НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут прикладного системного аналізу

Кафедра системного проектування

До захисту допущено:
Завідувач кафедри
_____ А.І. Петренко
«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інтелектуальні сервіс-орієнтовані розподілені обчислювання»

спеціальності 122 «Комп'ютерні науки»

на тему: «Відображення в реальному часі параметрів екологічного стану міста»

Виконала:
студентка IV курсу, групи ДА-72
Пашкова Єлізавета Олександрівна _____

Керівник:
магістр ком. наук, асистент кафедри СП,
Мироненко Сергій Сергійович _____

Консультант з економічного розділу:
доцент, кандидат наук,
Рощина Надія Василівна _____

Рецензент:

_____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2021 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут прикладного системного аналізу

Кафедра системного проектування

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Інтелектуальні сервіс-орієнтовані розподілені обчислювання»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ А.І. Петренко

«___»_____20__ р.

ЗАВДАННЯ

на дипломну роботу студенту

Пашкова Єлізавета Олександрівна

1. Тема роботи «Відображення в реальному часі параметрів екологічного стану міста», керівник роботи Мироненко Сергій Сергійович, магістр комп. наук, асистент кафедри СП, затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи

Мова програмування JavaScript, використання для веб-розробки фреймворку React з бібліотекою візуалізації D3.js .

4. Зміст роботи:

1. Провести огляд документації систем та публікацій з теми роботи.
2. Розробити ескізний проект інтерфейсу відображення метеорологічної інформації регіону.
3. Визначити джерела метеорологічної інформації, що можуть бути використані в проекті та виконати аналіз засобів взаємодії із ними.

4. Визначити вимоги до ресурсів системи для відображення інформації в реальному часі.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

1. Презентація до захисту роботи.

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Надія Василівна, доцент, кандидат наук		

7. Дата видачі завдання_____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	03.02.2021	
2	Збір відповідної інформації	17.02.2021	
3	Аналіз наукової літератури.	05.03.2021	
4	Аналіз вимог завдання, вибір методів і засобів розв'язання поставленої задачі	29.03.2021	
5	Проектування архітектури системи та аналіз прийнятих рішень	16.04.2021	
6	Проектування інтерфейсу та архітектури.	23.04.2021	
7	Розробка програмного продукту.	14.05.2021	
8	Проведення функціонально-вартісного аналізу програмного продукту.	29.05.2021	
9	Оформлення дипломної роботи.	31.05.2021	
10	Отримання допуску до захисту на подача роботи в ДЕК.	10.06.2021	

Студент Пашкова Єлізавета Олександрівна
Керівник Мироненко Сергій Сергійович

* Якщо визначені консультанти. Консультантом не може бути зазначено керівника дипломної роботи.

АНОТАЦІЯ

бакалаврської дипломної роботи Пашкової Єлізавети Олександрівни на тему «Відображення в реальному часі параметрів екологічного стану міста»

Протягом посиленого розвитку технічного процесу ми маємо великі можливості отримувати інформацію атмосфери для сприятливого проживання в тих чи інших містах. Це в більшості стосується людей ,які не можуть проживати в країнах з певним кліматом ,який не підходить по одним з параметрів чи мегаполісах, оскільки вони дуже забруднені. Людям з хворобами легень доволі тяжко доводиться справлятися з цією проблемою, оскільки, наприклад, не мають можливості переїхати в іншу країну. Тому за допомогою показників атмосфери в конкретному місті ми можемо побачити, де найбільш очищене повітря від діоксиду вуглицю, оксіду сірки та інших тяжких та рідких частинок. Метою цієї роботи є розробка додатку з відображенням параметрів забруднення повітря за допомогою фреймворку React та бібліотеки D3.js.

При створенні бакалаврської роботи було створено та реалізовано систему реального часу, що показує показники забруднення міста.

Загальний обсяг роботи: 76 сторінок, 20 рисунків, 9 таблиць, 24 бібліографічних посилань.

Ключові слова: реактивне програмування, React, D3.js.

АННОТАЦИЯ

бакалаврской дипломной работы Пашковой Елизаветы Александровны на тему «Отображение в реальном времени параметров экологического состояния города»

В течение усиленного развития технического процесса мы имеем большие возможности получать информацию атмосферы для благоприятного проживания в тех или иных городах. Это в большинстве касается людей, которые не могут проживать в странах с определенным климатом, который не подходит по одним из параметров или мегаполисах, так как они очень загрязнены. Людям с болезнями легких довольно тяжело приходится справляться с этой проблемой, поскольку, например, не имеют возможности переехать в другую страну. Поэтому с помощью показателей атмосферы в конкретном городе мы можем увидеть, где наиболее очищенный воздух от диоксида вуглицю, оксиды серы и других тяжких и жидких частиц. Целью данной работы является разработка приложения с отображением параметров загрязнения воздуха с помощью фреймворка React и библиотеки D3.js.

При создании бакалаврской работы была создана и реализована система реального времени, показывает показатели загрязнения города.

Общий объем работы 76 страниц, 20 рисунков, 9 таблиц, 24 библиографических ссылок.

Ключевые слова: реактивное программирования, React, D3.js.

ANNOTATION

bachelor's thesis of Pashkova Yelizaveta Alexandrovna
on the topic "Real-time display of parameters of the ecological state of the city»

During the intensive development of the technical process, we have great opportunities to get information about the atmosphere for favorable living in certain cities. This mostly applies to people who cannot live in countries with a certain climate that is not suitable for some of the parameters or megacities ,because they are very polluted. It is quite difficult for people with lung diseases to cope with this problem, because, for example, they do not have the opportunity to move to another country. Therefore, using atmospheric indicators in a particular city, we can see where the most purified air is from carbon dioxide, sulfur oxide and other heavy and liquid particles. The purpose of this work is to develop an application that displays air pollution parameters.

When creating a bachelor's thesis, a real-time system was created and implemented that shows indicators of urban pollution.

Total volume of work: 76 pages, 20 figures, 9 tables, 24 bibliographic references.

Keywords: jet programming, React, D3.js.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

Фреймворк - каркас з готових рішень, який спрощує процес розробки і підтримку проектів.

HTML – HyperText Markup Language, мова гіпертекстової розмітки сторінки.

CSS – Cascading Style Sheets, мова стилю мов, за допомогою якого ми можемо керувати виглядом зовнішнім.

JS – JavaScript, це мова програмування, яка застосовується для управління вмістом HTML-документа.

SVG – Scalable Vector Graphics, формат на векторі, зроблений спеціально для веб-розробників.

SPA – Single Page Application, односторінковий додаток, використовує єдиний HTML-документ для всіх веб-сторінок

API – Application Programming Interface, сукупність різних функцій, зроблений у вигляді інтерфейсу для створення нових додатків, за допомогою якого інша програма може взаємодіяти з іншою.

JSX – розширення JS, яке використовується для фреймворку React.

DOM – Document Object Model, представляє документ у вигляді дерева тегів, іншими словами API для HTML документів.

ВСТУП

Останнім часом вимоги до великих додатків стали набагато більше ніж це було 5 років тому. У ті часи ми мали гігабайти даних, десятки серверів, офлайнове обслуговування, які могли тривати годинами, а час відгуку доходило до декількох секунд. Зараз же час відгуку знизився до мілісекунд і стовідсоткового аптайму, дані виросли до петабайтів і це не межа. [1]

Спершу це стосувалося великих компаній-гігантів, проте тепер такі вимоги ставлять у багатьох галузях. Першими хто почав робити внесок новими практиками були телекомунікаційні компанії, тепер же хід дійшов і до інших.

З новими вимогами прийшли і нові технології. У минулому, більшість рішень робили зусилля на створення контейнерів і керованих серверів. Застосовувалися неефективні і дорогостоючі рішення для додавання нових серверів, а масштабування було за рахунок багатопоточності і купівлі більш крутих пристроїв. Однак з прогресом еволюціонували не тільки підвищення вимог, але архітектура самих додатків.

Завдання даної дипломної роботи полягає в дослідженні даних реального часу якості повітря в містах України, проектування і розробки програми, візуалізація даних на карті. Дане завдання включає в себе вирішення проблем, а саме:

• Зробити аналіз літератури по темі веб-розробки додатків і їх фреймворків, додаткових бібліотек.

• Проаналізувати альтернативи і вибрати підходящий варіант.

• Створити додаток.

1 Актуальність моніторингу екології

1.1. Вступ

Технології вплинули не тільки в кращу сторону на світ в цілому, але так само спостерігається і зворотний ефект,на жаль. Через велику кількість різних заводів в атмосфері міститься велика кількість діоксиду вуглецю, шкідливих важких і легких частинок. Отже, забруднення повітря прямо впливають на навколишнє середовище і особливо на наше з вами здоров'я. Так само впливає на здоров'я тварин, рослин, води і навіть грунту. [2]

У зв'язку з цим вельми актуальна інформація про онлайн-моніторинг якості повітря. Щорічно 7 мільйонів людей помирають від забруднення повітря, а мільярди – без потреби страждають від поганої якості повітря. У кожній країні існує велика кількість станцій для вимірювання найрізноманітніших параметрів повітря, грунту, води та інше. Наша увага спрямована саме на якість повітря.

Можна навіть не виходячи з дому вносити лепту у внесок вивчення і збереження природи і навколишнього середовища. Це можливо за допомогою участі в проектах краудсорсингу та цивільних наук у світі, що використовують різноманітні технології такі як онлайн-карти, дрони, супутниковий-моніторинг. Краудсорсинг-це залучення великої кількості людей для вирішування найрізноманітніші проблем, що діють на добровільних засадах. За допомогою інтернету цей напрямок став глибоко розвиватися і застосовуватися в самих різних областях. Таким є і платформа https://www.iqair.com / для отримання інформації про якість повітря в режимі реального часу. Цей сайт приваблює велику кількість громадян, організацій та урядів світу. [3]

1.2. Дані реального часу

Раніше різні організації збирали дані тривалий час, зараз же все відбувається набагато швидше, ніж будь-коли. Статистика свідчить, що до 2030 року підключених пристроїв до мережі буде перевищувати 24, 1 млрд, надалі зростання все так само буде зростати, тому потрібно вміти їх обробляти і аналізувати швидше, ніж зараз. Якщо розглядати широко, то дані збираються для якоїсь користі людини або групи людей, наприклад, компанії. За допомогою них розвиток бізнесу стрімко підвищується в рази. [4]

Після збору даних аналітика перетворює дані в практичний висновок. Час в такому випадки є дуже важливим фактором. Оперативна аналітика (аналітика реального часу) підказує нам про моменти виходу з ладу будь-якого обладнання і попереджає персонал заздалегідь. Розсилка акцій покупцям, коли вони знаходяться поблизу конкретного магазину так само заслуга оперативної аналітики. Таким способом можна і виявляти шахрайство з банківськими картками, до завершення транзакції і тим самим зберегти свої гроші. Типи аналітики між собою різняться: аналітика за запитом і потокова. Перший тип аналітики очікує запит від користувача або системи і видає аналітичний результат. Потокова ж працює безперервно, попереджає користувачів або робить відгуки на дії.

Потокова аналітика має досить великий попит з урахуванням того, що дані і швидкість передачі їх з кожним роком зростає, як і потреба в обробці даних зразу після отримання їх. Бізнес-аналітик якраз і використовують даний механіх для прийняття своєчасних рішень, тим самим на ринку підвищуючи конкурентну перевагу.

Кожен потік даних має свою цінність, вона може зникнути через тиждень, день, годину і навіть пару секунд. Для прикладу, дані для управління безпілотною вантажівкою стають непотрібними, коли вони застарівають і

стають небезпечними в даному випадку. Так само стосується і обладнання, що застарівають, і може статися непоправні дії через збій.

За допомогою аналітики усуваються больові точки компаній: надаються послуги клієнтам і продукти саме в той момент часу, коли вони їм потрібні.

Відправною точкою розширеної аналітики є прогнозована аналітика, де висновок ґрунтується не тільки на загальній інформації, але і на інформації реального часу. Прикладами застосування даної аналітики є прогнозна, когнітивна і розпорядча. Не дивлячись на те, яку аналітику використовує компанія, важливо застосовувати комплексний підхід обробки даних, на підставі архітектури аналітики реального часу. Загальна риса всіх типів є збір, зберігання, аналіз і захист даних, що дозволяє поширювати інформацію всередині компанії для миттєвого прийняття рішень.

Аналітика реального часу допомагає більше і швидше отримувати користь з оброблених даних. Для більш точних прогнозів створюється в кожній галузі своя інфаструктура для більш ймовірних прогнозів і впевнених рішень.

