МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Комп'ютерна графіка і Web-дизайн»

Група: 4КГ-08

Дипломний проект

здобувача освіти денної форми навчання
КГ.08.24.000.ДП

ТРАНСЬКОГО
МИКИТИ ОЛЕКСІЙОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітньо-професійна програма: «**Комп'ютерна графіка і Web-дизайн»**

Група: **4КГ-08**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту на тему:

Розробка набору віджетів для візуалізації числових даних у вигляді діаграм

Проектний матеріал складається з пояснювальної записки на 80 сторінках та графічного (презентаційного) матеріалу на 16 аркушах (слайдах)

Дипломник _____ (Транський М.О.)

Керівник _____ (Закроєв Ю.М.)

Консультанти:

з економічного розділу _____ (Канський М.Ю.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Краснокутська К.Г.)

Захист «20» червня 2025 р. Протокол ЕК № 1

Оцінка ЕК 4 (добре) / 80 б.

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Комп'ютерна графіка і Web-дизайн»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"19" 05 2025 р.

ЗАВДАННЯ

на дипломний проект

Здобувачеві освіти *Транському Микиті Олексійовичу*

(прізвище, ім'я, по батькові)

1. Тема проекту *Розробка набору віджетів для візуалізації числових даних у вигляді діаграм*

затверджена наказом по коледжу від "14" листопада 2024 р. № 246

2. Термін здачі закінченого проекту 16.06.25.

3. Вихідні данні до проекту *віджет має дозволяти візуалізовувати числові дані за допомогою лінійчатої, стовпчастої діаграми та діаграми розкиду; передбачити меню налаштувань віджету для детальної конфігурації графічних діаграм; передбачити можливість керувати станом віджету і його життєвим циклом; передбачити можливість розширення функціоналу застосунку; загальні вимоги до ергономіки візуального інтерфейсу web-застосунку*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

Огляд методів візуального представлення інформації; Аналітичний огляд програмних засобів для візуалізації числових даних; Вибір та аналіз засобів розробки; Розробка моделі життєвого циклу компоненту; Розробка UML-діаграми класів; Розробка структури застосунку; Реалізація застосунку обраною мовою програмування; Встановлення та тестування розробленого набору для візуалізації числових даних; Економічні розрахунки; Заходи ТБ

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Порівняння програмних засобів для візуалізації числових даних; Схема фреймворку; Архітектура бібліотек; Схема життєвого циклу компоненту; Блок-схема алгоритму обробки подій компоненту; UML-діаграма класів; Схема зв'язку основних компонентів; Схема зв'язку шаблонів розміщення; Модель віджету; Схема зв'язку віджетів; Вигляд головного екрану у робочому режимі; Створення лінійчатої, стовпчастої діаграми, діаграми розкиду

6. Консультанти по проекту, із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Закроєв Ю.М.*		
Економічний розділ	*Канський М.Ю.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання 15.05.25р

Керівник *Закроєв Ю.М.* _____
(підпис)

Завдання прийняв до виконання *Транський М.О.* _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту	Термін виконання етапів дипломного проекту (роботи)	Відмітка пр виконання
1	*Вступ. Постановка мети та задач проектування*	15.05.25	
2	*Аналіз методів візуального представлення інформ.*	16.05.25	
3	*Аналітичний огляд програмних засобів для*	17.05.25	
	візуалізації числових даних		
4	*Вибір та аналіз засобів розробки*	19.05.25	
5	*Розробка моделі життєвого циклу компоненту*	22.05.25	
6	*Розробка UML-діаграми класів*	26.05.25	
7	*Розробка структури набору віджетів*	01.06.25	
8	*Реалізація застосунку мовою програмування*	06.06.25	
9	*Встановлення та тестування розробленого*	10.06.25	
	застосунку для візуалізації числових даних		
10	*Аналіз отриманих результатів роботи застосунку*	11.06.25	
	для візуалізації числових даних у вигляді діаграм		
11	*Економічні розрахунки та питання з охорони праці*	12.06.25	
12	*Підготовка графічної частини проекту*	14.06.25	
13	*Підготовка проекту до захисту та тестування ПЗ*	16.06.25	

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КГ 08. 24 000. 00 ДП	Дипломний проект		
А4			КГ 08. 24 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КГ 08. 24 000. 00 ДП		
Розробив		Транський М.О.		16.06	Розробка набору віджетів для візуалізації числових даних у вигляді діаграм	Літ.	Аркуш 4 / Аркушів 80
Перевірив		Закроєв Ю.М.		16.06		Н Д П	
Н. контр.		Петрашова В.І.		16.06.05		ВСП "ОТФК ОНТУ" гр.4КГ-08	
Затверд.		Кривченко Ю.В.		16.06			

ЗМІСТ

ВСТУП

В сучасному світі інформація відіграє важливу роль, супроводжуючи людину в кожному аспекті її діяльності. Щодня ми стикаємося з великим обсягом даних, що належать до різних сфер життя, мають різні формати та структуру. Частина цієї інформації є надзвичайно цінною, інша ж може не мати практичної користі. Особливу роль у сприйнятті інформації відіграє зорове сприйняття, оскільки саме візуальний контент є найбільш зрозумілим та зручним для аналізу. Тому візуалізація даних набуває особливого значення, оскільки дає змогу швидко та ефективно обробляти числову інформацію.

Візуалізація даних може здійснюватися за допомогою різних способів і засобів. Серед найпопулярніших засобів є лінійні графіки, гістограми, кругові та стовпчасті діаграми, точкові графіки й теплові карти. Ці інструменти допомагають перетворювати сухі числові дані в наочні графічні представлення, що сприяє кращому розумінню тенденцій, закономірностей та взаємозв'язків між показниками. Правильний вибір засобу візуалізації залежить від типу даних, які необхідно відобразити, та від цілей аналізу.

Основною метою цього дипломного проєкту є створення застосунку, який дозволить візуалізувати числові дані у вигляді діаграм. Додаток повинен забезпечувати інтерактивність, зручність використання та підтримувати широкий набір функцій для фільтрації, перетворення та відображення даних.

Проєкт передбачає створення бібліотеки віджетів, яка дозволить користувачам не лише візуалізувати статистичні дані, але й налаштовувати їхній зовнішній вигляд. Серед передбачених способів візуалізації можуть бути графіки різних типів, діаграми, теплові карти та інші інтерактивні елементи, що надають можливість налаштовувати візуальне відображення даних відповідно до вимог користувача. Розробка застосунку буде здійснена за допомогою JavaScript та фреймворку Vue.js, що забезпечить гнучкість і динамічність інтерфейсу. Для користування додатком буде необхідний лише доступ до браузера та обліковий запис Google. Інтуїтивно зрозумілий інтерфейс і можливість збереження та відновлення даних забезпечать комфортне використання системи.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Огляд методів візуального представлення числової інформації

1.1.1 Основні функції даних та їх значення

Дані є одним із ключових ресурсів сучасного суспільства. Вони відіграють важливу роль у всіх сферах життя: від економіки й бізнесу до науки, медицини та соціальних мереж. Дані можуть бути як кількісними (числовими), так і якісними (категоріальними), і їхнє правильне використання дозволяє ухвалювати обґрунтовані рішення, виявляти закономірності та тенденції, а також прогнозувати події.

Основними функціями даних є:

- Описова функція: надання інформації про стан об'єкта чи явища;
- Аналітична функція: аналіз і порівняння даних для виявлення взаємозв'язків;
- Прогностична функція: використання історичних даних для прогнозування майбутніх тенденцій;
- Управлінська функція: допомога у прийнятті стратегічних рішень на основі об'єктивної інформації.

1.1.2 Процес візуалізації даних та його значення

Візуалізація даних є процесом перетворення числової або текстової інформації в графічні елементи, що дозволяють легше сприймати й аналізувати інформацію. Завдяки візуальному представленню, дані стають доступнішими для розуміння, оскільки людині легше аналізувати графічні структури, ніж складні таблиці або масиви чисел.

Роль візуалізації даних є багатогранною:

- Полегшення розуміння: Візуальні об'єкти, такі як графіки чи діаграми, дозволяють швидко зрозуміти основні тенденції та закономірності в даних;
- Покращення аналізу: Завдяки візуалізації можна порівнювати різні

набори даних і легко знаходити відхилення або аномалії;

– Прийняття рішень: Візуальні інструменти допомагають у швидкому ухваленні рішень на основі наочних фактів і показників;
– Підвищення комунікації: Візуалізація допомагає ефективно передавати складну інформацію, роблячи її зрозумілою для широкого кола людей.

1.1.3 Елементи візуалізації даних

Візуальне представлення даних може відбуватися через різноманітні графічні елементи, які дозволяють користувачу краще зрозуміти і проаналізувати дані. Серед таких елементів найбільш популярними є:

– Таблиці: Відображають числові дані у сітці з рядків і стовпців. Зручні для подання великої кількості інформації (рис.1.1);
– Графіки: Включають лінійні графіки, гістограми, кругові діаграми, які візуально відображають зміни та співвідношення між даними (рис.1.2);
– Діаграми та карти: Кругові діаграми для відображення часток, картограми для географічного представлення даних, теплові карти для виділення інтенсивності даних (рис.1.3).

	A	B	C	D	E
4	*Кількість елементів у кожному кластері*	(94,55 ± 2,04879)	(88,432432 ± 1,391547)	-	30,388889 ± 0,320963
5		2,166886%	1,573571%		1,056187%
6	*Nзаповн*	(1399,875 ± 0,149435)	(1399,923077 ± 0,136923)	(600,0 ± 0,0)	1304,578947 ± 40,843677
7		0,010675%	0,009781%	0,000000%	3,130794%
8	*Доля кластерної системи*	(0,55995 ± 6,0E-5)	(0,559969 ± 5,5E-5)	(0,24 ± 0,0)	0,521832 ± 0,016337
9		0,010675%	0,009781%	0,000000%	3,130794%
10	*Радіус-вектор центру мас r0*	(39,099064 ± 0,525628)	(37,480224 ± 0,487531)	(37,987454 ± 0,222554)	37,523127 ± 0,274222
11		1,344349%	1,300768%	0,585861%	0,730808%
12	*Квадрат радіусу гірації R^2(s)*	(2,528547439805223E9 ± 1,01517708203421E8)	(2,356424332770548E9 ± 6,31688055403113E8)	-	1,5047029686755E7 ± 455116,847801
13		4,014863%	26,807059%		3,024629%
14	*Ступінь анізотропії*	(0,967227 ± 0,007533)	(0,970415 ± 0,007905)	(0,946944 ± 0,013476)	0,961554 ± 0,006237
15		0,778835%	0,814572%	1,423125%	0,648625%
16	*Фрактальна розмірність*	(1,835315 ± 0,01571)	(1,819221 ± 0,031052)	(1,777139 ± 0,039441)	1,284684 ± 0,039146
17		0,855962%	1,706862%	2,219333%	304,715300%
18	*Перша кор. розмірність*	(1,419763 ± 0,019042)	(1,411306 ± 0,018672)	(1,528527 ± 0,028404)	0,798531 ± 0,022156
19		1,341187%	1,323009%	1,858229%	2,774655%
20	*Вторая кор. розмірність*	(1,292363 ± 0,021423)	(1,278862 ± 0,031079)	(1,427893 ± 0,026243)	0,613011 ± 0,026913
21		1,657629%	2,430219%	1,837883%	4,390218%
22	*Коеф. пропорційності k2*	(1,357239 ± 0,089214)	-	(0,618858 ± 0,090546)	-
23		6,573190%		14,631095%	
24	*Потужність неск. Кластеру*	-	-	(0,24 ± 0,0)	-
25				0,000000%	
26	*Сер. зн. кв. радіусу гірації*	-	-	(7,8212997170644449E12 ± 7,593179225444022E10)	-
27				0,970833%	
28	*Сер. зн. довжини зв'язності*	-	-	(1,5642599434128898E13 ± 1,518635845088804E11)	-
29				0,970833%	
30	*Коеф. пропорційності k3*	-	-	(989674,122111 ± 172476,001156)	-

Лист1

Рисунок 1.1. Приклад візуалізації даних за допомогою таблиці у MS Excel

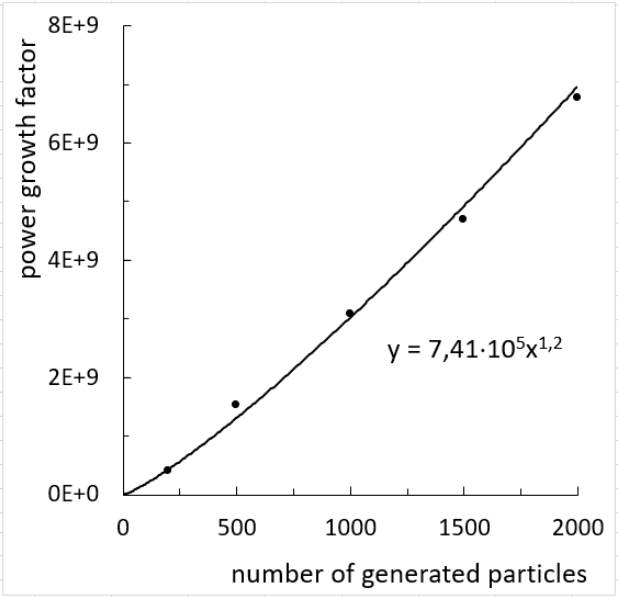


Рисунок 1.2. Приклад візуалізації даних за допомогою діаграми у MS Excel

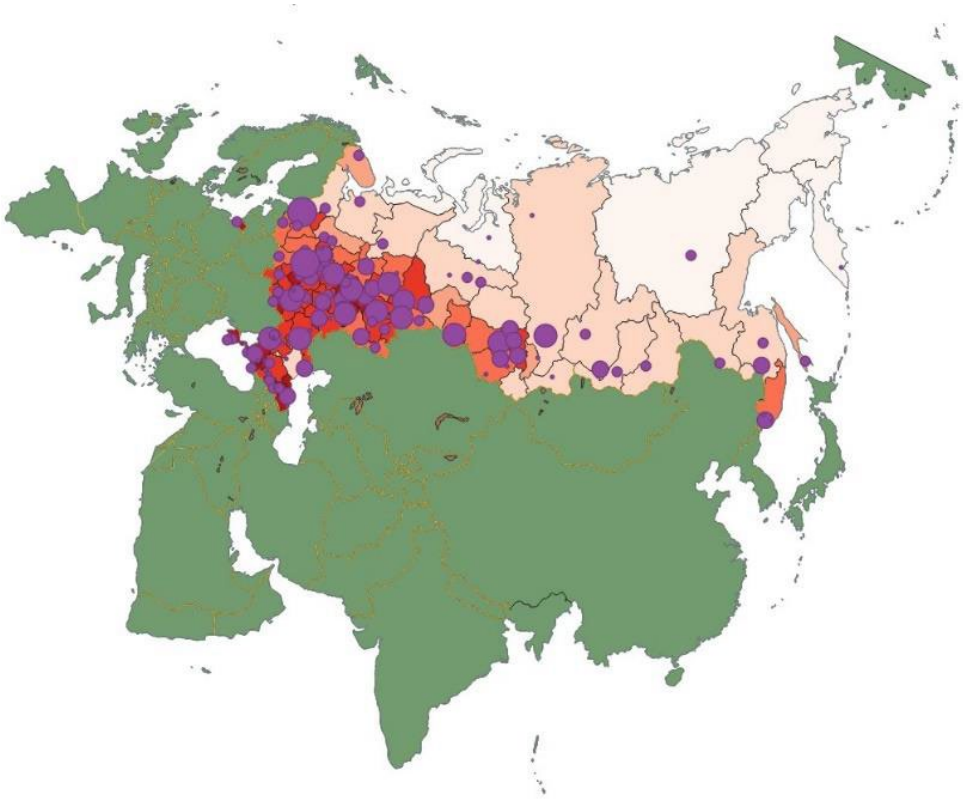


Рисунок 1.3. Приклад картограми для географічного представлення даних у QGIS

1.1.4 Типи діаграм для візуального представлення числової інформації

Візуалізація даних є важливим інструментом для ефективного представлення та інтерпретації числової інформації. Різні методи візуалізації використовуються для демонстрації певних типів даних, залежно від їхньої природи та мети аналізу. У цьому підрозділі розглянемо найбільш поширені методи візуалізації числових даних, їхні особливості та сфери застосування.

Різні типи діаграм допомагають ефективно візуалізувати числові дані, в залежності від специфіки задачі та структури даних:

- Лінійні графіки є одним із найпростіших і найбільш поширених методів візуалізації даних, особливо для відображення зміни показників протягом часу. Графік будується за допомогою ліній, які з'єднують точки, що відповідають значенням змінних на осі X та осі Y. Лінійні графіки часто використовуються для показу тенденцій, сезонних коливань, прогнозування майбутніх показників. Вони надають змогу легко помітити закономірності та тренди в даних. Приклад: Лінійний графік може показувати динаміку продажів компанії протягом останніх років, дозволяючи виявити як зростання, так і спад у різні періоди;
- Стовпчасті діаграми використовуються для порівняння величин між кількома категоріями. Кожен стовпець представляє окрему категорію, а його висота пропорційна значенню величини, що порівнюється. Цей метод візуалізації корисний для порівняння кількостей, доходів, попиту чи інших показників між різними групами або часовими періодами. Приклад: Стовпчаста діаграма може використовуватися для порівняння кількості проданих товарів у різних регіонах або для оцінки доходу кожного місяця за рік;
- Кругові діаграми використовуються для відображення часткових співвідношень між елементами в загальній структурі. Кожен сегмент діаграми відповідає певній категорії і має розмір, пропорційний частці, яку ця категорія займає в загальному обсязі. Кругові діаграми дозволяють легко побачити, як розподіляються ресурси або частки ринку між різними компонентами. Приклад: Кругова діаграма може показувати розподіл ринку між різними компаніями або частки бюджету, виділені на різні витрати;
- Точкові діаграми (Scatter Plot) використовуються для відображення взаємозв'язку між двома числовими змінними. На графіку кожна точка представляє пару значень змінних. Такий метод є ефективним для

виявлення кореляцій, залежностей або закономірностей між змінними, що дозволяє будувати прогнози або проводити порівняльний аналіз. Приклад: Точкова діаграма може показувати зв'язок між рекламними витратами та доходами компанії, що дозволить виявити ефективність маркетингових заходів;

- Гістограми є засобом для демонстрації розподілу значень числових даних за інтервалами. Вони дозволяють зрозуміти, як часто певні значення зустрічаються у вибірці, що допомагає оцінити варіативність і щільність даних. Гістограми особливо корисні для аналізу великих обсягів даних і виявлення відхилень. Приклад: Гістограма може відображати розподіл оцінок студентів за екзаменом, де кожен стовпець відповідає певному діапазону оцінок;
- Теплові карти є ефективним інструментом для представлення великих обсягів даних через кольорові градації. Вони дозволяють швидко ідентифікувати області з найвищою або найнижчою щільністю даних. Теплові карти часто використовуються в статистичних дослідженнях, соціології та географії для аналізу розподілу даних за певними просторовими параметрами. Приклад: Теплова карта може показувати розподіл температури по території країни або активність користувачів на веб-сайті;
- Картограми є особливим видом візуалізації, який поєднує статистичні дані з географічними картами. Цей метод дозволяє відображати числові дані в географічному контексті, що надає змогу краще зрозуміти поширення або концентрацію певних явищ на території. Приклад: Картограма може демонструвати населення різних регіонів або кількість захворювань у країнах;
- Діаграми деревовидної структури (Treemaps) використовуються для представлення ієрархічних даних, де кожен прямокутник відповідає певній категорії і має розмір, пропорційний до величини категорії в загальній структурі. Цей метод дозволяє аналізувати вклад окремих компонентів у

загальний обсяг. Приклад: Діаграма може використовуватися для відображення частки продажів кожного товару в загальному доході компанії.

1.1.5 Застосування лінійних структур даних під час візуалізації

Лінійні структури даних є важливим компонентом під час візуалізації. До таких структур належать:

- Масиви: Послідовність елементів, що мають фіксовану довжину;
- Черги: Структура даних, у якій доступ до елементів можливий лише за принципом "перший увійшов – перший вийшов";
- Стеки: Структура, яка працює за принципом "останній увійшов – перший вийшов". Такі структури часто використовуються для організації та обробки даних перед їхньою візуалізацією.

1.1.6 Візуалізація на основі числових даних

Статистичні дані складають основу багатьох візуалізацій. Їх аналіз дає змогу виявити загальні тенденції, кореляції та аномалії в даних. Основні статистичні показники, які використовуються під час аналізу:

- Середнє значення: Показує середню величину вибірки;
- Медіана: Визначає середнє значення в упорядкованій послідовності даних;
- Дисперсія: Відображає ступінь розкиду даних відносно середнього значення;
- Кореляція: Показує взаємозв'язок між двома змінними.

1.1.7 Інтерактивність візуалізації

Окрім статичних зображень, сучасні додатки часто передбачають інтерактивність візуалізації. Це дозволяє користувачам взаємодіяти з графіками, змінювати фільтри, налаштовувати параметри відображення та отримувати додаткову інформацію про дані. Такі функції забезпечують не лише зручність, але й глибший аналіз інформації.

1.1.8 Сучасні технології візуалізації числових даних

Серед найпоширеніших технологій для візуалізації числових даних використовують:

- JavaScript-бібліотеки: D3.js, Chart.js, Highcharts, які дозволяють створювати інтерактивні діаграми різної складності;
- Фреймворки для веб-розробки: Vue.js, React, Angular, які інтегруються з візуалізаційними бібліотеками та забезпечують гнучкість у створенні інтерфейсів для роботи з даними.

1.2 Аналітичний огляд програмних засобів для візуалізації числових даних

Візуалізація статистичних даних є ключовим етапом аналізу, який дозволяє перетворювати складні числові набори у графічну форму для полегшення їх розуміння та інтерпретації. У цьому підрозділі буде проведено огляд програмних засобів для візуалізації статистичних даних, починаючи від загальнодоступних інструментів, таких як Microsoft Word і Excel, до спеціалізованих професійних застосунків і бібліотек для розробки складних візуалізацій.

1.2.1 Інструмент Microsoft Word

Microsoft Word — це текстовий редактор, який містить базові засоби для візуалізації даних, такі як графіки та діаграми. Хоча Word не є професійним інструментом для обробки великих обсягів статистичних даних, його можливості достатні для створення простих графічних відображень даних у звітах, дипломних проектах або презентаціях (рис.1.4). Основні можливості:

- Побудова базових діаграм (лінійні, стовпчасті, кругові діаграми);
- Інтеграція діаграм із текстовими документами;
- Можливість швидкого редагування даних у діаграмах.

Приклад використання: Word дозволяє створювати прості діаграми, що відображають, наприклад, кількість реалізованих продуктів за категоріями у вигляді стовпчастої діаграми, яку можна включити в текстовий документ.

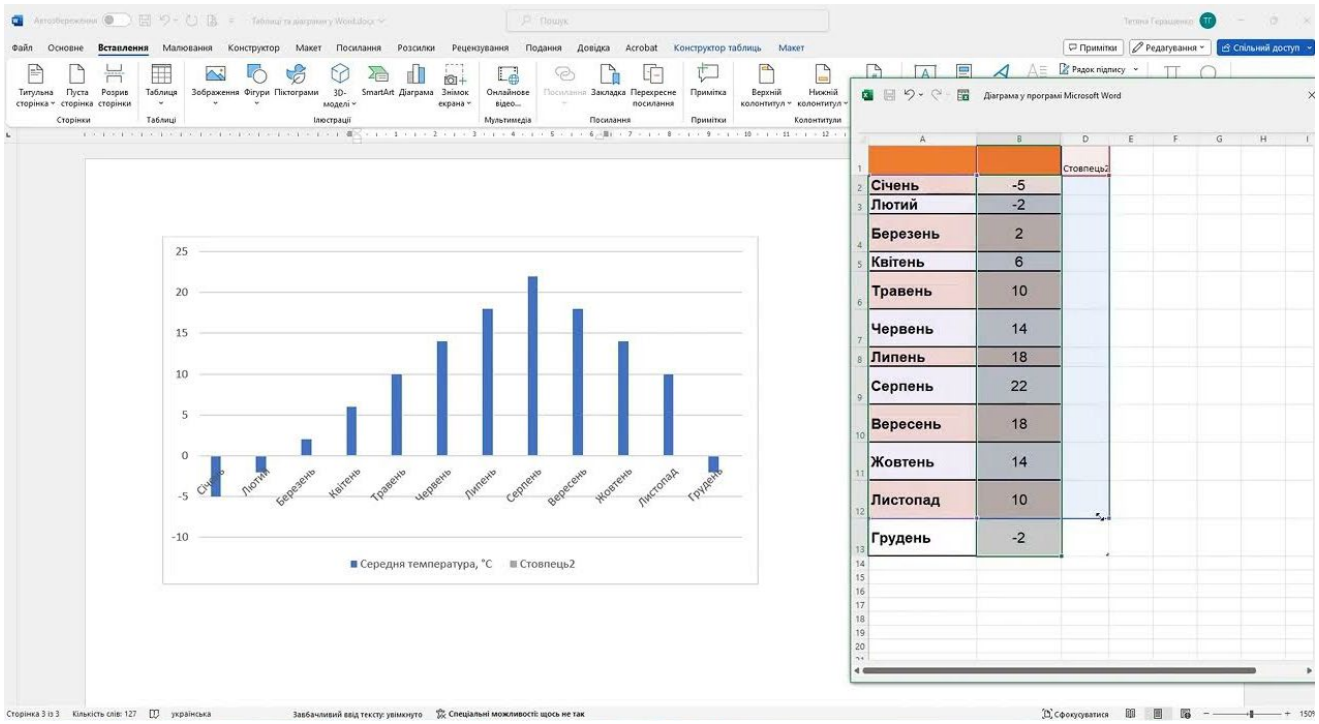


Рисунок 1.4. Створення діаграм засобами Microsoft Word

1.2.2 Інструмент Microsoft Excel

Microsoft Excel є більш потужним інструментом для обробки та візуалізації даних, ніж Word, оскільки він включає в себе безліч функцій для роботи з таблицями, формулами та діаграмами. Excel дозволяє користувачам будувати складні графіки і використовувати різні типи візуалізацій для аналізу числових даних (рис.1.5). Основні можливості:

- Створення широкого спектру діаграм (лінійні графіки, стовпчасті, гістограми, кругові діаграми, точкові графіки тощо);
- Підтримка великих обсягів даних і складних математичних розрахунків;
- Інтерактивні можливості, зокрема фільтрація, сортування даних;
- Використання функцій для аналізу даних, таких як таблиці зведень (Pivot Tables).

