка. За цим слідує зміщення, яке вказує, як далеко назад у вихідному буфері почати копіювання. Додаткові байти (якщо такі є) довжини відповідності приходять в кінці послідовності

FastLZ - це реалізація алгоритму Lempel-Ziv 77 (LZ77) стиснення даних без втрат. Він підходить для стиснення рядів тексту/абзаців, послідовностей

необроблених піксельних даних або будь-яких інших блоків даних з великою кількістю повторів. Він не призначений для використання для зображень, відео та інших форматів даних, які зазвичай вже в оптимальному стиснутому вигляді.

У центрі уваги FastLZ є дуже швидке стиснення та декомпресія, роблячи це за рахунок коефіцієнта стиснення

FastLZ підтримує будь-який стандартний компілятор ANSI C/C90, включаючи такі популярні, як GCC, Clang, Intel C++ Compiler, Visual Studio і навіть Tiny CC. FastLZ добре працює на багатьох архітектурах (32-розрядні та 64-розрядні, великі і маленькі байти), від Intel/AMD, ARM та MIPS

1.5.2 QuickLZ

QuickLZ є найшвидшою у світі бібліотекою стиснення, яка досягає 308 МБ/с на ядро. Бібліотека претендує на звання найшвидшого алгоритму стиснення у світі, він все ще має розрив з LZ4, особливо швидкість декомпресії.[13]

Library	Level	Compressed size	Compression Mbyte/s	Decompression Mbyte/s
QuickLZ C 1.5.0	1	47.9%	308	358
QuickLZ C 1.5.0	2	42.3%	131	309
QuickLZ C 1.5.0	3	40.0%	31	516
QuickLZ C# 1.5.0	1	47.9%	133	132
QuickLZ Java 1.5.0	1	47.9%	127	95

Рисунок 1.5 – Швидкодія бібліотек

1.5.3 Snappy

Snappy розроблений Google на основі ідей LZ77 і відкритий у 2011 році. Швидкість стиснення становить 250 МБ/с, а швидкість декомпресії становить 500 МБ/с за допомогою одного потоку. Бібліотека має швидкість, а ступінь стиснення не високу.

Швидке кодування орієнтоване не на біти, а на байти (з потоку випускаються або споживаються лише цілі байти). Формат не використовує ентропійний кодер , як -от дерево Хаффмана або арифметичний кодер .

Snappy широко використовується в таких проєктах Google, як Bigtable , MapReduce , а також у стисканні даних для внутрішніх систем Google RPC . Його можна використовувати в проєктах з відкритим кодом, таких як MariaDB ColumnStore[8], Cassandra , Couchbase , Hadoop , LevelDB , MongoDB , RocksDB , Lucene , Spark і InfluxDB .

Декомпресія перевіряється на виявлення будь-яких помилок у стисненому потоці. Snappy не використовує вбудований асемблер і є портативним. Коефіцієнт стиснення на 20–100% нижчий, ніж у gzip.[14]

1.5.4 Compress

Compress - це програма для стиснення оболонки Unix на основі алгоритму стиснення LZW. У порівнянні з більш сучасними утилітами стиснення, такими як gzip і bzip2 , стиснення виконується швидше і з меншим використанням пам'яті, за рахунок значно нижчого коефіцієнта стиснення .

Алгоритм LZW, що використовується для стиснення, дозволяє досягти одного з найкращих ступенів стиснення серед інших існуючих методів стиснення графічних даних, за повної відсутності втрат або спотворень у вихідних файлах. В даний час використовується у файлах формату TIFF, PDF, GIF, PostScript та інших, а також частково в багатьох популярних програмах стиснення даних[15]

1.5.5 ZIP

ZIP – це формат архівного файлу, який підтримує стиснення даних без втрат . ZIP-файл може містити один або кілька файлів або каталогів, які могли бути стиснуті. Формат файлу ZIP дозволяє використовувати ряд алгоритмів стиснення.

zlib розроблено як безкоштовна , універсальна, юридично необтяжена, тобто не охоплена жодними патентами, бібліотека стиснення даних без втрат для використання практично на будь-якому комп’ютерному обладнанні та операційній системі. Формат даних zlib сам по собі переносний на різні платформи. На відміну від методу стиснення LZW, який використовується в Unix compress і у форматі зображення GIF, метод стиснення, який зараз використовується в zlib, по суті, ніколи не розширює дані.

Швидкість компресії 55 Mbyte/s, декомпресії 234 Mbyte/s [16].

Gzip – це бібліотека, яка використовується для стиснення та розпакування файлів заснований на алгоритмі DEFLATE, який є комбінацією LZ77 і кодування Хаффмана. DEFLATE був задуманий як заміна LZW та інших алгоритмів стиснення даних, обтяжених патентами , які на той час обмежували зручність стиснення та інших популярних архіваторів.

Формат gzip використовується для стиснення HTTP, методики, яка використовується для прискорення надсилання HTML та іншого вмісту у всесвітній мережі. Це один із трьох стандартних форматів для стиснення HTTP, як зазначено в RFC 2616.

bzip2 — це безкоштовна бібліотека для стиснення файлів з відкритим вихідним кодом, яка використовує алгоритм Берроуза–Уілера. Вона стискає лише окремі файли.

Bzip2 використовує кілька шарів методів стиснення, накладених один на одного, які відбуваються в такому порядку під час стиснення та у зворотному порядку під час декомпресії [17].

1.5.6 RAR

RAR — це власний формат архівного файлу, який підтримує стиснення даних, виправлення помилок і розширення файлів. Він був розроблений у 1993 році російським інженером -програмістом Євгеном Рошалем, а програмне забезпечення ліцензовано win.rar GmbH.

Розширення назв файлів, які використовуються в RAR, .rarпризначені для набору томів даних і .rev для набору томів відновлення. Попередні версії RAR розділяли великі архіви на кілька менших файлів, створюючи «багатотомний архів». Числа були використані в розширеннях файлів менших розмірів, щоб зберегти їх у належній послідовності. Перший файл використовував розширення .rar, потім .r00 для другого, а потім .r01, .r02 і т.д.

Файли RAR можна створювати лише за допомогою комерційного програмного забезпечення WinRAR

Безперервний архів - це архів RAR, упакований спеціальним способом, при якому всі файли, що стискаються, розглядаються як один послідовний потік даних. Безперервна архівація підтримується лише у форматі RAR, для формату ZIP такого типу упаковки немає. Метод стиснення для архівів RAR – звичайний або безперервний – вибирається користувачем.

Швидкість стиснення від обраних умов стиснення[9]

1.5.7 7z

7z — це формат стисненого архівного файлу, який підтримує кілька різних алгоритмів стиснення , шифрування та попередньої обробки даних. Формат 7z спочатку з'явився як реалізований архіватором 7-Zip .

На даний момент визначено такі методи стиснення:

LZMA – варіант алгоритму LZ77, що використовує ковзний словник довжиною до 4 ГБ для видалення дублікатів рядка. За етапом LZ слід ентропійне кодування з використанням кодера діапазонів на основі ланцюга Маркова та бінарних дерев.

LZMA2 – модифікована версія LZMA, що забезпечує кращу підтримку багатопоточності та менше розширення даних, що не стискаються. [5]

PPMd – PPMdH (PPMII/cPPMII) Дмитра Шкаріна 2002 року з невеликими змінами: PPMII є покращеною версією алгоритму стиснення PPM 1984 року (прогноз за частковим збігом) [18].

DEFLATE – Стандартний алгоритм на основі 32 КБ LZ77 і кодування Хаффмана. Deflate можна знайти в кількох форматах файлів, включаючи ZIP, gzip, PNG і PDF. 7-Zip містить з нуля кодер DEFLATE, який часто перевершує стандартну версію zlib де-факто за розміром стиснення, але за рахунок використання ЦП.

Набір інструментів для повторного стиснення під назвою AdvanceCOMP містить копію кодера DEFLATE з реалізації 7-Zip; ці утиліти часто можна використовувати для додаткового стиснення розмірів існуючих файлів gzip , ZIP , PNG або MNG .

Формат 7z підтримує шифрування за алгоритмом AES з 256-бітним ключем. Ключ генерується з наданої користувачем парольної фрази за допомогою алгоритму, заснованого на хеш-функції SHA-256 . SHA-256 виконується 218 (262144) разів, що викликає значну затримку на повільних ПК перед початком стиснення або вилучення [18]. Ця техніка називається розтягуванням ключів і використовується, щоб ускладнити пошук парольної фрази методом грубої сили . Поточні атаки на основі графічного процесора та користувацькі апаратні атаки обмежують ефективність цього конкретного методу розтягування клавіш тому важливо вибрати надійний пароль. Формат 7z надає можливість шифрувати імена файлів архіву 7z.

1.5.8 Xz

Xz – це формат стиснення даних XZ Utils. Конструкція базується на форматах даних compress , gzip і bzip2 . Таким чином, це не архівний файл , тому як вміст підтримує лише один файл.

Файли ідентифікуються розширенням назви файлу .xz, стиснені архіви tar мають розширення імені файлу .txz або .tar.xz .

Формат підтримує кілька алгоритмів стиснення (так званих фільтрів) з розширюваного списку, які також можна комбінувати один з одним. Еталонна

реалізація переважно використовує алгоритм Лемпеля-Зіва-Маркова (LZMA/LZMA2).

Вміст захищено контрольними сумами , які також підтримують ряд алгоритмів контрольної суми з розширюваного списку. Потік даних можна стиснути в кілька незалежних блоків і записати у файл. Для окремих блоків також можна використовувати різні алгоритми стиснення. Як і у файлах gz і bz2, кілька файлів xz можна об’єднати в дійсний новий за допомогою конкатенації, а до кратних чотирьох можна додати нульові байти (дійсний файл xz має кількість байтів, що ділиться на чотири), наприклад. В. для досягнення певних розмірів файлів. Формат підтримує потокову передачу , тому його можна обробляти через канали [19] .

1.5.9 LZJB

LZJB - алгоритм стиснення даних без втрат , винайдений Джефом Бонвіком в 1998 для стиснення аварійних дамп програм і даних у файловій системі ZFS . Заснований на методі стиснення з використанням словника. Цей алгоритм включає безліч виправлень до алгоритму LZRW1 , який є варіантом LZRW, що є членом сімейства алгоритмів стиснення Lempel-Ziv. Цей алгоритм націлений збільшення швидкості стиснення [20].

1.6 Висновки до розділу 1

В Розділі 1 було охарактеризовано головні відмінності бібліотек стиснення. Розглянуто як новітні так і застарілі методи та алгоритми. Було виявлено, що в різних сфера доцільно використовувати окремі бібліотеки для досягнення максимальної ефективності стиснення та найбільшої швидкості.

Також бібліотеки різняться типами стиснення, а саме з втратами та без втрат інформації. З втратами найчастіше використовується для відео та звуку. Без втрат для файлів, де інформація не може бути зменшена через втрату цілісності.

У звязку з цим вирішено у системі використовувати по 4 бібліотеки з втратами і стільки же без втрат. Також, таке розділення потребує додаткової фільтрації в залежності від типу файлу.

РОЗДІЛ 2
ВИБІР ТА РОЗРОБКА ТЕХНОЛОГІЧНОЇ БАЗИ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ

У розробці програмного забезпечення , якщо потрібно вибрати мову для проєкту, спочатку потрібно поставити кілька запитань, перш ніж приймати рішення. Наприклад, що це за проєкт? масштабованість програми, складність програми, бюджет розробки, ліміт часу розробки, безпека програми, доступні ресурси тощо . Команда проєкту завжди хоче, щоб програма залишалася на тривалий час і відповідала потребам клієнта, навіть якщо бізнес зміни відбуваються пізніше.