1.3 Висновки до розділу

На даний момент існує актуальність теми онлайн-моніторингу стану якості повітря, в особливості для людей з проблемами дихальних шляхів. Розглянуті наступні теми в даному розділі:

- вимоги новітніх технологій, що ставляться для систем
- актуальність онлайн-моніторингу екології
- важливість даних реального часу
- як саме аналітика допомагає раціонально використовувати дані реального часу

2 Фреймворки веб-програмування

2.1. Вступ

Серед найбільш популярних мов програмування, враховуючи лідируючу позицію обширності використання, займає мова JavaScript. З самого початку використовувалась тільки на стороні клієнта, але через деякий час почали використовувати і на стороні сервера. Якщо сказати більш широко, то це мова є мовою Інтернету. Через це збільшується кількість готових рішень для найрізноманітніших завдань, прискорюючи і спрощуючи сам процес розробки. Часто кажуть на фреймворки бібліотеками , але різниця між ними в тому, що перші включають в себе бібліотеки з готовими шаблонами , створюючи патерни проектування ,в той час бібліотеки надають набір готових класів, методів для вирішення класичних завдань. Зазвичай фреймворки складаються з модулів, які між собою строго пов'язані. [5]

Загалом кажучи, фреймворки мають написаний код для використання в стандартних завданнях. Вважається, що це основа для створення веб-сайтів і веб-додатків. Написання веб-сторінок цілком можливо і без фреймворків, але прискорення в часі полегшує роботу над завданнями. Більшість фреймворків безкоштовні і навіть з відкритим вихідним кодом. Отже, це дозволяє писати менше коду вручну, оскільки вже є готові рішення: шаблони та функції. Також вони більш адаптовані для дизайну веб-сайту. [6]

2.2 Реактивне програмування

Останнім часом в безліч статей можна помітити такий термін як «реактивний». Реактивне програмування не тільки стало дуже популярним, але і в цілому тепер керує процесом розробки програмного забезпечення. Великі гіганти як Microsoft, Facebook, Google, не тільки використовують даний підхід реактивності, а й впроваджують нові фреймворки за даним принципом.

Реактивне програмування - це програмування з використанням асинхронних потоків даних, які відправляються користувачеві, коли йому вони стають необхідні, таким чином на подібні зміни можуть швидко реагувати асинхронно. [7]

Потік-це послідовність поточних змін, що впорядковано за часом. Видає три різних сигналу: завершення, помилку, визначення типу. Прослуховування потоку називається підпискою. Використовуючи реактивне програмування ми можемо стежити за будь-якою змінною або функцією, і при змінах їх значення повідомляти іншу функцію, що б пішов ланцюжок змін в програмі.

Абстракція з таким підходом підвищується в рази. Концентрувати доводиться вже не на підтримці коду, що є, а на написання бізнес-логіки. Тому реактивність підходить при наявності великої кількості подій в додатку.

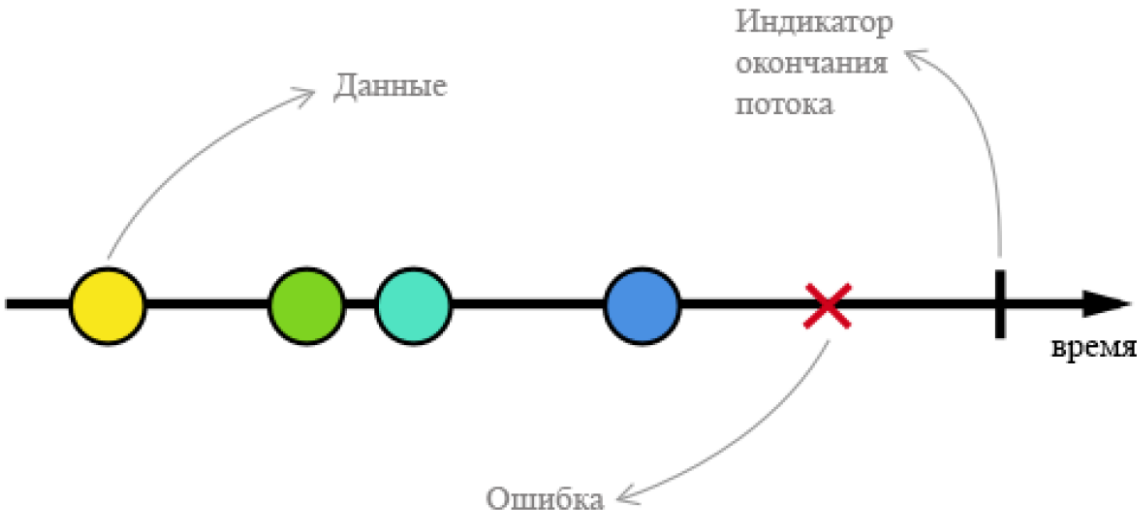


Рисунок 1 – Схема представлення потоків даних [1]

```
--a---b-c---d---X---|->

a, b, c, d - генерируемые значения
X - ошибка
| - сигнал завершения
---> - временная ось
```

Рисунок 2 – візуалізація події, обробки натискання на мишку

В даному прикладі (рис. 2) ми розглянемо як в цілому працює концепція реативності. Візьмемо подію обробки натискання на мишку. Головна особливість потоків в тому , що коли ми викликаємо одну з функцій, він повертає новий потік на базі батьківського. Це називається незмінюваністю, що дозволяє викликати ланцюжок функцій.

```
  clickStream: ---c----c--c----c------c-->
               vvv map(c становится 1) vvv
               ---1----1--1----1------1-->
               vvvvvvvvv scan(+) vvvvvvvvv
counterStream: ---1----2--3----4------5-->
```

Рисунок 3 – візуалізація події, представлення по функціям

Кожне значення, яке прийшло до нас з кліком, ми замінюємо на одиницю за допомогою функції map. Функція scan поєднує попередній клік з поточним і акумулює, внаслідок чого функція counterStream показує кількість кліків (рис. 3). Що б обробити подію «подвійний клік» потрібно приймати велику кількість натискань за подвійні. В імперативному підході нам би знадобилося інтервали, стан зберігання змінних і самі змінні. У реактивному

програмування це завдання вирішує в 4 рядки коду.

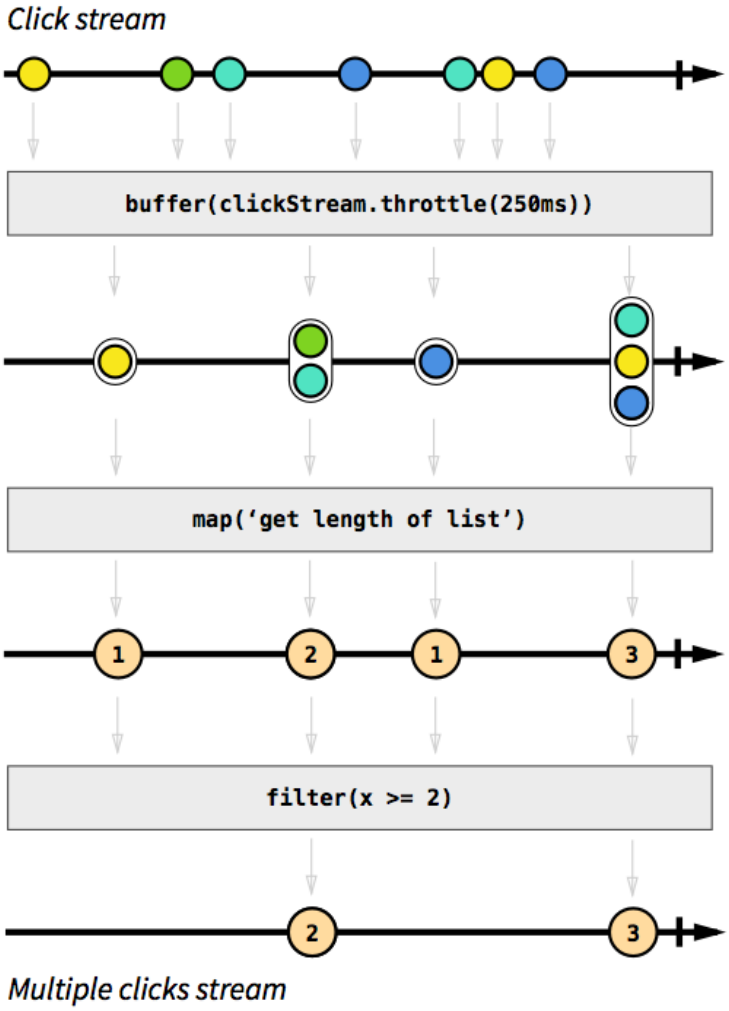


Рисунок 4 – обробка події

По початку ми збираємо кліки в список. Якщо за певний час не відбулося жодного натискання, ми застосуємо функцію map для кожного зі списку, для підрахунку його довжини. Далі відбувається фільтрація всіх кліків що менше 1. За допомогою підписки ми можемо надалі використовувати кліки (рис.4).

Міняти код в повній мірі немає сенсу, якщо це не принесе нам будь-які бонуси. Але реактивне програмування дає значні переваги:

Полегшення при зворотньому виклику

- Зворотній виклик гарантує, що функція не буде запускатися до завершення завдання, а буде запускатися відразу після завершення завдання. Це допомагає нам розробляти асинхронний код JavaScript і захищає від проблем і помилок. У JavaScript спосіб створення функції зворотного виклику полягає в тому, щоб передати її в якості параметра іншої функції, а потім викликати її назад відразу після того, як щось сталося або якесь завдання завершилося. Цей механізм має досить багато коду, тому можна виконувати асинхронний виклик за допомогою зворотних викликів.

- Механізм обробки помилок

При роботі зі складними завданнями і HTTP-викликами помилки обробки є серйозною проблемою, особливо за відсутності будь-якого стандартного механізму. Але реактивне програмування допомагає зробити це простіше.

- Простий спосіб для асинхронних операцій

Потокові і асинхронні операції взаємопов'язані. Додаток викликає зовнішній REST API або базу даних, можемо виконувати цей виклик асинхронно. Якщо це зробити, поточний потік не блокується. Ми можемо робити безліч запитів, створюючи один або кілька потоків.

- Один і той же API для всіх операцій

- Функціональний спосіб

Реактивне програмування робить код більш зрозумілим для прочитання, бо в ньому все більш функціонально.

- Більш придатний код для тестування і лагодження

Якщо слідувати принципам реактивного програмування, то на пошук проблем і лагодження коду йде менше часових витрат.

2.2.1. Принципи реактивного підходу

Є так само термін реактивна система, для опису стилю архітектури для доставки реагуючих і реактивних додатків на системному рівні. Всі описи були прописані в деякому реактивному маніфесті. У нього входять 4 основних характеристики.()

Рисунок 5 – принципи реактивного підходу

1.Чуйність-обробляти події за якомога менший проміжок часу.

2.Стійкість- збої в системи не принесуть системі великих пошкоджень.

3. Еластичність-мати можливість масштабувати в різних напрямках.

4.Управління повідомленнями-встановлений між компонентами кордон, що забезпечувати малу зв’язність , прозорість розміщення (рис.5).

Хоч надалі реативное програмування забезпечує дійсно спрощену модель управлінням асинхронними потоками, створити подібного типу систему досить складно. Для полегшення даного процесу були розроблені фреймворки. Надалі ми будемо розглядати один з них - React.

2.3. SPA-додаток

Слід розглянути також і SPA-додаток на базовому рівні для подальшого розуміння. Це web-додаток, які використовують HTML-документ як оболонку для всіх-сторінок, що дозволяє веб-сторінкам обмінюватися інформацією з користувачами через HTML, CSS, JS. [8]

SPA-архітектура полягає в таких елементах як: View, Model і Storage. З View користувач здійснює взаємодію, для зберігання інформації використовують Model. Являє собою набір функцій і даних, необхідні для роботи з подіями. І зберігається вся інформація в пам'яті саме в Storage. Для підтримки цілісності інформації постійно відслідковуються дії в моделі.

Особливість односторінкових додатків полягає в модульній структурі. За допомогою такого підходу спрощується рефакторинг і тестування, покращує і спрощує обслуговування програми. Дана структура дозволяє розділяти додаток на окремі пакети, кожен пакет-файл і модуль.