Приклад використання: За допомогою Excel можна створити динамічний графік, що показує залежність між витратами на рекламу і прибутком компанії. Крім того, можна аналізувати дані за допомогою зведених таблиць і швидко змінювати параметри візуалізації.

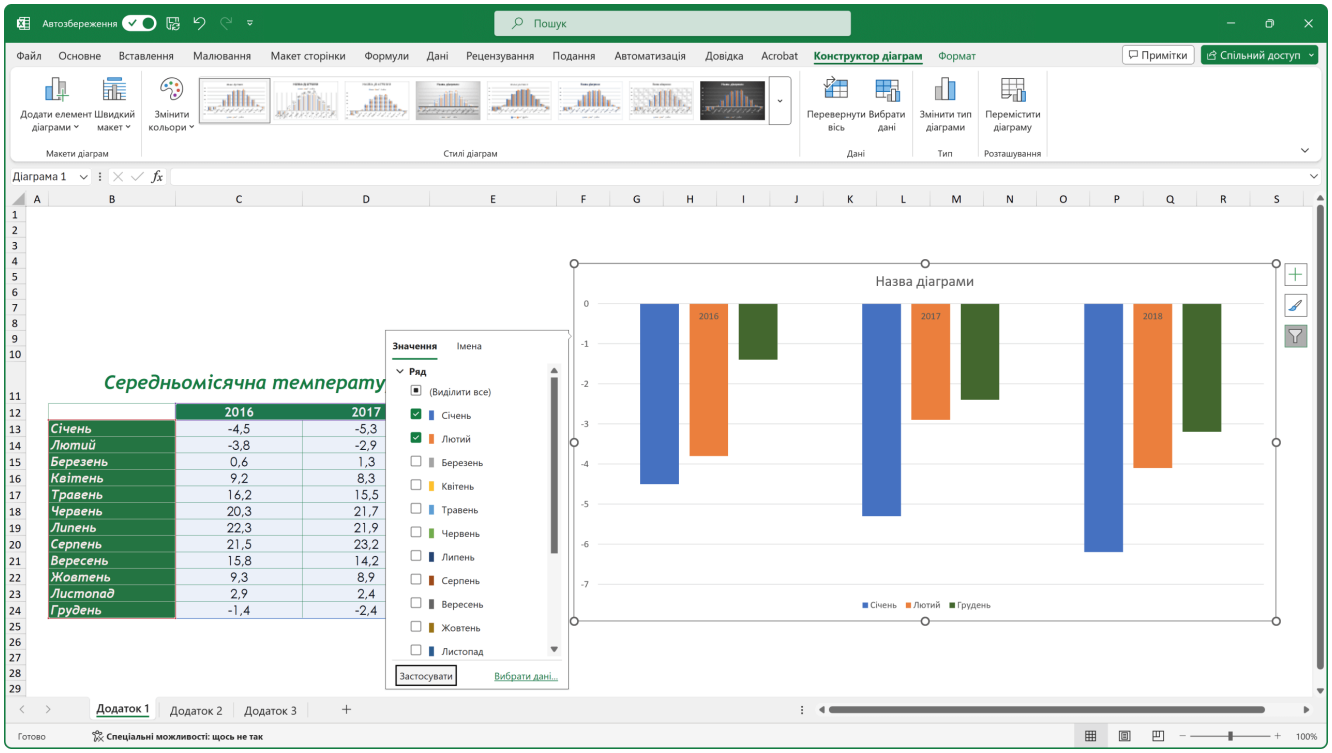


Рисунок 1.5. Створення діаграм засобами Microsoft Excel

1.2.3 Інструмент Tableau

Tableau є професійним інструментом для візуалізації даних, який дозволяє користувачам аналізувати великі обсяги даних і створювати складні інтерактивні візуалізації. Tableau особливо корисний для бізнес-аналітики, маркетингу та сфер, де необхідно ефективно працювати з числовими показниками (рис.1.6).

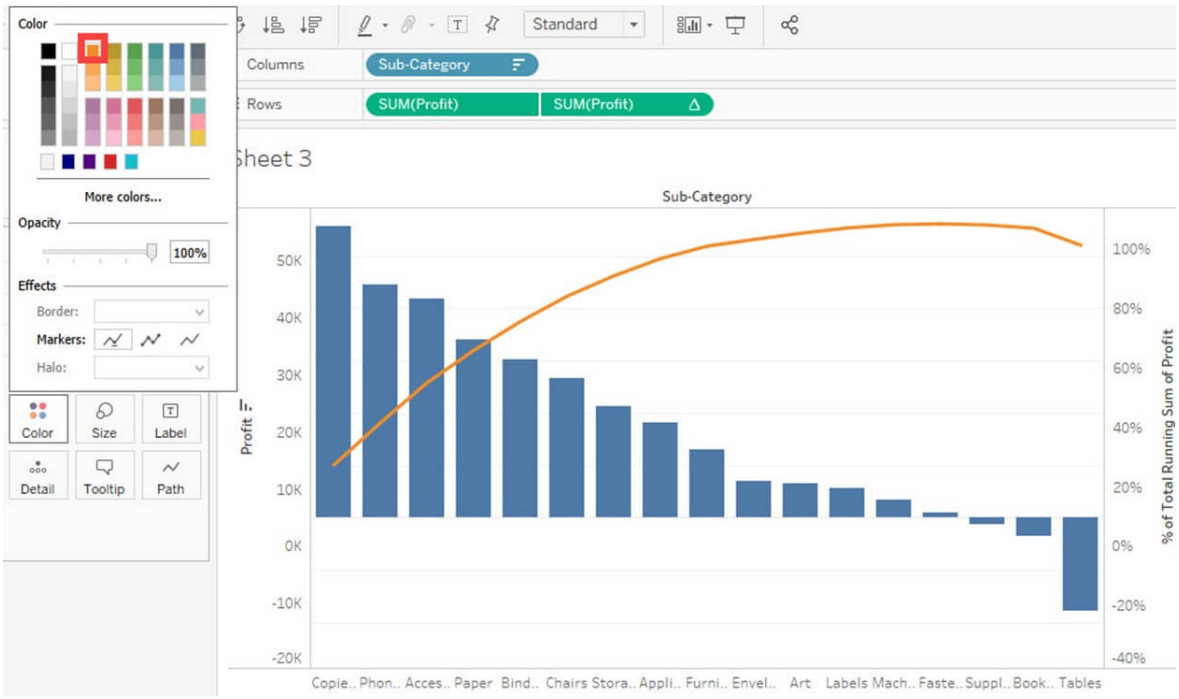


Рисунок 1.6. Створення діаграм засобами Tableau

Основні можливості:

– Підтримка інтерактивних дашбордів (панелей управління);
– Широкий вибір типів діаграм і можливостей для налаштування;
– Інтеграція з різними джерелами даних (бази даних, хмарні сховища);
– Можливість автоматичного оновлення даних у реальному часі;
– Гнучкі інструменти для фільтрації та сортування даних.

Приклад використання: За допомогою Tableau можна створити інтерактивний дашборд для відображення ключових бізнес-показників, таких як прибуток, витрати, рентабельність інвестицій у різних регіонах світу, з можливістю глибокого аналізу кожного параметра.

1.2.4 Інструмент Power BI

Power BI — це інструмент від Microsoft, призначений для бізнес-аналітики і візуалізації даних. Він дозволяє створювати інтерактивні звіти та панелі, що забезпечують ефективний аналіз даних у режимі реального часу (рис.1.7).

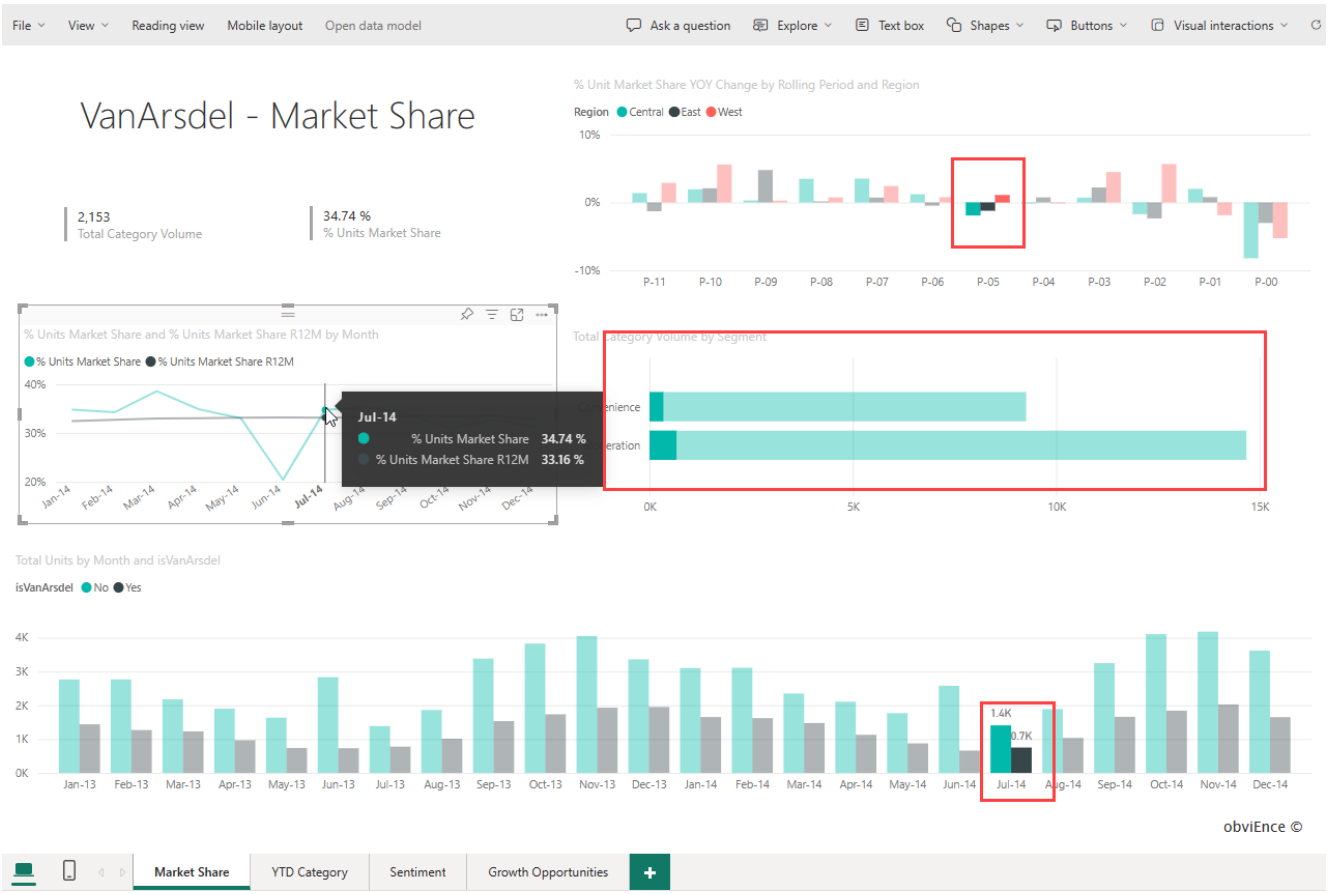


Рисунок 1.7. Створення діаграм засобами Power BI

Основні можливості Power BI:

– Інтерактивна візуалізація даних за допомогою діаграм і графіків;
– Підтримка інтеграції з різними джерелами даних (Excel, SQL-сервер, API);
– Можливість публікації звітів і доступу до них через хмару;
– Створення та налаштування панелей управління для швидкого огляду ключових показників.

Приклад використання: Power BI дозволяє створити звіт для аналізу продажів товарів у різних регіонах і автоматично оновлювати дані в режимі реального часу, що дозволяє менеджерам швидко реагувати на зміни.

1.2.5 Інструмент Google Data Studio

Google Data Studio є безкоштовним інструментом від Google для візуалізації даних і створення інтерактивних звітів. Він дозволяє користувачам підключатися до різних джерел даних і будувати на їх основі дашборди (рис.1.8).

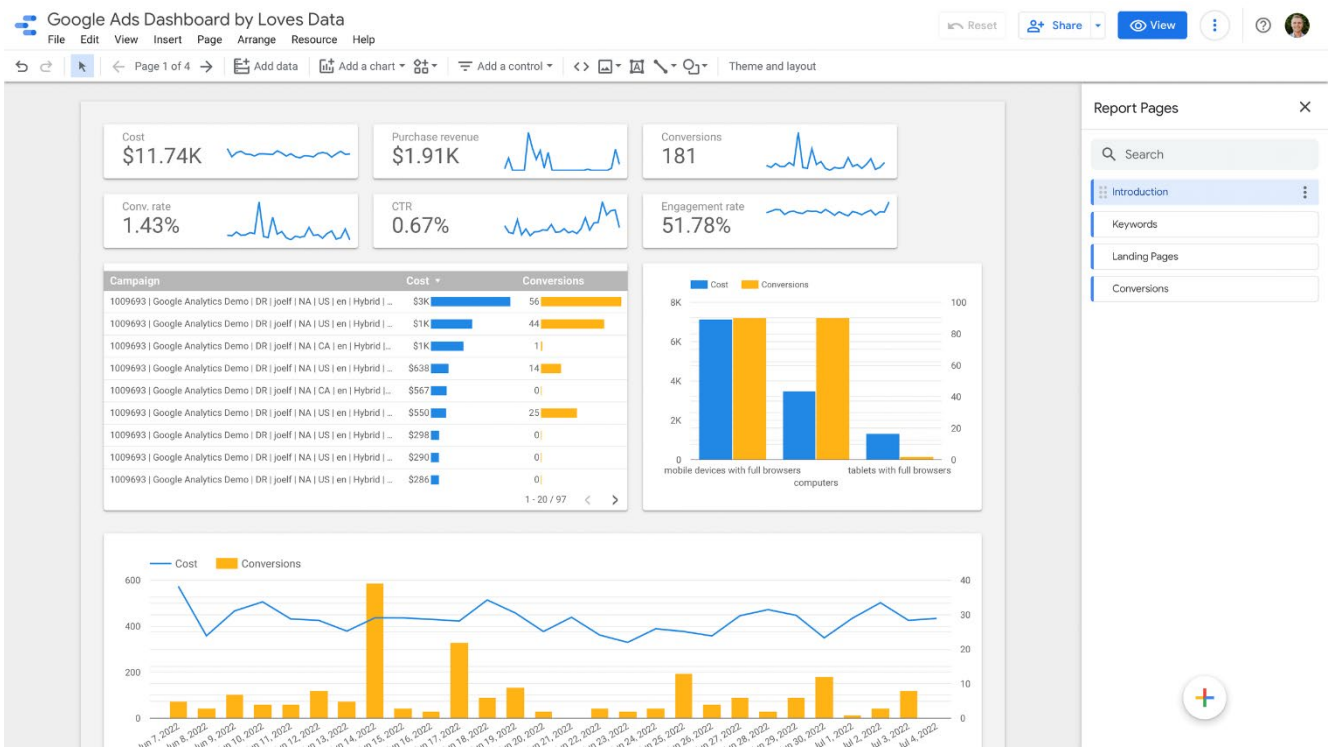


Рисунок 1.8. Створення діаграм засобами Google Data Studio

Основні можливості Google Data Studio:

– Інтерактивні графіки та діаграми;

– Підтримка підключення до таких джерел даних, як Google Analytics, Google Sheets, SQL-бази даних;
– Спільне використання звітів із іншими користувачами;
– Можливість налаштування зовнішнього вигляду звітів.

Приклад використання: За допомогою Google Data Studio можна створити інтерактивний звіт для аналізу поведінки користувачів на веб-сайті за допомогою даних із Google Analytics.

1.2.6 Інструмент D3.js

D3.js (Data-Driven Documents) — це JavaScript-бібліотека для створення динамічних, інтерактивних візуалізацій даних, заснованих на веб-технологіях. Вона дозволяє перетворювати дані в графічні об'єкти, інтегровані в HTML-документи, за допомогою технології SVG (Scalable Vector Graphics), рис.1.9.

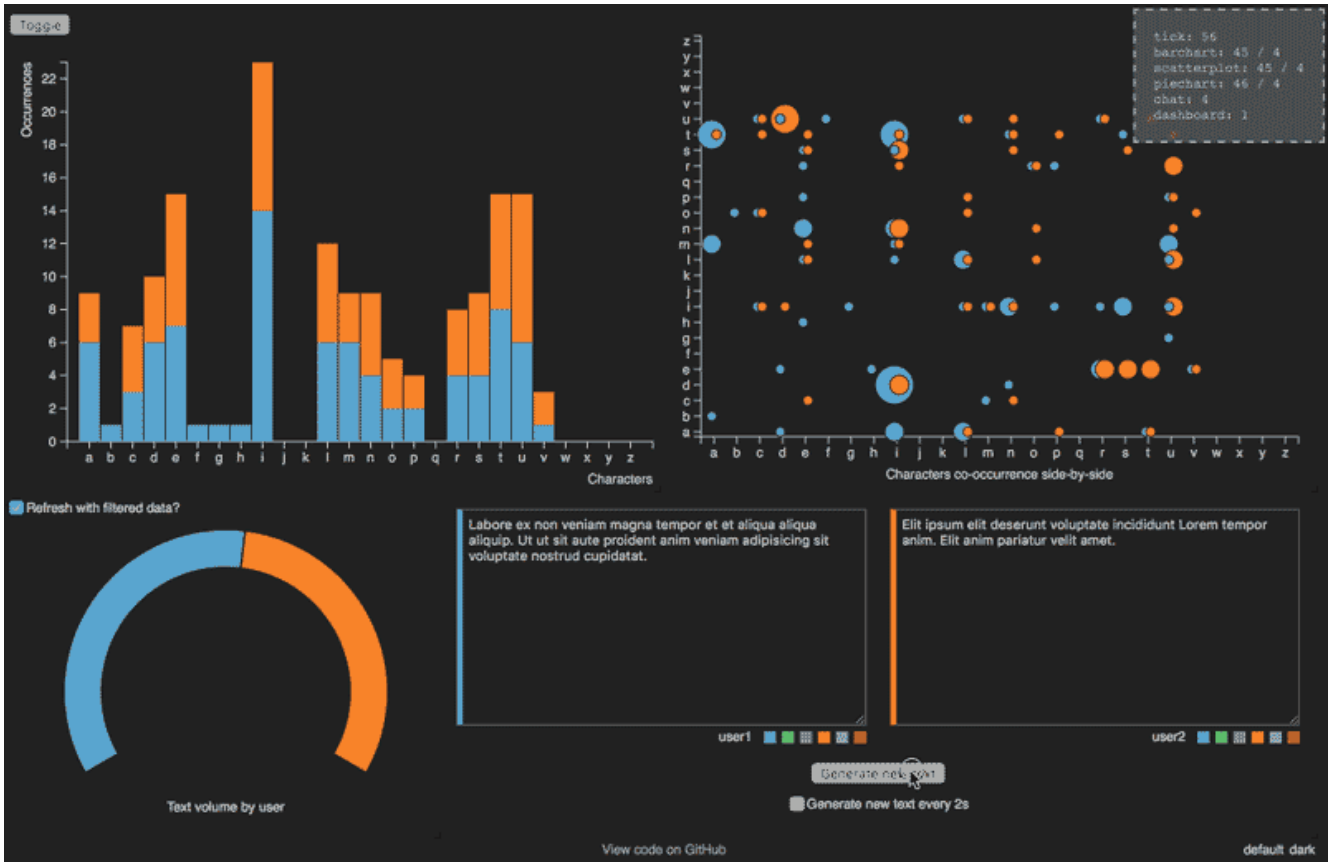


Рисунок 1.9. Створення діаграм засобами D3.js

Основні можливості D3.js:

– Побудова складних інтерактивних графіків, карт, діаграм;
– Повний контроль над виглядом і стилями графічних елементів;

– Підтримка анімації та динамічних змін даних;

– Можливість інтеграції з іншими веб-ресурсами та бібліотеками.

Приклад використання: D3.js використовується для створення інтерактивної картограми, що показує розподіл чисельності населення по країнах світу, з можливістю деталізувати дані при наведенні курсора.

1.2.7 Інструмент Matplotlib

Matplotlib — це бібліотека для мови Python, яка дозволяє створювати статичні, анімовані і інтерактивні графіки. Ця бібліотека є популярною серед дослідників та науковців, які працюють із великими наборами даних (рис.1.10).

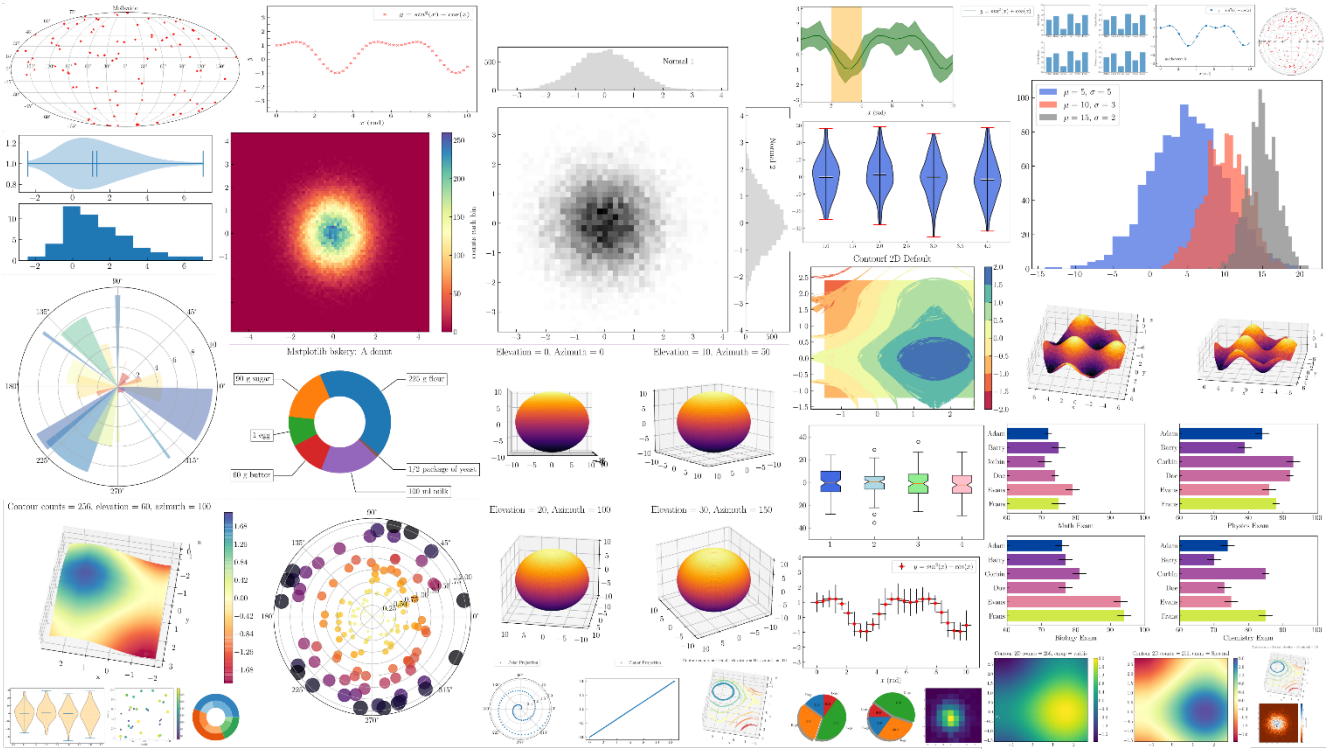


Рисунок 1.10. Створення діаграм засобами Matplotlib

Основні можливості Matplotlib:

– Створення двовимірних графіків і діаграм (лінійні, точкові, гістограми тощо);

– Підтримка інтеграції з іншими інструментами для аналізу даних, такими як NumPy і Pandas;

– Можливість експорту графіків у різних форматах (PNG, PDF, SVG).