2.1 Вибір технологій

Вибір мови для проєкту в корпоративному IT-секторі є однією з важливих проблем, з якими борються технологічні менеджери. Вибираючи мови програмування, найважливіше, що повинно враховуватись, це екосистема, спільнота та доступність для програмістів.

Багато розробників помиляються і вибирають мови програмування лише тому, що вони більш популярні, модні та круті. Якщо вибрати Lisp для проєкту лише тому, що це дуже чиста і красива функціональна мова, згодом це може обернутися неправильним рішенням. Тому добре уникнути цієї помилки.

Ось контрольний список, який можна використовувати, щоб уникнути помилки.

1. Популярність і розмір спільноти

Чим більш відомим і визнаним буде фреймворк, тим більше він буде «живим», розвиваючись і повноцінним: нові ідеї, кількість і якість плагінів тощо.

2. Філософія

Це сама суть фреймворку: це основний критерій, щоб гарантувати, що він відповідатиме вашим потребам. Інструмент, розроблений професіоналами для власних потреб, очевидно, відповідатиме вимогам інших професіоналів.

3. Стійкість

Перш ніж вибрати каркас, переконайтеся, що він зможе не відставати від вас протягом тривалого часу. Це спрощує як обслуговування, так і оновлення ваших програм.

4. Підтримка

Ще один критерій, який не варто нехтувати – це легкість пошуку відповідей на свої запитання та отримання допомоги. Визначити доступну підтримку: від видавця. Від спільноти (списки розсилки)? Від сервісних компаній (розробка, підтримка , навчання)?

5. Техніка

Щоб не потрапити в лабіринт, завжди краще вибрати сумісне рішення; такий, який поважає передові практики з точки зору розробки (шаблони проектування).

6. Безпека

Будь-який застосунок потенційно вразливий. Щоб мінімізувати ризик, завжди краще вибрати фреймворк, здатний забезпечити функції безпеки

7. Документація

Абсолютно необхідно оцінити характер, обсяг і якість існуючої літератури про фреймворк: добре задокументований інструмент і легший у використанні, і більш оновлений.

8. Ліцензія

Ліцензії важливі просто тому, що вони можуть мати значний вплив на ваші програми. Наприклад, програма, розроблена з використанням ліцензованої GPL фреймворка, обов'язково підпадає під дію GPL. З іншого боку, це не стосується фреймворку, ліцензованого MIT.

9. Наявність ресурсів на ринку [21]

Щоб на етапі розробки або в довгостроковій перспективі можно було знайти технічну команду для обслуговування та оновлення. Треба, переконайтися, що навички, необхідні для інструменту, який використовується, доступні на відкритому ринку.

Згідно з опитуванням розробників Stack Overflow за 2022 рік, JavaScript є найпоширенішою мовою у світі (рис. 2.1).

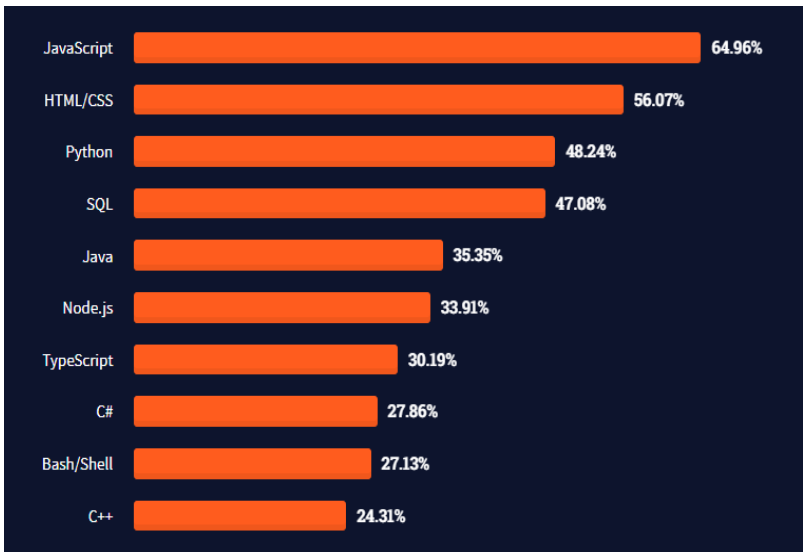

Рисунок 2.1 – Найпопулярніші мови програмування 2022

З розробкою численних фреймворків на основі різних мов програмування розробка веб-додатків стала значно простішою. Веб-фреймворки спеціально розроблені з попередньо вбудованими функціями та функціями, щоб забезпечити надзвичайно високу ефективність та продуктивність під час створення програм. Завантажувальна здатність веб-сайтів, розроблених із відповідною структурою, значно підвищується.

Більшість фреймворкових методологій включають найкращі методи розробки програмного забезпечення. Численні веб-фреймворки мають попередньо вбудований або зовнішній інтегрований механізм тестування, який тут же перевіряє код, зменшуючи кількість помилок в остаточному коді.

Популярність фреймворків в опитуванні Stack Overflow за 2022 рік наступна (рис. 2.2)

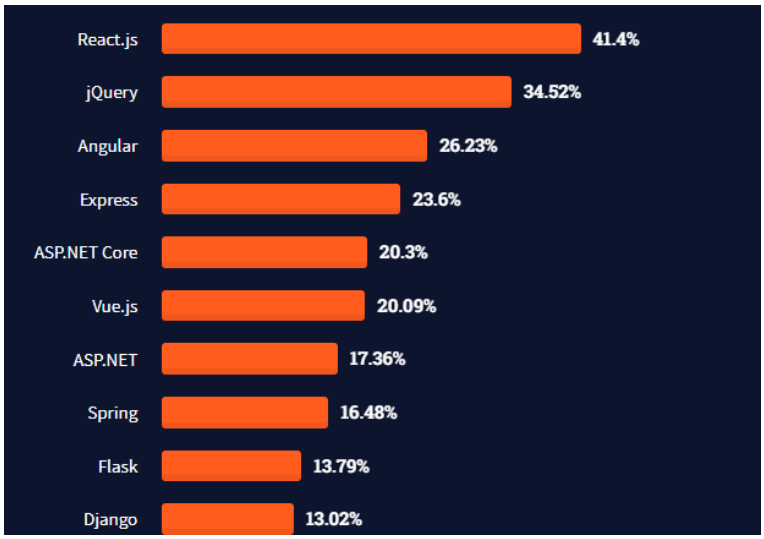

Рисунок 2.2 – Найпопулярніші фреймворки 2022

Також, фреймворки включають сотні готових компонентів, створених і регулярно оновлюваних спільнотою розробників. Ця величезна підтримка співтовариства програмістів гарантує, що ваш проєкт не застрягне між ними, і може бути розроблено найкраще можливе рішення для бізнес-задач.

Більшість веб-фреймворків постачаються з попередньо написаними шаблонами та об’єктами, які можна використовувати для виконання зайвих завдань програмування. Ці інструменти економлять час розробників і дозволяють їм зосередитися на основній частині програмування, забезпечуючи швидкі та більш продуктивні результати.

На опитуванні Stack Overflow за 2022 рік одними з найбільш зручними фреймворками стали Vue.js (скоріш зручний: 64,41%, не зручний 35,59%), фреймворк для створення Front частини сайту та Express (скоріш зручний: 62,07%, не зручний 37,93%), написані на мові JavaScript (рис. 2.3)

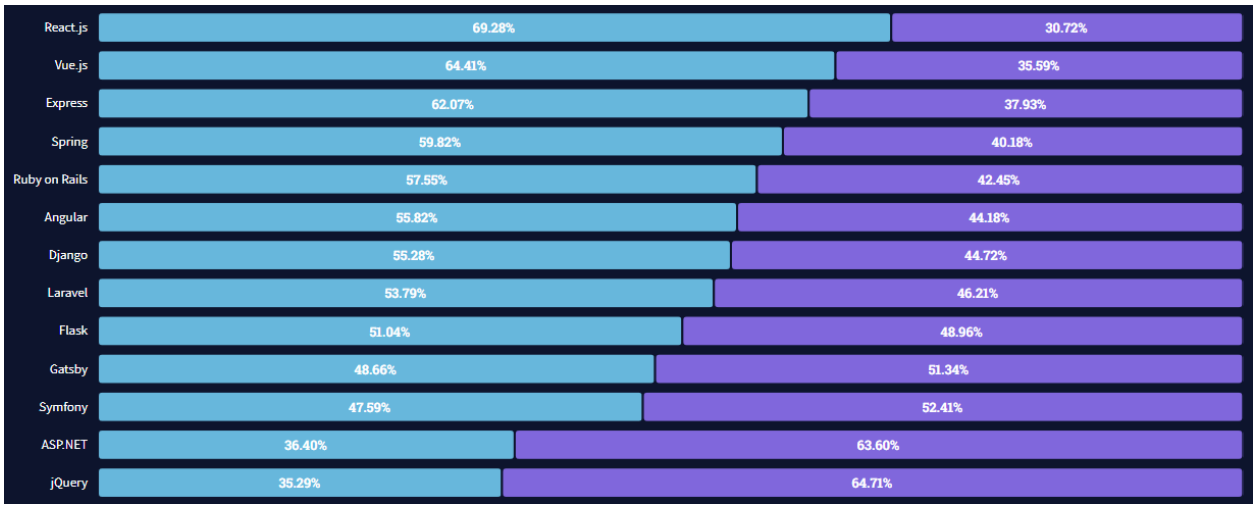

Рисунок 2.3 –Рейтинг зручності фреймворків 2022

2.2 Історія Javascript

Початкова мета цієї нової мови програмування полягала в тому, щоб зробити Інтернет повноцінною платформою додатків, що означає, що JavaScript працюватиме як на клієнті, так і на сервері. Однак це не було повним успіхом, оскільки знадобилося багато років, щоб його серйозно сприймали як мову бекенда. Але він швидко процвітав у інтерфейсі, ставши стандартною мовою програмування в Інтернеті. З часом його прийняли всі веб-браузери, від Explorer до Firefox і Chrome.

Хоча JavaScript народився поспішно, і різні примхи зашкодили б засвоєнню мови в перші роки його існування. Тим не менш, деякі з його потужних функцій були частиною його з самого початку. Вони визначать JS і дозволять йому розвиватися.

Наступним великим кроком після публічного релізу стала стандартизація ECMA, яка стала «конвенцією» за JavaScript. Це привело мову до ширшої аудиторії та дозволило іншим потенційним реалізаторам, таким як Microsoft, мати право вплинути на її еволюцію.

З появою JavaScript стало можливим задовольнити різну аудиторію:

- авторів компонентів і професіоналів корпоративного рівня з Java
- дизайнерів з JavaScript.

Назва Java Script походить від спроби подолати хвилю популярності Java та прискорити прийняття. Сьогодні вже не знайти багато спільного між обома мовами.

Як вже згадувава, це також принесло динамічні функції в Інтернет. Що це саме означає? Ось кілька речей, які можна бачити, які є результатом JavaScript:

- Автозавершення
- Завантаження нового вмісту або даних на сторінку без перезавантаження сторінки
- Ефекти прокрутки та спадні меню
- Анімація елементів сторінки, таких як вицвітання, зміна розміру або переміщення
- Відтворення аудіо та відео
- Перевірка введення з форм

JavaScript — це мова сценаріїв, яка вставляється безпосередньо в HTML-код сторінки. Це єдина мова програмування такого роду, яку можуть зрозуміти веб-браузери[22]. Браузери можуть читати Javascript, інтерпретувати його, а потім запускати програму, створюючи потужний функціонал на стороні клієнта.