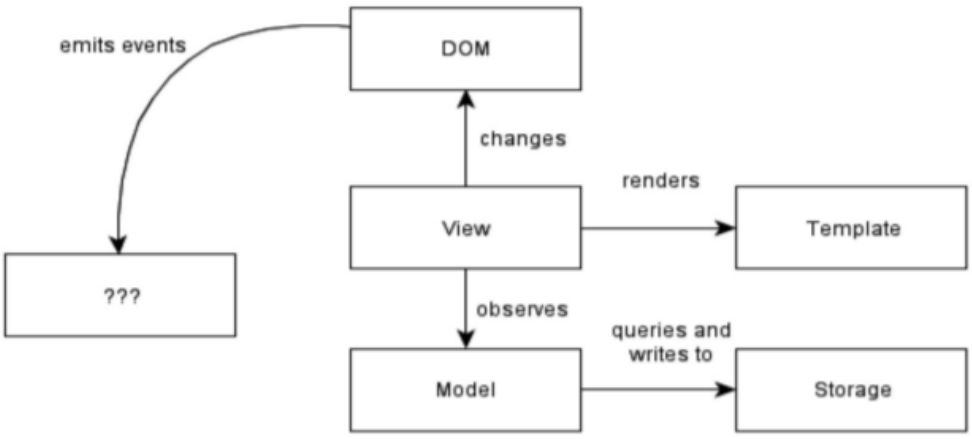


Рисунок 6 – архітектура SPA-додатку

2.4. Різновидності веб-фреймворків

Розглянемо різноманітні фреймворки JS [9]:

1.React

На даний момент часу він є лідером у цій галузі. Спершу Facebook Facebook називався Ads, так як співробітники Facebook почали працювати над розробкою, що б спростити роботу. Концепція React полягає в тому, що компоненти могли багаторазово використовуватися . Кожен компонент відповідає за певну частину сторінки. Використовує синтаксис XML - JSX (поєднує в собі JS і HTML). Даного фреймворку додатки вміщує на одній веб-сторінці і не потребують постійного оновлення з веб-браузера.

Перевага:

- гнучке рішення використання
- зрозуміла модель компонентів
- рендеринг може бути і на стороні сервера

Недоліки:

- потреба в інструменті збірки

- є деяка проблема з несумісністю з іншими бібліотеками, так як використовує DOM.

2.Angular

Angular фреймворк від Google Компанії, який був створений для клієнтських додатків. Зроблено в першу чергу для розробки SPA (Single Application Page) сторінок. Він є спадкоємцем іншого фреймворку AngularJS, але потрібно сприймати як окремо новий фреймворком так як в ньому присутній двустроннє зв'язування, що дозволяє нам в одному місці міняти

дані, при зміні даних моделі в іншому. Особливо фреймворку входить те, що в якості мови програмування використовує TypeScript. Контент оновлюється динамічно. [10]

Переваги:

- універсальний фреймворк - будь-який тип програми за допомогою нього може створюватися.
- функціональність дозволяє розробникам збільшувати можливість HTML.
- є можливість робити посилання даних до HTML-вмісту

Недоліки:

- не обробляє маніпуляції з DOM.

3.Vue

Фреймворк для створення користувацьких інтерфейсів. Його відмінність від інших полягає в тому, що він цілком підійде для поступового впровадження. Основне завдання, яке він вирішує, в першу чергу, це завдання рівня представлення, що полегшує взаємодію з іншими бібліотеками. Він також підходить і для SPA додатків. Так само як і інші, концепція полягає в компонентах. Тобто сторінку представляє як дерево компонентів. Компонент можливо використовувати в шаблоні іншого компонента. Найбільша перевага даного фреймворку це відкритий вихідний код.

Переваги:

- малий розмір

- висока швидкість розробки

- оптимізована обробка HTML-блоків

Недоліки:

- оновлення залежить від одного розробника

- відстежує зміни при деяких умовах.

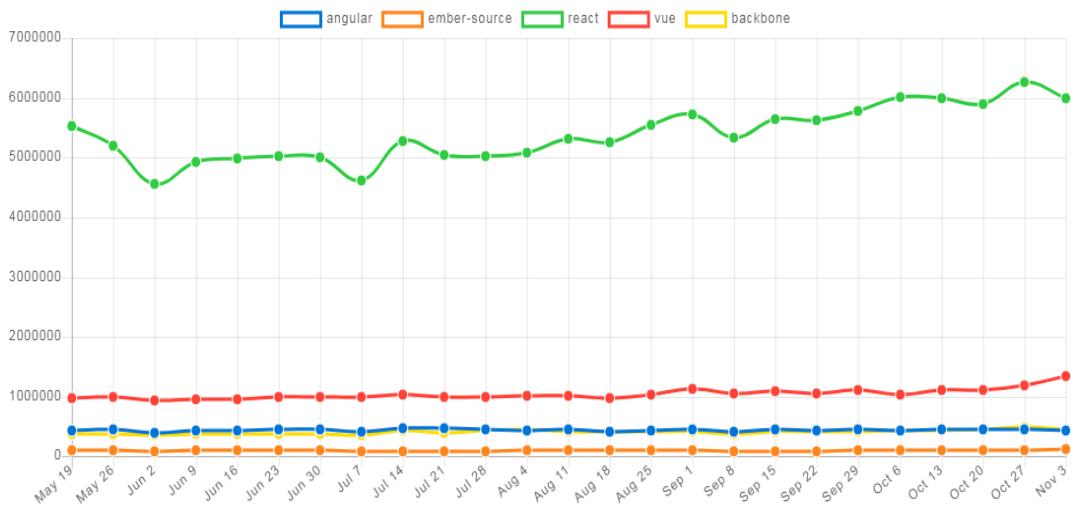


Рисунок 7 – Обсяги завантажень пакетів angular, ember-source, react, vue і backbone

Stats

	stars	forks	issues	updated	created	size
angular	59604	28878	464	Oct 22, 2019	Jan 6, 2010	minzipped size 61.9 KB
ember-source	21253	4196	326	Nov 7, 2019	May 26, 2011	minzipped size 325.2 KB
react	139001	26379	902	Nov 6, 2019	May 24, 2013	minzipped size 2.6 KB
vue	151663	22554	395	Nov 5, 2019	Jul 29, 2013	minzipped size 22.8 KB
backbone	27552	5675	88	Nov 6, 2019	Oct 1, 2010	minzipped size 13.5 KB

Рисунок 8 – Порівняння пакетів angular, ember-source, react, vue і backbone

2.5. React

Інтерфейс користувача в будь-якому веб-проекті є основним завданням.

Зовсім неважливо як за підсумком буде використовуватися сайт, важливо, що б користувачеві було зручно користуватися. У цьому також полягає робота не тільки веб розробників, але і так само UI дизайн сайту. Значну роль у зберіганні та обробці даних відіграють серверні компоненти, такі як PHP та база даних для динамічної генерації сторінки. Що стосується інтерфейсу користувача, то більшою мірою залежить від HTML, CSS і JS.

За допомогою JS ми можемо домогтися однакового відображення сторінок в різних браузерах. Це і є основне завдання, що змушує ламати голови вебмайстрів і власників сайтів.

React підтримує Facebook, Instagram і служить він для побудови користувацьких інтерфейсів. Головна мета його-забезпечити масштабованість проекту, простоту і високу швидкість розробки. Часто цей фреймворк використовують разом з Redux бібліотекою. Також надає callback-функції для відтворення HTML і деякі шаблони. Результат роботи React-це HTML.

Концепції React заключаються на:

Компоненти – зв’язка HTML і JS, які зберігають стан в памяті. Мають свою структуру та функціональність.

Елементи – це HTML розмітка, яка описує в DOM – елементи, наприклад h1, div.

JSX-на ньому пишуться React компоненти. За допомогою нього додається менше зусиль при управліннями елеметами і запуску в браузер, так як трансформується в JavaScript.

Virtual DOM - це деревоподібна структура елементів на JS, за ним стежить React і вимальовує поточні зміни в браузер, що б відповідав віртуальному. Оновлення робляться автоматично за допомогою diff-алгоритму. Таким чином позбавляється додаток від гальм завдяки

найменшу кількість операцій.

Для відтворення віртуального елемента ми повинні повідомити React про це і відмалювати. Це робиться функцією render (), що приймає віртуальний елемент і реальний вузол DOM. React вставляє віртуальний елемент в зазначений вузол і ми бачимо в браузері зображення.

Якщо властивості компонента постійно в процесі змін слід використовувати стан – це об'єкт всередині компонента, що зберігає змінені дані . У React надана функція useState, яка допомагає йому з цим.

Компоненти дозволяють розбивати на незалежні частини сайт, і в одну одиницю часу думати про конкретно один компонент, як він повинен себе вести при певних діях. Так само компоненти можна використовувати багаторазово, це велика перевага для величезних проектів з повторюваними частинах, позбавляє від написання додаткового коду, тобто можна їх розглядати як функцію, яку можемо викликати в певних частинах коду при певних відповідних обставинах.

Компоненти діляться на два типи: класові і функціональні. Відмінності між ними полягає в тому, що в класових ми можемо зберігати стан всередині, тобто він буде містити метод render (), що повертає React-елемент, а функціональні – ні. Функціональні звертаються до props і вже за допомогою них приймають в компоненту певні значення від батьківського компонента.

```
function Welcome(props) {
    return <h1>Hello, {props.name}</h1>;
}
```

Рисунок 9 – Компоненти-функції

Рисунок 10 – Компонент-клас

У своїх вихідних даних компоненнти можуть посилатися на інші компоненти

Фреймворк включає в себе ідею рективного програмування: стан в компоненті якщо змінюється, то всі залежні від нього також перемальовуються. В даному випадки використовується односпрямований зв'язок даних, що передаються зверху вниз по дереву. Такий підхід дає можливість організовувати прості зв'язки між компонентами. [11]

2.6. Висновки до розділу

В данному розділі ми детально вникли в різновидність веб-фреймворків, та дійшли висновку, що для даного проекту більш доречно використовувати React. Також розглянули переваги використовувати реактивне програмування та його основні концепції.

Особливості та архітектуру SPA-додатків. Різновиди веб-фреймворків, які на сьогодення найпопулярніші:

- React
- Vue
- Angular

Провели порівняльний аналіз між роботами фреймворків, та виявили найшвидший з них. Розібралися в особливостях React, його основних концепціях, чому саме його найчастіше використовую в проектах.

3 Візуалізація данних

3.1. Вступ

Кожна людина сприймає світ по різному через 5 органів почуттів. За допомогою них ми чуємо-бачимо-відчуваємо інформацію через канали сприйняття і робимо аналіз її. Але є і провідна модальність, тобто той канал, через який найкраще за все інформація сприймається і у кожної людини вона своя. [12]

Виділяють основних три типи сприйняття:

-візуальний

- аудіальний

- кінестетичний

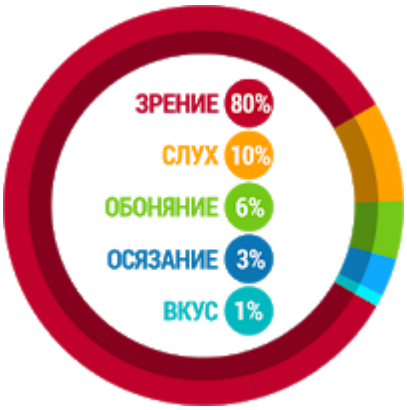


Рисунок 11 – різні типи сприйняття людиною у відсотковому відношенні

Для кожного типу передбачений свій підхід запам'ятовування інформації. Аудіали сприймаю за допомогою слуху, кінестетики через інші органи чуття: дотик, нюх і рухів (рис.11). Візуали більшу частину часу сприймають світ за допомогою зору і їх переважна більшість. [15] саме по візуальним образам проводяться багато досліджень на відміну від інших

сенсорних модальностей. Бачення виділяється з ряду інших ще й тим, що воно найважливіше, і в той же час найскладніше. Чому ж почали вивчати саме даний тип сприйняття, аргументують двома речами:

- існують сучасні технології, які підходять для даної мети краще, ніж для вивчення інших органів.
- є результатом того, як влаштоване наше суспільство, і, отже, на нього сильно впливає прийняття рішень людиною.

Комбінуючи висновки методологічно-структурному пояснення і культурного, можна зробити узагальнення, що візуальна частина, найбільш широко дає інформацію людині, ніж інші джерела. Отже, розуміння сприйняття візуалу на людину дозволяє дизайнерам змушувати відчувати людину те, що вони хочуть донести до глядача.