Приклад використання: Matplotlib дозволяє створити графік функції та

візуалізувати результати статистичного аналізу, наприклад, відображення розподілу ймовірностей у вигляді гістограми.

1.2.8 Порівняльний аналіз розглянутих програмних засобів

Аналіз розглянутих інструментів для візуалізації даних дозволив виокремити три основні категорії: локальні (десктопні) додатки, веб-орієнтовані платформи та програмні бібліотеки. Кожен із цих підходів має певні переваги та недоліки, залежно від специфіки використання, вимог до інтерактивності, обсягів даних та технічних знань користувачів. Порівняння програмних засобів для візуалізації статистичних даних наведене у табл.1.1.

Таблиця 1.1. Порівняння програмних засобів для візуалізації статистичних даних

Критерій	*Microsoft Word*	*Microsoft Excel*	*Tableau*	*Power BI*	*Google Data Studio*	*D3.js*	*Matplotlib*
Тип програми	Десктоп-ний застосунок	Десктоп-ний застосунок	Веб/ десктопний застосунок	Веб/десктоп-ний застосунок	Веб-застосунок	JavaScript бібліотека	Python бібліотека
Типи візуалізацій	Лінійні, стовпчасті, кругові діаграми	Лінійні, стовпчасті, гістограми, точкові	Лінійні, стовпчасті, картограми, аналітичні	Лінійні, стовпчасті, гістограми, точкові, KPI	Лінійні, стовпчасті, картограми	Всі типи (гнучка побудова за допомогою SVG)	Лінійні, гістограми, точкові, поверхневі
Інтерак-тивність	Відсутня	Обмежена інтерак-тивність	Повна інтерак-тивність	Повна інтерак-тивність	Інтерактив-ність за рахунок інтеграції з даними	Повна інтерактив-ність (реалізована через код)	Відсутня чи мінімальна (залежить від бібліотек)
Підтримка великих наборів даних	Обмежена	Обмежена	Висока	Висока	Середня	Висока	Висока
Підтримка візуалізації даних у реальному часі	Відсутня	Відсутня	Присутня	Присутня	Присутня	Можлива, але реалізація залежить від коду	Обмежена
Легкість у використанні	Проста у використанні	Проста для базових задач	Вимагає спеціальних знань	Вимагає базових навичок	Проста у використанні	Вимагає знань JavaScript	Вимагає знань Python
Ціна	Платний (Office Suite)	Платний (Office Suite)	Платний, але є безкоштовна пробна версія	Платний, але є безкоштовна версія	Безкоштовний	Безкоштовна бібліотека	Безкош-товна бібліотека
Гнучкість налаштувань	Мінімальна	Середня	Висока	Висока	Середня	Висока	Висока
Синхронізація з іншими джерелами даних	Відсутня	Є інтеграція з іншими файлами Excel	Підтримує інтеграцію з різними джерелами	Інтеграція з Excel, базами даних	Інтеграція з продуктами Google, базами даних	Залежить від реалізації користувача	Інтеграція через інші Python бібліотеки
Підтримка різних форматів даних	Таблиці, текст	Таблиці, CSV, XLSX	CSV, Excel, бази даних, API	CSV, Excel, бази даних, API	CSV, Google Sheets, API	JSON, CSV, API	CSV, Excel, JSON, бази даних

На основі табл.1.1 можна зазначити, що десктопні програмні засоби підходять для базових завдань і невеликих наборів даних. Веб-застосунки пропонують розширені функції та інтерактивність, проте можуть бути складними для налаштування. Бібліотеки для візуалізації даних забезпечують найбільшу гнучкість і можливості, але вимагають технічних знань для використання та налаштування.

1.3 Вибір та аналіз засобів розробки

Для розробки веб-застосунку, що візуалізує числові дані у вигляді діаграм, важливо обрати правильні інструменти, які забезпечать як високу продуктивність, так і гнучкість в реалізації. Основною метою є створення інтуїтивного інтерфейсу та надання можливостей для налаштування і динамічної обробки даних. У цьому розділі буде проаналізовано декілька мов програмування та бібліотек для візуалізації даних, щоб визначити найбільш доречні варіанти для вирішення поставленого завдання.

1.3.1 Аналіз мов програмування для фронтенд-розробки

1. Python (з бібліотекою Dash)

Python — популярна мова програмування з багатим екосистемою для аналізу та обробки даних. Одна з бібліотек, що дозволяє створювати веб-додатки для візуалізації даних, — це Dash, яка базується на Flask та Plotly (рис.1.11).

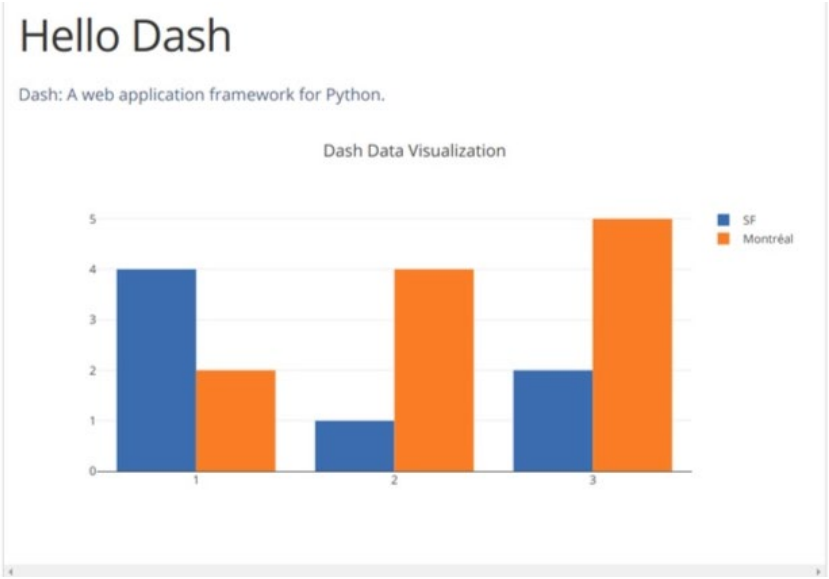


Рисунок 1.11. Візуалізація числових даних у Python з бібліотекою Dash

Переваги Python:

- Легка інтеграція з бібліотеками для аналізу даних, такими як Pandas, NumPy.
- Багато інструментів для роботи з числовими та статистичними даними.
- Dash надає широкий функціонал для інтерактивної побудови графіків та діаграм.

Недоліки Python:

- Низька продуктивність при великих обсягах даних або складних візуалізаціях.
- Обмежена інтеграція з іншими фронтенд технологіями, такими як сучасні фреймворки.
- Не завжди зручний для створення складних інтерфейсів, через що може потребувати додаткової роботи з HTML/CSS.

2. TypeScript

TypeScript — це надбудова над JavaScript, що додає статичну типізацію, дозволяючи уникнути багатьох помилок на етапі компіляції. Його переваги особливо помітні у великих проєктах з великим об'ємом коду, де важливо підтримувати структуру та ясність коду (рис.1.12).

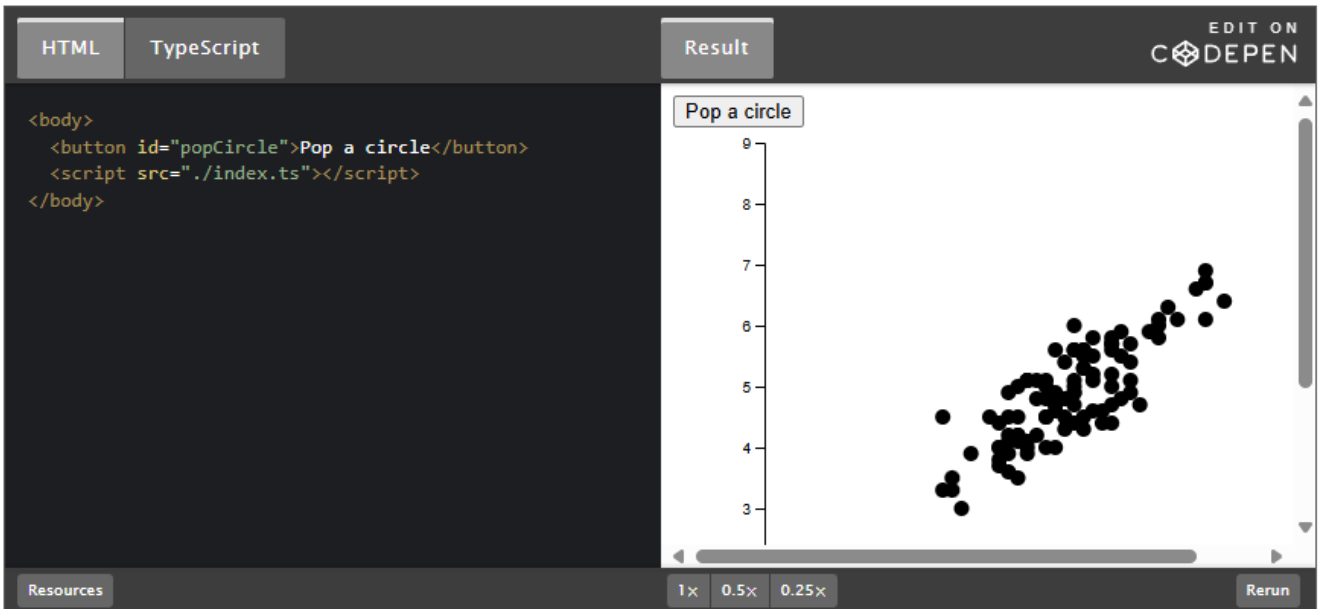


Рисунок 1.12. Візуалізація числових даних у TypeScript

Переваги TypeScript:

– Статична типізація допомагає запобігти багатьом помилкам.

– Сумісний з більшістю існуючих бібліотек та фреймворків для JavaScript.
– Підтримує сучасні інструменти для розробки, такі як React, Vue.js.
– Недоліки TypeScript:
– Потребує більше часу на налаштування та навчання у порівнянні з чистим JavaScript.
– Ускладнює процес розробки для невеликих проєктів через збільшення часу компіляції.

3. JavaScript

JavaScript є стандартом для фронтенд-розробки. Він підтримується всіма сучасними браузерами та має величезну екосистему бібліотек та інструментів. JavaScript також дозволяє створювати як прості, так і складні динамічні веб-додатки, зокрема для візуалізації даних.

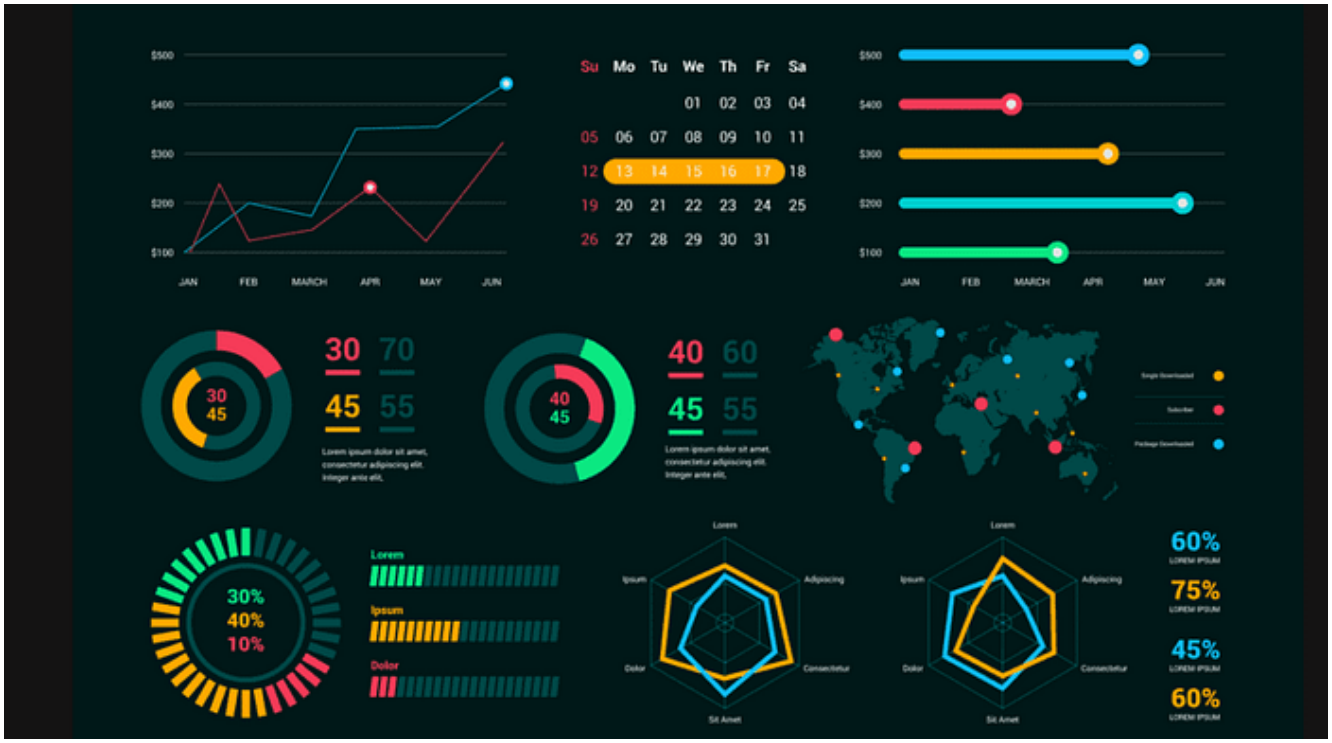


Рисунок 1.13. Візуалізація числових даних у JavaScript

Переваги JavaScript:

– Найпопулярніша мова для веб-розробки, підтримується всіма браузерами.
– Велика кількість бібліотек для візуалізації, таких як Chart.js, D3.js, Echarts.
– Простота використання та швидкість розробки завдяки існуючим рішенням та фреймворкам.

Недоліки JavaScript:

- Динамічна типізація може спричиняти помилки під час виконання, особливо в проєктах із великим кодом.
- Відсутність механізму перевантаження методів.
- Для складних задач може потребувати додаткових інструментів або бібліотек.

1.3.2 Огляд клієнтських фреймворків

Для реалізації веб-застосунку з візуалізацією даних важливо обрати фреймворк, який забезпечить швидкість розробки, легкість підтримки коду та високі можливості для інтерактивної взаємодії з даними.

Angular — це фреймворк для розробки SPA (односторінкових додатків), створений компанією Google. Він має багато вбудованих функцій для управління станом додатку, модульності та залежностями між компонентами (рис.1.14).

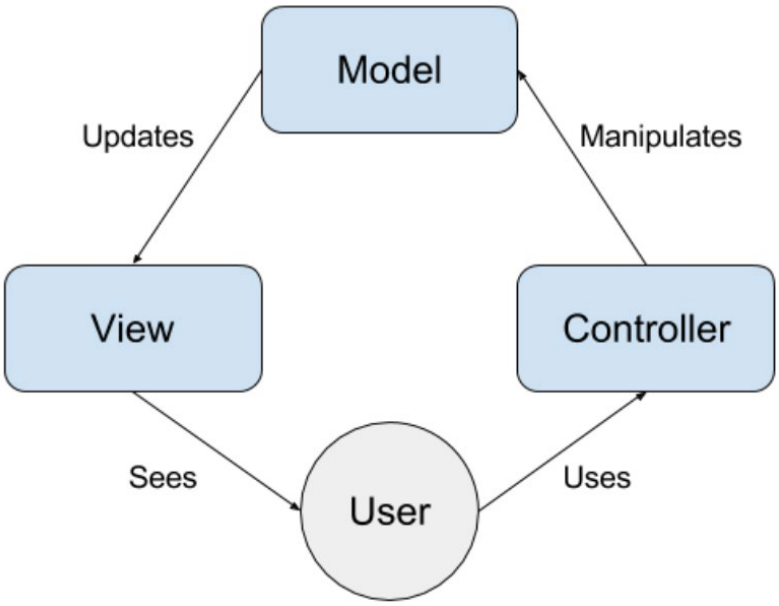


Рисунок 1.14. Схема MVC Angular з розділенням логіки, представлення і даних

Переваги Angular:

- Підтримка складних архітектур додатків із великими обсягами даних;
- Вбудовані засоби для взаємодії з API та управління станом;
- Забезпечує автоматичну синхронізацію даних між моделями та інтерфейсом (Data binding);

Недоліки Angular:

- Високий поріг входження через складність і велике число концепцій;
- Велика вага фреймворку, що може негативно вплинути на продуктивність.

React — це бібліотека для розробки інтерфейсів, яка використовує компонентний підхід і віртуальний DOM для оптимізації продуктивності. Створена компанією Facebook, вона має широку популярність серед розробників (рис.1.15).

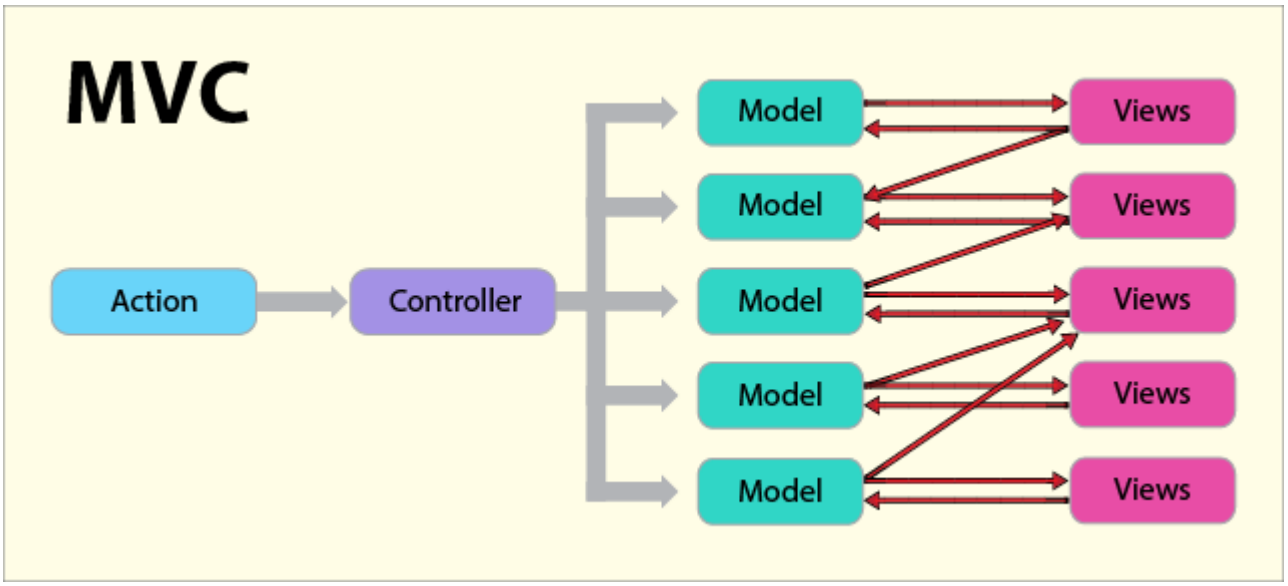


Рисунок 1.15. Схема MVC React з розділенням логіки, представлення і даних

Переваги React:

- Легкий у використанні для невеликих та середніх проєктів;
- Швидка робота завдяки віртуальному DOM;
- Велика кількість додаткових бібліотек та компонентів для розширення функціональності.

Недоліки React:

- Потребує більше зусиль для керування станом додатку у великих проєктах;
- Може бути складним для інтеграції з іншими технологіями.

Vue.js — це прогресивний фреймворк для створення інтерфейсів користувача, який має низький поріг входження і добре підходить для швидкої розробки інтерактивних веб-додатків. Він підтримує компонентну архітектуру та реактивне зв’язування даних, що забезпечує високу продуктивність та зручність розробки (рис.1.16).

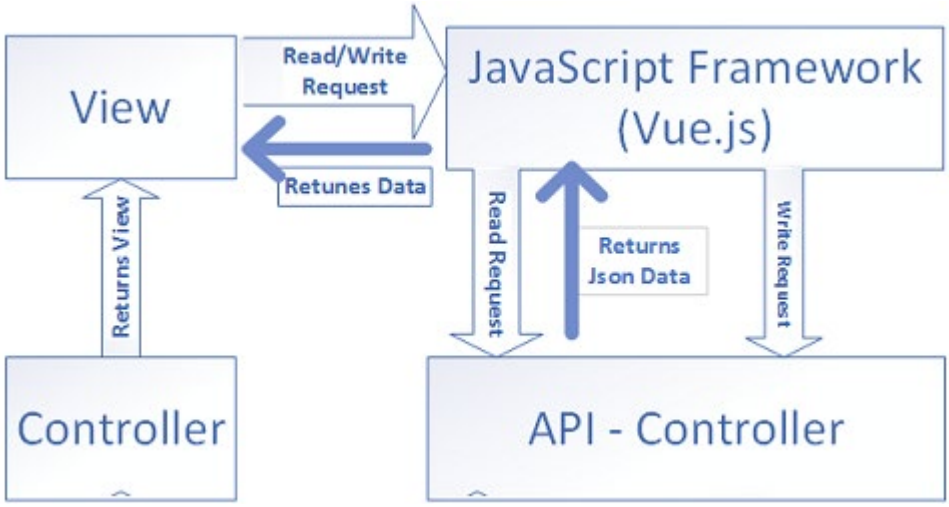


Рисунок 1.16. Схема MVC Vue.js з розділенням логіки, представлення і даних

Переваги Vue.js:

- Легкість вивчення та використання;
- Реактивне зв'язування даних між компонентами;
- Підтримка великої кількості плагінів та модулів для розширення функціоналу;
- Гнучка інтеграція з існуючими проєктами;

Недоліки Vue.js:

- Може бути складним у великих проєктах через необхідність управляти подіями між компонентами;
- Менша спільнота розробників порівняно з React або Angular.

1.3.3 Огляд бібліотек для візуалізації даних

Chart.js — проста та потужна бібліотека для візуалізації даних у вигляді діаграм. Вона підтримує основні типи діаграм (лінійні, стовпчикові, кругові тощо) і добре підходить для невеликих проєктів (рис.1.17).

Переваги Chart.js:

- Простота у використанні та налаштуванні;
- Легка вага бібліотеки;
- Підтримка анімацій та інтерактивності;

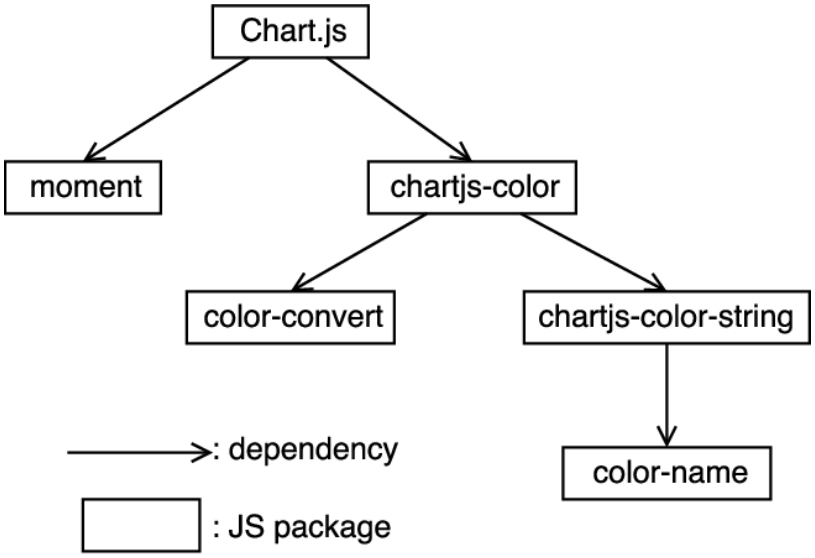


Рисунок 1.17. Ланцюжки залежностей у Chart.js

Недоліки Chart.js:

- Обмежені можливості для кастомізації та розширення;
- Мало функцій для обробки великих даних.

D3.js — бібліотека для створення складних і гнучких візуалізацій на основі даних. Вона дозволяє повністю контролювати процес побудови графіків та діаграм за допомогою SVG, HTML і CSS (рис.1.18).

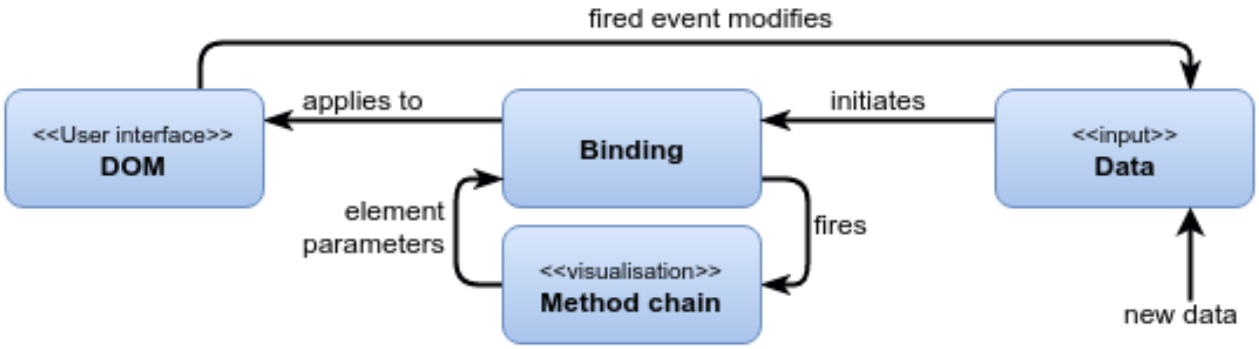


Рисунок 1.18. Схема обробки даних у D3.js

Переваги D3.js:

- Гнучкість у налаштуванні та створенні кастомних візуалізацій;
- Підтримка великої кількості типів графіків;
- Можливість інтеграції з іншими бібліотеками;

Недоліки D3.js:

- Велика складність у використанні;
- Не має готових компонентів для простих задач, що уповільнює розробку.