Він отримав такий статус, оскільки є відкритим, стандартизованим. JavaScript добре підходить для Інтернету завдяки своїй динамічності та тісній інтеграції з DOM.

JavaScript також сумісний з іншими мовами. Це надзвичайно важливо, оскільки веб-сервери працюють на різних мовах, будь то PHP, Python, Ruby, Java або .NET. Оскільки JavaScript, який запускається у браузері, на 100% відокремлений від того, як генеруються веб-сторінки HTML, користувачі завжди матимуть такий самий багатий досвід, як працює JS, незалежно від мови сервера.

ECMAScript 6 це оновлення додало новий синтаксис для написання складніших програм і багато інших функцій, які визначать наступну еру JavaScript.

Довгий час веб-сайти в основному працювали на базі PHP CMS, наприклад WordPress. Більшу частину логіки обробляв серверний код. Але все змінюється — «статичні» сайти повертаються. Однак вони не схожі на статично згенеровані веб-сайти 90-х, про які я згадував раніше [23].

Сучасні браузери тепер мають можливість зробити їх інтерактивними та повністю динамічними. Однак особливістю, яку вони поділяють зі своїми предками, є абстракція розробки бекенда. Логіка обробляється на стороні клієнта, безпосередньо в браузері, завдяки JavaScript.

Деякі з найвідоміших веб-додатків сьогодні створені за допомогою JS. Великі корпорації вже використовують Javascript у повну силу, такі компанії як Facebook, Gmail, Twitter та багато інших.

Наприклад, у Facebook JavaScript забезпечує оновлення статусу та більшість інтерактивності користувачів. Без нього він не мав би особливої привабливості.

Ці технічні гіганти насправді створили власні рамки JavaScript, і тепер вони дозволяють тисячам розробників створювати власні веб-програми. Наприклад Angular, який підтримується Google, і React, який підтримує FB [23]. Також один із нових Vue, який, навіть не підтримується технічним центром, завершує тріаду важливих фреймворків JS.

Рисунок 2.4 – Логотипи найбільших JavaScript фреймворків

Окрім скорочення часу та зусиль, необхідних для розробки сайтів і програм на основі JS, ці фреймворки допомогли сформувати новий веб-досвід. Наприклад, односторінкові програми (SPA). SPA – це веб-сайт, який взаємодіє з користувачами, динамічно переписуючи сторінку в браузері, а не завантажуючи цілі нові сторінки з сервера, завдяки чому вони поводяться більше як настільні програми.

Розробники можуть використовувати JavaScript для отримання даних з інших джерел і відображення їх на власному сайті. Однією з концепцій, яка більш ніж будь-коли просувається у веб-розробці, є модульність — використання різних інструментів для виконання конкретних завдань. Що ж, тепер легко створювати такі стеки завдяки API та JavaScript[24].

2.3 JavaScript на стороні сервера

З самого початку були невдалі спроби запустити JavaScript на стороні сервера. Багато хто думав, що вона просто ніколи не стане стабільною мовою бекенда, аж до появи Node.js.

Сьогодні це середовище виконання JS є популярним інструментом для забезпечення роботи веб-серверів. Це означає, що розробники JS можуть

використовувати Node.js для написання коду як на стороні клієнта, так і на стороні сервера на JavaScript, не покладаючись на зовнішні веб-сервери.

Основна ідея Node.js: використовувати неблокуючий, керований подіями ввод-вивод, щоб залишатися легким і ефективним перед обличчям додатків реального часу з інтенсивним використанням даних, які працюють на розподілених пристроях.

У порівнянні з традиційними методами веб-обслуговування, коли кожне з’єднання (запит) породжує новий потік, займаючи системну оперативну пам’ять і в кінцевому підсумку вичерпуючи доступну кількість оперативної пам’яті, Node.js працює з одним потоком, використовуючи неблокуючий I/O викликів, що дозволяє йому підтримувати десятки тисяч одночасних з’єднань, що утримуються в циклі подій [24].

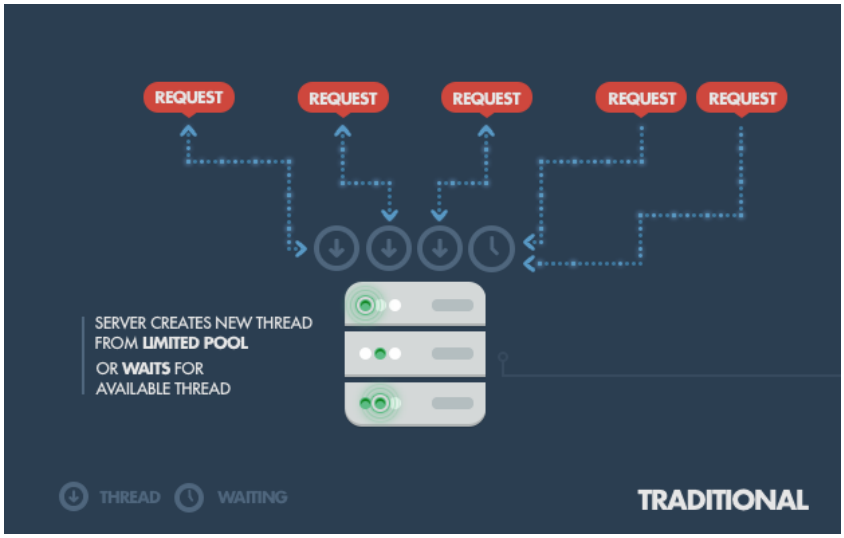

Рисунок 2.5 – Традиційна обробка процессів

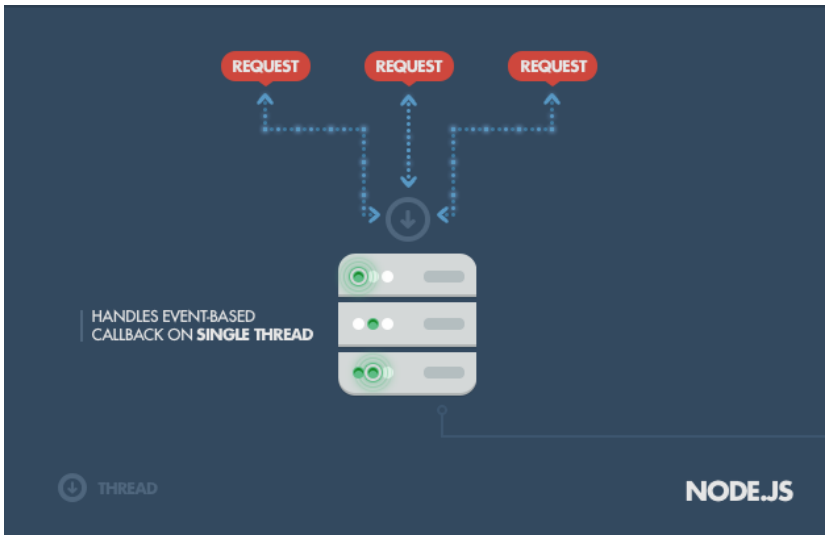

Рисунок 2.6 –Обробка процессів Node.js

2.4 Сфери застосування

Зростаюча популярність JavaScript принесла з собою багато змін, і обличчя веб-розробки сьогодні кардинально відрізняється. Те, що ми сьогодні можемо робити в Інтернеті за допомогою JavaScript, запущеного на сервері, а також у браузері, було важко уявити лише кілька років тому, або вони були інкапсульовані в середовищі пісочниці, як-от Flash або Java-аплети.

Зараз JavaScript має наступні сфери застосування:

1. Веб-додатки

Оскільки браузери з кожним днем постійно вдосконалюються, JavaScript набув популярності для створення надійних веб-додатків. Ми можемо зрозуміти це на прикладі Google Maps. У Картах користувачеві потрібно просто клацнути та перетягнути мишею; деталі видно лише одним клацанням миші. За цими поняттями стоїть використання JavaScript.

2. Веб-розробка

JavaScript зазвичай використовується для створення веб-сторінок. Це дозволяє нам додавати динамічну поведінку до веб-сторінки та додавати спеціальні ефекти на веб-сторінку. На веб-сайтах він в основному використовується для перевірки. JavaScript допомагає нам виконувати складні дії, а також дозволяє взаємодіяти веб-сайтів із відвідувачами. Використовуючи

JavaScript, також можна завантажувати вміст у документ без перезавантаження веб-сторінки.

3. Мобільні програми

Зараз для доступу до Інтернету широко використовуються щоденні мобільні пристрої. Використовуючи JavaScript, ми також можемо створити застосунок для невеб-контекстів. Функції та використання JavaScript роблять його потужним інструментом для створення мобільних додатків [26].

React Native — це широко використовувана платформа JavaScript для створення мобільних додатків. Використовуючи React Native, ми можемо створювати мобільні додатки для різних операційних систем. Нам не потрібно писати різні коди для операційних систем iOS та Android. Нам потрібно лише написати його один раз і запустити на різних платформах.

4. Ігри

JavaScript також використовується для створення ігор. Він має різноманітні бібліотеки та фреймворки для створення гри. Гра може бути 2D або 3D. Деякі ігрові движки JavaScript, такі як PhysicsJS, Pixi.js, допомагають нам створити веб-гру. Ми також можемо використовувати WebGL (бібліотеку веб-графіки), яка є JavaScript API для відтворення 2D та 3D зображень у браузерах.

5. Презентації

JavaScript також допомагає нам створювати презентації як веб-сайт. Бібліотеки, такі як RevealJs і BespokeJs, можна використовувати для створення веб-коди слайдів. Вони простіші у використанні, тому ми можемо легко зробити щось дивовижне за короткий час.

Reveal.js використовується для створення інтерактивних і красивих колод слайдів за допомогою HTML. Ці презентації чудово підходять для мобільних пристроїв і планшетів. Він також підтримує всі кольорові формати CSS. BespokeJS включає анімовані марковані списки, адаптивне масштабування та широкий спектр функцій.

6. Серверні програми

Велика кількість веб-додатків мають серверну частину. JavaScript використовується для створення вмісту та обробки HTTP - запитів. JavaScript також може працювати на серверах через Node.js. Node.js забезпечує середовище, що містить необхідні інструменти, необхідні для роботи JavaScript на серверах [27]..

2.5 Переваги та недоліки JavaScript

JavaScript постійно розвивається, як і його екосистема. Майбутнє починається з того, що створюється менше нових інструментів, а великі гравці стають більш зрілими та отримують широке застосування. Це вже можно спостерігати в області JS-фреймворків, де React і Vue.js беруть перевагу.

2.5.1 Переваги JavaScript

Виконання логіки на стороні клієнта прискорює роботу користувачів. З кодом, що запускається безпосередньо в браузері, потреба у викликах сервера абстрагується, отже, скорочується час завантаження. Навіть за наявності сервера той факт, що JS є асинхронним, означає, що він може спілкуватися із сервером у фоновому режимі, не перериваючи взаємодії користувача, що відбувається у інтерфейсі.

З самого початку JavaScript приніс інтерактивність інтерфейсу користувача в Інтернет. Тепер він робить те ж саме для додатків будь-якого типу, допомагаючи розробити найбільш привабливий UX. Сьогодні такі фреймворки, як Vue.js, виводять переходи й анімацію на новий рівень.