Якщо розглядати зорове сприйняття фізіологічно, то воно відбувається, коли світло виявляється на сітківці ока. Фоторецептори, що знаходяться на сітківці, перетворюють світло в ряд електрохімічних сигналів, які передаються в подальшому в мозок. Були проведені дослідження Массачусетського інституту, що передача сигналів відбувається всього за 13 мілісекунд. [12]

Мозок постійно знаходиться в процесі розуміння світу, комбінуючи-розділяючи-порівнюючи ті образи, що до нього приходили раніше. Він це робить для заповнення прогалин інформації створюючи загальний обрис картини світу. З допомогою таких гештальт-принципів (як люди сприймають візуал) дизайнери створюють зрозумілий UI для графічних інтерфейсів, дозволяючи визначати що за візуальні елементи будуть доречні в даному випадки і як ефективно управляти увагою людини, змінювати поведінку користувача, коли це потрібно. [13]

3.2 Гештальти сприйняття даних людиною

Гештальтом називають групу принципів, за допомогою яких ми можемо впливати на сприйняття інформації людиною. За допомогою цього методу намагаються пояснити як саме людина відділяє чи комбінує інформацію.

Розглянемо деякі з них [13]:

- Близькість

Більш зв'язаними об'єкти сприймаються якщо між ними мало простору . Таким чином комбінують об'єкти по групах.

- Загальні області

Згрупованими об'єктами вважаються так само якщо вони знаходяться в одній загальній області.

- Схожість

Елементи, які мають загальні характеристики по візуалу, сприймаються більш пов'язаними .

- Замкнуте / завершення

Складні елементи, що ми бачимо намагаємося побачити впізнавані форми. До того ж, якщо об'єкт неповний ми в мозку його намагаємося закінчити

- Симетричність

Симетричність дає нам відчуття порядку в зображенні і сприймаються як належать один одному.

Існують і інші принципи ,але це основні. За допомогою правильного розташованого візуалу ми можемо спілкуватися з користувачами , та комфортно використовувати продукт, тим самим досягати більшого успіху компанії.

Правила запам'ятовування тексту і зображенням різняться між собою. Феномену дали назву ефект перевагу образу. Були проведені дослідження, які показали наскільки точно людина переглядаючи всього 10 секунд картину, а їх було дві з половиною тисячі, зможе відтворити надалі. Дослідження вразило, точність виявилося дорівнює 90 відсоткам. Через рік точність розпізнавання знизилася і дійшла до 63 відсотка. У будь-яких експерементах на тривале запам'ятовування візуал був в лідерах. І ця тенденція все так само зберігається. [14]

Що стосується Інтернету, то кожен сайт бореться за утримання уваги користувача. Більшість людей не люблять сайти з сухим текстом, вони женуться за візуалом, чим менше тексту і більше картинок, тим простіше віддають їм перевагу, оскільки простіше інформація сприймається. Це вплинуло і на такі технології як створення діаграм і графіків, як візуалізація інформації з більш простим поглинанням. Передача інформації може бути як красивою, так і функціональною. Інтерактивність графіків і творчість уяви дає можливість виробляти насичені і глибокі дослідження, одночасно приголомшуючи аудиторію естетикою.

3.3. Бібліотеки візуалізації в JavaScrip

У світі кількість даних постійно зростає і з'являється потреба їх обробляти. І чим більше її стає, тим очевидніше стає той факт, що дані для більш швидкого сприйняття потрібно візуалізувати. Розробники задумувалися над тим як же все таки об'єднати петабайти інформації баз даних в одному місці і прийшли до рішення використовувати діаграми, графіки, дашборди, які

до всього були б ще й інтерактивними. За допомогою них зменшується швидкість сприйняття інформації і сил на сам перегляд, на підставі цього зробити свідомий інформаційний вибір чого небудь.[15,16]

Останнім часом візуалізація даних прогресує і існує безліч відповідних бібліотек. Раніше, в 2000-і роки, в візуалізації були найбільш популярні растрові зображення. Для інтерактивних діаграм використовували на той момент популярні плагіни Flash і Silverlight, але було безліч недоліків: швидкість завантаження була повільною ,велике споживання батареї і системних ресурсів.

Далі прийшла ера використання мобільних пристроїв і дані плагіни звелися нанівець. З'явилися екрани з дуже високою роздільною здатністю і потреба використовувати векторні діаграми, оскільки кількість контенту за допомогою жестів переважала, до того ж були б незалежні від дозволу екрану. Це і послідувало поштовх для розвитку нових фронтенд-технологій, які могли запускатися на різних платформах.

На сьогоднішній день першість у візуалізації тримає JavaScript і SVG. Діаграми стали виглядати дуже чітко на різних пристроях, незалежно від розширення екрану, відпала потреба використовувати додаткові плагіни для інтерактиву і анімацій, до того ж працюють у всіх браузерах.

Безліч бібліотек для візуалізації на JavaScript було створено, кожна з них мають свої плюси і мінуси, як і кожен інструмент у своїй сфері. Для більш ефективної роботи і вибору бібліотеки потрібно визначиться з метою його використання. Розглянемо деякі з них.

- AmCharts

Являє собою просте і гнучке рішення для візуалізації даних. Кількість діаграм різноманітний і широка площа можливостей для кастомізації.

Основними можливостями виділяється :

- Дрилдаун аналіз даних (провалюватися на більш деталізований рівень) та інші інтерактивні опції.

- Непогана анімація графіків.

- Експорт графіків в зображення.

Можна використовувати і безкоштовно, але брендинг буде заважати виду графіка, для цього потрібно буде купити платну ліцензію.

- AnyChart

Легка JavaScript бібліотека візуалізації з рендерингом SVG . Підтримує більше 70 типів графіків і діаграм.

Основними можливостями виділяється :

- Велика кількість варіантів як завантажити дані JS API, JSON,CSV, XML.

- Підтримка ранніх версій браузерів

- Доступний ряд встановлених варіантів виду графіків, є можливість налаштовувати колірні схеми під себе.

Як і в попередньому випадком, безкоштовна версія є, але з вотермаркою. Для використання в комерційних проектах потрібно придбати ліцензію.

- Chart.js

Досить популярна серед розробників, все через те, що має в собі базове рішення для графіків і діаграм, не перевантажена додатковими складними

ефектами, індивідуальними налаштуваннями. Графіки виглядають інформативно і акуратно. З легкістю дозволяє комбінувати різні типи діаграм.

Основними можливостями виділяється:

- Можна розширювати за допомогою додаткових плагінів.

- Адаптивні графіки при роботі в мережі.

- 8 основних типів графіків і діаграм (Line, Area. Bar та інші).

Хороша бібліотека з відкритим вихідним кодом, можна використовувати у вільному доступі.

- Chartist.js

Головна мета даної бібліотеки: звести все до мінімуму,тому має при собі всього лише три елементарних графіки і не вимагає багато чого в оформленні виду даних. Проста бібліотека без додаткових замудрених фіч.

Основними можливостями виділяється :

- 3 типи графіків: Line, Bar, Pie.

- Є можливість дозавантажувати плагіни.

- SVG використовує для відтворення діаграм і графіків.

- Побудований за допомогою SASS

- Чуйний і незалежний від DPI.

Відкритий код з вільним використанням.

- D3.js

Найпотужніша і популярна з усіх бібліотек. Вона більше схожа на фреймворк ,ніж на звичайну бібліотеку, як по функціоналу ,так і по початку з нею роботи, але є достатня кількість посібників, які згладжують даний невеликий мінус. З її допомогою можна створювати по істині вражаючі і оригінальні візуалізації. Якщо дивитися по цифрам, то щотижня скачується приблизно по 1 444 538 разів, в той час як інша популярна бібліотека Chart.js 1 351 478 разів. Так само може проводити маніпуляції DOM, надаючи можливість підлаштовувати в сучасних браузерах правильну UI архітектуру.

Основними можливостями виділяється :

- Велика кількість посібників для вивчення деталей роботи з бібліотекою.

- Генерація кривих.

- Має найбільшу кількість типів графіків і діаграм, ніж переважна більшість бібліотек.

- Drag-and-drop GUI (дозволяє користувачеві затиснути елемент і перетягнути його на інше місце).

Що робить цю бібліотеку дійсно ідеальною, так це те, що бібліотека абсолютно безкоштовна для застосування в будь-яких цілях.

3.4. D3.js

У сучасному світі велику роль відіграє популяризації концепцій та ідей за допомогою візуалізації даних. Багато з побутових приладів, такі як смартфон, ноутбук та інші інтернет речі, виробляють велику кількість даних. Отже виникає потреба в зберіганні та аналізі їх.[17]

D3 одна з найпотужніша і гнучка система візуалізації даних. Гнучкість

візуалізації коштує дорого, оскільки крива навчання дещо крута. Зазвичай дані не дають просто візуалізувати, у багатьох випадках потрібно знати, де знайти дані, як їх очистити, вивчити і вибрати кращу візуалізацію для їх представлення.

Візуалізація дає швидке розуміння ситуації на графіку, і так само дозволяє побачити те, що в іншому випадки залишилося б непоміченим, або викликати додаткові питання, які без графіка ми б не сформулювали.

Бібліотека була створена в 2007 році, її використовують для візуалізації , обробки даних і вирішують з її допомогою нестандартні і складні завдання побудови графіків і діаграм. Назва розшифровується як Data-Driven Documents, що дає нам зрозуміти, що бібліотека акцентує свою увагу на управлінні даними. Однак основний функціонал бібліотеки спрямований на візуалізацію.[18]

Бібліотека працює з будь-яким браузером, включаючи мобільні пристрої. Крива навчання компенсується високою гнучкістю і дозволяє не прив'язуватися до якоїсь певної візуальної форми. Під гнучкістю бібліотеки так само мають на увазі і маніпулювання будь-якою частиною об'єктної моделі документа, це дає величезну перевагу перед іншими інструментами.

Позиціонує себе як низкорівнева бібліотека. З моменту появи цієї бібліотеки стала де-факто для побудови складних візуальних даних в Інтернеті.

Існує безліч способів представлення даних в залежності від мети:

-Відображення на лінійному графіку

-Порівняння значень на гістограмі

-Розподіл подій на тепловій карті.

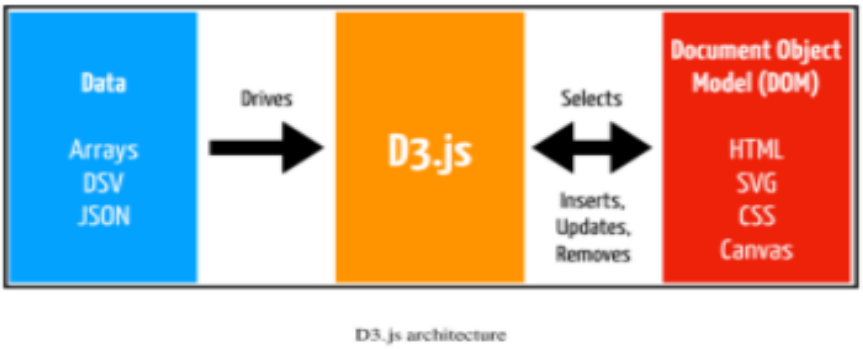
-Відсотки від загального числа на круговій діаграмі.

На відміну від інших великих візуальних бібліотек, ця бібліотека не патерн з готовими рішеннями. D3.js надає великий API (http: //bit.ly/pro-d3-api), який включає в себе, серед інших функцій:

- Математичні функції для розрахунку складних креслень (d3-chord, d3-hierarchy)
- Функції для роботи з масивами даних і структурами даних (d3-array і d3-collection)
- Функції для вибору і перетворення елементів DOM (D3-selection)
- Примітивні функції для створення елементів SVG (d3-shape)

D3 завантажує дані і приєднує їх до DOM. Потім пов'язує дані з елементами DOM і перетворює елементи, при необхідності переходячи між станами. Вони дозволяють вибирати елементи DOM за допомогою селекторів CSS і працюють над ними. Ми можемо змінити атрибути та стилі або додати нові вкладені елементи всередині обраного елемента.

Архітектура бібліотеки виглядає таким чином.(рис.) Так як бібліотекою керує концепція «дані керують документами», то за допомогою змін даних, таких як додавання, змінювання чи видалення , ми впливаємо на те, як саме діаграма буде виглядати[19]:



D3.js architecture

Рисунок 12 – D3.js архітектура

Дані надаються або у вигляді масиву JavaScript, або локально, є також варіант завантажити з файлу окремо. Для вибору елементів HTML або SVG застосовується селектори CSS , пов'язуючи їх з елементами даних,за необхідністю можна додавати графічні елементи автоматично.

Селектори застосовуються для виділення об’єкту, який потім застосовують для зміни стилю елемента, чи додавання будь-яких атрибутів до обраних елементів. Привязка масиву до виділеного обєкту поєднує кожний елемент даних з елементами виділення. Доступ до виділених елементів отримується за допомогою зворотніх функцій, що використовується в методах, які задають значення для стилів , атрибутів та перетворень.