ECharts — це потужна бібліотека для візуалізації даних, яка підтримує широкий спектр діаграм і можливостей для інтерактивної взаємодії з даними. Вона легко інтегрується з Vue.js і дозволяє створювати динамічні графіки з можливістю налаштування стилів (рис.1.19).

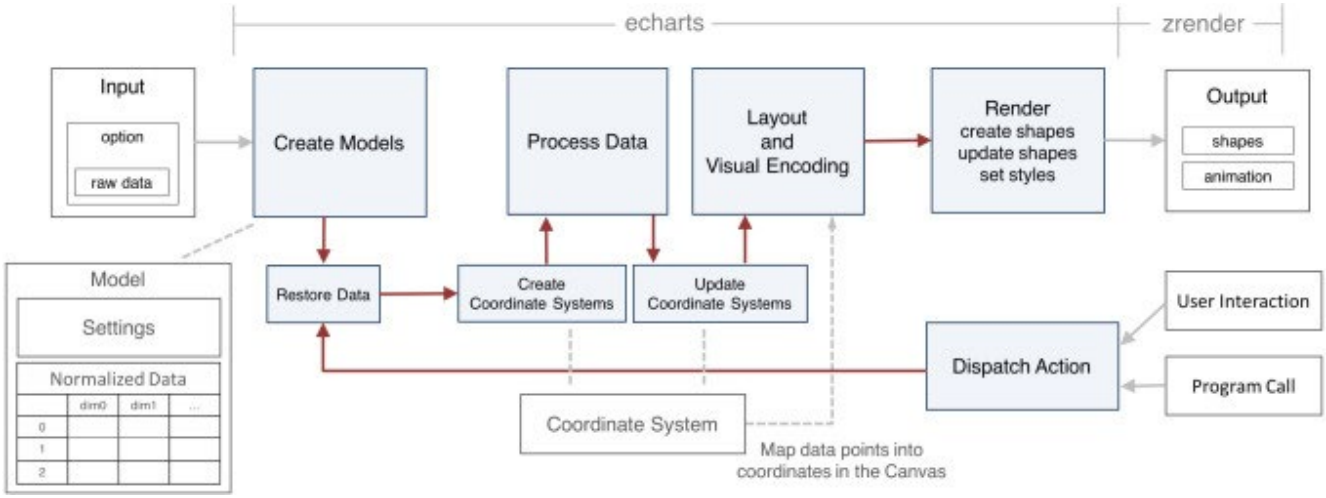


Рисунок 1.19. Архітектура бібіліотеки ECharts

Переваги ECharts:

- Широкий набір типів діаграм;
- Підтримка інтерактивності та обробки подій;
- Висока продуктивність при роботі з великими обсягами даних;

Недоліки ECharts:

- Вища складність налаштування порівняно з Chart.js;
- Деякі типи діаграм потребують додаткового налаштування.

Для реалізації веб-застосунку для візуалізації числових даних було обрано JavaScript, оскільки він надає гнучкі можливості для інтерактивної візуалізації та легко інтегрується з сучасними бібліотеками. Vue.js як фреймворк забезпечить швидку розробку, а ECharts надасть потужний набір інструментів для створення складних діаграм із можливістю інтерактивної взаємодії з даними.

1.4 Розробка моделі життєвого циклу компоненту для візуалізації числових даних

1.4.1 Визначення вимог до розробки компоненту

Для розробки віджету, який буде відповідати вимогам нашого застосунку для візуалізації числових даних, ключовими аспектами є продуктивність, гнучкість та

легкість у налаштуванні та інтеграції. Обрані інструменти для реалізації — JavaScript, Vue.js, та ECharts — забезпечують потужні можливості, але потребують певної структури для контролю життєвого циклу компоненту.

Основними вимогами до компоненту є:

- Продуктивність: Компонент має швидко завантажуватися та рендеритися, забезпечуючи плавну роботу навіть при великих обсягах даних;
- Інтерактивність: Віджет повинен підтримувати інтерактивні можливості, такі як масштабування, фільтрація, або взаємодія з діаграмами;
- Масштабованість: Компонент повинен бути адаптованим для різних типів діаграм, які надаються бібліотекою ECharts, дозволяючи гнучко змінювати відображення даних без необхідності значних змін у коді;
- Легкість інтеграції: Віджет має бути легко інтегрований у будь-який інший Vue.js додаток без значних змін у конфігурації.

Для того, щоб ці вимоги були дотримані, необхідно побудувати чітку модель життєвого циклу компоненту, яка буде підтримувати процес створення, оновлення та знищення компонентів, а також дозволить ефективно керувати змінами у даних.

1.4.2 Створення схеми життєвого циклу

Компонент на основі Vue.js слідує стандартному життєвому циклу, який включає різні етапи від ініціалізації до знищення компоненту. Це дозволяє ефективно контролювати процес взаємодії з даними та Document Object Model.

Основні етапи життєвого циклу компоненту для візуалізації числових даних (рис.1.22) включають:

- beforeCreate: Ініціалізація компоненту. Дані та події ще не встановлені.
- created: Дані компоненту готові, але його шаблон ще не відмальовано.
- beforeMount: Виконується перед відображенням компоненту. Використовується для зміни Document Object Model структури перед її рендерингом.
- mounted: Після відображення компоненту. На цьому етапі можна взаємодіяти з Document Object Model та доступними даними.
- beforeUpdate: Виконується перед оновленням даних у компоненті,

наприклад, при зміні реактивних властивостей.

- updated: Виконується після оновлення компоненту та доступу до оновленого Document Object Model.
- beforeDestroy: Перед демонтажем компоненту. Забезпечує доступ до даних та подій перед знищенням.
- destroyed: Компонент повністю видалено з Document Object Model.

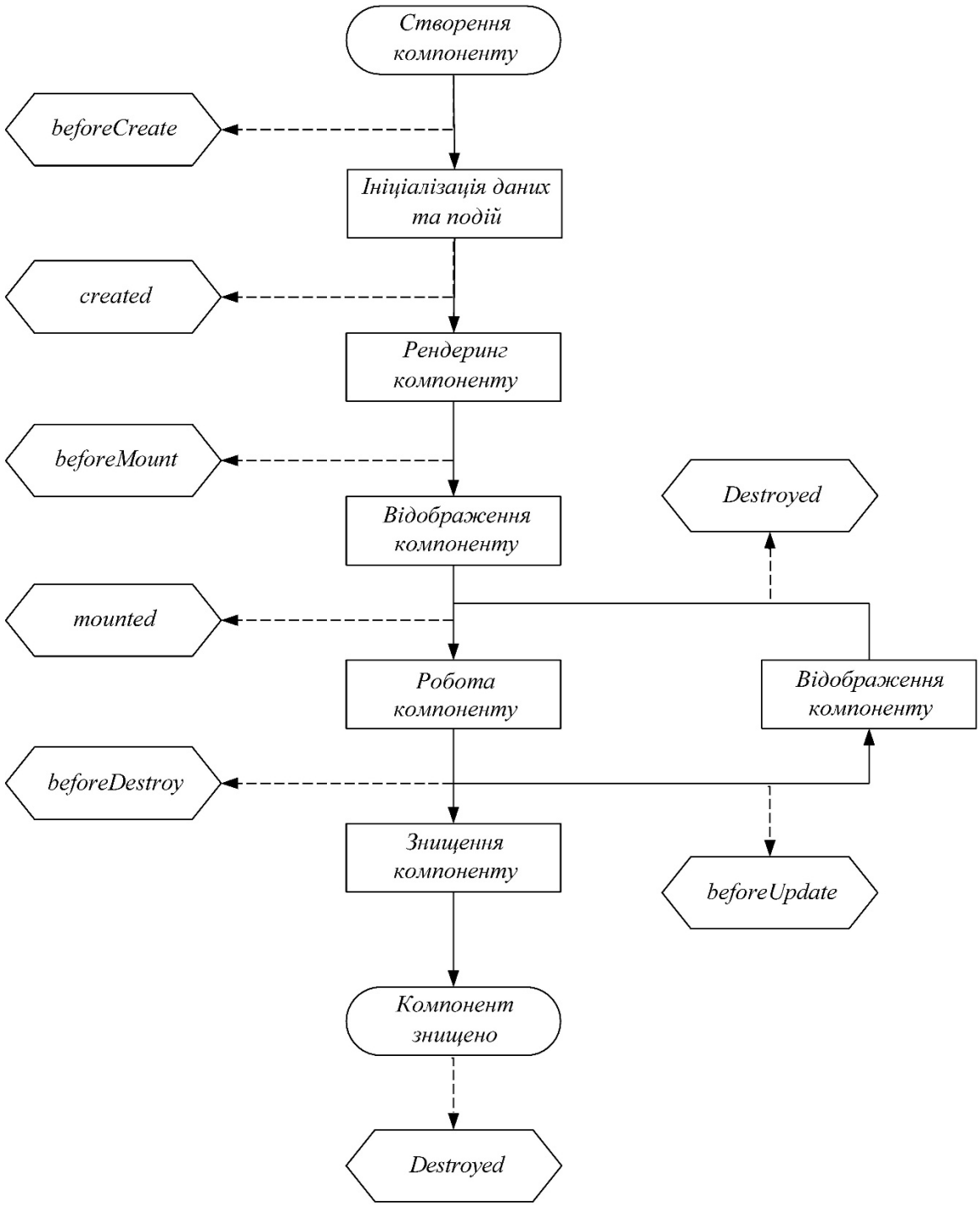


Рисунок 1.22. Схема життєвого циклу компоненту для візуалізації числових даних

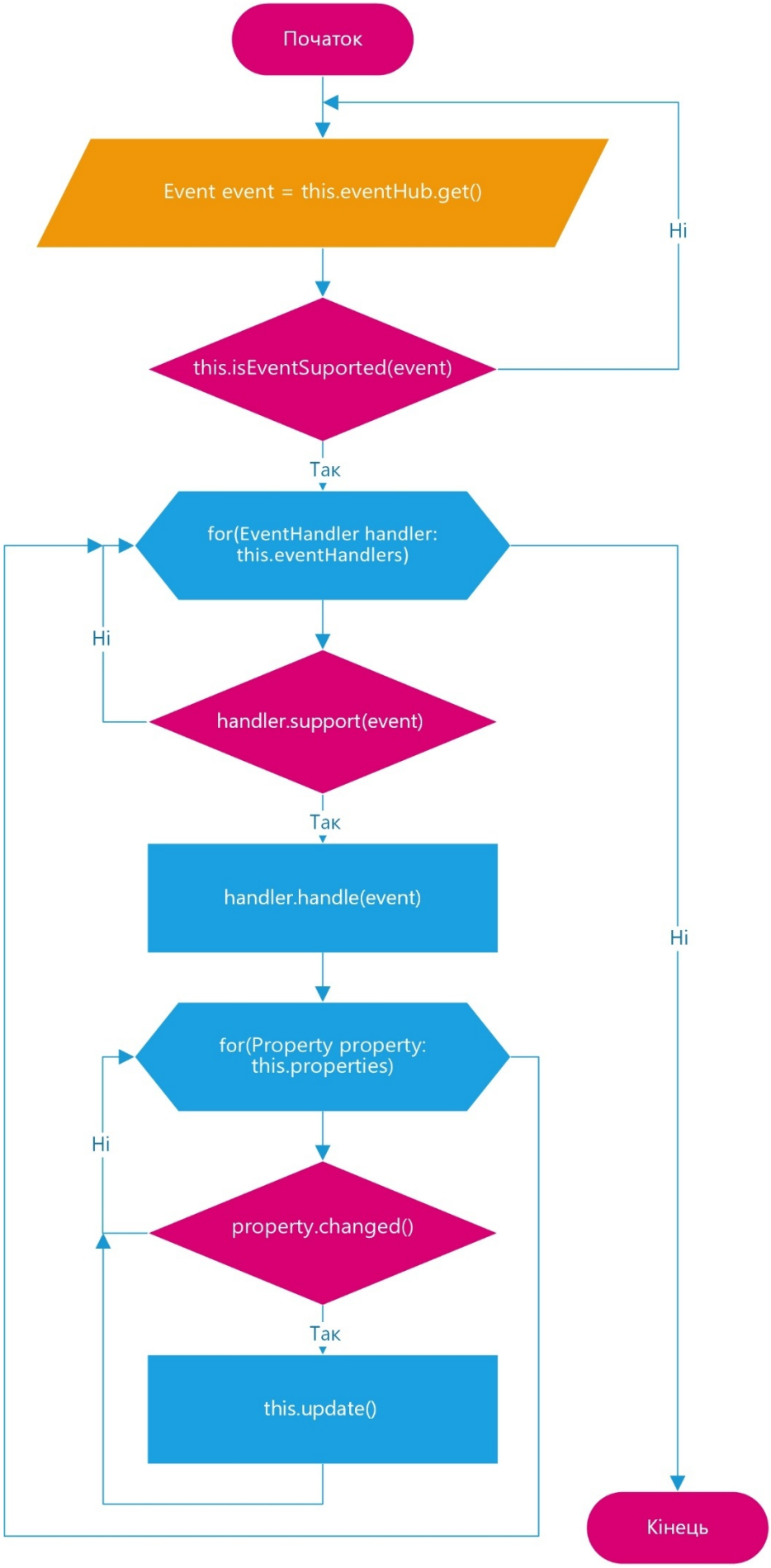


Рисунок 1.23. Блок-схема алгоритму обробки подій компоненту через Vue.js

Схема цього життєвого циклу ілюструє послідовність подій і дозволяє контролювати, коли і як відбувається взаємодія з компонентом під час його роботи у веб-застосунку. Ці етапи дозволяють нам забезпечити максимальну продуктивність та контроль за станом компоненту під час візуалізації даних.

На основі цих етапів створюється модель віджету, яка може бути розширена для різних типів діаграм, що надаються бібліотекою ECharts, забезпечуючи тим самим високу гнучкість для різноманітних сценаріїв використання.

1.4.3 Інтеграція подій та їх обробка

Для інтерактивної візуалізації числових даних, компонент повинен підтримувати підписку на події та обробку змін у даних у реальному часі. Vue.js дозволяє легко обробляти події та реактивність даних, що особливо важливо при роботі з великими обсягами динамічної інформації.

За допомогою ECharts можна налаштувати різні типи інтерактивних діаграм з можливістю реагування на дії користувача (наприклад, наведення на елементи, клік або масштабування). Реалізація подібної функціональності потребує гнучкого налаштування обробників подій, які можуть бути легко інтегровані в життєвий цикл компоненту через Vue.js (рис.1.23).

У підсумку, дана модель життєвого циклу компоненту дозволить реалізувати віджет для візуалізації числових даних, що відповідає вимогам гнучкості, продуктивності та інтерактивності, зручно інтегрується в існуючі рішення на базі Vue.js та ECharts, та забезпечує простоту налаштування для різних типів візуалізації даних.

1.5 Розробка UML-діаграми класів для візуалізації числових даних

Для розробки набору віджетів, що візуалізує числові дані у вигляді діаграм, важливо спроектувати ефективну структуру класів, яка забезпечить гнучкість, модульність і простоту розширення. Обрані технології JavaScript, Vue.js, та бібліотека ECharts надають необхідний інструментарій для реалізації цього завдання, а побудована UML-діаграма класів дозволяє чітко розуміти архітектуру застосунку (рис.1.24).

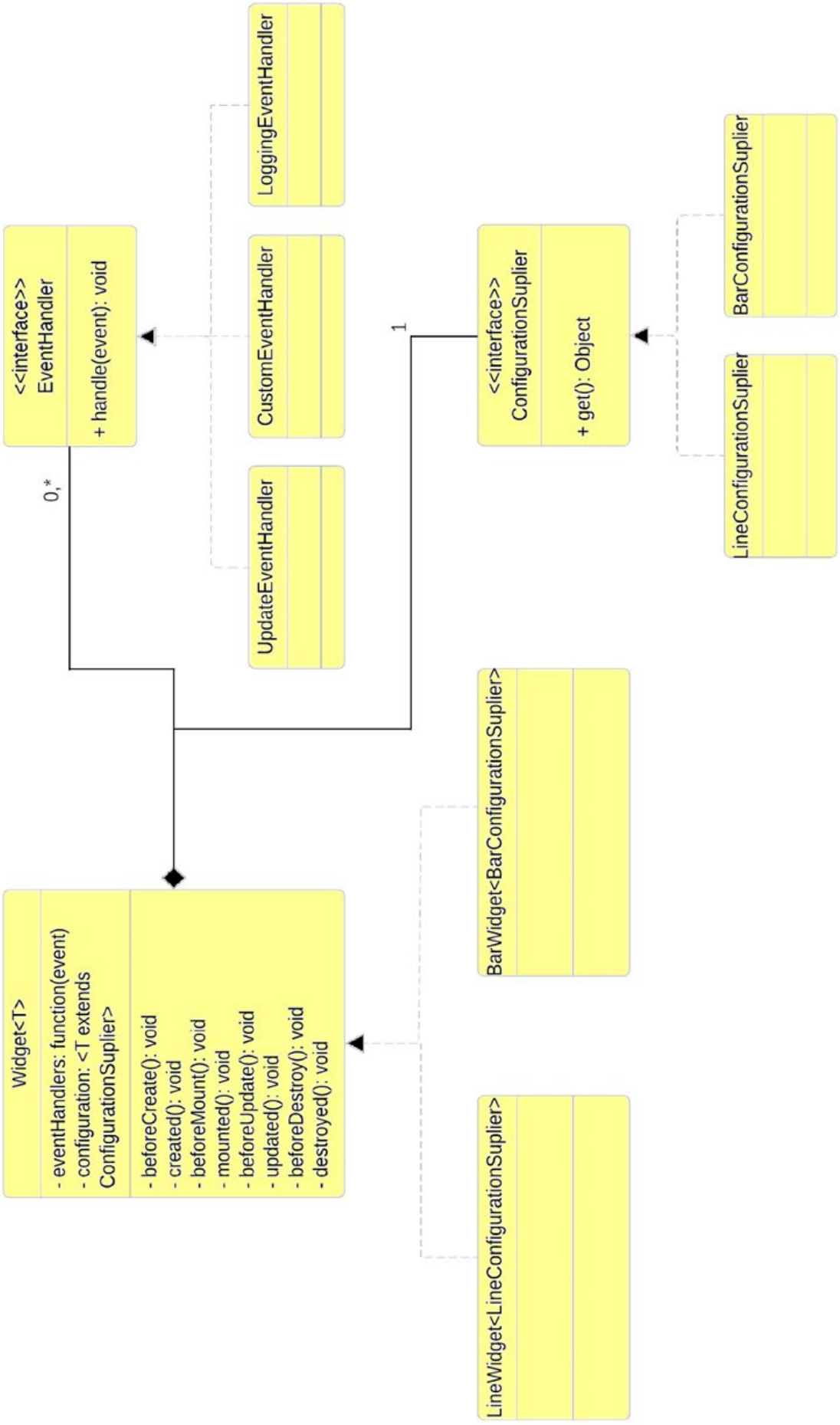


Рисунок 1.24. UML-діаграма класів для візуалізації числових даних

На основі розробленої UML-діаграми (див. рис. 1.24) можна виділити кілька ключових елементів:

1. Базовий клас Widget<T>:
   - Цей клас є загальним компонентом для всіх типів віджетів. Він містить основний набір методів, які контролюють життєвий цикл компоненту, такі як beforeCreate, created, beforeMount, mounted, beforeUpdate, updated, beforeDestroy, та destroyed;
   - Клас також включає механізм обробки подій за допомогою об'єкта eventHandlers. Це дозволяє керувати подіями у віджетах через відповідні обробники;
   - Конфігурація віджетів задається через параметр configuration, що робить клас гнучким та придатним для використання з різними типами діаграм, такими як лінійчаті діаграми та гістограми;
2. Інтерфейс EventHandler:
   - Відповідає за обробку подій, таких як оновлення віджету або спеціальні події користувача. Кожен обробник реалізовує метод handle(event), який дозволяє працювати з різними типами подій;
   - У діаграмі представлені кілька специфічних реалізацій обробників подій: UpdateEventHandler, CustomEventHandler, та LoggingEventHandler, що забезпечують обробку подій відповідно до контексту застосування;
3. Конфігураційний інтерфейс ConfigurationSupplier:
   - Цей інтерфейс визначає, яким чином має бути отримана конфігурація для кожного типу віджету. Клас Widget<T> використовує цей інтерфейс для генерації специфічних конфігурацій під час ініціалізації віджету;
   - Для кожного типу діаграми (лінійчатої або гістограми) розроблено окремі класи конфігурацій: LineConfigurationSupplier та BarConfigurationSupplier, що визначають специфічні параметри для кожного типу візуалізації;

4. Специфічні класи віджетів: LineWidget та BarWidget є реалізаціями базового класу Widget<T>, де кожен з них відповідає за рендеринг лінійчатої та стовпчикової діаграм відповідно. Кожен з цих класів отримує свою конфігурацію через відповідні класи LineConfigurationSupplier та BarConfigurationSupplier.

Основною ідеєю цієї UML-діаграми є перевикористання базових компонентів та інтерфейсів для створення віджетів різних типів діаграм. Це дозволяє значно зменшити кількість повторного коду та спростити підтримку та розширення застосунку. Замість створення окремого компонента для кожного виду діаграм, можна легко адаптувати існуючий базовий клас, що призведе до зменшення обсягу коду та підвищення гнучкості. Також за рахунок чіткої структури класів та використання інтерфейсів, досягнуто модульності системи, що дозволяє легко додавати нові типи віджетів або обробників подій у майбутньому, без суттєвих змін у вже існуючому коді.

Розроблена UML-діаграма класів для застосунку з візуалізації числових даних демонструє ефективну і модульну архітектуру, яка базується на перевикористанні компонентів і гнучкості конфігурацій. Це забезпечує можливість розширення застосунку з мінімальними зусиллями та спрощує підтримку в довгостроковій перспективі.

1.6 Розробка структури набору віджетів для візуалізації числових даних у вигляді діаграм

Створення структури набору віджетів для візуалізації числових даних базується на ефективному використанні сучасних веб-технологій, таких як JavaScript, Vue.js, та бібліотека ECharts для візуалізації. Правильна організація файлової структури та розбиття на компоненти дозволяє створити гнучкий, модульний та легко розширюваний застосунок.

1.6.1 Управління структурою та збіркою проекту

Для управління залежностями та автоматизації збірки проекту використовується пакетний менеджер npm. У файлі package.json в кореневій директорії проекту знаходяться всі кінцеві та розробницькі залежності, а також скрипти для збірки і тестування. У файлі package-lock.json зафіксовані версії всіх

пакетів і хеші, що гарантує сталість залежностей на різних середовищах.

Для контролю версій та співпраці над проектом використовується система контролю версій Git. Тимчасові файли, бібліотеки та згенеровані файли ігноруються через файл .gitignore, що зменшує ризик непотрібних конфліктів під час комітування змін.

У кореневій директорії знаходяться такі важливі папки та файли:

- /config – містить усі конфігураційні файли для середовищ розробки, тестування та продакшену, а також файли для маршрутизації та інтернаціоналізації;
- /assets – містить медіа, стилі та інші статичні ресурси, необхідні для роботи застосунку;
- /src – головна директорія для коду програми, де розміщено основні частини застосунку, такі як компоненти, шаблони та логіка бізнес-процесів.

1.6.2 Основні директорії проекту та їх призначення

У директорії /src структуру застосунку побудовано наступним чином:

- /components – основні компоненти, які використовуються для створення окремих частин інтерфейсу, таких як діаграми та панелі керування;
- /mixins – містить загальні методи та функції, які можуть бути повторно використані в різних компонентах. Тут знаходяться логіка обробки подій, управління життєвим циклом компонентів, функції для міжнародної підтримки (інтернаціоналізації);
- /state – зберігає описи моделей даних, що використовуються в додатку. Це дозволяє централізовано управляти станом додатку і забезпечити передачу даних між різними компонентами.