JavaScript стоїть за будь -яким хорошим адаптивним веб-дизайном. Все більше і більше розробникам доводиться адаптувати свій дизайн для різних браузерів і пристроїв. Поєднуючи HTML5, CSS3 і JavaScript, вони можуть робити це в межах однієї кодової бази.

Для розробників JS легко освоїти і швидко приступити до активної розробки. Його синтаксис простий і гнучкий для новачків. Це також спрощує розробку складних програм, дозволяючи розробникам спростити композицію програми. Численні фреймворки та пакети також певною мірою полегшують життя розробникам.

JavaScript шалено популярний. Якщо популярність не завжди дорівнює якості механізму загалом, це принаймні означає одну важливу річ: можна знайти рішення будь-якої проблеми в громаді. У веб-розробці це важлива деталь. Якщо потрібно наймати розробників, це також великий плюс, оскільки пул кандидатів величезний.

Якщо використовувати JavaScript на стороні сервера. У цьому випадку універсальність коду у вашому стеку є великою перевагою, яку слід пам'ятати. Використовуючи повний стек JavaScript, дає можливість написати веб-програму, яка безперебійно відображається як у браузері, так і на сервері

Такі інструменти, як webpack , допомагають повторно використовувати код на обох кінцях і залишатися узгодженими на всіх рівнях системи.

Функція async/await повністю змінила спосіб написання асинхронного коду, зробивши його виглядом і поведінкою трохи більше схожим на синхронний код.

Усе вищесказане робить JavaScript стек чудовим для наступних випадків використання.

Випадок використання 1: програми в режимі реального часу

Додатки для спільної роботи (Trello, Google Docs), живий чат, обмін миттєвими повідомленнями та онлайн-ігри — це приклади RTA, які отримують переваги від архітектури Node.js.

Ці програми функціонують протягом періоду часу, який користувачі сприймають як негайні й поточні. Специфікації Node.js є рішенням для низької затримки, необхідної для ефективної роботи цих програм.

Це полегшує обробку кількох клієнтських запитів, дозволяє повторно використовувати пакети бібліотечного коду, а синхронізація даних між клієнтом і сервером відбувається дуже швидко [27].

Випадок використання 2: односторінкові програми

SPA – це веб-програми, які завантажують одну сторінку HTML і динамічно оновлюють цю сторінку, коли користувач взаємодіє з програмою. Велика частина роботи виконується на стороні клієнта в JavaScript.

Незважаючи на те, що це чудова еволюція у веб-розробці, у них виникають деякі проблеми, коли справа доходить до візуалізації. Наприклад, це може негативно вплинути на вашу ефективність SEO . Рендеринг на стороні сервера в середовищі Node.js є популярним варіантом вирішення цієї проблеми.

Випадок використання 3: масштабованість

Node.js ніколи не стане більшим, ніж потрібно. Його принадність полягає в тому, що він досить мінімалістичний, щоб його можна було налаштувати залежно від випадку використання. З точки зору продуктивності, це ключ.

Навіть його назва підкреслює , що він створений для збирання кількох невеликих розподілених вузлів , які спілкуються один з одним.

Модульність Node дозволяє створювати невеликі програми без необхідності мати справу з роздутою, надмірною екосистемою. Програміст вибирає інструменти, необхідні для роботи, а потім масштабує їх за потребою в проєкті.

2.5.2 Недоліки JavaScript

Навіть до сьогодні не зрозуміло, в якій мірі пошукові системи можуть сканувати JavaScript. Незважаючи на те, що Google стверджує, що в основному це так, не варто ризикувати, якщо не хочеться зіткнутися з проблемами

сканування/SEO. Ця проблема не позбавлена рішення, оскільки є способи обробки JavaScript так, щоб сканери бачили це.

Занадто багато JavaScript. Роздуття проектів за допомогою JS принесе ведмежу послугу в довгостроковій перспективі, оскільки в кінцевому підсумку спричинить проблеми з продуктивністю. Щоб уникнути цього, потрібно вставляти JavaScript лише тоді, коли це необхідно, а не використовувати його скрізь.

Тисячі пакетів, які становлять екосистему JS, дозволяють розробникам швидко працювати, не винаходячи велосипед для кожного нового завдання. Однак вони також викликають те, що деякі називають «пеклом залежності». Потрібно навчитися боротися з цими часто необхідними залежностями, щоб вони не становили клопоту для вас і людей, які використовують ваші проекти.

2.6 TypeScript

JavaScript (також відомий як ECMAScript) почав своє життя як проста мова сценаріїв для браузерів. Очікувалося, що на момент винаходу його використовуватимуть для коротких фрагментів коду, вбудованого у веб-сторінку — написання більше кількох десятків рядків коду було б дещо незвичайним. Через це ранні веб-браузери виконували такий код досить повільно. Проте з часом JS ставав все більш популярним, і веб-розробники почали використовувати його для створення інтерактивних програм.

Розробники веб-браузерів відповіли на це збільшення використання JS, оптимізувавши свої механізми виконання (динамічна компіляція) та розширивши можливості з ним (додавання API), що, у свою чергу, змусило веб-розробників використовувати його ще більше. На сучасних веб-сайтах ваш браузер часто запускає програми, які охоплюють сотні тисяч рядків коду. Це тривалий і поступовий розвиток «мережі», починаючи з простої мережі статичних сторінок і перетворюючись на платформу для різноманітних програм усіх видів [29].

Більше того, JS став досить популярним, щоб його можна було використовувати поза контекстом браузерів, наприклад, реалізувати сервери JS за допомогою node.js. Природа JS «запускати будь-де» робить його привабливим вибором для кросплатформної розробки. Сьогодні багато розробників використовують лише JavaScript для програмування всього свого стека!

Підсумовуючи, у нас є мова, яка була розроблена для швидкого використання, а потім перетворилася на повноцінний інструмент для написання програм з мільйонами рядків. Кожна мова має свої особливості — дивацтва та несподіванки, і скромний початок JavaScript робить її багато з них.

TypeScript — це мова, яка є надмножиною JavaScript: отже синтаксис JS є законним TS. Синтаксис відноситься до того, як ми пишемо текст для формування програми [27].

TypeScript не вважає будь-який код JavaScript помилкою через його синтаксис. Це означає, що ви можете взяти будь-який робочий код JavaScript і помістити його у файл TypeScript, не турбуючись про те, як саме він написаний. Однак TypeScript є типізованим наднабором, що означає, що він додає правила щодо того, як можна використовувати різні типи значень.

Засіб перевірки типів TypeScript призначений для того, щоб дозволити коректним програмам запуститися, одночасно виявляючи якомога більше поширених помилок.

Якщо перемістити деякий код з файлу JavaScript у файл TypeScript, можно побачити помилки типу залежно від того, як написаний код. Це можуть бути проблеми з кодом або надто консервативний TypeScript. У посібнику[27] продемонстровано, як додати різний синтаксис TypeScript для усунення таких помилок.

TypeScript також є мовою програмування, яка зберігає поведінку JavaScript під час виконання. Наприклад, ділення на нуль у JavaScript створює

Infinity замість того, щоб викликати виняток під час виконання. Як принцип, TypeScript ніколи не змінює поведінку коду JavaScript під час виконання.

Це означає, що якщо ви перемістите код з JavaScript на TypeScript, він гарантовано працюватиме так само, навіть якщо TypeScript вважає, що код містить помилки типу.

Підтримка такої ж поведінки під час виконання, що й JavaScript, є основоположною обіцянкою TypeScript, оскільки це означає, що ви можете легко переходити між двома мовами, не турбуючись про незначні відмінності, які можуть призвести до припинення роботи вашої програми.

Як тільки компілятор TypeScript закінчить перевірку коду, він стирає типи, щоб створити результуючий «скомпільований» код. Це означає, що після компіляції коду звичайний JS-код не має інформації про тип.

Це також означає, що TypeScript ніколи не змінює поведінку вашої програми на основі типів, які він виводить. Суть полягає в тому, що, хоча ви можете бачити помилки типу під час компіляції, сама система типів не має жодного відношення до того, як ваша програма працює під час її запуску.

Нарешті, TypeScript не надає жодних додаткових бібліотек часу виконання. Ваші програми використовуватимуть ту саму стандартну бібліотеку (або зовнішні бібліотеки), що й програми JavaScript, тому немає додаткової специфічної для TypeScript рамки для вивчення.

2.7 Структура проєкту

Опираючись на розглянутий матеріал для побудови системи стиснення файлів. Обрано створити на JavaScript стеку. Веб застосунок поділенний на Back-end та Front-end частину.

Для Front-end частини обрано Vue.js фреймворк – це платформа JavaScript для створення інтерфейсів користувача. Він побудований на основі стандартних HTML, CSS і JavaScript і надає декларативну і компонентну

модель програмування, яка допомагає вам ефективно розробляти інтерфейси користувача, будь то прості чи складнір.

Для Back-end частини обрано Express фреймворк це мінімалістичний та гнучкий веб-фреймворк для програм Node.js [29], що надає широкий набір функцій для мобільних та веб-додатків. Маючи у своєму розпорядженні безліч службових методів HTTP та проміжних обробників, створити надійний API можна швидко та легко. Express надає тонкий шар фундаментальних функцій веб-застосунків, які не заважають працювати з давно знайомими та улюбленими функціями Node.js.

Для наочної побудови моделі работи було використано програмне забезпечення для створення блок-схем (рис. 2.6) і онлайн-схем draw.io [28]

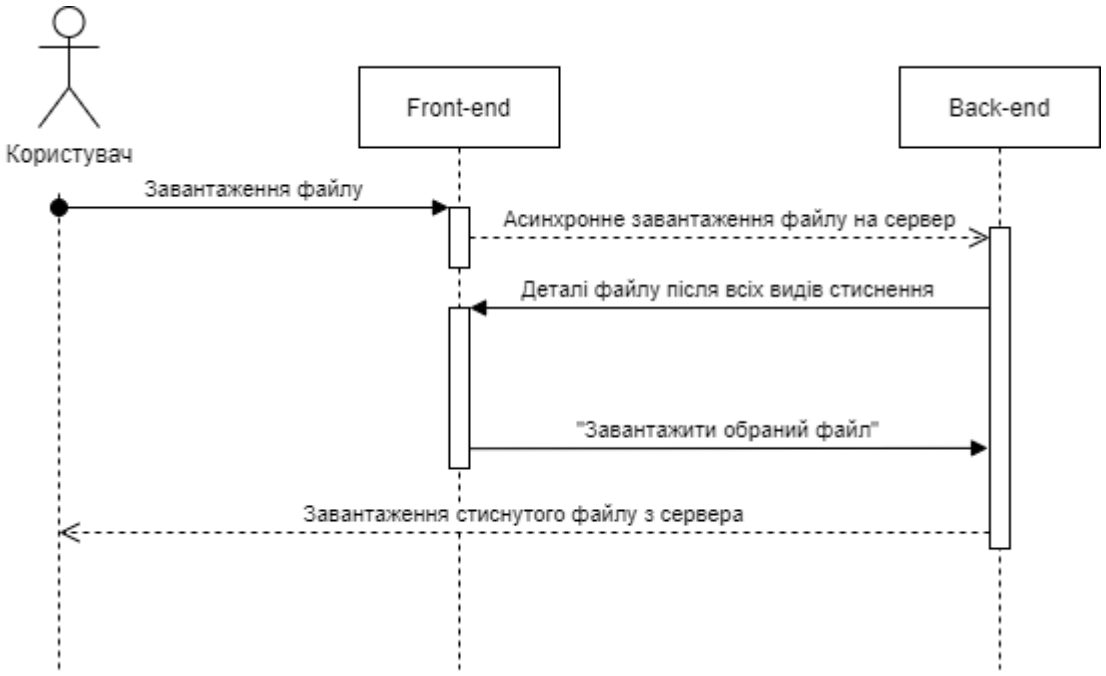

Рисунок 2.7 – Діаграма послідовностей у програмованній системі

2.8 Висновки до розділу 2

У другому розділі описано технології що використані в побудові системи стиснення. Наведенні плюси та мінуси платформи, а також аргументовано використання JavaScript стеку, що дає змогу без великих втрат часу редагувати як back-end частину так і front-end частину.