Зручно використовувати виділення ще не існуючих елементів в DOM. Прив’язка цього виділення до даних автоматично створює по одному елементу для кожного елемента даних. Можемо надалі використовувати дані для додавання вмісту, стилів та атрибутів для нових елементів.

D3.js зберігає дані та їх елементи в sync, при оновленні з меншою кількістю даних надлишкові елементи поміщаються в окремий масив, тому їх просто можно видалити. Якщо набір даних зростає ,існуючі елементи оновлюються, а відсутні створюються і додаються відповідно з наявними даними.

Документація достатньо добре пояснює як зробити ті чи інші речі, на головній сторінці можна натиснути на один з шестикутників, на якому намалювана конкретна візуалізація якогось графіку, і перед вами зявиться його вихідний код. Більшість з них публікуються на GitHub.

3.4.1 Модулі

Не має потреби завантажувати цілу бібліотеку D3. Оскільки бібліотека модульная, тому якщо в програмі використовується тільки окремі функції, ми можемо включити тільки їх. Розмір зменшеного пакета за замовчуванням становить близько 240 КБ, але ми можемо створити діаграму, використовуючи всього 13 КБ, якщо не потрібно багато інших модулів.

Але якщо потрібно осі, карти та інші функції, в такій ситуації потрібно більше модулів і залежностей. Можна використовувати пакет за замовчуванням, або налаштувати середовище розробки, в якому можна встановити кожний модуль за допомогою npm, оскільки він автоматично включає будь-які залежності.

Навіть якщо постійно тільки і використовуєте пакет за замовчуванням, бажано знати про модулі, які він містить , які не є частиною пакета за замовчуванням, оскільки вони повинні бути імпортовані окремо у випадку, якщо потріні їх функції. Вся документація також організована по модулям, і знання модульної структури дозволить налаштувати продуктивність програми, якщо буде потреба створити користувальницький пакет. Модулі мають свої власні системи управління версіями. Можливо, буде потрібно завантажувати модуль окремо , якщо потрібно використовувати функцію, включену в нову основну версію, що не включена в комплект за замовчуванням.

На наступній діаграмі (рис.13) показані модулі, доступні в d3js версії 5, із зазначенням прямих залежностей (транзитивні залежності не показані).

Наприклад, якщо вам потрібно використовувати функцію з модуля d3-scale, ви повинні імпортувати всі прямі залежності.

У цьому випадку слід або використовувати npm, або імпортувати пакет за замовчуванням.

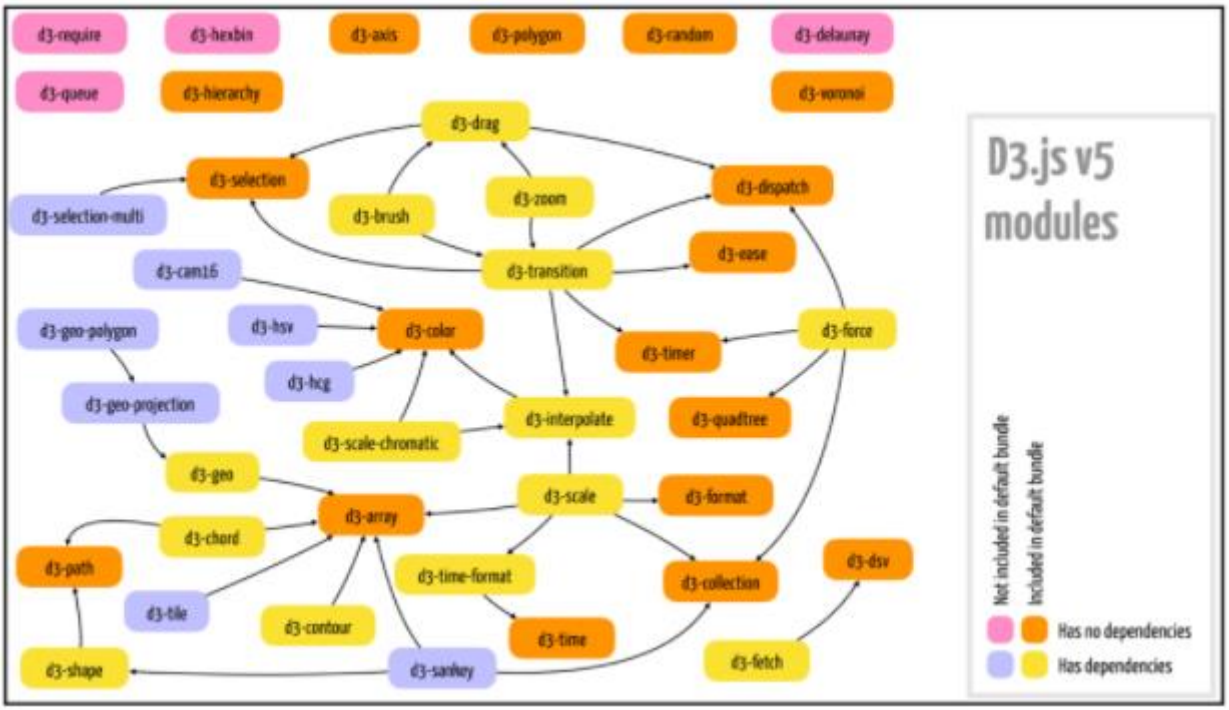


Рисунок 13 – Взаємозв'язок модулів в D3.js

У наступних таблицях міститься короткий огляд всіх модулів, доступних в поточній версії, класифікованих відповідно до їх призначення. Наведені нижче модулі використовуються для створення, обробки, перетворення, аналізу та форматування даних у вигляді чисел, тексту, масивів, об'єктів і файлів. Всі вони включені в значення за замовчуванням(табл.3):

d3-array	Кілька утиліт масиву, що розширюють основні функції ES6,

	оптимізовані для використання з наборами даних.
d3-collection	Карти та набори, оптимізовані для використання з наборами даних; функції для колекцій об'єктів і вкладених даних.
d3-random	Генератори випадкових чисел
d3-dsv	Функції синтаксичного аналізу для даних, розділених роздільниками.
d3-interpolate	Кілька функцій для інтерполяції чисел, кольорів, рядків і так далі
d3-scale	Функції генератора для зіставлення вимірювань даних з графічними вимірювання.
d3-time	API для операцій з часом (інтервали, діапазони тощо)
d3-format	Методи форматування чисел, що залежать від локалі. Залежність: ніхто.
d3-time-format	Locale-чутливі методи форматування дати і часу.

Таблиця 3 -Модулі з методами для маніпуляції, трансформації,

парсінгу та форматування дат.

3.5. Висновки до розділу

В данному розділі ми розглянули різновиди бібліотек для візуалізації даних, та дійшли висновку, що для покриття всіх потреб проекту доречно використовувати бібліотеку D3.js. Вона, на відміну від інших великих візуальних бібліотек, не патерн з готовими рішеннями. Ви самі керує тим, як потрібно, щоб графік виглядав. Тому рівень входження достатньо високий.

Також було розглянуто як саме людина сприймає інформацію, чим керуватися створюючи комфортний інтерфейс продукту. Порівняли основні бібліотеки сьогодення,а саме:

- AmCharts
- AnyChart
- Chart.js
- Chartist.js
- D3.js

4 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

4.1 Вступ

Візуалізація головне завдання при розробці веб складової програми. Вибір був зроблений на бібліотеку D3.js і використання веб-фреймворку React. Бібліотека містить в собі великий список можливостей, побудова різних типів діаграм (лінійні, кругові і тд.), так само включає роботу з картографічними даними. На користь вибору D3.js є обґрунтовані причини, наведені нижче[20]:

1. Працює з DOM, що дає нам додавати ,видаляти, в цілому видозмінювати елементи в додаток. Великий плюс так само присутній у відмінній читабельності коду.

2. Важливим фактором так само грає, що бібліотека має відкритий код, що дає можливість її використовувати в комерційних цілях.

Бібліотека виявилася відмінним вибором з боку функціоналу, так і за формальними ознаками. Функціонал розбит на блоки, тому ми використовували лише те, що потрібно в даній роботі, щоб не перевантажувати додаток,а саме: d3-geo, d3-scale, d3-selection.

Що стосується фреймворку React, це ідеально рішення, так як маємо можливість масштабувати проект і підвищувати швидкість розробки програми.

4.2 Проблематика програмного продукту

React і D3 мають деякі недоліки в спільному використанні : кожен з них бере під контроль елементи Dom-дерева. D3.js спершу дані довантажує і чіпляє за DOM. Далі ці дані зв'язуються з елементами переходячи між станами. Вже скомбіновані дані діляться на три типи: елементи, які потрібно створити,

елементи, що потрібно оновити і ті, які необхідно видалити. React ж має віртуальний будинок, який представляє поточний стан програми і за допомогою алгоритму диффа оптимізує роботу ререндеринга.[21]

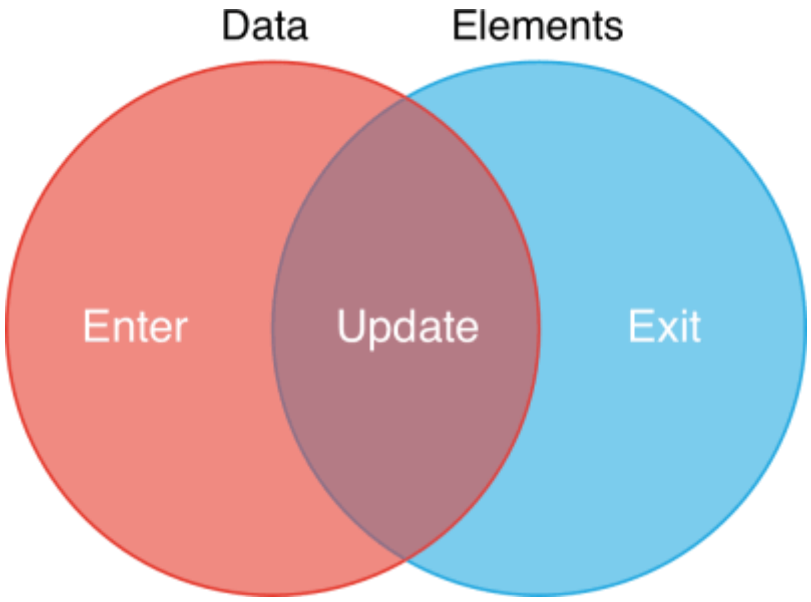


Рисунок 14 – Комбінування елементів

Візуалізуючи компоненти, кожному елементу ми повинні присвоїти унікальний ключ-значення, за допомогою якого реактор з'ясовує, чи потрібно перерендерити елемент. Цей алгоритм перевіряє ключ і дає зрозуміти чи потрібно додати або видалити елемент. Це дуже схоже на приєднання даних за допомогою D3.js. Ця бібліотека вирішує головну проблему: маніпулюванням документом на основі даних, що були надані. Таким чином створює гнучкість для всіх можливих веб-стандартів: HTML, SVG і CSS.

Кожен з робочих інструментів віддають перевагу чистим функціям ,тобто дають постійно однаковий результат з одними тими ж вхідними даними, незалежні ні від чого, і компонентам без внутрішнього стану в них. Щоб дозволити працювати даним інструментам потрібно щоб вони ніколи не мали можливості ділити DOM управління спільно.

Для усунення даної проблеми ми використовували React для структури (рендеринга HTML-контейнера), в той час як D3.js всі наступні операції всередині створеного контейнера, що був створений React.

Даний метод має свої плюси і мінуси.

Плюс:

- Є можливість використовувати D3.js з усім функціоналом для конкретного Dom-елемента.
- Для раніше створений компонентів D3 легко впроваджується в роботу нові

Мінус:

- Для тривалих проектів не підходить, так як важко-тестований код.
- Оптимізація React не використовується у всій своїй потужності.

4.3 Загальний вигляд інтерфейсу

За допомогою зору ми отримуємо 80% інформації сприйняті з навколишнього світу. Отже можна зробити висновок, що найефективніший спосіб розуміння даних – створити візуальні графіки. У даній роботі було створено веб-додаток для візуалізації даних на карті за допомогою D3.js і GeoJSON. Створила інтерактивну карту, що відображає якість повітря України в реальному часі.[23] Створюючи карту нам знадобилися координати меж областей. Далі ми приступили до розробки програми. Веб-додаток розроблялося на мові JavaScript, на компоненти розбивати мені допоміг фреймворк React, а візуалізацію даних було зроблено за допомогою бібліотеки D3.js. Для відтворення карти використовувалася мова розмітки векторної графіки SVG. Переваги даного формату в тому, що стиль зображення ми можемо коригуватися за допомогою CSS. Карта відображає динамічні дані. Popup для відображення даних в конкретній натиснутому місці відображає

рівень забруднення повітря. За допомогою шкали, що знаходиться зверху праворуч екрану веб-додатки ми можемо зрозуміти наскільки забруднене місце в даний момент часу. Наводячи на конкретний контур області висвічується назва цієї області.