1.6.3 Структура та категорії компонентів

Компоненти, що відповідають за візуалізацію діаграм, поділені на кілька підкатегорій:

- /core – містить базові компоненти для віджетів. Тут розміщені основні класи і інтерфейси, які забезпечують функціонування кожного віджету (рис.1.25);

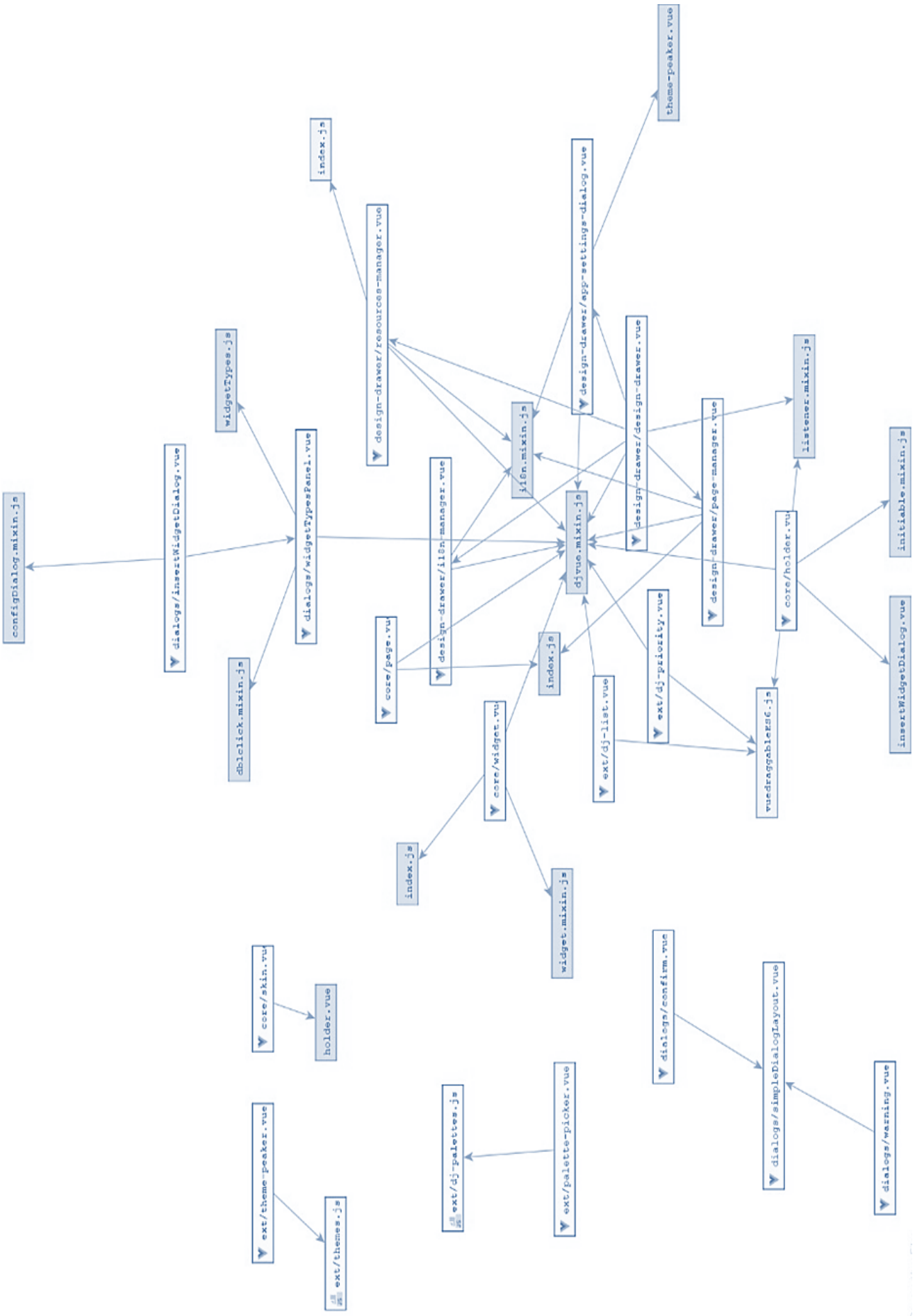


Рисунок 1.25. Схема зв’язку основних компонентів для віджетів

- /layouts – шаблони розміщення компонентів та віджетів на сторінках застосунку. Вони визначають структуру сторінки та позиціонування віджетів (рис.1.26);
- /widgets – тут зберігаються усі віджети, кожен з яких відповідає за певний тип діаграми.

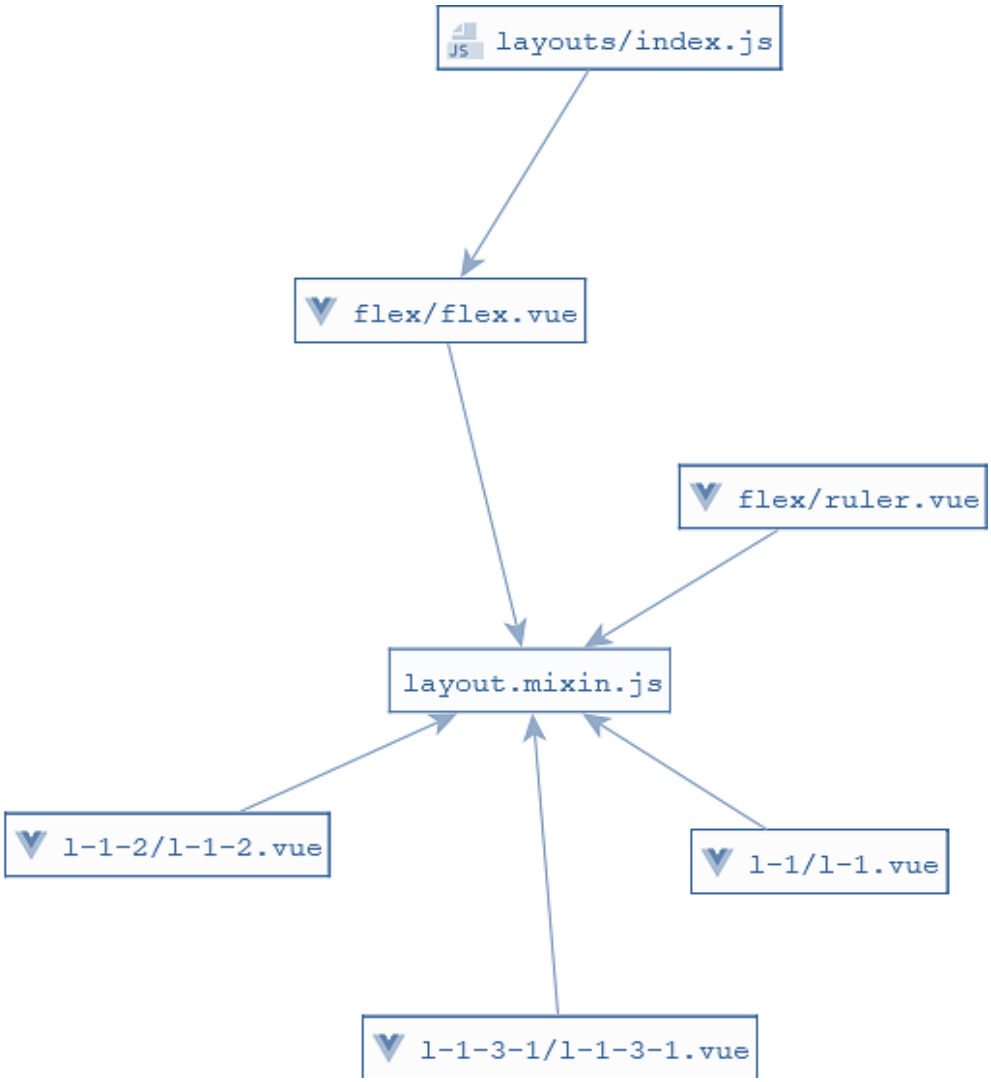


Рисунок 1.26. Схема зв’язку шаблонів розміщення компонентів та віджетів

1.6.4 Структура та склад віджетів

Кожен віджет складається з трьох основних файлів:

- chart.type.js – відповідає за визначення типу діаграми, іконок, стилів та методів конфігурації для цієї діаграми;
- chart.vue – містить шаблон віджету та основну логіку роботи з діаграмами, включаючи обробку подій, керування станом та оновлення візуалізації;
- snippet.js – включає різноманітні конфігурації для діаграми, що дозволяє

користувачам обирати потрібний тип візуалізації та налаштовувати віджет відповідно до своїх потреб.

На прикладі лінійчатої діаграми (див. рис. 1.27), віджет може відображати стандартні лінійчаті діаграми, складені діаграми або діаграми з областями. Кожен тип можна легко конфігурувати, що забезпечує гнучкість та мінімізацію коду.

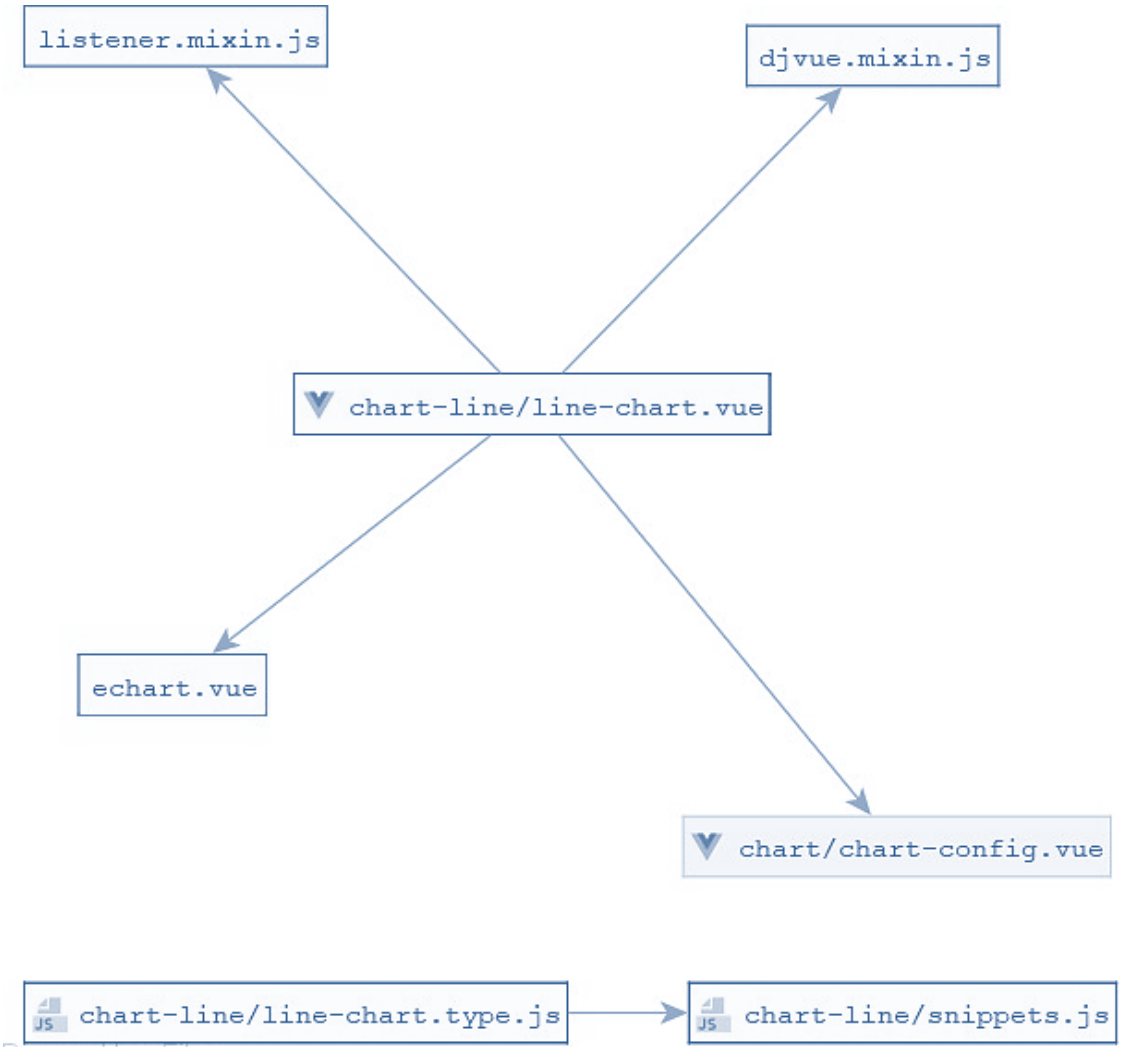


Рисунок 1.27. Модель кінцевого віджету для візуалізації лінійчатих діаграм

Структура застосунку побудована таким чином, щоб забезпечити максимальну гнучкість і повторне використання компонентів. Поділ на директорії core, layouts та widgets дозволяє підтримувати ясність архітектури та простоту розширення функціональності.

На рис. 1.27 наведено приклад структури лінійчатого віджету, що демонструє, як різні компоненти працюють разом для досягнення високого рівня інтерактивності та динамічності у візуалізації даних.

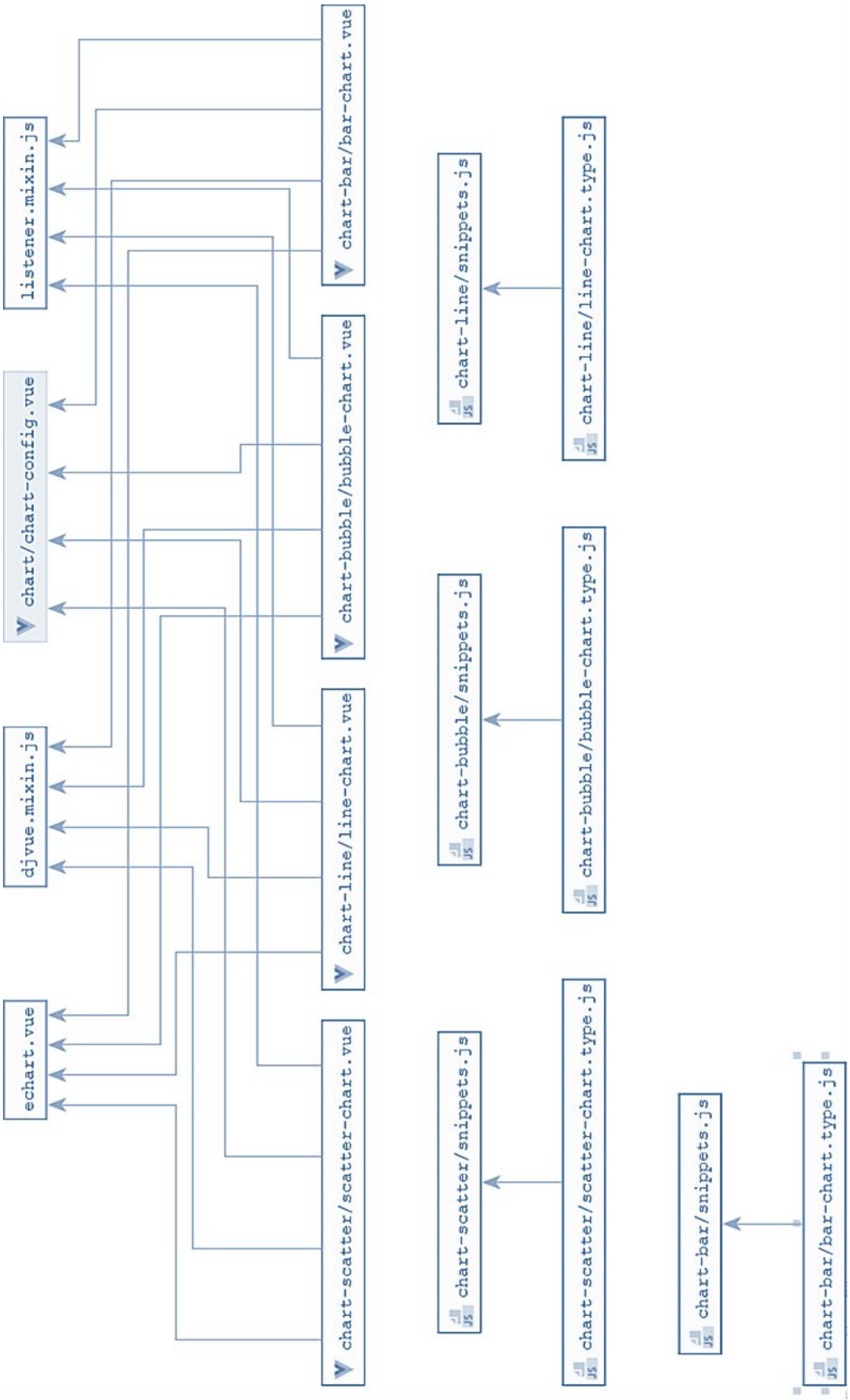


Рисунок 1.28. Схема зв’язку віджетів для розробленого набору

Запропонована структура проекту для візуалізації числових даних є гнучкою та ефективною, що дозволяє легко керувати як віджетами, так і загальними компонентами. Завдяки використанню Vue.js та ECharts, можна з легкістю адаптувати віджети під різні типи діаграм та конфігурацій, забезпечуючи оптимальну візуалізацію для користувачів (рис.1.28).

1.7 Реалізація проекту мовою програмування JavaScript

Реалізація застосунку виконувалась на основі побудованих вище діаграм та алгоритмів. Основною метою було створити динамічний інтерфейс користувача та забезпечити обробку даних у відповідності до поставлених завдань. Нижче наведено покроковий аналіз основних компонентів коду, що було розроблено мовою програмування JavaScript (Додаток А).

1.7.1 Структура та основні компоненти коду

На початковому етапі була визначена загальна структура застосунку та виконано її розподілення на кілька ключових модулів:

1. HTML-документ – структура сторінки, що містить основні елементи інтерфейсу, такі як поля введення, кнопки та контейнер для відображення результатів;
2. CSS-стилі – забезпечують візуальне оформлення елементів інтерфейсу;
3. JavaScript – основна логіка обробки подій та взаємодії з користувачем.

Код JavaScript забезпечує взаємодію з HTML-документом за допомогою Document Object Model, що дозволяє відстежувати події, змінювати елементи на сторінці, а також обробляти та відображати результати.

1.7.2 Обробка подій та взаємодія з користувачем

Однією з ключових функцій програми є обробка подій від користувача, таких як натискання кнопок або введення даних. Для цього використовуються обробники подій, що прив'язані до конкретних елементів інтерфейсу. Наприклад, для кнопки "Submit" був доданий обробник подій, що відповідає за зчитування введених користувачем даних та їх подальшу обробку:

javascript

```
КопіюватиРедагувати
document.getElementById('submitButton').addEventListener('click', function() {
    const inputData = document.getElementById('inputField').value;
    processData(inputData);
});
```

У цьому фрагменті коду відбувається звернення до кнопки за допомогою її ідентифікатора та прив'язування до неї обробника події "click", який викликає функцію processData() з параметром, що містить введені користувачем дані.

1.7.3 Логіка обробки даних

Функція processData() є центральною частиною логіки програми. Вона приймає введені дані, обробляє їх у відповідності до розроблених алгоритмів, а потім генерує результати, що будуть відображені на сторінці. Для обробки даних можуть бути використані різні функції, що відповідають за окремі етапи алгоритму:

```
javascript
КопіюватиРедагувати
function processData(data) {
    // Перевірка вхідних даних
    if (validateData(data)) {
        const result = performCalculations(data);
        displayResult(result);
    } else {
        displayError('Некоректні дані. Введіть правильні значення.');
    }
}
```

Функція validateData() перевіряє коректність введених даних, а performCalculations() виконує основні обчислення згідно з алгоритмом. Результати обробки виводяться на екран за допомогою функції displayResult().

1.7.4 Відображення результатів

Для відображення результатів користувачу використовується спеціальний блок у HTML-документі, куди JavaScript динамічно додає необхідну інформацію. Наприклад, якщо дані оброблені коректно, результат буде відображено наступним чином:

```
javascript
КопіюватиРедагувати
```

function displayResult(result) {

document.getElementById('resultContainer').innerHTML = 'Результат: ' + result;

}

Якщо введені дані некоректні, користувач побачить повідомлення про помилку:

javascript

КопіюватиРедагувати

function displayError(message) {

document.getElementById('resultContainer').innerHTML = '' + message + '';

}

1.7.5 Взаємодія між компонентами

Всі компоненти програми, включаючи обробку подій, перевірку даних та відображення результатів, взаємодіють між собою завдяки використанню функцій та глобальних змінних. Також було дотримано принципу інкапсуляції, що забезпечує ізольованість функцій та дозволяє легко змінювати або доповнювати окремі частини коду без ризику порушення логіки всієї програми.

Таким чином реалізується основна логіка взаємодії з користувачем та обробки даних на основі попередньо розроблених алгоритмів та діаграм. Завдяки використанню DOM, обробників подій та функцій для обробки даних, застосунок працює динамічно та надає користувачеві зручний інтерфейс для отримання результатів у реальному часі.

1.8 Встановлення та тестування розробленого набору віджетів для візуалізації числових даних у вигляді діаграм

1.8.1 Встановлення набору віджетів

Для встановлення розробленого застосунку з візуалізації числових даних, необхідно виконати наступні кроки:

1. Завантаження проекту: Спочатку користувач повинен завантажити або склонував репозиторій проекту з платформи керування версіями, наприклад,

GitHub, за допомогою команди:

2. git clone <URL проекту>

3. Встановлення залежностей: Додаток використовує ряд залежностей та бібліотек, таких як D3.js для побудови діаграм, які потрібно встановити. Для цього необхідно виконати команду:

4. npm install

Ця команда виконає завантаження всіх необхідних пакетів та бібліотек, перелічених у файлі package.json.

5. Запуск додатку: Після встановлення всіх залежностей, для запуску застосунку можна скористатися командою:

6. npm start

Після виконання цієї команди додаток відкриється у браузері за стандартною адресою localhost:3000, де користувач зможе взаємодіяти з інтерфейсом програми.

7. Доступ до веб-інтерфейсу: Додаток надає інтуїтивний інтерфейс для користувачів, що дозволяє завантажувати дані у форматі CSV, обирати тип діаграм для візуалізації та конфігурувати параметри відображення.

1.8.2 Тестування коректності розробленого набору віджетів

Тестування є важливою частиною розробки програмного забезпечення, що дозволяє переконатися у коректності функціонування системи. Для тестування даного додатку було застосовано кілька підходів.

Для тестування функціональних можливостей додатку були створені тести з використанням популярного фреймворку для тестування – Jest. Це дозволило протестувати основні функції додатку, такі як завантаження даних, обробка помилок, та візуалізація даних у вигляді діаграм. Наприклад, були протестовані наступні аспекти:

1. Обробка користувацького введення: тести перевіряли, чи правильно обробляються введені користувачем дані у форматі CSV, а також чи коректно вони перетворюються у відповідні масиви для подальшої обробки;

2. Коректність побудови діаграм: було протестовано відображення різних типів діаграм, таких як лінійчата діаграма, стовпчаста діаграма та діаграма розкиду.

3. Обробка помилок: фреймворк Jest перевіряв, як додаток обробляє неправильні або неповні дані, та чи відображаються відповідні повідомлення про помилки для користувача.

На рисунку 1.29 представлено приклад тестового звіту, що був створений за допомогою Jest після виконання тестів.

```
PASS  app/actions/__tests__/activity_list.spec.js
PASS  app/actions/__tests__/messaging.spec.js
PASS  app/actions/__tests__/authentication.spec.js
PASS  app/actions/__tests__/payment.spec.js
PASS  app/actions/__tests__/fellow.spec.js
PASS  app/actions/__tests__/workflow.spec.js
PASS  app/actions/__tests__/environment.spec.js
PASS  app/actions/__tests__/loading.spec.js
PASS  app/reducers/mixpanel/__tests__/mixpanel.spec.js
PASS  app/reducers/loading/__tests__/loading.spec.js
PASS  app/components/__tests__/messaging.spec.js
PASS  app/components/__tests__/workflow.spec.js
PASS  app/components/custom/carousel/__tests__/carousel.spec.js (9.183s)
PASS  app/components/custom/v2/task_container/__tests__/task_container.spec.js (9.251s)
PASS  app/components/custom/v2/navigation_bar/__tests__/home_nav_bar.spec.js (10.412s)
PASS  app/components/custom/v2/activity_list/__tests__/task_row.spec.js (10.017s)
PASS  app/components/custom/v2/navigation_bar/__tests__/back_nav_bar.spec.js (10.745s)
PASS  app/components/custom/v2/navigation_bar/__tests__/cancel_icon.spec.js (9.212s)
PASS  app/components/custom/headers/__tests__/edit_nav_bar.spec.js (9.852s)
PASS  app/components/custom/headers/__tests__/modal_close_icon.spec.js (9.623s)
PASS  app/components/custom/headers/__tests__/modal_header.spec.js (8.961s)
PASS  app/components/custom/headers/__tests__/save_icon.spec.js (9.784s)
PASS  app/components/authentication/__tests__/password.spec.js
PASS  app/components/authentication/__tests__/enter_login.spec.js (11.327s)

Test Suites: 22 passed, 22 total
Tests:       229 passed, 229 total
Snapshots:   229 passed, 229 total
Time:        14.562s
Ran all test suites.
```

Рисунок 1.29. Тестовий звіт з використанням фреймворку Jest

Крім юніт-тестування, було проведено інтеграційне тестування, яке перевіряло взаємодію між різними компонентами програми. Це дозволило переконатися, що компоненти, такі як завантаження даних та їх візуалізація, коректно працюють у зв’язці. Інтеграційне тестування було виконано вручну

шляхом перевірки таких елементів, як:

- Коректність побудови діаграм при зміні типу відображення.
- Можливість збереження діаграм у різних форматах.
- Налаштування кольорів та стилів діаграм.

1.8.3 Використання розробленого набору віджетів та створення діаграм

Після успішного встановлення та запуску застосунку, його основний інтерфейс представлений стартовою сторінкою, де користувач має можливість завантажувати числові дані для візуалізації.