Так як обидва сервери працюють на одній базі, а саме Node.js, це дозволяє використовувати сервіс максимально швидко.

Використання Typescript дає чуйний та надійний код, якій швидко адаптується.

Також, завдяки розділенню на серверну (back-end) так клієнську (front-end) частини, є можливість легкого розширення функціоналу та інтеграції бібліотек стиснення до майбутніх проєктів.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ

Проєктування програмного забезпечення – етап життєвого циклу програмного забезпечення, під час якого досліджується структура і взаємозв'язки елементів системи, що розробляється. Результатом цього етапу є перш за все документ (набір документів), що містить в собі достатню кількість інформації для реалізації системи.

Express-застосунок найчастіше використовується як бекенд-застосунок в архітектурі клієнт-сервер, тоді як клієнт може бути написаний на Vue.js або іншому популярному зовнішньому рішенні, а сервер може бути написаний на Express. Обидві сутності призводять до архітектури клієнт-сервер (взаємозв’язок інтерфейсу та серверної частини), тоді як бекенд знадобиться для бізнес-логіки, яка повинна бути відокремленна від клієнтської частини – інакше вона була б доступна у браузері.

Однак клієнтська частина використовуется як представлення данних після обробки, а серверний застосунок за їх обробку. Ці умови не можна так легко обміняти. У той час як інтерфейсний застосунок – це те, що можна побачити у браузері, бекенд виконує бізнес-логіку, яку не слід відображати у браузері, і часто також підключається до бази даних. У нашому випадку до каталогів системи.

3.1 Реалізація Backend частини

Спочатку вирішено створювати Api частину що дає змогу потім побудувати представлення базуючись на вихідні данні Json з back-end частини.

Так як Back-end побудований по моделі API, тому використано 2 сервери на Node.js для Front та Back частини. Які будуть спілкуватися між собою за допомогою Json формату. Також, таке розділення дає змогу використовувати сервер стиснення (Backend) незалежно від Front серверу, що дозволяє отримувати доступ до технологій іншим, а так же дає змогу розширення.

API — це абревіатура від Application Programming Interface, який є програмним посередником, який дозволяє двом додаткам спілкуватися один з одним. Кожного разу, коли ви використовуєте програму, ви надсилаєте миттєве повідомлення на сторону сервера, який працює по моделі API, та отримуєте відповідно інформацію, ви використовуєте API.

API дають змогу продукту чи службі взаємодіяти з іншими продуктами та послугами, не знаючи, як вони реалізовані. Це може спростити розробку додатків, заощадивши час і гроші. Коли розробляються нові інструменти та продукти або керуються існуючі, API надають гнучкість; спрощене проектування, адміністрування та використання; і надає можливості для інновацій.

На початку проєкту створемо роутер для опису 3 шляхів нашого серверу. Головна сторінка виводить стандартну сторінку з шаблону, для того щоб зрозуміти що сервіс працює правильно.

```
router.get( path: '/', handlers: function (request: any, response: any)
    response.render('index')
});
```

Рисунок 3.1 – Головна сторінка

3.1.1 Завантаження даних на сервер

Наступна сторінка завантаження файлів на сервер. Для обробки помилок та для того щоб коректно обробляти виключення було створенно Try Catch обгортку

```
router.post( path: '/upload', handlers: async (req: any, res: any) => {
    try {
        if (!req.files) {
            res.send({
                status: false,
                message: 'No file uploaded'
            });
```

Рисунок 3.2 – Обробка помилки виконання запросу без файлу

Далі після перевірки на наявність файлу від користувача виконується його завантаженя на сервер у папку uploads. Так як на сайті не використовується авторизація то для кожного користувача зі сторони серверу створюється унікальний токен, який використовується для відокремлення користувацьких данних.

```
let file = req.files.file;
let path = './uploads/' + req.body.token + '/' + file.name;
file.mv(path);
const com = new Compressions( dirPath: "./uploads/" + req.body.token, file.name);
await com.zlibCompress();
await com.lz4Compress();
await com.lzwCompress()
await com.zipCompress()
```

Рисунок 3.3 – Завантаження та стиснення файлів.

У класі Compressions створенно окремі методи для кожного виду стиснення.

```
public async lzwCompress() {
    const lzw = require("lzw");
    const libPath = this.createLibFolder( libName: 'lzw');
    const content = await readFile(this.fullPath);
    const compressed = lzw.compress(content);
    fs.writeFileSync( file: libPath + '/' + this.fileName + '.lzw', compressed)
}

public async zipCompress() {
    var Zip = require('node-zip');
    var zip = new Zip;
    const libPath = this.createLibFolder( libName: 'zip');
    const content = await readFile(this.fullPath);
    zip.file(this.fileName, content)
    var options = {base64: false, compression: 'DEFLATE'};
    fs.writeFileSync( file: libPath + '/' + this.fileName + '.zip', zip.generate(options), options: 'binary',);
}
```

Рисунок 3.4 – Методи стиснення без втрат

Методи компресії без втрат побудовані за единим принципом (Отримання даних > Стиснення бібліотекою > Запис у новому форматі).

Після стиснення сервер сбирає розмір файлів у байтах (рис 3.7)

```
let memory = {
    gz: com.getSize( path: com.dirPath + '/gz/' + com.fileName + '.gz'),
    lz4: com.getSize( path: com.dirPath + '/lz4/' + com.fileName + '.lz4'),
    lzw: com.getSize( path: com.dirPath + '/lzw/' + com.fileName + '.lzw'),
    zip: com.getSize( path: com.dirPath + '/zip/' + com.fileName + '.zip'),
    image: null,
}
```

Рисунок 3.5 – Отримання розмірів данних

Далі виконується стиснення з втратами, якщо файл відповідає вимогам

```
if (path.endsWith('.jpg') || path.endsWith('.png') || path.endsWith('.svg') || path.endsWith('.gif')) {
    await com.imageCompress(req.body.quality);
    memory.image = com.getSize( path: com.dirPath + '/images/' + com.fileName);
}
```

Рисунок 3.6 – Перевірка для зображень

В методі стисненя зображень для кожного виду використовується своя бібліотека для отримання найкращого результату, а також використовується параметр якості, який задає користувач при завантаженні.

```
const {compress} = require('compress-images/promise');
const input = this.fullPath;
const libPath = this.createLibFolder( libName: 'images');
const output = libPath + '/';

const processImages = async () => {
    const result = await compress({
        source: input,
        destination: output,
        enginesSetup: {
            jpg: {engine: "mozjpeg", command: ["-quality", quality]},
            png: {engine: "pngquant", command: ["--quality=1-" + quality, "-o"]},
            svg: {engine: "svgo", command: "--multipass"},
            gif: {engine: "gifsicle", command: ["--colors", "64", "--use-col=web"]},
        }
    });
}
await processImages();
```

Рисунок 3.7 – Метод стиснення зображень

Після закіння всіх видів стиснення сервер збирає усі данні формує лінки на файли (рис 3.8) та відправляє відповідь (рис 3.9)

```
public getLink(fullUrl: string, compress: string, token: string) {
    return fullUrl + "/download?token=" + token + '&compress=' + compress + '&name=' + this.fileName
}
```

Рисунок 3.8 – Метод формування лінку

```
var fullUrl = req.protocol + '://' + req.get('host');
res.send({
    status: true,
    message: 'File is uploaded',
    data: [
        {name: "gz"...},
        {name: "lz4"...},
        {name: "lzw"...},
        {
            name: "zip",
            size: memory.zip,
            link: com.getLink(fullUrl, compress: "zip",req.body.token)
        },
        {
            name: "image",
            size: memory.image,
            link: com.getLink(fullUrl, compress: "image", req.body.token)
        },
    ],

});
```

Рисунок 3.9 – Формування відповіді сервера

3.1.2 Отримання даних з серверу

Шлях для отримання файлів формує файл та надсилає його користувачеві. За допомогою фреймворку не потрібно вказувати для кожного типу файлу заголовки, тому завантаження здійснюється за допомогою команди download.

```
router.get( path: '/download', handlers: function (req: any, res: any) {
    const file = `./uploads/${req.query.token}/${req.query.compress}/${req.query.name}${req.query.compress == 'images' ? '' : '.' + req.query.compress}`;
    res.download(file);
});

module.exports = router;
```

Рисунок 3.10 – Шлях завантаження файлу

3.2 Реалізація Frontend частини

Nuxt.js — це інтерфейс-фреймворк, створений на основі Vue.js, який пропонує чудові функції розробки, такі як рендеринг на стороні сервера, автоматично створені маршрути, покращене керування мета-тегами та покращення SEO.

Nuxt.js пропонує багато переваг для розробників інтерфейсу, але також є одна ключова особливість, яка зробила остаточне рішення використовувати цей фреймворк – покращення SEO. Говорячи про соціальний обмін, Nuxt.js має чудове керування мета-тегами

Щоб покращити SEO, Nuxt.js використовує SSR (Server Side Rendering). SSR отримує дані AJAX і відтворює компоненти Vue.js у рядки HTML на сервері (Node.js). Він надсилає їх безпосередньо до браузера, коли вся асинхронна логіка буде виконана, а потім, нарешті, подає статичну розмітку в повністю інтерактивну програму на клієнті. Ця функція дозволяє чудово аналізувати елементи DOM за допомогою аналізатора Google SEO. SEO-парсер аналізує елементи DOM з величезною швидкістю відразу після завантаження DOM веб-сайту.

З іншого боку, типові додатки SPA, створені за допомогою таких фреймворків, як Vue.js, React, Angular тощо, витягують дані із бекенда за допомогою AJAX після завантаження DOM, і тому SEO-парсер не може проаналізувати всі елементи DOM, оскільки там ще не надано. Вибір AJAX є асинхронним, тоді як аналіз SEO ні.