Рисунок 15 – Скріншот проекту

4.4. Порівняльний аналіз

Є також і інші фреймворки, але вони використовуються набагато менше, ніж даних три лідера. Проведемо порівняння між ефективністю рішення кожного з них.

Одним з важливих характеристик веб-додатків це продуктивність. На цей показник вплинути може вибір використовуваного готового рішення. Для тестування використовували js-framework-benchmark.Тест проводился на ноутбуке ASUS X54A:

– Тип процесора: Intel Celeron B815 1.6ГГц

– Оперативна пам’ять (RAM): 2ГБ

– Операційна система: Windows 7 Ultimate

– Браузер: Версія 74.0.3729.131 (32-bit)

Результати тестування в таблиці. JavaScript в таблиці означає, що ніякий з фреймворків не використовується, тобто певна початкова точка відліку. Час вказано в мілісекундах. Тестування створюється за допомогою ключа тобто створюється асоціація між даними DOM-елемента і домену за допомогою "ключа". Якщо дані змінюються, DOM-елемент с цим ключем буде оновлений. Після вставки або видалення елемента в масиві даних в DOM відбуваються відповідні зміни (табл. 1).

Дія	JavaScript	React	AngularJS	Vue.js
Створення рядків створення 1000 рядків	427.869	524.823	510.866	408.723
Видалення рядків видалити 1рядок з 1000	278.304	495.127	366.996	308.221
Поміняти рядки місцями поміняти місцями 2 рядки з 1000	43.649	51.832	245.982	36.876
Вибір рядків вибір 1 рядок з 1000	3.889	3.723	4.556	3.281
Додати рядки додати 1000 рядків	480.490	466.323	985.012	411.998
Очищення рядків Видалення всіх рядків	55.489	158.789	446.008	46.097

Таблиця 1. Результати тестування бібліотеки і фреймворків

За даними результатами велику швидкість в даному виді тестування справив Vue. React і Angular повільніше, але результати досить близькі. Тільки за результатами продуктивності вибір фреймворку не буде проводитися, тому кожне рішення має свої плюси і мінуси. Спиратися варто на те, що за проект,

з яким колективом будете працювати(табл.2).

Назва фреймворку	Рендеринг	Архітектура	Направленість даних	Зворотня сумісність
React	замінюються тільки ті частини сторінки, які відрізняються від результатів обробки поточного DOM дерева	-	Одностороння спрямованість даних.	Повна сумісність між версіями.
Vue	замінюються тільки ті частини сторінки, які відрізняються від результатів обробки поточного DOM дерева	MVC	Одностороння спрямованість даних.	Повна зворотня сумісність.
Angular	рендеринг відбувається як на сервері так і на клієнті.	MVC	Двостороння спрямованість даних.	Залежність від попередніх версій і компонентів.

Таблиця 2. Порівняння фреймворків і бібліотек

4.5. Висновки до розділу

React і D3.js є відмінними інструментами, які допоможуть нам впоратися

з DOM і його проблемами. Вони, безумовно, можуть працювати разом, і ми уповноважені вибирати, де провести грань між ними. Велика екосистема бібліотек існує, щоб допомогти нам використовувати D3.js. Багато цікавих варіантів є для React-D3.js теж, і обидві бібліотеки знаходяться в постійній еволюції, так що проекти, що поєднують обидва буде важко йти в ногу з часом. [22]

5 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, який призначений для аналізу постави користувача. Інтерфейс користувача був розроблений за допомогою мови розмітки JSX у середовищі розробки WebStorm .

Програмний продукт призначено для використання в браузерах під управлінням операційних систем Windows.

Нижче наведено аналіз різних варіантів реалізації модулю задля вибору оптимального, враховуючи при цьому як економічні фактори, так і ті характеристики продукту, що впливають на швидкодію і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.[24]

Метою ФВА є забезпечення правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на їх реалізацію.

Фактично цей метод працює за таким алгоритмом:

- визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі проводиться оптимізація послідовності скороченням тих кроків, що не впливають на цінність і, відповідно, витрат.

- для кожної функції визначаються повні річні витрати та кількість робочих часів.

- для кожної функції, засновуючись на оцінках попереднього пункту, визначається кількісна характеристика джерел витрат.

- наступним кроком проводиться кінцевий розрахунок витрат на виробництво продукту.

5.1 Постановка задачі техніко-економічного аналізу

У даній роботі застосовується метод ФВА для проведення техніко-економічний аналізу розробки системи аналізу нелінійних нестаціонарних процесів. Основні проектні рішення стосуються всієї системи загалом, через що кожна окрема підсистема теж має їм задовольняти. Через це фактичний аналіз являє собою аналіз функцій програмного продукту, який призначений для збору, обробки та проведення аналізу гетероскедастичних процесів в економіці та фінансах.

Відповідно до цього варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

- програмний продукт повинен функціонувати в браузерах на базі операційної системи Windows не нижче 7 із стандартним набором компонент.

- забезпечувати достатню швидкість обробки наявних об’ємів даних у реальному часі;

- забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;

- передбачати мінімальні витрати на впровадження програмного продукту.

5.1.1 Обґрунтування функцій програмного продукту

Головна функція F_0– розробка програмного продукту, який аналізує процес за вхідними даними та будує його модель для подальшого прогнозування. За результатом поставленої мети, можемо виділити наступні основні функції ПП:

F_1 – вибір бібліотеки візуалізації;

F_2 – вибір фреймворку для веб-розробки;

F_3 – вибір мови програмування.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

а) Chart.js;

б) D3.js;

Функція F_2:

а) React;

б) Vue.

Функція F_3:

а) мова програмування JavaScript;

б) мова програмування C++

5.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (Таблиця 5.1).

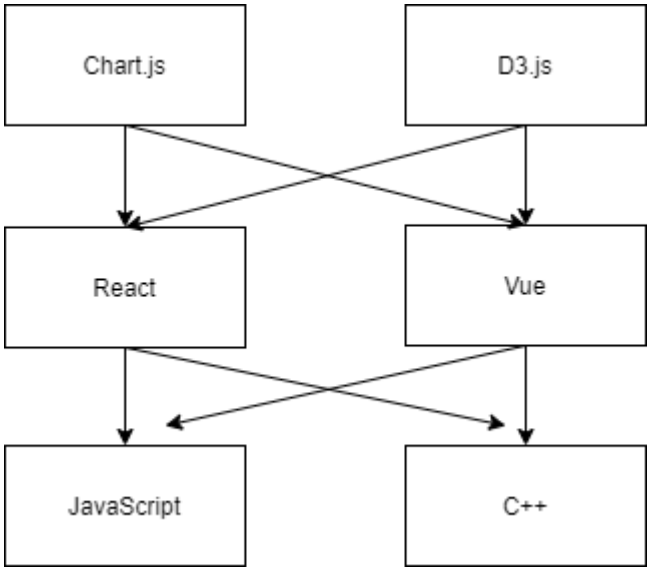


Рисунок 5.1 – Морфологічна карта

Морфологічна карта показує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 5.1 – Позитивно-негативна матриця

Основні функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F1	*А*	Адаптивні графіки при роботі в мережі.	Дає можливість створювати тільки основні базові типи діаграм.
	Б	Пропонує великий вибір варіантів візуалізації	Досить високий поріг входження

F2	А	Дозволяє повторно використовувати частини вже написаного коду	Досить важка для вивчення.
	Б	Підтримує як сторонні бібліотеки, так і має безліч вбудованих модулів для вирішення повсякденних завдань.	Відсутність підтримки великих проектів
F3	*А*	Поширеність	Проблема роботи з типами даних .
	Б	Універсальність	Язык С++ является сложным для изучения и для компиляции.

Аналіз позитивно-негативної матриці дає змогу дійти висновку, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, оскільки вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція *F1:*

Різноманітність реалізації графіків є дуже важливим фактором, тому варіант а) має бути відкинутий.

Функція *F2:*

Оскільки повторне використанная коду є критично важливою для браузеру, то варіант б) має бути відкинутий

Функція *F3:*

Відмінність в об’ємі коду програми є незначною, обидві мови прекрасно підходять для написання додатку, тому обидва варіанти а) та б) вважаємо гідними розгляду.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1б – F2а – F3а

2. F1б – F2а – F3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

5.2. Обґрунтування системи параметрів ПП

5.2.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.[24]

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- *X1* – швидкодія мови програмування;
- *X2* – обсяг пам’яті для збереження даних;
- *X3* – час обробки даних;
- *X4* – потенційний обсяг програмного коду.

X1: Відображає швидкодію виконуваних операцій в залежності від обраної мови програмування.

X2: Показує обсяг пам’яті в оперативній пам’яті , який є необхідним для збереження та обробки даних під час виконання програми.

X3: Відображає час, який витрачається на необхідні дії.

X4: Показує кількість рядків коду, що буде створено розробником під час створення програмного продукту.

5.2.2. Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 4.2.

Таблиця 5.2 – Основні параметри ПП

Назва Параметра	**Умовні позначення**	**Одиниці виміру**	**Значення параметра**		
			гірші	**середні**	**кращі**
Швидкодія мови програмування	X1	Оп/мс	3000	8000	18000
Обсяг пам'яті для збереження даних	X2	Мб	32	16	8
Час обробки даних алгоритмом	X3	мс	500	220	50
Потенційний обсяг програмного коду	X4	кількість рядків коду	1500	1000	500