На рис. 1.30 показано головний екран застосунку після запуску, на якому відображаються можливості вибору файлу з даними та вибору типу діаграми.

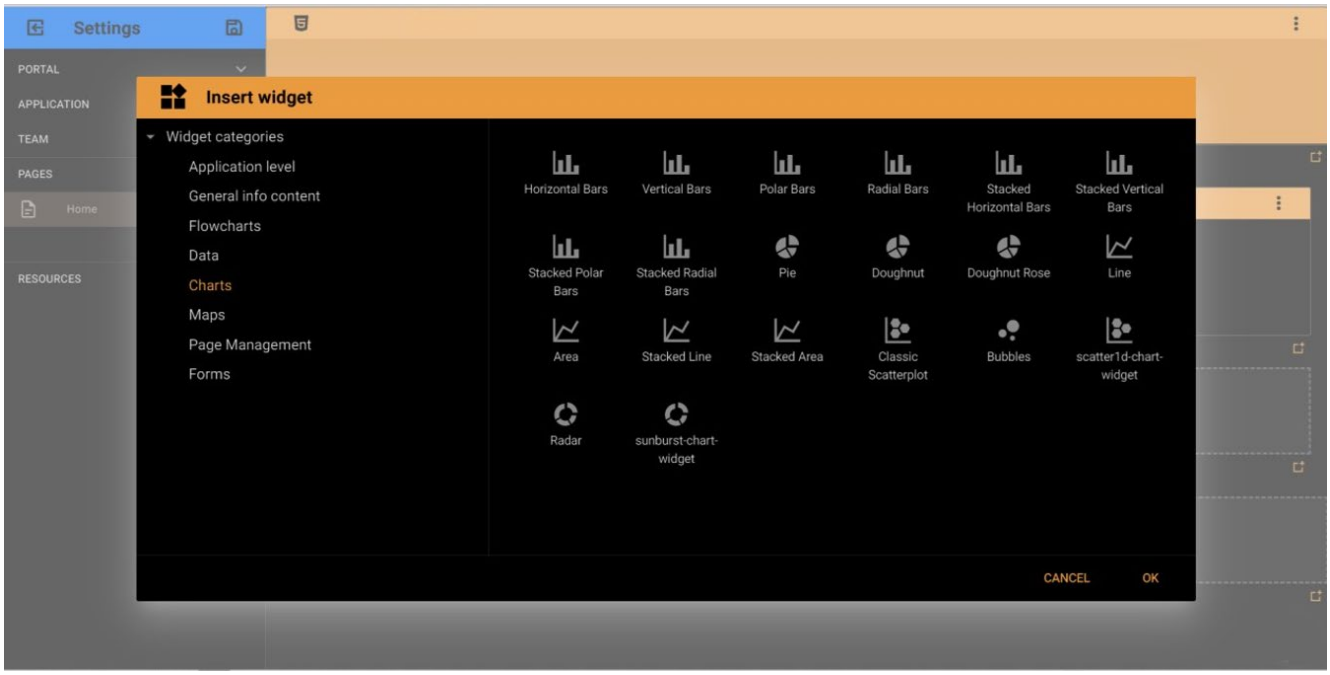


Рисунок 1.30. Вигляд головного екрану застосунку та вікна обирання віджетів

Для початку роботи із застосунком користувач може завантажити файл з даними у форматі CSV, після чого застосунок запропонує обрати тип діаграми для візуалізації цих даних.

На рис. 1.31 наведено приклад лінійчатої та стовпчастої діаграми, побудованих на основі тестових даних. Застосунок також надає можливість гнучко налаштовувати візуальні параметри діаграм, змінюючи палітру кольорів, стилі ліній та інші параметри.

Рисунок 1.31. Створення лінійчатої та стовпчастої діаграми

На рис. 1.32 наведено екран налаштувань віджету, де користувач може змінювати опції відображення діаграми.

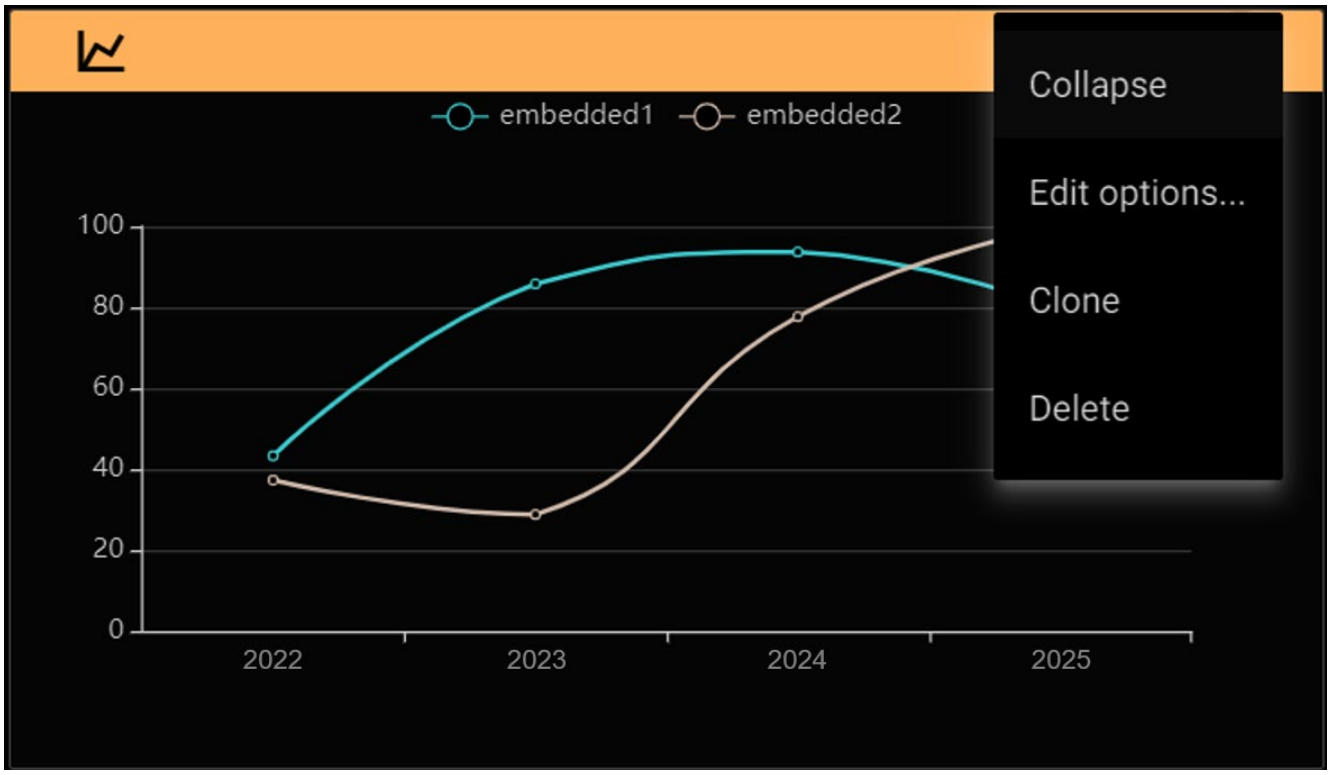


Рисунок 1.32. Управління віджетом з спливаючого меню

Крім того, користувач може додавати або видаляти віджети, що відповідають за різні типи діаграм, а також редагувати їх конфігурації в реальному часі (рис.1.33).

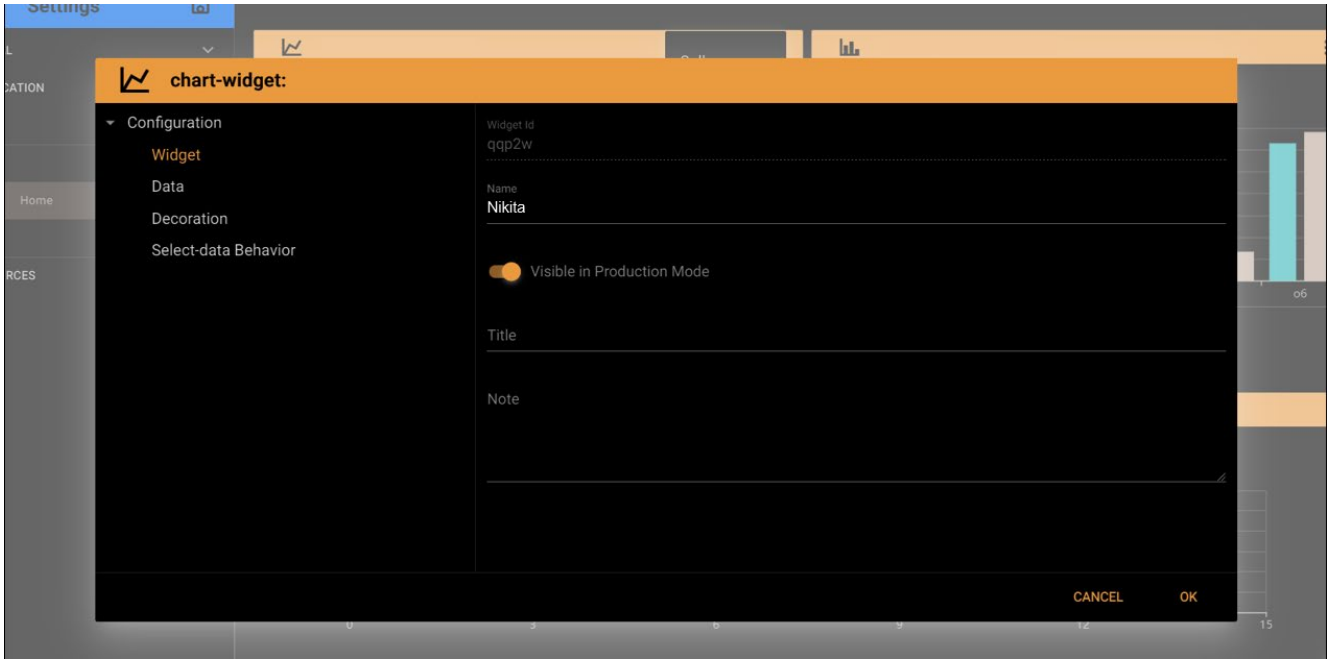


Рисунок 1.33. Екран конфігурування властивостей віджету

Тестування застосунку дозволило переконатися у стабільності його роботи та коректності відображення числових даних у вигляді діаграм (рис.1.34). Використання фреймворку Jest для юніт-тестування забезпечило надійність функцій, а інтеграційне тестування показало, що всі компоненти системи коректно взаємодіють між собою.

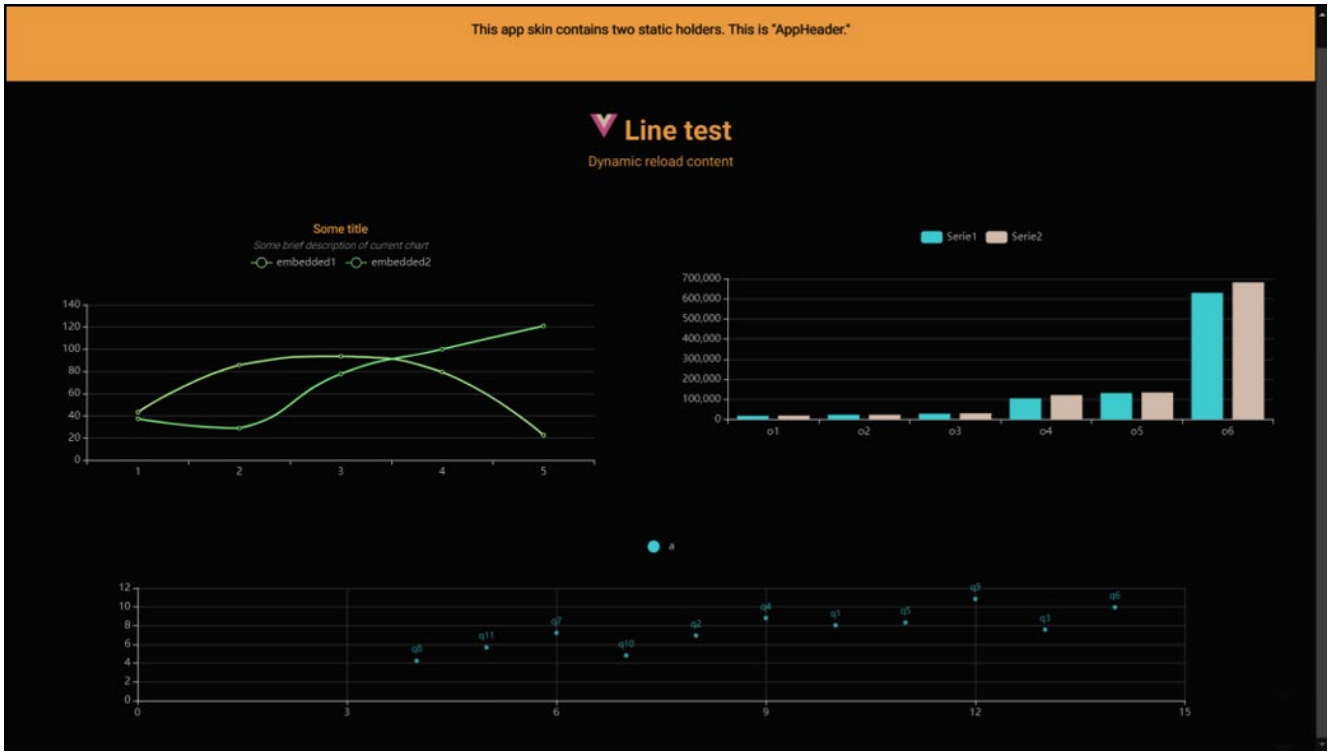


Рисунок 1.34. Інтерфейс для відображення даних у робочому режимі

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

Темою даного дипломного проекту є розробка набору віджетів для візуалізації числових даних у вигляді діаграм.

Основною метою даного дипломного проекту є створення набору віджетів, які забезпечують візуалізацію числових даних у форматі діаграм. Ефективність розробленого програмного продукту залежить від його якості та оптимізації процесу створення. Показники якості ПП можна оцінювати за такими критеріями: зручність для користувачів, ефективність використання ресурсів, відповідність встановленим вимогам програмного забезпечення. Оцінювання якості також передбачає аналіз трудомісткості та розрахунок вартості розробки.

2.2 Визначення трудомісткості розробки програмного забезпечення

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт.

Таблиця 2.1. Каталог аналогів

Найменування ПП	Обсяг функції ПП – V_o, ум. машинних команд
1. ПП СУБД	2500 – 9800
2. Комплексні системи ведення БД	940 – 7450
3. ПП введення інформації	1070 – 5850

На основі отриманих даних із довідника встановлюється базова норма часу для розробки аналогічного програмного забезпечення. Вихідне значення (229 людогодин) коригується за допомогою поправочного коефіцієнта, що враховує умови розробки, зокрема роботу в умовах комп’ютера, де коефіцієнт (Кк) може змінюватися від 0,7 до 0,8. В обраному прикладі використовується значення 0,8, що забезпечує більш консервативну оцінку:

Тар = 229 × 0,8 = 183,2 люд.-годин.

Сама ідея полягає в тому, щоб адаптувати загальну трудомісткість до реальних умов, забезпечуючи точніший прогноз часових витрат, який можна

використовувати для планування проекту та розподілу ресурсів:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \quad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \quad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \quad (2.3)$$

Li – питома вага і-го етапу розробки (див. табл. 2.2.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.3.);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.4).

Таблиця 2.2. Значення питомих коефіцієнтів трудомісткості і-го етапу в загальній трудомісткості розробки ПП

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ (L_1)	0,14	0,13	0,13
ТП (L_2)	0,15	0,16	0,12
РП (L_3)	0,58	0,59	0,63

Для нашого варіанта виділено сірим кольором.

Таблиця 2.3. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	Ступінь новизни	Значення $К_н$
А	Принципово нові ПП	1,75 – 1,2
Б	ПП – розвиток параметричного ряду	1,0 – 0,8
В	ПП, що має аналог	0,7

Для зазначеного варіанта виділено у таблиці кольором.

Таблиця 2.4. Значення коефіцієнта використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПП типовими програмами, %	Значення $К_т$
60 і вище	0,65
40-60	0,75
20-40	0,85
до 20	0,95

Нижче наведено розширений опис розрахунку трудомісткості окремих етапів розробки програмного забезпечення для варіанту, а також узагальнення

отриманих даних у вигляді таблиці (Таблиця 2.5).

1. Трудомісткість технічного завдання (ТТЗ):
   Для обчислення трудомісткості цього етапу використовується формула
   $$T_{\text{тз}} = T_a \times L_1 \times K_n,$$
   де:
   - (T_a = 183,2) люд/годин – скоригована базова норма часу,
   - (L_1 = 0,12) – коефіцієнт, що відображає відносну складність або обсяг даного етапу,
   - (K_n = 0,7) – поправочний коефіцієнт, який враховує специфіку умов розробки.

   Обчислення дає:
   $$183,2 \times 0,12 \times 0,7 = 15,39 \text{ люд/годин}.$$
   До цього етапу також відведено 2 сторінки документації.
2. Трудомісткість розробки технічного проекту (ТП): Застосовується подібний метод із використанням параметрів для цього етапу:
   $$T_{\text{тп}} = T_a \times L_2 \times K_n,$$
   де (L_2 = 0,11) відображає обсяг та специфіку робіт на етапі технічного проекту, а коефіцієнт (K_n = 0,7) залишається тим самим.

   Обчислення:
   $$183,2 \times 0,11 \times 0,7 = 17,42 \text{ люд/годин}.$$
   За цей етап виділено 28 сторінок документації.
3. Трудомісткість розробки робочого проекту (РП):
   Оскільки цей етап є найбільш ресурсозатратним, у формулі враховано додатковий коефіцієнт, що описує специфіку робіт:
   $$T_{\text{рп}} = T_a \times L_3 \times K_n \times K_t,$$
   де:
   - (L_3 = 0,61) – коефіцієнт, що характеризує обсяг робіт на етапі робочого проекту,
   - (K_n = 0,7) – базовий поправочний коефіцієнт,
   - (K_t = 0,7) – додатковий коефіцієнт, який враховує зростання складності.

Обчислення:

$$183{,}2 \times 0{,}61 \times 0{,}7 \times 0{,}7 = 54{,}76 \text{ люд/годин.}$$

Для цього етапу виділено 37 сторінок документації.

4. Пояснювальна записка: Окрім основних етапів розробки, у подальших розрахунках враховано також обсяг пояснювальної записки, який становить 30 сторінок. Цей показник враховується при підготовці фінальної документації та демонстрації результатів проекту.

Таблиця 2.5. Розрахунок трудомісткості за етапами

Етап розробки	Формула	Обчислення	Трудомісткість (люд/годин)	Обсяг документації (стор.)
Технічне завдання (ТТЗ)	$T_{тз} = T_a \times L_1 \times K_n$	$183{,}2 \times 0{,}12 \times 0{,}7 = 15{,}39$	15,39	2
Технічний проект (ТП)	$T_{тп} = T_a \times L_2 \times K_n$	$183{,}2 \times 0{,}11 \times 0{,}7 = 17{,}42$	17,42	28
Робочий проект (РП)	$T_{рп} = T_a \times L_3 \times K_n \times K_t$	$183{,}2 \times 0{,}61 \times 0{,}7 \times 0{,}7 = 54{,}76$	54,76	37
Пояснювальна записка	–	–	–	30

2.3 Розрахунок ціни програмного продукту

1. Розрахунок основної заробітної плати виконавців

Основну заробітну плату виконавців обчислено за окремими видами робіт із врахуванням трудомісткості кожного етапу та погодинної тарифної ставки. Це дозволяє точно визначити прямі витрати на оплату праці, що безпосередньо впливають на собівартість розробки. Дані наведено в таблиці 2.6:

Таблиця 2.6. Розрахунок основної заробітної плати виконавців

Найменування робіт	Трудомісткість робіт (години)	Погодинна тарифна ставка (грн)	Розрахунок (грн)
1. Розробка ПП	149,11	39,26	5384,36
2. Контроль керівника	67,8	39,26	2583,18
3. Нормоконтроль	14,55	39,26	554,36
Усього	—	—	8521,90

Сума основної заробітної плати становить 8521,90 грн, що є базовою

статтею витрат на оплату праці розробників і контролерів проекту.

2. Розрахунок матеріальних витрат

Матеріальні витрати враховують необхідні ресурси для оформлення документації та інших матеріальних потреб проекту. У нашому випадку основним матеріальним ресурсом є папір, а також враховано транспортно–заготівельні витрати, що відповідають 10% від вартості паперу. Дані наведено в таблиці 2.7:

Таблиця 2.7. Розрахунок матеріальних витрат

Найменування матеріальних витрат	Тип, модель	Кількість	Ціна одиниці (грн)	Вартість (грн)
Папір	Лист А4	50	2,05	210,0
Транспортно–заготівельні витрати	—	—	—	21,0 (10% від 210,0)
Усього	—	—	—	231,0

Таким чином, загальна сума матеріальних витрат дорівнює 231,0 грн. Це дозволяє забезпечити повне покриття витрат, пов’язаних із закупівлею та транспортуванням матеріалів, необхідних для оформлення документації по проекту.

3. Формування калькуляції планової собівартості

Для складення повної собівартості розробки програмного продукту об'єднуються всі розраховані статті витрат. Це дозволяє отримати комплексну картину витрат і визначити оптову (кошторисну) вартість, до якої додається прибуткова складова. Дані представлені в таблиці 2.8:

Таблиця 2.8. Формування калькуляції планової собівартості

Стаття витрат	Значення (грн)	Формула розрахунку
1. Матеріали	231,0	V_m (див. табл. 2.7)
2. Основна заробітна плата	8521,90	$З_о$ (див. табл. 2.6)
3. Додаткова заробітна плата	852,19	8521,90 × 0,14 (14% від основної заробітної плати)
4. Відрахування до єдиного фонду соціального внеску	2062,30	0,22 × (8521,90 + 852,19)
5. Накладні витрати	3408,76	0,4 × 8521,90
Повна собівартість (Спов)	15076,15	231,0 + 8521,90 + 852,19 + 2062,30 + 3408,76

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

3.1 Вступ

Проблематика створення безпечних і нешкідливих умов праці існує так само давно, як і людство. Проте в сучасних умовах вона набуває особливої актуальності, адже вартість кожної аварії значно зростає. Згідно зі статтею 3 Конституції України, людина та її здоров'я визнаються найціннішими надбаннями держави.

Покращення умов і охорона праці стали одним із ключових напрямків підвищення як матеріального, так і культурного рівня життя народу, що сприяє підвищенню якості і продуктивності праці, покращенню соціально-економічних показників виробництва та зниженню витрат, пов'язаних із травматизмом, професійними захворюваннями і аварійними ситуаціями.

Держава виступає гарантом створення безпечних та нешкідливих умов праці для працівників підприємств, установ і організацій усіх форм власності.

3.2 Аналіз умов праці і забезпечення безпеки при виконанні основних видів робіт на об'єкті

Під час роботи на виробництві людину може впливати як один окремий, так і комплекс небезпечних та шкідливих виробничих чинників. Оскільки тема дипломного проєкту пов'язана з розробкою програмного забезпечення, зосередимо увагу саме на негативних факторах, які впливають на діяльність програміста. Виявлення та аналіз шкідливих і небезпечних виробничих факторів слід починати з перевірки дотримання вимог, встановлених санітарними правилами та нормами для виробничих приміщень і робочих місць.

3.3 Розробка заходів з охорони праці

Людина, що працює, проводить на виробничому майданчику значну частину свого життя. Тому для її нормальної життєдіяльності у виробничих умовах необхідно створити санітарні умови, які сприятимуть плідній роботі без перевтоми та дозволять зберегти здоров'я. Для цього енергетичні витрати під час роботи мають бути компенсовані відпочинком та сприятливими умовами

навколишнього середовища. Такі умови забезпечуються організацією для працівника зручного робочого місця із чистим повітрям, відповідно нормованим рівнем освітлення як у приміщенні, так і на робочому місці, а також ефективним захистом від шуму, вібрацій, впливу шкідливих речовин та випромінювання.

3.3.1 Виробничі приміщення

Для приміщень, призначених для роботи з ВДТ, доцільно обрати розташування вікон із орієнтацією на північ або північний схід. На вікнах мають бути встановлені регульовані жалюзі або штори, які забезпечують можливість їх повного закриття. У приміщеннях з ВДТ слід передбачити наявність побутових приміщень для відпочинку, психологічного розвантаження тощо. Площа одного робочого місця для дорослих операторів не повинна бути меншою за 6 кв.м., а об’єм — не менше 20 куб.м.

При кольоровому оформленні виробничих та допоміжних приміщень потрібно враховувати орієнтацію їхніх вікон щодо сторін світу та використовувати гармонійне поєднання кольорів. Стелі у всіх приміщеннях мають бути виконані в білому кольорі, а стіни — пофарбовані матовою фарбою.

Об’ємно-планувальні рішення для будівель та приміщень, де використовуються візуальні дисплейні термінали (ВДТ), повинні відповідати вимогам ДСанПІН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами ЕОМ».