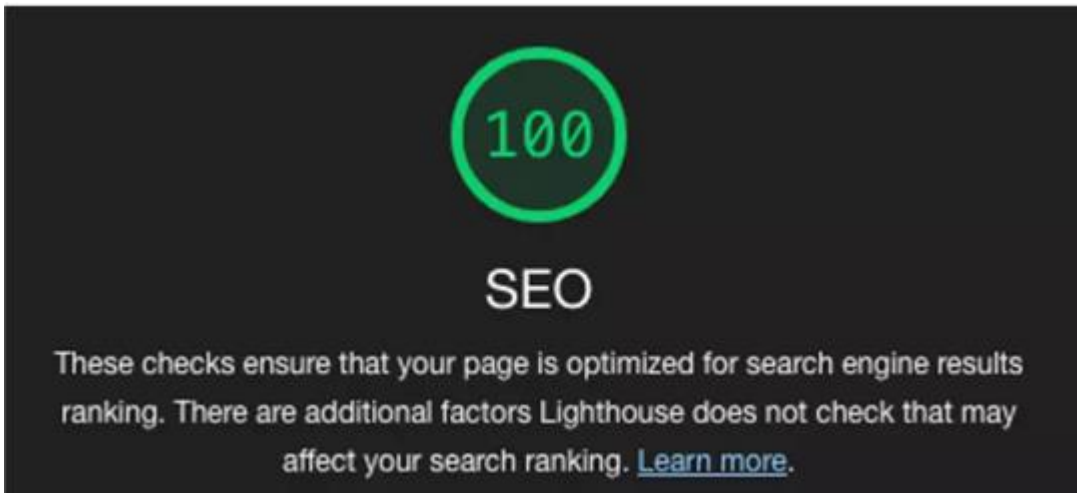

Рисунок 3.11 – Оцінка SEO-аудиту Nuxt.js

Рисунок 3.12 – Оцінка SEO-аудиту Vue.js

Nuxt.js і Vue.js по-різному обробляють логіку. Основна відмінність полягає в тому, що Vue завжди працює на стороні клієнта, а Nuxt — ні, і це може викликати серйозні проблеми в деяких випадках. Наприклад, якщо ви хочете вибрати елемент DOM відразу після завантаження програми, є ймовірність, що програма працює на стороні Node.js, і, звичайно, у Node.js немає елементів DOM.

Те ж саме станеться під час доступу до локального сховища браузера. Це основна причина, чому Nuxt використовує файли cookie в локальному сховищі, оскільки вони завжди доступні.

Nuxt.js створює власний маршрутизатор на основі структури папок, тоді як у Vue.js це потрібно робити вручну, але треба памятати, що обидва принципи мають як плюси, так і мінуси. Переваги автоматично згенерованого маршрутизатора полягають у тому, що його легше та швидше створити. Ви просто створюєте каталог і файли, а Nuxt виконує всю роботу. Але мінуси в тому, що він менш керований і керований, ніж написаний вручну.

Також підтримка Typescript для Nuxt.js значно покращилася, і тепер є нові стартові набори та конфігурації, які можна використовувати, не турбуючись про проблеми, пов’язані з Typescript

Nuxt.js має дуже схожу архітектуру на Vue.js. Є лише дві основні відмінності:

- Маршрутизатор
- Основний компонент App.vue

Nuxt генерує логіку маршрутизатора та його маршрути на основі каталогу та структури файлів для сторінок. Наприклад, якщо створити каталог і файл «about/index.vue», Nuxt.js автоматично створює маршрут «/about» для цієї сторінки. Немає необхідності визначати або налаштовувати маршрути в будь-якому іншому місці програми.

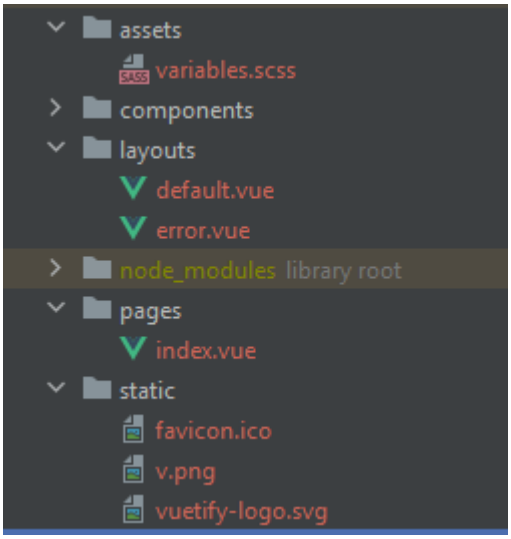

Рисунок 3.13 – Коренева структура програми Nuxt.js

Nuxt.js є основою проектів Vue.js і дає міцні структури для створення життєздатних проектів з достатньою гнучкістю. Основними функціями Nuxt.js є:

- Рендеринг на стороні сервера
- Доступ до створення односторінкового додатка (SPA)
- Надійна система маршрутизації
- Попередні процесори: Sass, Less, Stylus тощо.
- Статична обслуговування файлів
- Автоматичне розділення коду
- Розширення модульної архітектури
- Мінімізація та об'єднання CSS і JS

Ось що відбувається, коли користувач відвідує програму Nuxt.js або переходить на одну з її сторінок за допомогою <nuxt-link>:

- Коли користувач відвідує програму, якщо у nuxtServerInit визначено в дії, Nuxt.js викличе її та оновить стан додатку.
- Далі він виконує будь-яке існуюче проміжне програмне забезпечення для сторінки, яку відвідують. Nuxt спочатку перевіряє nuxt.config.js файл на наявність глобального проміжного програмного забезпечення, потім перевіряє відповідний файл макета (для запитаної сторінки) і, нарешті, перевіряє сторінку та її дочірні програми на наявність проміжного програмного забезпечення.
- Якщо маршрут, який відвідується, є динамічним і для нього існує метод validate(), маршрут перевіряється .
- Потім Nuxt.js викликає методи asyncData() і fetch() для завантаження даних перед відтворенням сторінки. Метод asyncData() використовується для отримання даних і відтворення їх на стороні сервера, тоді як fetch() метод використовується для заповнення сховища перед відтворенням сторінки.
- На останньому кроці відтворюється сторінка (містить усі належні дані).

Ці дії правильно зображені в цій схемі, отриманій з документів Nuxt:

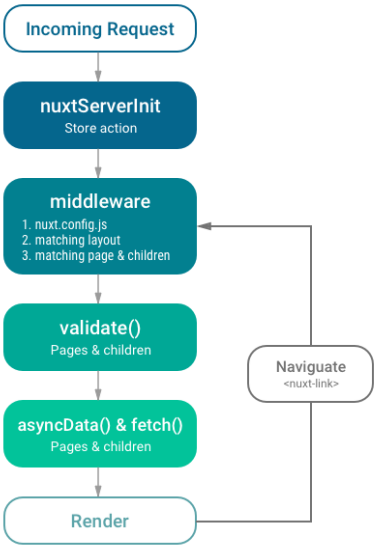

Рисунок 3.14 – Обробка запиту Nuxt.js[31]

Найпростіший спосіб розпочати роботу з Nuxt.js — використовувати шаблон, створений командою Nuxt. Щоб швидко встановити його в проєкт за допомогою vue-cli

```
vue init nuxt/starter <project>
```

Рисунок 3.15 – Команда швидкого старту

Також було додано бібліотеку Vuetify [32] для створення сучасного інтерфейсу системи. Vuetify — це повна структура інтерфейсу користувача, побудована на основі Vue.js. Мета проекту — надати розробникам інструменти, необхідні для створення багатих і привабливих користувацьких можливостей. На відміну від інших фреймворків, Vuetify розроблено з нуля так, щоб його було легко навчитися та корисно опанувати завдяки сотням ретельно розроблених компонентів

Vuetify використовує мобільний підхід до дизайну, що означає, що ваша програма просто працює з коробки — на телефоні, планшеті чи настільному комп'ютері.

З моменту свого першого випуску в 2014 р.Vue.js став одним із найпопулярніших фреймворків JavaScript у світі. Однією з причин такої популярності є широке використання компонентів, які дозволяють розробникам створювати стислі модулі для використання та повторного використання в їхньому додатку. Бібліотеки інтерфейсу користувача — це колекції цих модулів, які реалізують конкретні вказівки щодо стилю та надають необхідні інструменти для створення великих веб-додатків.

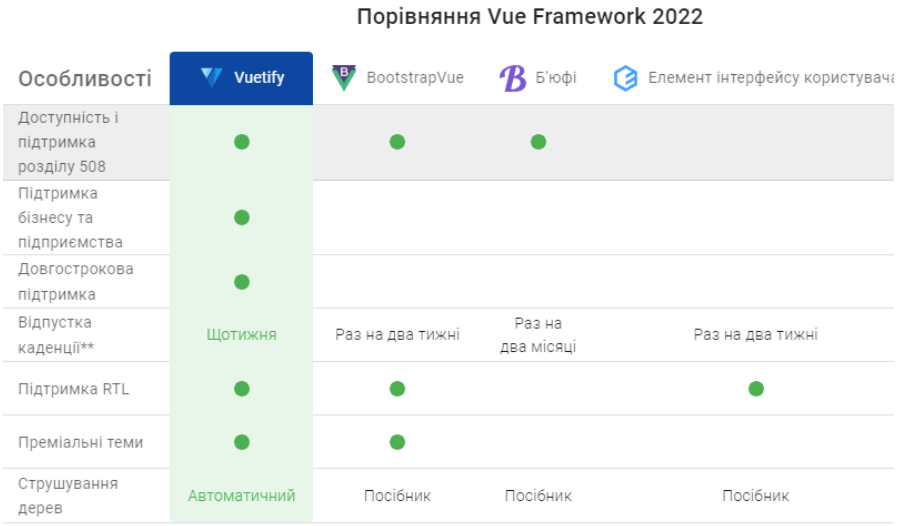

Рисунок 3.16 – Порівняння бібліотек інтерфейсів 2022

Vuetify розроблено точно відповідно до специфікації матеріального дизайну[33] з кожним компонентом, ретельно розробленим, щоб бути модульним, чуйним і продуктивним. Налаштуйте свою програму унікально та динамічно. Також, можна створити стилі ваших компонентів за допомогою змінних SASS.

У готовий шаблон додатку додамо файл макету сайту для відображення навігації на сайті та додано контейнер для даних на сторінці. А також футер с датою.

```
<v-app dark>
  <v-app-bar
    :clipped-left="clipped"
    fixed
    app
  >
    <router-link
      to="/"
      custom
    >
      Home
    </router-link>
  </v-app-bar>
  <v-main>
    <v-container>
      <Nuxt/>
    </v-container>
  </v-main>
  <v-footer
    :absolute="!fixed"
    app
  >
    <span>© {{ new Date().getFullYear() }}</span>
  </v-footer>
</v-app>
```

Рисунок 3.17 – Код шаблон default.vue

3.2.1 Створеня шаблону

Наступне створено головну сторінку index.vue. Спочатку було створено шаблон сторінки. Сторінка поділенна на 3 блоки:

- Заголовок
- Функціональний (кнопка підтвердження, вибір файлу та поле якості)
- Таблиця результатів.

У заголовку описано головний сенс системи та можливості.

```
<div v-if="!table" class="text-center">
  <h1>The <span class="accent">Progressive</span>
    compressive app</h1>
  <p class="tagline">What do you want to compress?<br></p>
  <p>We can compress anything you want</p>
</div>
```

Рисунок 3.18 – Код заголовоку

Для досягнення візуальної естетики вітального повідомлення було додано стилі в секції <Style>

```
<style scoped>
h1 {
  font-size: 76px;
  font-weight: 900;
  letter-spacing: -1.5px;
  max-width: 960px;
  margin: 0 auto;
}

:root {
  --vt-font-family-base: Inter, -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Oxygen, Ubuntu, Cantarell,
  --vt-font-family-mono: Menlo, Monaco, Consolas, "Courier New", monospace;
}

.accent {
  background: -webkit-linear-gradient(315deg, #42d392 25%, #647eff);
  background-clip: text;
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;
}
</style>
```

Рисунок 3.19 – Вигляд вітального повідомлення

Також для адаптиву інтерфейсу використано потужну сітку flexbox для мобільних пристроїв. За допомогою сітки можно створювати макети будь-яких форм і розмірів завдяки системі з дванадцяти стовпців, п'яти чутливим рівням за замовчуванням, змінним і Sass змінними, а також десяткам попередньо визначених класів.