За даними таблиці 4.2 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

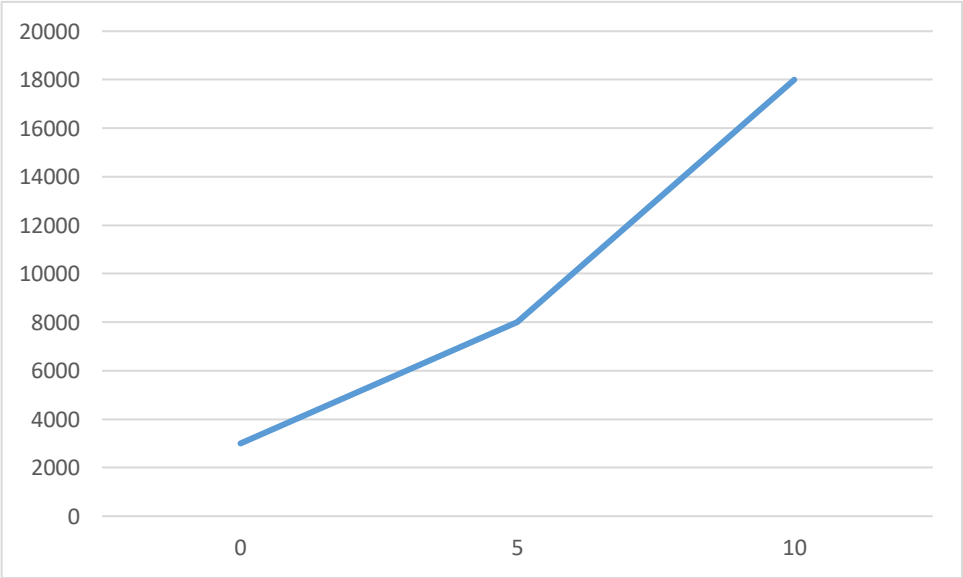


Рисунок 5.2 – X1, швидкодія мови програмування

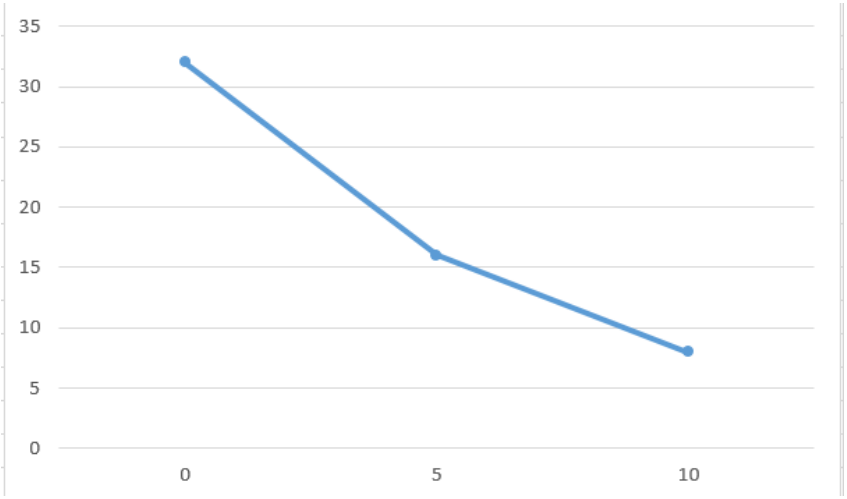


Рисунок 5.3 – X2, об'єм пам'яті для збереження даних

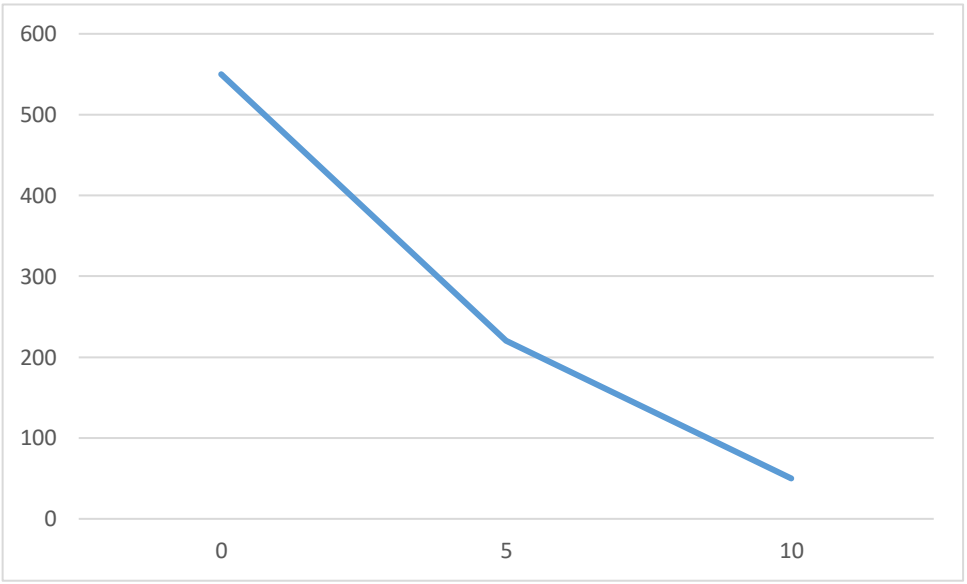


Рисунок 5.4 – X3, час обробки даних алгоритмом

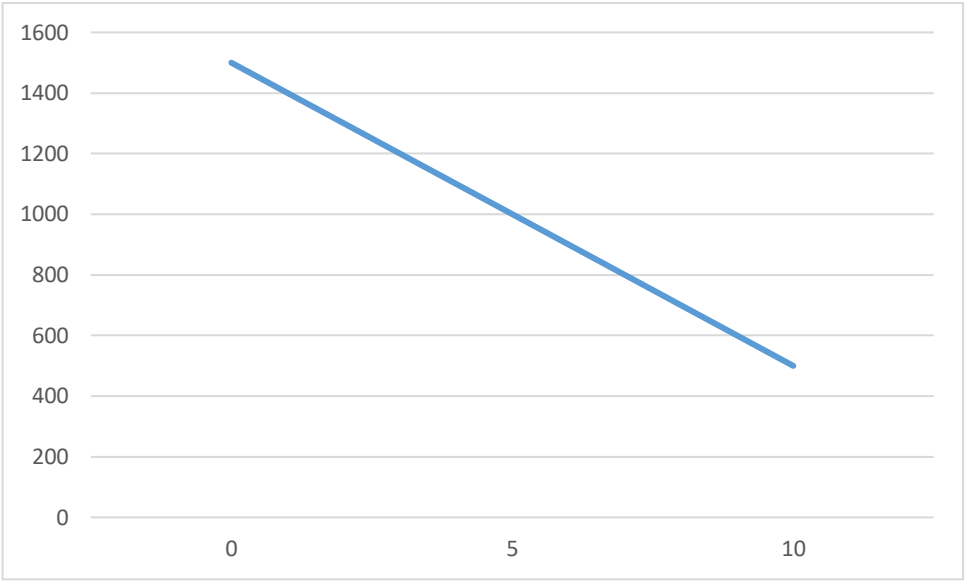


Рисунок 5.5 – X4, потенційний об'єм програмного коду

5.2.3. Аналіз експертного оцінювання параметрів

Наступним етапом після детальної бесіди й аналізу є оцінка кожним окремим експертом ступеню важливості кожного параметру для конкретної сформованої цілі – розробка програмного продукту, який допомагає користувачу слідкувати за своєю поставою.

Важливість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 6 людей. Визначення коефіцієнтів значимості передбачає:

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого використання;
- визначення оцінки попарного пріоритету параметрів;
- обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 5.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	4	5	4	4	4	4	4	29	5	25
X2	Об'єм пам'яті для збереження даних	Мб	5	5	5	4	3	4	4	30	-2	4
X3	Час обробки даних алгоритмом	Мс	3	3	2	3	2	1	1	15	-15	225
X4	Потенційний об'єм програмного коду	кількість строк коду	3	2	4	4	6	6	6	31	12	144
	Разом		15	15	15	15	15	15	15	105	0	398

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 105,$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 26{,}25.$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 398.$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3-n)} = \frac{12 \cdot 398}{7^2(5^3-5)} = 0{,}81 > W_k = 0{,}67 \text{ [27]}$$

Ранжирування можна рахувати достовірним, оскільки коефіцієнт узгодженості, що було знайдено в процесі, перевищує нормативний, що дорівнює 0,67.

Наступний етапом іде попарне порівняння всіх вище зазначених параметрів, для чого можемо скористатися результатами ранжування, проведемо попарне порівняння всіх параметрів. Результати буде занесено до таблиці 4.4.

Таблиця4.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	<	=	<	<	<	<	<	<	0,5
X1 і X3	<	<	<	<	<	<	<	<	0,5
X1 і X4	<	<	<	<	<	<	<	<	0,5
X2 і X3	<	<	<	<	<	<	<	<	0,5
X2 і X4	>	>	>	>	>	>	>	>	1,5
X3 і X4	>	>	>	>	>	>	>	>	1,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij}визначається по формулі:

$$a_{ij} = \begin{cases} \\ 1{,}5 \text{ при } X_i > X_j \end{cases}$$

1.0 при $X_i = X_j$

0.5 при $X_i < X_j$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \text{де } b_i = \sum_{i=1}^{N} a_{ij}.$$

Відносні оцінки потрібно розраховувати декілька разів, до тих пір, поки наступні значення не будуть ненабагато відрізнятися від попередніх (а саме, різниця буде менше, ніж два відсотки). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{вi}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \text{де } b_i' = \sum_{i=1}^{N} a_{ij} b_j.$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує двох відсотків, тому на цій кількості ітерацій можемо зупинити розрахунки.

Таблиця 5.5 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{вi}}$	b_i^1	$K_{\text{вi}}^1$	b_i^2	$K_{\text{вi}}^2$
X1	1,0	1,5	0,5	0,5	3,5	0,147	8,25	0,148	33,125	0,154
X2	1,5	1,5	0,5	1,5	5,0	0,277	15,25	0,264	58,125	0,268
X3	1,5	1,5	0,5	1,5	5,0	0,322	20,25	0,352	76,875	0,355
X4	1,5	1,5	0,5	1,0	4,5	0,254	11,25	0,254	43,875	0,223
Всього:					18	1	55	1	212	1

5.3. Аналіз рівня якості варіантів реалізації функцій

Далі визначимо рівень якості кожного з варіантів виконання основних функцій окремо.

Абсолютні значення параметрів *X2*(обсяг пам'яті для збереження даних) та *X1* (швидкодія мови програмування) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X3 (*час обробки даних) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 600 мс або варіанту б) 100мс.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (Таблиця 5.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j} ,$$

де n – кількість параметрів;$K_{вi}$– коефіцієнт вагомості i–го параметра;B_i – оцінка i–го параметра в балах. [27]

Таблиця 5.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1)	А	8000	3,6	0,148	0,469
F2(X2)	А	16	3,4	0,264	0,833
F3(X3,X4)	А	500	2,4	0,352	0,851
	Б	50	1	0,254	0,207

За даними з таблиці 4.6 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}],$$

визначаємо рівень якості кожного з варіантів:

K_{K1} = 0,469+ 0,833+ 0,851= 2,15

K_{K2} = 0,469+ 0,833+ 0,207= 1,51

З розрахунків очевидно, що кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

5.4. Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмного коду.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується додаткова інформація, а завдання 2 використовує інформацію у вигляді даних.

Спочатку проведемо розрахунок норм часу на розробку та програмування для кожного з наведених завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_М \cdot K_{СТ} \cdot K_{СТ.М}, \quad (5.1)$$

де T_P – трудомісткість розробки ПП;

$К_П$ – поправочний коефіцієнт;

$К_{СК}$ – коефіцієнт на складність вхідної інформації;

$К_М$ – коефіцієнт рівня мови програмування;

$К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення. [27]

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює:

- $Т_Р$ =80 людино-днів.
- Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1,86.
- Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань дорівнює одиниці: $К_{СК}$ = 1.0.
- Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0,77.

Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

$Т_1$ = 80·1,86·1,0·0,77 = 114,58 людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто

- $Т_Р$ =42 людино-днів,

- $К_П$ =1,2,
- $К_{СК}$ = 1,3,
- $К_{СТ}$ =0,69:

$Т_2$ = 42 · 1,2 · 1,3 · 0,69 = 45,21 людино-днів.

Для третього завдання (використовується алгоритм третьої групи складності, ступінь новизни Г):

$Т_р$ = 35 людино-днів;

$К_п$ = 0,9;

$К_{ст}$ = 0,84;

$Т_3$ = 35 • 0,9 • 0,84 = 26,46.

Для четвертого завдання (використовується алгоритм першої групи складності, ступінь новизни В):

$Т_р$ = 64 людино-днів;

$К_п$ = 0,79;

$К_{ст}$ = 0,8;

$Т_о$ = 64 • 0,79 • 0,8 = 40.45.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$Т_I$ = (**114,58** + **45,21** + **26,46.**) · 8 = 1490 людино-годин;

$Т_{II}$ = (**114,58** + **45,21** + **40.45**) · 8 = 1601.92 людино-годин;

Найбільш високу трудомісткість має варіант два.

В розробці беруть участь три програмісти з окладом 21500 грн., один дизайнер з окладом 12000грн. Визначимо зарплатню в гривнях за годину за наступною формулою:

$$С_Ч = \frac{М}{T_m \cdot t} грн.,$$

де М – місячний оклад працівників;T_m – кількість робочих днів тиждень;t – кількість робочих годин в день.

$$\mathbf{С_Ч} = \frac{21500 * 3 + 12000}{3 \cdot 21{,}3 \cdot 8} = 149{,}65 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$С_{ЗП} = С_Ч \cdot Т_i \cdot К_Д$,

де $С_Ч$– величина погодинної оплати праці програміста;$Т_i$ – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $\mathbf{С_{ЗП}} = 149{,}65 \cdot 1490 \cdot 1.2 = 267574{,}24$ грн.

II. $\mathbf{С_{ЗП}} = 149{,}65 \cdot 1601.92 \cdot 1.2 = 287672{,}79$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $\mathbf{С_{ВІД}} = С_{ЗП} \cdot 0.22 = 169474{,}5 \cdot 0.22 = 37284{,}39$ грн.

II. $\mathbf{С_{ВІД}} = С_{ЗП} \cdot 0.22 = 184294{,}93 \cdot 0.22 = 40544{,}89$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна електронно-обчислювальна машина обслуговує одного програміста з окладом 21500 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$\mathbf{C_Г} = 12 \cdot М \cdot K_З = 12 \cdot 21500 \cdot 0{,}2 = 51600 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$\mathbf{C_{ЗП}} = C_Г \cdot (1 + K_З) = 51600 \cdot (1 + 0.2) = 61920 \text{ грн.}$$

Відрахування на єдиний соціальний внесок:

$$\mathbf{C_{ВІД}} = C_{ЗП} \cdot 0.22 = 61920 \cdot 0{,}22 = 13622{,}4 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 8000 грн.

$$\mathbf{C_А} = K_{ТМ} \cdot K_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 8000 = 2300 \text{ грн.},$$

де $K_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $K_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$\mathbf{C_Р} = K_{ТМ} \cdot Ц_{ПР} \cdot K_Р = 1.15 \cdot 9000 \cdot 0.05 = 517{,}5 \text{ грн.},$$

де $K_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу персонального комп’ютеру за рік розраховуємо за формулою:

$$\mathbf{T_{ЕФ}} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot K_В = (365 – 104 – 9 – 16) \cdot 8 \cdot 0.9 = 1699{,}2 \text{ годин,}$$

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування; t –кількість робочих годин в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни. [27]

Витрати на оплату електроенергії розраховуємо за формулою:

$$\mathbf{С_{ЕЛ}} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1699,2 * 0,34 * 0,5 * 3,51 = 1013,91 \text{ грн.},$$

де $N_С$ – середньо-споживча потужність приладу; КЗ– коефіцієнтом зайнятості приладу; ЦЕН – тариф за один КВт-годин електроенергії.