3.3.2 Мікроклімат робочої зони працівників, вентиляція

Ключовими нормативними документами, в яких наведено норми мікроклімату, є санітарні норми та стандарти безпеки праці (ДСанПІН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами ЕОМ»).

Виробничий мікроклімат – найважливіший чинник, що впливає на продуктивність і безпеку праці – характеризується температурою, вологістю повітря, швидкістю його руху та інтенсивністю радіації, і має відповідати стандартам ГОСТ 12.1.005-88 і СНиП 2.04.05-86. Оптимальні параметри

становлять:

t° = 18 – 24 °C (температура);

w = 40–60% (вологість повітря);

v = 0,1–0,2 м/с (швидкість руху).

Для підтримання нормального складу повітря, який відповідає гігієнічним вимогам, і для видалення з нього шкідливих газів, пару і пилу використовується вентиляція. У дипломному проекті передбачено встановлення припливно-витяжної системи вентиляції, а також можливе застосування кондиціонерів.

3.3.3 Освітлення робочого місця, шум, вібрація

Для освітлення приміщення, де працює програміст, застосовується комбіноване освітлення, тобто поєднання природного та штучного світлового потоку. Для загального освітлення робочої зони програміста використовують газорозрядні лампи типу ЛД.

Нормативними документами встановлено необхідний рівень освітлення робочого місця — ЕН = 300 лк (для робіт високої точності, коли найменший розмір об'єкта розрізнення складає 0,3–0,5 мм).

У робочих приміщеннях основним джерелом акустичних шумів є шуми ПЕОМ. Крім того, ЕОМ виступають джерелами шумів електромагнітного походження (через коливання елементів).

Для усунення або зменшення негативного впливу шуму доцільно ізолювати робочі приміщення, розташовуючи їх у тих частинах будинку, які найбільш віддалені від міського шуму — зазвичай у глибині будівлі зі вікнами, що виходять у двір. Необхідно також перевіряти герметичність корпусів комп’ютерів і своєчасно замінювати вентилятори охолодження.

3.3.4 Організація робочого місця користувача ПК

Обладнання та організація робочого місця з ВДТ мають гарантувати, що конструкції всіх елементів робочої зони та їхнє взаємне розміщення відповідають ергономічним вимогам з урахуванням характеру та специфіки трудової діяльності (ГОСТ 12.2.032-78, ГОСТ 22.269-76, ГОСТ 21.889-76).

Конструкція робочого місця та просторове розташування всіх його складових (сидіння, органів керування, засобів відображення інформації) має відповідати антропометричним, фізіологічним і психологічним нормам, а також особливостям характеру роботи. Конструкція меблів повинна забезпечувати можливість індивідуального регулювання відповідно до зросту працюючих для підтримання комфортної пози. Робочий стіл необхідно пофарбувати матовою фарбою. Дисплей встановлюють таким чином, щоб його верхній край знаходився на рівні очей на відстані приблизно 70 см, що відповідає припустимим межам від 60 до 90 см, а частота його мерехтіння складає fмер = 100 Гц, що задовольняє умову fмер > 70 Гц.

Розміщення робочого місця перпендикулярно до віконних прорізів здійснюють з метою усунення прямої та відбитої мерехтливості екрана, що виникає від вікон і приладів штучного освітлення, зокрема ламп накалювання. Обладнання та організація робочого місця з ВДТ мають забезпечувати відповідність конструктивних рішень усіх елементів робочої зони та їх взаємного розташування ергономічним вимогам з урахуванням специфіки і особливостей трудової діяльності (ГОСТ 12.2.032-78, ГОСТ 22.269-76, ГОСТ 21.889-76).

Для захисту людини від впливу ЕМ-опромінення використовують різні засоби та заходи – тимчасовий захисний режим, захист за допомогою відстані, екранізацію джерела випромінювання, екранування робочих місць, застосування засобів індивідуального захисту та відділення зон випромінювання.

3.4 Пожежна безпека

Пожежа – це процес неконтрольованого горіння, який розгортається поза спеціально облаштованим вогнищем і веде до значної матеріальної шкоди. Основними причинами виникнення пожеж та вибухів на підприємствах є порушення правил і норм пожежної безпеки, а також невиконання положень Закону «Про пожежну безпеку». За стан пожежної безпеки на підприємстві відповідають керівники, начальники цехів, майстри та інші посадові особи, які мають забезпечити своєчасне виявлення ризиків і впровадження необхідних заходів для їх усунення.

До можливих причин виникнення пожеж у приміщеннях належать:

1) коротке замикання проводки, яке може статися через зношеність або несправність електроустаткування;

2) використання побутових електрорадіоприладів, якщо їх експлуатація не відповідає встановленим стандартам безпеки;

3) недотримання умов протипожежної безпеки, що унеможливлює своєчасне реагування на початкову стадію загоряння.

Для гасіння пожеж на робочому місці застосовують вуглекислотні та порошкові вогнегасники. Вуглекислотні вогнегасники, які випускаються як ручні (наприклад, ВВК-5), ефективні для гасіння електричних і рідинних пожеж, тоді як порошкові вогнегасники (типи ВП-2, ВП-5, ВП-10 тощо) здатні ліквідувати пожежі, викликані горінням різноманітних матеріалів. Наявність первинних засобів пожежогасіння, їх кількість та їх правильне розміщення мають відповідати вимогам ГОСТ 12.4.009-75 і ISO3941-77, що гарантує оперативність і ефективність реагування у випадку надзвичайної ситуації.

У приміщенні повинні бути реалізовані всі заходи пожежної безпеки згідно з вимогами НАПБ А.0.001-95 «Правила пожежної безпеки в Україні». Це включає не лише технічне оснащення, але й організаційні заходи, зокрема розробку чіткого плану евакуації на випадок пожежі. Час евакуації відповідає вимогам СНиП 2.01.02-85, забезпечуючи швидке виведення персоналу, а максимальна відстань розташування робочих місць від евакуаційних виходів відповідає вимогам СНиП 2.09.02-85, що мінімізує ризики для життя і здоров’я людей у разі надзвичайної ситуації.

ВИСНОВКИ

У процесі виконання дипломного проекту було розроблено набір віджетів для візуалізації числових даних у вигляді діаграм, який відповідає поставленим цілям і завданням. Основними технологіями реалізації продукту виступили JavaScript та Vue.js, що дозволило досягти високої інтерактивності й гнучкості у роботі з різними типами даних. Найбільший виклик під час розробки полягав у забезпеченні легкої інтеграції додатку з іншими програмними рішеннями та мінімізації розміру кінцевого продукту.

Розроблений набір віджетів надає можливості візуалізації статистичних даних за допомогою широкого спектру діаграм, серед яких лінійчаті, стовпчасті, кругові, діаграми розкиду та інші. Застосунок дозволяє користувачам конфігурувати діаграми через зручний користувацький інтерфейс, змінювати їх вигляд і параметри без необхідності прямого втручання в код. Важливою особливістю є можливість перевикористання діаграм – збережені конфігурації можуть бути використані в інших частинах додатку або навіть в інших проєктах. Також, користувач може повноцінно керувати станом віджетів, що відповідають за візуалізацію даних: додавати, видаляти, редагувати та змінювати їхню структуру. Гнучка архітектура застосунку забезпечує його легке розширення, що дозволяє в майбутньому додавати нові типи діаграм та інші функції.

Розроблений набір віджетів успішно вирішує задачу зручної візуалізації різних структур даних, забезпечуючи можливість як програмної, так і користувацької конфігурації. Особливо важливо відзначити його здатність інтегруватися з іншими програмами та сервісами, що значно розширює потенціал для бізнесу, освіти та наукових досліджень. Крім того, застосунок може бути використаний на локальних машинах, серверах або в хмарних середовищах, що забезпечує його універсальність. Щодо можливості подальшої модернізації, існують кілька напрямів для вдосконалення, серед яких розширення набору діаграм, інтеграція нових типів, а також оптимізація продуктивності для роботи з великими обсягами даних. Можливе розширення функціоналу аналітики через додавання функцій обробки та аналізу даних, виявлення аномалій або трендів.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Коваленко А. В. Програмування на JavaScript: сучасні підходи до розробки веб-додатків – Київ: Основа, 2021. – 350 с.
2. Гончарук М. П. Фреймворк Vue.js: створення інтерактивних веб-застосунків – Львів: Видавництво "Львівська політехніка", 2020. – 280 с.
3. Баранюк С. О. Візуалізація даних – Харків: Ранок, 2022. – 310 с.
4. Іваненко В. О. Алгоритми та структури даних для веб-розробки / В. О. Іваненко. – Київ: Видавництво Київського університету, 2019. – 290 с.
5. Степаненко Д. В. Технології сучасної веб-розробки: JavaScript, CSS та HTML5 – Одеса: Наукова думка, 2021. – 370 с.
6. Журбенко С. І. Розробка веб-додатків з використанням JavaScript та Vue.js – Дніпро: Видавництво ДНУ, 2020. – 320 с.
7. Кравченко О. М. Методи візуалізації даних для аналітичних систем – Київ: Видавництво НТУУ "КПІ", 2022. – 255 с.
8. Павлюк А. С. Інструменти для обробки та візуалізації великих обсягів даних – Харків: ХНУРЕ, 2021. – 300 с.
9. Вознюк М. В. Тестування веб-додатків з використанням Jest – Київ: Видавництво НАН України, 2020. – 275 с.
10. Олійник С. Г. Інтеграція веб-додатків з використанням сучасних хмарних технологій – Львів: Видавництво ЛНУ, 2019. – 240 с.
11. Іванов І. Д. JavaScript для початківців: основи та практика – Київ: Видавництво Академії наук України, 2020. – 280 с.
12. Черненко П. В. Контейнеризація додатків за допомогою Docker та Kubernetes – Запоріжжя: Видавництво "Прометей", 2021. – 310 с.
13. Шевченко О. Л. Аналіз даних та візуалізація у веб-додатках – Київ: Видавництво КНТЕУ, 2022. – 295 с.
14. Петров С. В. Фреймворки JavaScript для швидкої розробки веб-додатків – Вінниця: ВНТУ, 2019. – 270 с.
15. Мельник І. О. Інтеграція інтерфейсів користувача у веб-застосунках – Полтава: Видавництво "Полтавський університет", 2021. – 245 с.

ДОДАТОК А.