Рисунок 3.20 – Вигляд вітального повідомлення

Далі було створенно функціональну частину:

- Поле для завантаження файлу
- Кнопку для відправки файлу на сервер
- Поле для заданя сили стиснення картинок

```
<div class="mt-15">
  <v-row justify="center" align="center">
    <div class="col-4 text-center">
      <v-file-input
        label="File input"
        truncate-length="3" @change="onInputChanged"/>
      <v-text-field
        label="Image compress %"
        v-model="quality"
      ></v-text-field>
      <v-btn @click="submitFile()" class="text-button text-center">Compress</v-btn>
    </div>
  </v-row>
</div>
```

Рисунок 3.21 – Функціональні меню

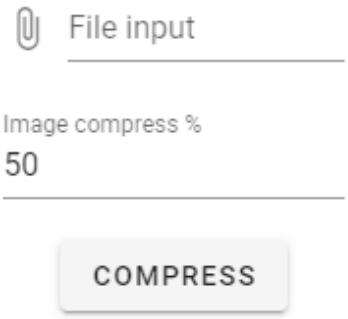

Рисунок 3.22 – Вигляд функціонального меню

Після додано таблицю яка буде замінятись після отримання результатів стиснення. Також, додано посилання на завантаження файлів та їх розмір.

```
<div v-if="table" >
  <v-data-table
    :headers="headers"
    :items="body"
    :items-per-page="5"
    class="elevation-1"
  >
    <template #item.link="{ item }">
      <a target="_blank" :href="item.link">
        download
      </a>
    </template>
  </v-data-table>

</div>
```

Рисунок 3.23 – Таблиця результатів

3.2.2 Функціональна частина

Після створення шаблону додаються скрипти для досягнення функціональності. Перше було створенно змінні в функції data.

```
data() {
  return {
    file: '' as Blob | string,
    quality: 50 as number,
    table: false as boolean,
    headers: [{'text':'Compress name','value':"name"},{'text':'Size','value':"size"},{'text':'Link','value':"link"}],
    body:[]
  }
},
```

Рисунок 3.24 – Структура змінних

Змінні:

- file – збереження файлу користувача та завантаження на сервер.
- quality – якість стиснення
- table – змінна статусу
- headers – заголовки таблиці
- body збереження данних стиснення

Створенно 2 методи. Перший onInputChanged створенно для запису файлу з шаблону до змінної. Головний метод submitFile запоковує файл та відправляє на сервер.

```
methods: {
  onInputChanged(file: any): void {
    this.file = file;
  },
  async submitFile() {
    const formData = new FormData();
    formData.append( name: 'file', this.file);
    formData.append( name: 'token', String(Math.random().toString( radix: 36).substr( from: 2,  length: 9)));
    formData.append( name: 'quality', String(this.quality))
    const headers = {'Content-Type': 'multipart/form-data'};
    const data = (await axios.post( url: this.$config.backUrl + '/upload', formData,  config: {headers})).data

    this.body = data.data;
    this.table = !this.table;

  }
}
```

Рисунок 3.25 – Функціональна частина

Після отримання данних (рис. 3.26), заповнються данні та відображається таблиця результатів стиснення (рис. 3.27).


```
▼data: [{name: "gz", size: 104,…}, {name: "lz4", size: 161,…}, {name: "lzw", size: 2
  ▼0: {name: "gz", size: 104,…}
      link: "http://localhost:3001/download?token=2lkm8rh4j&compress=gz&name=test.txt'
      name: "gz"
      size: 104
  ▶1: {name: "lz4", size: 161,…}
  ▶2: {name: "lzw", size: 2384,…}
  ▶3: {name: "zip", size: 200,…}
  ▶4: {name: "image", size: null,…}
 message: "File is uploaded"
 status: true
```


Рисунок 3.26 – Таблиця результатів стиснення

Compress name	Size,Byte	Link
gz	271989	download
lz4	273201	download
lzw	324515	download
zip	272443	download
image		download

Rows per page: 5 1-5 of 5

Рисунок 3.27 – Таблиця результатів стиснення

Загальний код з системою можна побачити в Додатку В

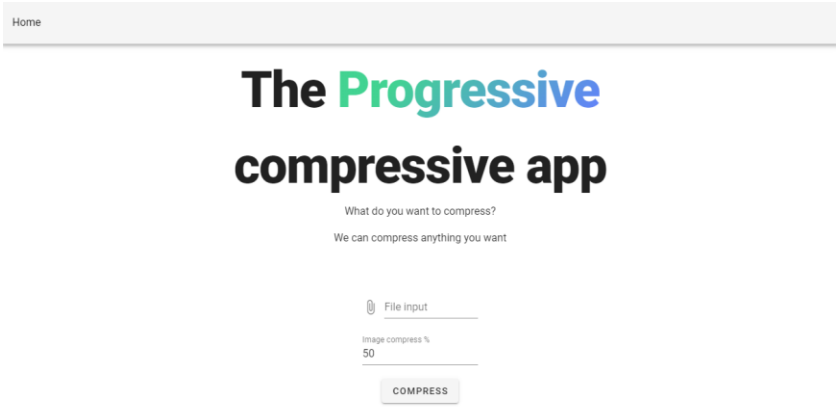

Рисунок 3.28 – Загальний вигляд сторінки

3.3 Тестування системи

Будь-який програмний продукт має бути протестованим для виявлення дефектів і помилок, припущених на стадії інженерії ПЗ. Тестування програмного забезпечення — це перевірка того, чи відповідають фактичні результати очікуваним. Процес передбачає запуск та виконання компонента програмного забезпечення або компонента системи для оцінки однієї або декількох властивостей.

Для тестування стиснення файлів без втрат обрано файл test.txt 30 000 байт. Після завантаження файлу отримано наступні результати (рис. 3.29). Результат завантаження стиснутого файлу з сереверу можна побачити в Додатку Г

Compress name	Size,Byte	Link
gz	104	download
lz4	161	download
lzw	2384	download
zip	200	download
image		download

Rows per page: 5 1-5 of 5

Рисунок 3.29 – Результати стиснення

Для тестування стиснення зображеня було обрано файл ImageTest.png з розміром 8000 байт.

image	6026	download
image	5363	download

Рисунок 3.30 – Стиснення на 10% – сверху; 90% – снизу

Після стиснення на стороні сервера було створенно папку с токеном з представлення. Також, для кожного виду стиснення створюється папка.

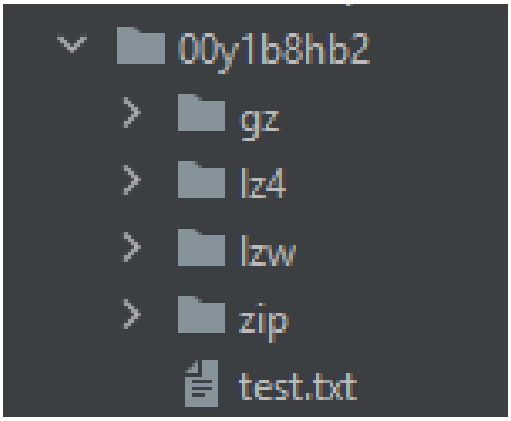

Рисунок 3.31 – Структура проєкту

3.4 Висновки до розділу 3

У 3 розділі спроектовано систему для дослідження та моделювання стиснення файлів різними бібліотеками. В загальному обсязі веб сервер використовує 4 різні типи стиснення файлів без втрат, а саме формати (zip,lz4,lzw,gz), а також 4 види стиснення з втратами для 4 видів зображеннь, а саме формати (jpg,png,svg,gif).

Так як проєкт побудований на моделі API + SPA, це дає змогу для швидкого маштабування та розширення системи. Також, це дає можливість використовувати сервіс стиснення з будь-якої іншої платформи, яка може здійснити веб запит.

Front-end частина так і Back-end частина використовує новітні фреймворки, тому виконує поставленні задачі максимально чітко та швидко, що дає новітнії та легкорозуміємий інтерфейс для користувача.

Працездатність сервісу перевіренно, та надано звіт по стисненню файлів у форматі з втратами та без.

ВИСНОВКИ

Метою бакалаврської кваліфікаційної роботи є розробка системи моделювання та дослідження різних бібліотек стиснення шляхом розробки відповідного вебзастосунку.

При виконанні завдання було реалізовані наступні етапи:

- опис предметної області;
- огляд та аналіз бібліотек;
- постановка задачі;
- вибір технології створення вебзастосунків;
- вибір найкращих інструментів та методології для вирішення поставленої задачі;
- проєктування діаграми використання вебзастосунку;
- розробка структури вебзастосунку;
- проєктування завантаження файлів до вебзастосунку;
- опис програмної реалізації;
- проведення тестування на різних об'єктах та аналіз даних вебзастосунку.

У першому розділі було знайдено та проаналізовано бібліотеки стиснення, проведено аналіз предметної області, серед яких виявлено переваги та недоліки.

У другому розділі було проаналазовано та підібрано найбільш влучні технології створення вебзастосунку, сучасні та зручні інструменти, необхідні методології та обрано кращі з них для даного вебзастосунку. Найкращою методологією для даного проєкту стала розробка вебзастосунку за допомогою комбінації фреймворків Express та Nuxt.js (Vue.js).

У третьому розділу було спроєктовано взаємодії користувача з вебзастосунком та сервером за допомогою діаграм відповідно до процесу. Це допомогло створити незалежні частини сервісів, а також дало можливість для подальшого розширення систем.

У розробленому вебзастосуноку реалізовано мінімалістичний та зручний інтерфейс для користування.

Функціональні можливості розробленого застосунку:

- Стиснення будь-яких данних у 4 формати архіву без втрат.
- Стиснення фотографій 4 форматів з заданям % стиснення

Серед інструментів обрано:

1. Сервер Node.js [29];
2. Фреймворк Express 4.1 [30];
3. Фреймворк Nuxt.js 2.1 [31];
4. Кодовий редактор PhpStorm [34];

Зібрано статистичні дані ефективності стиснення даних загального спектору форматів, а також фотографій.

Враховуючи вищенаведене, всі задачі роботи було виконано, а мету роботи, що полягала у створенні системи моделювання та дослідження бібліотек стиснення досягнуто.

У спеціальному розділі з охорони праці було розглянуто питання охорони праці в університеті «Чорноморський національний університет імені Петра Могили», виконана інтегральна оцінка умов праці, проведені заходи, спрямовані на їх покращення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Інформатика: Комп'ютерна техніка. Комп'ютерні технології. — Київ : Академія, 2002. — 704 с.

2. Руденко В.Д., Макарчук О.М., Патланжоглу М.О. Практичний курс інформатики. — Київ : Фенікс, 1996. — С. 418.

3. Ярмуш О.В., Редько М.М. Інформатика і комп’ютерна техніка. — Київ : Вища освіта, 2006. — С. 359.

4. Гуржій А.М., Поворознюк Н.І., Самсонов В.В. Інформатика та інформаційні технології. — Харків : ООО «Компанія СМІТ», 2003. — С. 352.

5. 10 Best Node.js Compression Libraries URL: https://openbase.com/categories/js/best-nodejs-compression-libraries (дата звернення 17.06.2022)

6. Welch, Terry A.. «A technique for high performance data compression». IEEE Computer. 17 (6): 8–19. doi:10.1109/MC.1984.1659158 (1984)

7. Введення в кілька бібліотек стиснення URL: https://russianblogs.com/article/2355264154/(дата звернення 17.06.2022)

8. Copeland, Jack, «ColumnStore Storage Architecture mariadb», Oxford: Oxford University Press, pp. 100–115, ISBN 978-0-19-284055-4 2010

9. RAR 5.0 archive format URL: https://www.rarlab.com/technote.html (дата звернення 17.06.2022)

10. Клименко О.Ф., Головко Н.Р Як працює стиснення GZIP Київ : КНЕУ, 2002. — С. 534.