Накладні витрати розраховуємо за наступною формулою:

$$\mathbf{С_Н} = Ц_{ПР} \cdot 0.67 = 9000 \cdot 0,67 = 6030 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$$

$$С_{ЕКС} = 61920 + 13622,4 + 2300 + 517,5 + 1013,91 + 6030 = 85403,81 \text{ грн.}$$

Собівартість однієї машино-години роботи електронно-обчислювальної машини дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 85403,81 / 1699,2 = 50,26 \text{ грн/час.}$$

Оскільки в наведному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, з огляду на обраний варіант реалізації, складає:

$$С_М = С_{М\text{-}Г} \cdot Т$$

I. $\mathbf{С_М} = 50,26 * 1490 = 74887,4$ грн.;

II. $\mathbf{C_м}$ = 50,26 * 1601,92 = 80512,5 грн.;

Накладні витрати складають 67% від заробітної плати:

$C_Н = C_{ЗП} \cdot 0,67$

I. $\mathbf{C_Н}$ = 267574,24 * 0,67 = 179274,74 грн.;

II. $\mathbf{C_Н}$ = 287672,79 * 0,67 = 192740,77 грн.;

Отже, вартість розробки програмного продукту за варіантами становить:

$C_{ПП} = C_{ЗП} + C_{ВІД} + C_М + C_Н 74887,4$

I. $\mathbf{C_{ПП}}$ = 267574,24 + 37284,39 + 74887,4 + 179274,74 = 559 020,77 грн.;

II. $\mathbf{C_{ПП}}$ = 287672,79 + 40544,89+ 80512,5 + 192740,77 = 601 470,95грн.;

5.5 Вибір кращого варіанта ПП техніко-економічного рівня

Тепер проведено обчислення коефіцієнту техніко-економічного рівня за наступною формулою:

$К_{ТЕРj} = ККj / СФj,$

КТЕР1 = 2,15/ 581973,51 = 0,037 ·10-4;

КТЕР2 = 1,51 / 625687,84 =0.024·10-4;

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня **КТЕР1**= 0,037 ·10-4

5.6. Висновки до розділу

В наведеному вище розділі було проведено повний функціонально-вартісний аналіз програмного продукту, що було розроблено в процесі написання дипломної роботи. Процес аналізу може бути умовно поділений на дві частини.

Перша частина присвячена дослідженню ПП з технічної точки зору:

- було сформульовано основні функції ПП та сформовано множину варіантів їх реалізації;
- ґрунтуючись на обчислених значеннях параметрів, а також експертних оцінок їх важливості, було проведено визначення коефіцієнту технічного рівня, який далі дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

В другій частині функціонально-вартісного аналізу було показано процес вибору із альтернативних варіантів реалізації той, який є найбільш економічно обґрунтованим. Поставлена мета вимагала провести порівняння запропонованих варіантів реалізації, яке в даній частині поводилось із використанням коефіцієнта ефективності, для обчислення останнього було проведено розрахунок наступних додаткових параметрів: трудомісткість, витрати на заробітну плату та накладні витрати.

На підставі проведеного функціонально-вартісного аналізу програмного комплексу, що розроблюється, сформулюємо висновок, що з того набору альтернатив, який залишився після першого відбору двох варіантів виконання програмного комплексу найбільш оптимальним є перший варіант реалізації програмного продукту. Він показав найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 0,037 \cdot 10\text{-}4$.

Цьому варіанту реалізації програмного продукту притаманні наступні характеристики:

- бібліотека візуалізації – D3.js;
- фреймворк для веб-розробки - React;
- мова програмування – JavaScript.

Відповідно до зазначеної вище інформації, можемо зробити висновок, що наведений варіант виконання програмного комплексу надає кінцевому користувачу інтуїтивно зрозумілий та зручний у використанні графічний інтерфейс, достатньо гарне функціональне наповнення, прийнятну та таку, що відповідає сучасним критеріям, швидкодію.

ВИСНОВКИ

Дана дипломна робота була розроблена для дослідження поєднання бібліотеки D3 з фреймворком React та як їх поєднати без втрати функціоналу для кожного з них. Для досягнення цієї мети було проаналізовано теоретичні знання та наявні методи реалізації ,а також прийнято рішення створити карту України для відображення якості повітря в реальному часі. Для цього був проведен аналіз альтернатив, які на данний момент наявні і наведено обґрунтоване пояснення.

Під час роботи було розглянуто наступні теми:

- Моніторинг екології за допомогою технологій;
- Фреймворки (визначення, альтернативи);
- Детально досліджено візуалізацію програмного продукту;
- Розробка додатків на мові програмування JavaScript
- Вартісний аналіз продукту

Результатом дипломної роботи, окрім детальне дослідження зазначених вище галузей знань, є проектування системи, яке містило в собі розробку відповідних діаграм (компонентів та станів). Комбінування частин проекту в єдиний продукт, за допомогою компонентного підходу, застосування хуків для передачі станів системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Rivu C. Reactive Programming in Kotlin [Електронний ресурс] / Chackraborty Rivu // Packt Piblishing Ltd.. – 2017. – Режим доступу до ресурсу: https://books.google.com.ua/books?id=ZMxPDwAAQBAJ&printsec=frontcover&dq=reactive+programming&hl=ru&sa=X&ved=2ahUKEwjXxq6N9a_wAhUmlYsKHVm4A_MQ6AEwAHoECAQQAg#v=onepage&q=reactive%20programming&f=false.

2. Загрязнения воздуха и здоровье [Електронний ресурс] // Haus & Luft. – 2017. – Режим доступу до ресурсу: http://haus-luft.com/ru/chistota-vozduha/zagryazneniya-vozduha-i-zdorove.

3. Технологии для экологии [Електронний ресурс] // Татьяна Честина. – 2019. – Режим доступу до ресурсу: https://bellona.ru/2019/03/07/tehnologii-dlya-ekologii/.

4. Как аналитика в реальном времени позволяет быстрее извлекать выгоду из данных [Електронний ресурс] // intel. – 2018. – Режим доступу до ресурсу: https://www.intel.ru/content/www/ru/ru/analytics/real-time-analytics.html.

5. Топ-5 JS-фреймворков для фронтенд-разработки в 2020 году. Часть 1 [Електронний ресурс] // Хабр. – 2020. – Режим доступу до ресурсу: https://habr.com/ru/company/ruvds/blog/476286/.

6. Популярные фреймворки JavaScript [Електронний ресурс] // vc.ru. – 2020. – Режим доступу до ресурсу: https://vc.ru/dev/147263-populyarnye-freymvorki-javascript.

7. Defining the term “reactive” [Електронний ресурс] // IBM. – 2020. – Режим доступу до ресурсу: https://developer.ibm.com/depmodels/reactive-systems/articles/defining-the-term-reactive/.

8. РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ ОДНОСТРАНИЧНОГО WEB-ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ REACT [Електронний ресурс] // Научное обозрение. – 2020. – Режим доступу до ресурсу: https://science-engineering.ru/ru/article/view?id=1278.

9. Топ-5 JS-фреймворков для фронтенд-разработки в 2020 году. Часть 1 [Електронний ресурс] // Хабр. – 2020. – Режим доступу до ресурсу: https://habr.com/ru/company/ruvds/blog/476286/.

10. АНАЛИЗ БИБЛИОТЕК / ФРЕЙМВОРКОВ ДЛЯ СОЗДАНИЯ SPA ПРИЛОЖЕНИЙ [Електронний ресурс] // НАУЧНЫЕ ДОСТИЖЕНИЯ И ОТКРЫТИЯ 2020. – 2020. – Режим доступу до ресурсу: https://www.elibrary.ru/download/elibrary_44341284_17092721.pdf.

11. Мухортова, Д. Д. Визуалы, аудиалы, кинестетики / Д. Д. Мухортова. — Текст : непосредственный // Молодой ученый. — 2016. — № 12 (116). — С. 787-789. — URL: https://moluch.ru/archive/116/31787/ (дата обращения: 30.05.2021). 789. — URL: https://moluch.ru/archive/116/31787/ (дата обращения: 05.06.2021).

12. Visual Perception [Електронний ресурс] // Interaction Design Foundation. – 2020. – Режим доступу до ресурсу: https://www.interaction-design.org/literature/topics/visual-perception.

13. Принципы гештальта в дизайне пользовательского интерфейса [Електронний ресурс] // Хабр. – 2018. – Режим доступу до ресурсу: https://habr.com/ru/company/cloud4y/blog/347444/.

14. Лучше тысячи слов [Електронний ресурс] // Издательство Сманн, Иванов и Фербер. – 2014. – Режим доступу до ресурсу: https://blog.mann-ivanov-ferber.ru/2014/07/30/luchshe-tysyachi-slov-pochemu-vizualnaya-informaciya-zapominaetsya-bystree/.

15. 10 лучших JavaScript библиотек для визуализации данных на графиках и диаграммах [Електронний ресурс] // Хабр. – 2019. – Режим доступу до ресурсу: https://habr.com/ru/post/457946/.

16. Лучшие библиотеки JavaScript для создания диаграмм [Електронний ресурс] // TECH. – 2019. – Режим доступу до ресурсу: https://techrocks.ru/2019/05/22/best-javascript-chart-libraries/.

17. D3: Data-Driven Documents [Електронний ресурс] // NPM. – 2021. – Режим доступу до ресурсу: https://www.npmjs.com/package/d3.

18. Tarek A. Practical D3.js [Електронний ресурс] / A. Tarek, S. Rayna // Apress. – 2016. – Режим доступу до ресурсу: https://books.google.com.ua/books?id=JjalDAAAQBAJ&pg=PA25&dq=d3.js&hl=ru&sa=X&ved=2ahUKEwiuse2kzo_wAhXlsosKHSBiCjkQ6AEwBXoECAUQAg#v=onepage&q&f=true.

19. Введение в D3.js [Електронний ресурс] // METANIT.NET. – 2015. – Режим доступу до ресурсу: https://metanit.com/web/d3js/1.1.php.

20. Learn D3.js [Електронний ресурс] // Packt Piblishing Ltd.. – 2019. – Режим доступу до ресурсу: https://books.google.com.ua/books?id=a1qWDwAAQBAJ&printsec=frontcover&dq=d3.js&hl=uk&sa=X&redir_esc=y#v=onepage&q=d3.js&f=false.

21. Зотов М. А. ВИЗУАЛИЗАЦИЯ АЛГЕБРАИЧЕСКИХ БАЙЕСОВСКИХ СЕТЕЙ С ПОМОЩЬЮ JAVASCRIPT БИБЛИОТЕКИ D3.JS† [Електронний ресурс] / М. А. Зотов, А. В. Иванова, А. А. Золотин // ЗОТОВ. – 2020. – Режим доступу до ресурсу: https://www.elibrary.ru/download/elibrary_30320169_66951424.pdf.

22. Объединяя react, D3, и их экосистемы [Електронний ресурс] // WordPressify. – 2018. – Режим доступу до ресурсу: https://wordpressify.ru/2018/02/obedinyaya-react-d3-i-ih-ekosistemy/.

23. ВЕБ-ПРИЛОЖЕНИЕ ДЛЯ ВИЗУАЛИЗАЦИИ ПРОСТРАНСТВЕННЫХ ДАННЫХ [Електронний ресурс] // (ГГУ им. Ф. Скорины, Гомель). – 2019. – Режим доступу до ресурсу: https://www.elibrary.ru/download/elibrary_44879969_37339150.pdf.

24. Пашін В. П. Методичні вказівки до виконання економіко-організаційного розділу дипломних проектів (робіт) бакалаврів і спеціалістів для студентів інституту прикладного системного аналізу / В. П. Пашін, В. В. Романов, Н. В. Єгорова. // НТУУ “КПІ”. – 2011. – С. 118.