11. Comparison of file archivers URL: https://en.wikipedia.org/wiki/Comparison_of_file_archivers (дата звернення 05.06.2022)

12. Lz4 repository URL: https://github.com/lz4/lz4 (дата звернення 17.06.2022)

13. Fastlz Google URL: https://code.google.com/archive/p/fastlz/ (дата звернення 17.06.2022)

14. Snappy URL: http://google.github.io/snappy/ (дата звернення 17.06.2022)

15. Compress URL: https://uk.upwiki.one/wiki/Compress (дата звернення 17.06.2022)

16. ZIP (Lempel Ziv Welch) URL: https://uk.wikipedia.org/wiki/Zip(дата звернення 17.06.2022)

17. bzip2 URL: https://linux-faq.ru/page/komanda-bzip2 (дата звернення 17.06.2022)

18. 7-Zip is free software URL: https://www.7-zip.org/ (дата звернення 17.06.2022)

19. XZ Embedded URL: https://tukaani.org/xz/embedded.html (дата звернення 17.06.2022)

20. Small & portable byte-aligned LZ77 compression URL: https://ariya.github.io/FastLZ/ (дата звернення 17.06.2022)

21. Why Is File Compression Important URL: https://www.techwalla.com/articles/why-is-file-compression-important (дата звернення 18.06.2022)

22. File compression pros and cons: what you need to know URL: https://www.thebusinesswomanmedia.com/organize-office-better/ (дата звернення 18.06.2022)

23. Advantages & Disadvantages of Using File Compression URL: https://smallbusiness.chron.com/file-compression-decompression-930.html (дата звернення 18.06.2022)

24. The State of Node.js & JavaScript for Backend Development URL: https://snipcart.com/blog/javascript-nodejs-backend-development (дата звернення 18.06.2022)

25. Ray Girvan, " Reasons Why JavaScript is Omnipresent in Modern Development" Archived 3 November 2012 at the Wayback Machine, Scientific Computing World, May/June 2003

26. Use Node.js. A Case-by-Case Tutorial URL: https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js (дата звернення 17.06.2022)

27. G. Wiet, V. Elisseeff, P. Wolff, J. Naudu «TypeScript for the New Programmer», p. 649. George Allen & Unwin Ltd, 2009

28. Flowchart Maker & Online Diagram Software Draw.io URL: https://app.diagrams.net/ (дата звернення 18.06.2022)

29. Node.js URL: https://nodejs.org/ (дата звернення 17.06.2022)

30. Express URL: https://expressjs.com/ (дата звернення 18.06.2022)

31. Nuxt js URL: https://nuxtjs.org/ (дата звернення 18.06.2022)

32. Vuetify URL: https://vuetifyjs.com/en/introduction/why-vuetify/#getting-started (дата звернення 18.06.2022)

33. Material Design URL: https://material.io/ (дата звернення 18.06.2022)

34. PhpStorm URL: https://www.jetbrains.com/ru-ru/phpstorm/ (дата звернення 18.06.2022)

ДОДАТОК А

КОД ПРОГРАМИ ROUTER

```
var express = require('express');
const {Compressions} = require("../services/compressions");
var router = express.Router();

router.get('/', function (request: any, response: any) {
    response.render('index')
});

router.post('/upload', async (req: any, res: any) => {
    try {
        if (!req.files) {
            res.send({
                status: false,
                message: 'No file uploaded'
            });
        } else {

            let file = req.files.file;
            let path = './uploads/' + req.body.token + '/' + file.name;
            file.mv(path);
            const com = new Compressions("./uploads/" + req.body.token,
file.name);
            await com.zlibCompress();
            await com.lz4Compress();
            await com.lzwCompress()
            await com.zipCompress()

            let memory = {
                gz: com.getSize(com.dirPath + '/gz/' + com.fileName + '.gz'),
                lz4: com.getSize(com.dirPath + '/lz4/' + com.fileName +
'.lz4'),
                lzw: com.getSize(com.dirPath + '/lzw/' + com.fileName +
'.lzw'),
                zip: com.getSize(com.dirPath + '/zip/' + com.fileName +
'.zip'),
                image: null,
            }
            if (path.endsWith('.jpg') || path.endsWith('.png') ||
path.endsWith('.svg') || path.endsWith('.gif')) {
                await com.imageCompress(req.body.quality);
                memory.image = com.getSize(com.dirPath + '/images/' +
com.fileName);
            }
            var fullUrl = req.protocol + '://' + req.get('host');
            res.send({
                status: true,
                message: 'File is uploaded',
                data: [
                    {
                        name: "gz",
                        size: memory.gz,
                        link: com.getLink(fullUrl,"gz",req.body.token)
                    },
                    {
```

```
                        name: "lz4",
                        size: memory.lz4,
                        link: com.getLink(fullUrl,"lz4",req.body.token)
                    },
                    {
                        name: "lzw",
                        size: memory.lzw,
                        link: com.getLink(fullUrl,"lzw",req.body.token)
                    },
                    {
                        name: "zip",
                        size: memory.zip,
                        link: com.getLink(fullUrl,"zip",req.body.token)
                    },
                    {
                        name: "image",
                        size: memory.image,
                        link: com.getLink(fullUrl, "image", req.body.token)
                    },
                ],

            });
        }
    } catch (err) {
        res.status(500).send(err);
    }
});

router.get('/download', function (req: any, res: any) {
    const file =
`./uploads/${req.query.token}/${req.query.compress}/${req.query.name}${req.qu
ery.compress == 'images' ? '' : '.' + req.query.compress}`;
    res.download(file);
});

module.exports = router;
```

ДОДАТОК Б

КОД ПРОГРАМИ BACKEND CLASS COMPRESSIONS

```
import * as fs from "fs";
import {readFile} from "fs/promises";

export class Compressions {

    private dirPath: string;
    private fullPath: string;
    private fileName: string;

    constructor(dirPath: string, fileName: string) {
        this.dirPath = dirPath;
        this.fileName = fileName;
        this.fullPath = dirPath + '/' + fileName;
    }

    public async zlibCompress() {
        const zlib = require('zlib');
        const libPath = this.createLibFolder('gz');
        const out = libPath + '/' + this.fileName + '.gz';
        const content = await readFile(this.fullPath);
        var compressed = zlib.gzipSync(content);
        fs.writeFileSync(out, compressed)
    }

    public async lz4Compress() {
        const lz4 = require('lz4');
        const libPath = this.createLibFolder('lz4');
        var input = fs.readFileSync(this.fullPath);
        var output = lz4.encode(input)
        fs.writeFileSync(libPath + '/' + this.fileName + '.lz4', output)
    }

    public async lzwCompress() {
        const lzw = require("lzw");
        const libPath = this.createLibFolder('lzw');
        const content = await readFile(this.fullPath);
        const compressed = lzw.compress(content);
        fs.writeFileSync(libPath + '/' + this.fileName + '.lzw', compressed)
    }

    public async zipCompress() {
        var Zip = require('node-zip');
        var zip = new Zip;
        const libPath = this.createLibFolder('zip');
        const content = await readFile(this.fullPath);
        zip.file(this.fileName, content)
        var options = {base64: false, compression: 'DEFLATE'};
        fs.writeFileSync(libPath + '/' + this.fileName + '.zip',
zip.generate(options), 'binary',);
    }

    public async imageCompress(quality: string) {
        const {compress} = require('compress-images/promise');
```

```
        const input = this.fullPath;
        const libPath = this.createLibFolder('images');
        const output = libPath + '/';


        const processImages = async () => {
            const result = await compress({
                source: input,
                destination: output,
                enginesSetup: {
                    jpg: {engine: "mozjpeg", command: ["-quality", quality]},
                    png: {engine: "pngquant", command: ["--quality=1-" +
quality, "-o"]},
                    svg: {engine: "svgo", command: "--multipass"},
                    gif: {engine: "gifsicle", command: ["--colors", "64", "--
use-col=web"]},
                }
            });
            const {statistics, errors} = result;
        }
        await processImages();
    }

    private createLibFolder(libName: string) {
        const zlibPath = this.dirPath + '/' + libName;
        if (!fs.existsSync(zlibPath)) {
            fs.mkdirSync(zlibPath);
        }
        return zlibPath
    }

    public getSize(path: string) {
        const stats = fs.statSync(path);
        return stats['size'];
    }

    public getLink(fullUrl: string, compress: string, token: string) {
        return fullUrl + "/download?token=" + token + '&compress=' + compress
+ '&name=' + this.fileName
    }
}
```

ДОДАТОК В

КОД ПРОГРАМИ FRONT

```
<template>
  <v-row justify="center" align="center">
    <v-col cols="12" sm="8" md="6">
      <div v-if="!table" class="text-center">
        <h1>The <span class="accent">Progressive</span>
          compressive app</h1>
        <p class="tagline">What do you want to compress?<br></p>
        <p>We can compress anything you want</p>
      </div>
      <div v-if="table" >
        <v-data-table
          :headers="headers"
          :items="body"
          :items-per-page="5"
          class="elevation-1"
        >
          <template #item.link="{ item }">
            <a target="_blank" :href="item.link">
              download
            </a>
          </template>
        </v-data-table>

      </div>
      <div class="mt-15">
        <v-row justify="center" align="center">
          <div class="col-4 text-center">
            <v-file-input
              label="File input"
              truncate-length="3" @change="onInputChanged"/>
            <v-text-field
              label="Image compress %"
              v-model="quality"
            ></v-text-field>
            <v-btn @click="submitFile()" class="text-button text-
center">Compress</v-btn>
          </div>
        </v-row>
      </div>
    </v-col>
  </v-row>
</template>

<script lang="ts">

import axios from "axios";
import Vue from "vue";

export default Vue.extend({
  name: 'IndexPage',
  data() {
    return {
      file: '' as Blob | string,
```

```
      quality: 50 as number,
      table: false as boolean,
      headers: [{'text':'Compress
name','value':"name"},{'text':'Size,Byte','value':"size"},{'text':'Link','val
ue':"link"}],
      body:[]
    }
  },
  methods: {
    onInputChanged(file: any): void {
      this.file = file;
    },
    async submitFile() {
      const formData = new FormData();
      formData.append('file', this.file);
      formData.append('token', String(Math.random().toString(36).substr(2,
9)));
      formData.append('quality', String(this.quality))
      const headers = {'Content-Type': 'multipart/form-data'};
      const data = (await axios.post(this.$config.backUrl + '/upload',
formData, {headers})).data

      this.body = data.data;
      this.table = !this.table;

    }
  }
})
</script>
<style scoped>
h1 {
  font-size: 76px;
  font-weight: 900;
  letter-spacing: -1.5px;
  max-width: 960px;
  margin: 0 auto;
}

:root {
  --vt-font-family-base: Inter, -apple-system, BlinkMacSystemFont, "Segoe
UI", Roboto, Oxygen, Ubuntu, Cantarell, "Fira Sans", "Droid Sans", "Helvetica
Neue", sans-serif;
  --vt-font-family-mono: Menlo, Monaco, Consolas, "Courier New", monospace;
}

.accent {
  background: -webkit-linear-gradient(315deg, #42d392 25%, #647eff);
  background-clip: text;
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;
}
</style>
```

ДОДАТОК Г

РОБОТА ЗАВАНТАЖЕННЯ СТИСНУТОГО ФАЙЛУ