Лістинг коду JSON-об'єктів. Шаблони компонентів на HTML. Скрипти компонентів на javascript

```
{
	"name": "widget",
	"version": "0.1.1",
	"description": "Платформа для відображення віджетів та статистики на основі Vuejs",
	"private": true,
	"scripts": {
		"postinstall": "gulp build", // Скрипт, що запускається після встановлення залежностей
		"start": "node --harmony ./node_modules/sails/bin/sails lift", // Запуск проєкту з Node.js
		"debug": "node --harmony debug ./node_modules/sails/bin/sails lift", // режим налагодження
		"test": "gulp test" // Запуск тестів через Gulp
	},
	"devDependencies": {
		"sails-hook-autoreload": "0.11.5" // перезавантаження сервера при зміні файлів
	},
	"dependencies": {
		"bcryptjs": "^2.4.3", // Пакет для хешування паролів
		"connect-mongo": "^2.0.3", // Підключення MongoDB для сесій
		"copy-object": "^1.0.0", // Модуль для копіювання об'єктів
		"del": "^3.0.0", // Пакет для видалення файлів і папок
		"djvue-client-modules": "file:./_modules", // Локальні модулі DJVue
		"express": "^4.16.4", // Фреймворк для створення серверів на Node.js
		"gravatar": "^1.8.0", // Підтримка аватарок Gravatar
		"gulp": "3.9.0", // Gulp — інструмент для автоматизації завдань
		"gulp-add-src": "0.2.0", // Плагін Gulp для додавання джерел
		"gulp-babel": "4.0.1", // Підтримка трансляції ES6 у ES5 через Babel
		"gulp-bower": "0.0.10", // Плагін для використання Bower з Gulp
		"gulp-cached": "1.0.4", // Підтримка кешування файлів у Gulp
		"gulp-changed": "1.2.1", // Обробка лише змінених файлів у Gulp
		"gulp-concat": "2.5.2", // Конкатенація файлів у Gulp
		"gulp-extend": "0.2.0", // Підтримка розширення файлів через Gulp
		"gulp-filter": "2.0.2", // Фільтрація файлів за допомогою Gulp
		"gulp-if": "1.2.5", // Умовне виконання завдань у Gulp
		"gulp-inline-source": "1.2.0", // Інлайнінг джерел (CSS, JS) у HTML
		"gulp-load-plugins": "0.10.0", // Автоматичне завантаження плагінів Gulp
		"gulp-markdown": "1.0.0", // Конвертація Markdown у HTML
		"gulp-minify-css": "1.1.0", // Мінімізація CSS-файлів
		"gulp-minify-html": "1.0.2", // Мінімізація HTML-файлів
		"gulp-modify": "0.1.1", // Підтримка модифікації файлів через Gulp
		"gulp-plumber": "1.0.0", // Обробка помилок у Gulp
		"gulp-rename": "1.2.2", // Перейменування файлів у Gulp
		"gulp-run": "1.6.7", // Запуск зовнішніх процесів через Gulp
		"gulp-shell": "0.4.1", // Виконання shell-команд через Gulp
		"gulp-size": "1.2.1", // Виведення розміру файлів у консолі під час роботи Gulp
		"gulp-sourcemaps": "1.5.2", // Генерація sourcemaps для налагодження
		"gulp-tap": "0.1.3", // Підтримка tap-файлів у Gulp
		"gulp-uglify": "1.2.0", // Мінімізація JavaScript-файлів
		"jspm": "0.15.4", // Менеджер JavaScript-пакетів JSPM
		"lodash-node": "^3.10.2", // Утиліти для роботи з об'єктами та масивами
		"mime": "^1.3.4", // Визначення типів MIME
		"newrelic": "1.18.4", // Інтеграція з New Relic для моніторингу
		"node-xlsx": "^0.6.0", // Робота з Excel-файлами у Node.js
		"oauth": "0.9.12", // Підтримка OAuth-автентифікації
		"object-hash": "0.5.0", // Генерація хешу для об'єктів
		"package-info": "^2.2.3", // Отримання інформації про пакет
		"passport": "0.2.1", // Підтримка автентифікації через Passport.js
		"passport-google-oauth": "0.2.0", // Авторизація через Google OAuth
		"passport-local": "1.0.0", // Локальна стратегія авторизації через Passport.js
		"path": "^0.12.7", // Модуль для роботи зі шляхами
		"rc": "1.0.1", // Читання файлів конфігурації
		"run-sequence": "1.1.0", // Запуск Gulp-завдань послідовно
```

```
                "sails": "~0.12.14", // Фреймворк Sails.js для Node.js
                "sails-hook-dev": "0.2.2", // Плагін для підтримки режиму розробки у Sails.js
                "sails-mongo": "^0.12.2", // Адаптер MongoDB для Sails.js
                "systemjs": "0.16.9", // Модульний завантажувач для JavaScript
                "traceur": "0.0.89", // Транспілятор Traceur для підтримки ES6
                "util": "^0.10.3", // Утиліти для роботи з об'єктами у Node.js
                "validator": "3.39.0", // Модуль для валідації даних
                "xlsjs": "^0.7.5", // Бібліотека для роботи з Excel-файлами
                "xlsx": "^0.8.0", // Ще одна бібліотека для роботи з Excel-файлами
                "xml2js": "^0.4.16" // Перетворення XML у JavaScript-об'єкти
        },
        "main": "app.js", // Головний файл додатку
        "engines": {
                "node": "8.12.x", // Версія Node.js
                "npm": "6.4.x" // Версія NPM
        }
}
<template>
   <v-card v-bind:class="{widget:!isProductionMode}" ma-1 flat style="background:transparent;">
      <!-- Компонент v-card використовує умову для класу в залежності від режиму розробки -->
      <v-toolbar dark card height="36px" :color="(!hasError)?'primary darken-1':'error darken-1'" v-
if="!isProductionMode">
         <!-- Панель інструментів, яка змінює колір в залежності від наявності помилки -->
         <v-tooltip top>
            <!-- Підказка з інформацією про віджет -->
            <v-avatar class="handle" size="32" tile slot="activator">
               <v-icon>{{config.icon}}</v-icon>
            </v-avatar>
            <span>{{config.type}}</span>
         </v-tooltip>
         <v-toolbar-title class="body-2 white--text">{{config.id}}"{{config.name}}"</v-toolbar-title>
         <!-- Назва віджету -->
         <v-spacer></v-spacer>
         <v-menu bottom left>
            <v-btn slot="activator" dark icon>
               <v-icon>more_vert</v-icon> <!-- Кнопка меню -->
            </v-btn>
            <v-list>
               <v-list-tile @click="collapsed = !collapsed">
                  <v-list-tile-title>{{(collapsed) ? "Expand" : "Collapse"}}</v-list-tile-title>
                  <!-- Плитка меню для розгортання або згортання віджету -->
               </v-list-tile>
               <v-list-tile @click.stop="configure()">
                  <v-list-tile-title>Edit options...</v-list-tile-title>
                  <!-- Плитка для редагування налаштувань віджету -->
               </v-list-tile>
               <v-list-tile @click="cloneWidget()">
                  <v-list-tile-title>Clone</v-list-tile-title>
                  <!-- Плитка для клонування віджету -->
               </v-list-tile>
               <v-list-tile @click="deleteWidget()">
                  <v-list-tile-title>Delete</v-list-tile-title>
                  <!-- Плитка для видалення віджету -->
               </v-list-tile>
            </v-list>
         </v-menu>
      </v-toolbar>
      <component v-if="config.type" style="width:100%" v-bind:is="config.type" ref="instance" :config="config"
@init="onInit"></component>
      <!-- Якщо тип віджету визначений, компонент відображається; якщо ні - повідомлення про помилку -->
      <h4 v-else class="error--text">Widget type not defined</h4>
   </v-card>
</template>
<script>
import components from "djvue/components/widgets/index.js"  // Імпорт компонентів для віджетів
import djvueMixin from "djvue/mixins/core/djvue.mixin.js"  // Імпорт основного міксину djvue
import widgetMixin from "djvue/mixins/core/widget.mixin.js"  // Імпорт міксину для віджетів
```

```
export default {
    mixins: [djvueMixin, widgetMixin],  // Використання міксинів
    name: "dj-widget",  // Назва компоненту
    props: ["type", "holder"],  // Вхідні властивості компоненту
    components,  // Компоненти, які можна використовувати в шаблоні
    data: () => {
        return {
            collapsed: false,  // Статус згортання віджету
            hasError: false  // Статус наявності помилки
        }
    },
    computed: {
        globalConfig() {
            console.log(JSON.stringify(this.app.currentPage.holders[this.holder], null, "\t"))
            return _.find(this.app.currentPage.holders[this.holder].widgets, (item) => item.id == this.config.id)
            // Знаходження конфігурації віджету за його ID
        }
    },
    methods: {
        configure() {
            this.$eventHub.emit("widget-reconfigure", this)  // Виклик події для повторної конфігурації віджету
        },
        cloneWidget() {
            this.$eventHub.emit("widget-clone", this)  // Виклик події для клонування віджету
        },
        deleteWidget() {
            this.$eventHub.emit("widget-delete", this)  // Виклик події для видалення віджету
        },
        onInit() {
            this._updateConfig();  // Оновлення конфігурації віджету
            this.onInitChild();  // Ініціалізація дочірніх компонентів
        }
    }
}
</script>
<!-- Підключення стилів для віджету та обробника -->
<style>
/* Додавання рамки для віджету */
.widget {
    border: 1px solid #bbb !important;
}
/* Встановлення курсора у вигляді вказівника для обробника */
.handle {
    cursor: pointer;
}
</style>
<!-- Шаблон з основним вмістом сторінки -->
<template>
    <div>
        <!-- Компонент заголовка сторінки -->
        <dj-holder name="AppHeader" type="skin"></dj-holder>
        <!-- Місце для відображення маршрутизованих компонентів -->
        <router-view></router-view>
        <!-- Компонент футера сторінки -->
        <dj-holder name="AppFooter" type="skin"></dj-holder>
    </div>
</template>
<script>
// Імпорт компонента holder
import holder from "djvue/components/core/holder.vue"
// Експорт основної конфігурації компонента
export default {
    components: {
        // Динамічний імпорт компонента dj-holder
        "dj-holder": () => import("djvue/components/core/holder.vue")
    }
```

```
}
</script>
<!-- Шаблон для відображення компонента в залежності від обраного макету -->
<template>
    <div>
        <!-- Динамічне завантаження компонента в залежності від layout -->
        <component :is="layout" :options="options"></component>
    </div>
</template>
<script>
// Імпорт списку доступних макетів
import layouts from "djvue/components/layouts/index.js"
// Імпорт міксину з функціональністю для компонента
import mixin from "djvue/mixins/core/djvue.mixin.js"
export default {
    // Підключення міксину для використання загальної функціональності
    mixins: [mixin],
    // Реєстрація компонентів макетів
    components: layouts,
    // Оголошення властивостей компонента
    props: {
        // Властивість типу макету, значення за замовчуванням - "layout-1-2"
        type: {
            default: "layout-1-2"
        },
        // Властивість для передачі додаткових налаштувань (не має значення за замовчуванням)
        "options": {}
    },
    // Оголошення локальних даних компонента
    data: () => ({
        // Початкове значення для layout
        layout: "empty"
    }),
    // Відслідковування змін маршруту
    watch: {
        // Спостереження за зміною маршруту
        '$route'(to, from) {
            // Оновлення поточної сторінки після зміни маршруту
            this.setCurrentPage(this.getPage(this.$route.params.page))
            this.layout = "empty"

            // Встановлення макету сторінки після оновлення
            this.$nextTick(() => {
                this.layout = this.getPage(this.$route.params.page).layout
            })
        }
    },
    // Оновлення заголовка сторінки після кожної зміни
    updated() {
        window.document.title =
            `${this.appName}${(this.getPage(this.$route.params.page).title ? "-" :
""")}${this.getPage(this.$route.params.page).title || ""}`
    },
    // Створення компонента
    created() {
        // Встановлення поточної сторінки при завантаженні компонента
        this.setCurrentPage(this.getPage(this.$route.params.page))
        this.layout = this.getPage(this.$route.params.page).layout
    }
}
</script>
<template>
    <!-- Основний контейнер для віджету, з умовними класами для режиму продакшн і прийняття віджетів
-->
    <div pa-2 mt-2 class="holder" v-bind:class="{producttion:isProductionMode, accepted:isAcceptWidget}">
        <!-- Закоментований код заголовка для віджету -->
        <!--
```

```
        <div class="holder-title">
            <h4 v-if="!isProductionMode"> Widget Holder: {{name}}</h4>
        </div>
        -->
            <!-- Розмітка для відображення елементів в колонку з обгорткою -->
        <v-layout column wrap>
            <!-- Компонент для перетягування віджетів -->
            <draggable class="list-group" element="div" v-model="widgets"
                    :options="dragOptions" :move="onMove"
                    @start="onStartDrag" @end="onEndDrag">
                <!-- Група перехідних елементів для анімацій при зміні списку віджетів -->
                <transition-group type="transition" name="holders" tag="div"
                            v-bind:class="{'empty-holder': isEmpty && !isProductionMode}">
                    <!-- Компонент для кожного віджету, з передачею конфігурації та даних -->
                    <dj-widget :config="widget" :holder="name"
                            v-for="widget in widgets" :key="widget.id"
                            class="list-group-item" @init="onInitChild"></dj-widget>
                </transition-group>
            </draggable>
        </v-layout>
        <!-- Панель для додавання нового віджету, що відображається в режимі не продакшн -->
        <v-layout align-center justify-end row fill-height>
            <v-btn icon small flat color="primary" class="ma-0"
                @click="insert()" v-if="!isProductionMode">
                <v-icon small class="primary--text">mdi-shape-square-plus</v-icon>
            </v-btn>
        </v-layout>
    </div>
</template>
<script>
<script>
import draggable from "modules/vue-draggable/vuedraggableES6.js"; // бібліотека для перетягування
елементів
import djvueMixin from "djvue/mixins/core/djvue.mixin.js" // Імпорт міксину для загальних функцій
import listenerMixin from "djvue/mixins/core/listener.mixin.js" // Імпорт міксину для обробки подій
import initiableMixin from "djvue/mixins/core/initiable.mixin.js" // Імпорт міксину для ініціалізації
import insertWidgetDialog from "djvue/components/core/dialogs/insertWidgetDialog.vue" // додавання віджету
// Реєстрація діалогового вікна на глобальному рівні
Vue.prototype.$dialog.component('insertWidgetDialog', insertWidgetDialog)
// Допоміжна функція для конвертації об'єкта в дерево
let toTree = (object) =>
_.keys(object).map(key => {
    return {
        name: (_.isObject(object[key])) ? key : `${key}: ${object[key]}`,
        children: (!_.isObject(object[key])) ? undefined : toTree(object[key])
    }
})
export default {
    // Використання кількох міксинів
    mixins: [djvueMixin, listenerMixin, initiableMixin],
    // Реєстрація компонентів
    components: {
        "dj-widget": () => import("./widget.vue"), // Динамічний імпорт компонента віджету
        draggable // Компонент для перетягування
    },
    // Оголошення даних компонента
    data() {
        return {
            isAcceptWidget: false // Стан, який вказує, чи приймається віджет
        }
    },
    // Оголошення властивостей компонента
    props: ["name", "type"],
    // Оголошення обчислюваних властивостей
    computed: {
        isEmpty() { return this.widgets.length == 0 }, // Перевірка на порожність списку віджетів
        dragOptions() {
```

```
        return {
            animation: 150, // Анімація при перетягуванні
            group: {
                name: "holders" // Група для перетягування
            },
            ghostClass: "ghost", // Клас для "привида" елемента
            dragClass: "drag", // Клас для елемента при перетягуванні
            handle: ".handle" // Клас для елемента, за допомогою якого можна перетягувати
        };
    },
    // Список віджетів з можливістю читання та запису
    widgets: {
        get() {
            if (this.type == "skin") return
            // Отримуємо віджети з поточної сторінки або з скіна
            return (this.app.currentPage.holders[this.name]) ?
                this.app.currentPage.holders[this.name].widgets : []
        },
        set(newValue) {
            // Оновлення вмісту хостера
            this.setHolderContent({
                page: (this.type == "skin") ? null : this.app.currentPage,
                holder: this,
                widgets: newValue
            })
        }
    }
},
// Оголошення методів
methods: {
    // Ініціалізація перед початком
    onBeforeInit() {
        if (this.type == "skin") {
            // Якщо тип "skin", формуємо список очікуваних віджетів
            this._waitList = this.app.skin.holders[this.name].widgets.map(item =>
                item.id)
        } else {
            // Для інших типів - перевіряємо поточну сторінку
            if(this.app.currentPage.holders[this.name]){
                this._waitList =
                    this.app.currentPage.holders[this.name].widgets.map(item => item.id)
            } else {
                this._waitList = []
            }
        }
        if (this._waitList.length == 0) {
            this.$emit("init", this) // Сповіщаємо про ініціалізацію компонента
        }
    },
    // Ініціалізація дітей після завантаження
    onChildsInitiated() {
        this.$emit("init", this.name) // Сповіщаємо про ініціалізацію за іменем хостера
    },
    // Додавання нового віджету
    insert() {
        this.$dialog.showAndWait(insertWidgetDialog) // Показуємо діалогове вікно для додавання віджету
        .then(initialWidgetConfig => {
            if (initialWidgetConfig) {
                initialWidgetConfig.id = this.$djvue.randomName() // Генеруємо нове ID для віджету
                this.widgets.push(this.$djvue.extend({}, initialWidgetConfig)) // Додаємо віджет
                this.setNeedSave(true) // Вказуємо, що потрібно зберегти зміни
            }
        })
    },
    // Перевірка, чи вже є такий віджет в хостері
    isHoldWidget(widget) {
        return !!_.find(this.widgets, w => widget.config && w.id == widget.config.id)
```

```
    },

    // Обробник початку перетягування
    onStartDrag() {
        this.emit("holder-accept", this) // Сповіщаємо про початок перетягування
        this.isDragging = true
    },
    // Обробник закінчення перетягування
    onEndDrag() {
        this.emit("holder-accept", null) // Сповіщаємо, що перетягування завершено
        this.isDragging = false
        this.setNeedSave(true) // Вказуємо, що потрібно зберегти зміни
    },
    // Обробник руху елементів при перетягуванні
    onMove({ relatedContext, draggedContext }) {
        this.emit("holder-accept", relatedContext.component.$parent) // Сповіщаємо про рух елементів
        return true
    }
},
// Знищення слухачів подій перед знищенням компонента
beforeDestroy() { this.off() },
// Створення компоненту
created() {
    this.on({
        event: "holder-accept",
        callback: (holder) => {
            this.isAcceptWidget = holder && (holder.name == this.name)
        }
    })
    // Обробка подій для клонування віджетів, видалення та оновлення конфігурацій
    this.on({
        event: "widget-clone", callback: (cloned) => {
            let widgetIndex = _.findIndex(this.widgets, w => w.id == cloned.config.id);
            let newWidget = this.$djvue.extend({}, this.widgets[widgetIndex])
            newWidget.id = this.$djvue.randomName();
            newWidget.name += "_clone_" + newWidget.id;
            this.widgets.splice(widgetIndex + 1, 0, newWidget)
        },
        rule: this.isHoldWidget
    })
    this.on({
        event: "widget-delete", callback: (deleted) => {
            if(deleted._delete()) {
                let widgetIndex = _.findIndex(this.widgets, w => w.id == deleted.config.id);
                if (widgetIndex > -1) this.widgets.splice(widgetIndex, 1)
                this.setNeedSave(true)
            }
        },
        rule: this.isHoldWidget
    })
    // Обробка події імпорту віджетів в холдер
    this.on({
        event: "holder-import-widgets", callback: (emitter, widgets) => {
            widgets = (_.isArray(widgets)) ? widgets : [widgets]
            this.widgets = this.widgets.concat(widgets)
            this.setNeedSave(true)
        },
        rule: this.isHoldWidget
    })
    // Обробка події оновлення конфігурації віджету
    this.on({
        event: "holder-update-widget-config", callback: (emitter, context) => {
            let widgetIndex = _.findIndex(this.widgets, w => w.id == context.widget.config.id);
            let newWidgets = JSON.parse(JSON.stringify(this.widgets));
            newWidgets[widgetIndex] = context.newConfig;
            this.widgets = newWidgets;
            this.setNeedSave(true)
```

```
            },
            rule: this.isHoldWidget
        })
    },
    // Слідкуємо за станом перетягування
    watch: {
        isDragging(newValue) {
            if (newValue) {
                this.delayedDragging = true;
                return;
            }
            this.$nextTick(() => {
                this.delayedDragging = false;
            });
        }
    }
}
</script>
<style scoped>
/* Стиль для елемента при перетягуванні */
.drag {
    opacity: 0; /* Зміна прозорості при перетягуванні */
}
/* Стиль для порожнього контейнера */
.empty-holder {
    border: 2px dashed #bbb !important; /* Додання пунктирного бордера */
    min-height: 100px !important; /* Мінімальна висота контейнера */
    background-color: #eee !important; /* Світло-сірий фон */
}
/* Стиль для анімації переміщення елементів */
.flip-list-move {
    transition: transform 0.5s; /* Плавний перехід */
}
/* Стиль для контейнера без руху */
.no-move {
    transition: transform 0s; /* Без анімації при відсутності руху */
}
/* Стиль для "привида" елемента при перетягуванні */
.ghost {
    opacity: 0 !important; /* Зміна прозорості до нуля */
}
/* Стиль для списку елементів */
.list-group {
    min-height: 20px; /* Мінімальна висота */
    width: 100%; /* Ширина на 100% */
}
/* Стиль для елементів списку */
.list-group-item {}
/* Стиль для іконок в елементах списку */
.list-group-item i {}
/* Стиль для контейнера хостера */
.container.holder.pa-2 {
    border: 2px solid #bbb !important; /* Оточення контейнера сірим бордером */
}
/* Стиль для контейнера в режимі "producttion" */
.container.holder.pa-2.producttion {
    margin-top: 1em !important; /* Зміщення контейнера вниз */
    border: none !important; /* Приховування бордера */
}
/* Стиль для заголовка контейнера */
.holder-title {
    background: #fafafa; /* Світлий фон */
    margin-top: -1.3em; /* Відступ верхній */
    color: #bbb; /* Світло-сірий колір тексту */
    width: fit-content; /* Автоматична ширина */
    padding: 0em 1em; /* Відступи */
}
```

```
/* Стиль для контейнера в режимі "accepted" */
.container.holder.pa-2.accepted {
    border-color: #00796B !important; /* Зміна кольору бордера */
}
/* Стиль для порожнього контейнера в режимі "accepted" */
.accepted .empty-holder {
    border-color: #00796B !important; /* Зміна кольору бордера для порожнього контейнера */
}
/* Стиль для заголовка в режимі "accepted" */
.accepted h4 {
    color: #00796B; /* Зміна кольору тексту заголовка */
}
</style>
import snippets from "./snippets.js"  // Імпортуємо файли з шаблонами для налаштування графіків.
export default {
  name: "line-chart-widget",  // Назва компонента.
  icon: "mdi-chart-line",  // Іконка компонента (матеріальний дизайн).
  // Метод для отримання початкових налаштувань компонента.
  getInitialConfig(snippet) {
    snippet = snippet || "Line";  // Якщо не передано значення snippet, використовуємо "Line" за
замовчуванням.
    let res = snippets[snippet] || snippets["Line"];  // Шукаємо в snippets конфігурацію для переданого типу
графіка.
    return res;  // Повертаємо конфігурацію.
  }
}
<template>
  <div>
    <!-- Контейнер для компонентів графіка -->
    <v-layout column justify-center>
      <!-- Заголовок графіка -->
      <h3 class="primary--text body-2 pa-0" style="text-align: center;">
        {{config.title}}  <!-- Виводимо заголовок графіка -->
      </h3>
      <!-- Примітка для графіка -->
      <p v-if="options" class="caption font-italic font-weight-light ma-0 pa-0" style="text-align: center;">
        {{config.note}}  <!-- Виводимо примітку, якщо вона є -->
      </p>
      <!-- Компонент для відображення графіка (echart) -->
      <echart v-if="options" :options="chartOptions" :height="options.widget.height"></echart>
      <!-- Виводимо графік, якщо налаштування (options) доступні -->
    </v-layout>
  </div>
</template>
import djvueMixin from "djvue/mixins/core/djvue.mixin.js";  // Імпортуємо міксин для основних функцій djvue.
import listenerMixin from "djvue/mixins/core/listener.mixin.js";  // Імпортуємо міксин для слухачів подій.
import ChartConfigDialog from "../widget-share/chart/chart-config.vue";  // для налаштування графіка.
import echart from "djvue/components/core/ext/echart.vue"  // Імпортуємо компонент для відображення
графіка.
Vue.prototype.$dialog.component('ChartConfigDialog', ChartConfigDialog)  // для налаштування графіка.
export default {
  name: "line-chart-widget",  // Назва компонента.
  icon: "mdi-chart-line",  // Іконка компонента (матеріальний дизайн).
  mixins: [djvueMixin, listenerMixin],  // Міксини, які додають додаткові функціональності до компонента.
  components: { echart },  // Оголошення локальних компонентів, використаних у шаблоні (наприклад,
echart).
  computed: {
    // Обчислювальне властивість для побудови налаштувань графіка.
    chartOptions() {
      if (!this.options) return;  // Якщо немає налаштувань, повертаємо порожнє значення.
      let res = JSON.parse(JSON.stringify(this.options));  // Копіюємо налаштування.
      // Якщо є дані для вибору елементів, фільтруємо серії графіка.
      if (this.config.dataSelectEmitters && this.config.dataSelectEmitters.length > 0) {
        let s = this.selection.filter(d => d.selected);  // Отримуємо вибрані елементи.
        res.series = this.series.filter(d => _.find(s, e => e.entity.id == d.selector));  // Фільтруємо серії по
елементам
      }
```

```
      res.legend.data = res.series.map(d => d.name);  // Оновлюємо легенду графіка.
      return res;  // Повертаємо оновлені налаштування.
    }
  },
  methods: {
    // Метод для оновлення даних графіка.
    onUpdate({ data, options }) {
      const tempOptions = JSON.parse(JSON.stringify(options));
      const tempData = JSON.parse(JSON.stringify(data));
      // tempOptions.legend.data = tempData.legend;  // (Закоментовано) Оновлення легенди.
      tempOptions.xAxis.data = tempData.xAxis;  // Оновлюємо осі графіка.
      // this.xAxis = tempData.xAxis; tempOptions.series = tempData.series; this.series = tempData.series;
this.options = tempOptions;  // (Закоментовано) Оновлення серій.
    },
    // Метод для пере налаштування компонента через діалогове вікно.
    onReconfigure(widgetConfig) {
      return this.$dialog.showAndWait(ChartConfigDialog, { config: widgetConfig });  // для зміни налаштувань
    },
    // Метод для вибору даних (наприклад, фільтрація або обрання елементів для графіка).
    onDataSelect(emitter, data) {
      this.selection = JSON.parse(JSON.stringify(data.selection));  // Оновлюємо вибір даних.
    }
  },
  props: ["config"],  // Оголошуємо властивість компонента config, яка передається при його використанні.
  mounted() {
    this.$emit("init");  // Генеруємо подію "init" після монтування компонента.
  },
  data: () => ({
    options: null,  // Налаштування графіка (початково порожнє).
    selection: [],  // Масив для зберігання вибраних даних.
    series: []  // Масив для зберігання серій даних графіка.
  })
}
</script>
export default {
"Line": {
  type: "line-chart-widget",
  name: "noname",
  icon: "mdi-chart-line",
  options: {
    widget: { visible: true, height: 300 },  // Віджет видимий і має висоту 300
    tooltip: {
      trigger: 'axis',
      axisPointer: {
        type: 'cross',
        label: { backgroundColor: '#6a7985' }  // Стиль підказки на осі
      }
    },
    legend: { data: ["1", "2"] },  // Легенда з двома елементами
    xAxis: { type: "category", data: ["2022", "2023", "2024", "2025"] },  // Ось X з категоріями
    yAxis: { type: "value" },  // Ось Y з числовими значеннями
    series: [
      { name: "1", type: "line", data: [43.3, 85.8, 93.7, 79.4], smooth: true },  // Перша лінія
      { name: "2", type: "line", data: [37.3, 28.8, 77.7, 100], smooth: true }  // Друга лінія
    ]
  },
  data: {
    source: "embedded",
    embedded: {
      "legend": ["embedded1", "embedded2"],  // Легенда для вбудованих даних
      "xAxis": ["2022", "2023", "2024", "2025"],  // Ось X
      "series": [
        { "name": "embedded1", "type": "line", "data": [43.3, 85.8, 93.7, 79.4], "smooth": true },
        { "name": "embedded2", "type": "line", "data": [37.3, 28.8, 77.7, 100], "smooth": true }
      ]
    }
  }
```

```
}
"Area": {
  type: "line-chart-widget",
  name: "noname",
  icon: "mdi-chart-line",
  options: {
    widget: { visible: true, height: 300 },
    legend: { data: ["1", "2"] },
    xAxis: { type: "category", data: ["2022", "2023", "2024", "2025"] },
    yAxis: { type: "value" },
    series: [
      { name: "1", type: "line", areaStyle: {}, data: [43.3, 85.8, 93.7, 79.4], smooth: true, areaStyle: { opacity: 0.2 } },
      { name: "2", type: "line", areaStyle: {}, data: [37.3, 28.8, 77.7, 100], smooth: true, areaStyle: { opacity: 0.2 } }
    ]
  },
  data: {
    source: "embedded",
    embedded: {
      "legend": ["embedded1", "embedded2"],
      "xAxis": ["2022", "2023", "2024", "2025"],
      "series": [
        { "name": "embedded1", "type": "line", "data": [43.3, 85.8, 93.7, 79.4], "smooth": true, areaStyle: { opacity: 0.2 }
},
        { "name": "embedded2", "type": "line", "data": [37.3, 28.8, 77.7, 100], "smooth": true, areaStyle: { opacity: 0.2 }
}
      ]
    }
  }
}
"Stacked Line": {
  type: "line-chart-widget",
  name: "noname",
  icon: "mdi-chart-line",
  options: {
    widget: { visible: true, height: 300 },
    legend: { data: ["1", "2"] },
    xAxis: { type: "category", data: ["2022", "2023", "2024", "2025"] },
    yAxis: { type: "value" },
    series: [
      { name: "1", type: "line", data: [43.3, 85.8, 93.7, 79.4], smooth: true, stack: "a" },
      { name: "2", type: "line", data: [37.3, 28.8, 77.7, 100], smooth: true, stack: "a" }
    ]
  },
  data: {
    source: "embedded",
    embedded: {
      "legend": ["embedded1", "embedded2"],
      "xAxis": ["2022", "2023", "2024", "2025"],
      "series": [
        { "name": "embedded1", "type": "line", "data": [43.3, 85.8, 93.7, 79.4], "smooth": true, "stack": "a" },
        { "name": "embedded2", "type": "line", "data": [37.3, 28.8, 77.7, 100], "smooth": true, "stack": "a" }
      ]
    }
  }
}
"Stacked Area": {
  type: "line-chart-widget",
  name: "noname",
  icon: "mdi-chart-line",
  options: {
    widget: { visible: true, height: 300 },
    legend: { data: ["1", "2"] },
    xAxis: { type: "category", data: ["2022", "2023", "2024", "2025"] },
    yAxis: { type: "value" },
    series: [
      { name: "1", type: "line", areaStyle: {}, data: [43.3, 85.8, 93.7, 79.4], smooth: true, stack: "a", areaStyle: {
opacity: 0.2 } },
```

```
    { name: "2", type: "line", areaStyle: {}, data: [37.3, 28.8, 77.7, 100], smooth: true, stack: "a", areaStyle: { opacity: 0.2 } }
   ]
  },
  data: {
   source: "embedded",
   embedded: {
    "legend": ["embedded1", "embedded2"],
    "xAxis": ["2022", "2023", "2024", "2025"],
    "series": [
     { "name": "embedded1", "type": "line", "data": [43.3, 85.8, 93.7, 79.4], "smooth": true, "stack": "a", areaStyle: { opacity: 0.2 } },
     { "name": "embedded2", "type": "line", "data": [37.3, 28.8, 77.7, 100], "smooth": true, "stack": "a", areaStyle: { opacity: 0.2 } }
    ]
   }
  }
}
}
```

ДОДАТОК Б. Слайди мультимедійної презентації

Критерій	Microsoft Word	Microsoft Excel	Tableau	Power BI	Google Data Studio	D3.js	Matplotlib
Тип програми	Десктопний застосунок	Десктопний застосунок	Веб/ десктопний застосунок	Веб/десктопний застосунок	Веб-застосунок	JavaScript бібліотека	Python бібліотека
Типи візуалізацій	Лінійні, стовпчасті, кругові діаграми	Лінійні, стовпчасті, гістограми, точкові	Лінійні, стовпчасті, картограми, аналітичні	Лінійні, стовпчасті, гістограми, точкові, KPI	Лінійні, стовпчасті, картограми	Всі типи (гнучка побудова за допомогою SVG)	Лінійні, гістограми, точкові, поверхневі
Інтерактивність	Відсутня	Обмежена інтерактивність	Повна інтерак-тивність	Повна інтерак-тивність	Інтерактивність за рахунок інтеграції з даними	Повна інтерактив-ність (реалізована через код)	Відсутня чи мінімальна (залежить від бібліотек)
Підтримка великих наборів даних	Обмежена	Обмежена	Висока	Висока	Середня	Висока	Висока
Підтримка візуалізації даних у реальному часі	Відсутня	Відсутня	Присутня	Присутня	Присутня	Можлива, але реалізація залежить від коду	Обмежена
Легкість у використанні	Проста у використанні	Проста для базових задач	Вимагає спеціальних знань	Вимагає базових навичок	Проста у використанні	Вимагає знань JavaScript	Вимагає знань Python
Ціна	Платний (Office Suite)	Платний (Office Suite)	Платний, але є безкоштовна пробна версія	Платний, але є безкоштовна версія	Безкоштовний	Безкоштовна бібліотека	Безкоштовна бібліотека
Гнучкість налаштувань	Мінімальна	Середня	Висока	Висока	Середня	Висока	Висока
Синхронізація з іншими джерелами даних	Відсутня	Є інтеграція з іншими файлами Excel	Підтримує інтеграцію з різними джерелами	Інтеграція з Excel, базами даних	Інтеграція з продуктами Google, базами даних	Залежить від реалізації користувача	Інтеграція через інші Python бібліотеки
Підтримка різних форматів даних	Таблиці, текст	Таблиці, CSV, XLSX	CSV, Excel, бази даних, API	CSV, Excel, бази даних, API	CSV, Google Sheets, API	JSON, CSV, API	CSV, Excel, JSON, бази даних

Порівняння програмних засобів для візуалізації числових даних

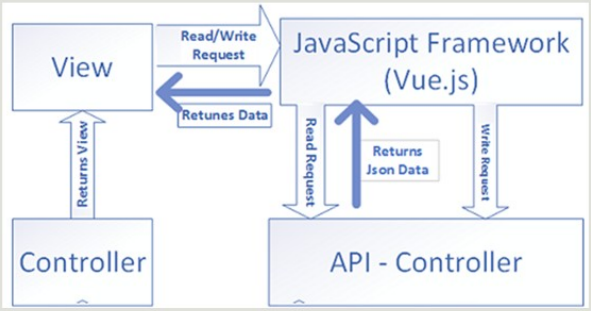
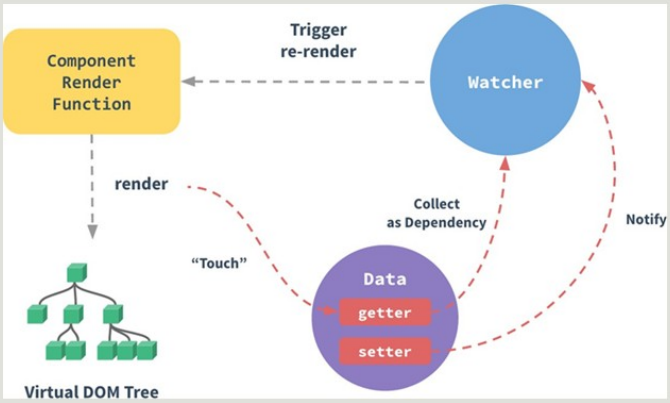
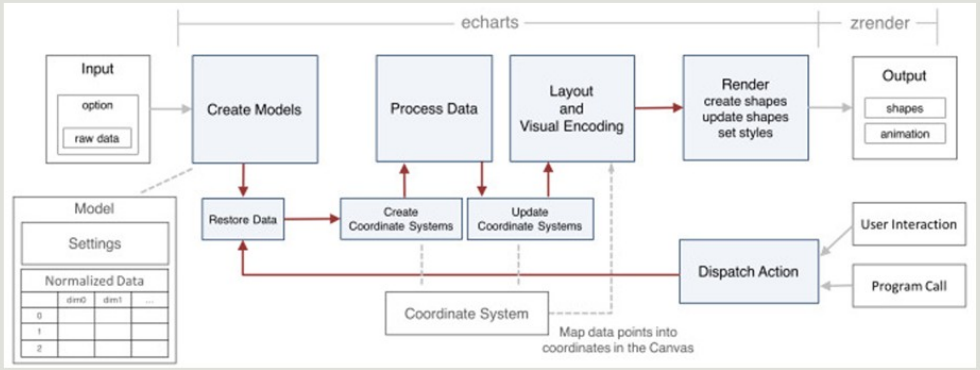


Схема MVC Vue.j



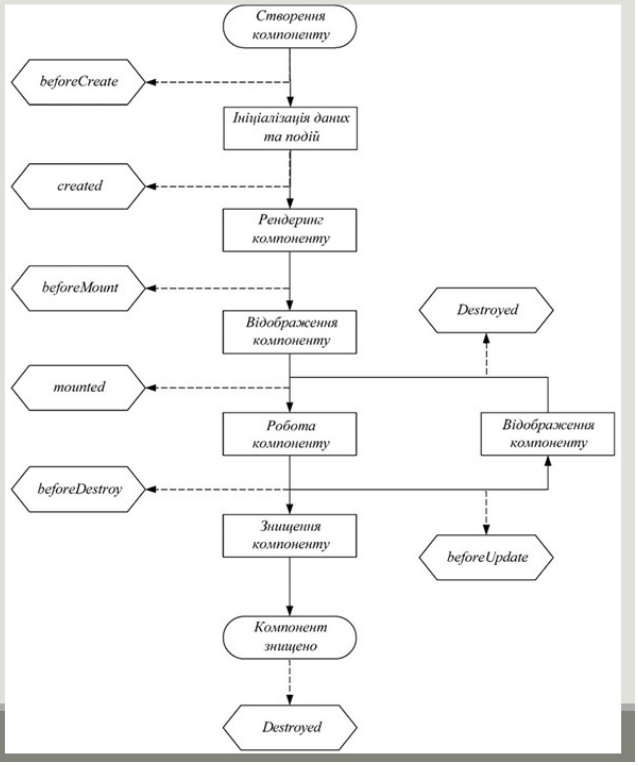
Реактивне зв’язування у Vue.js

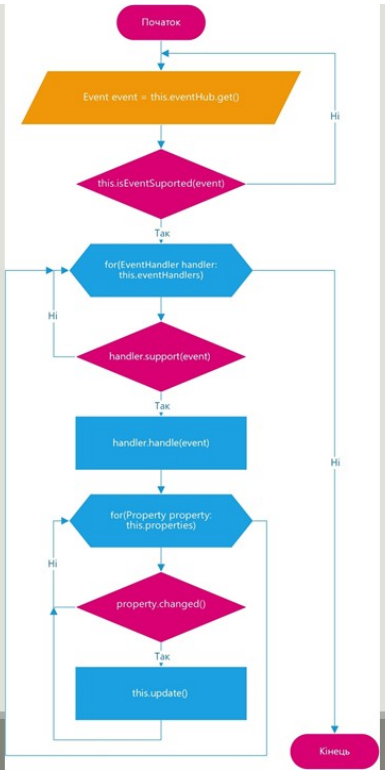


Архітектура бібліотеки ECharts

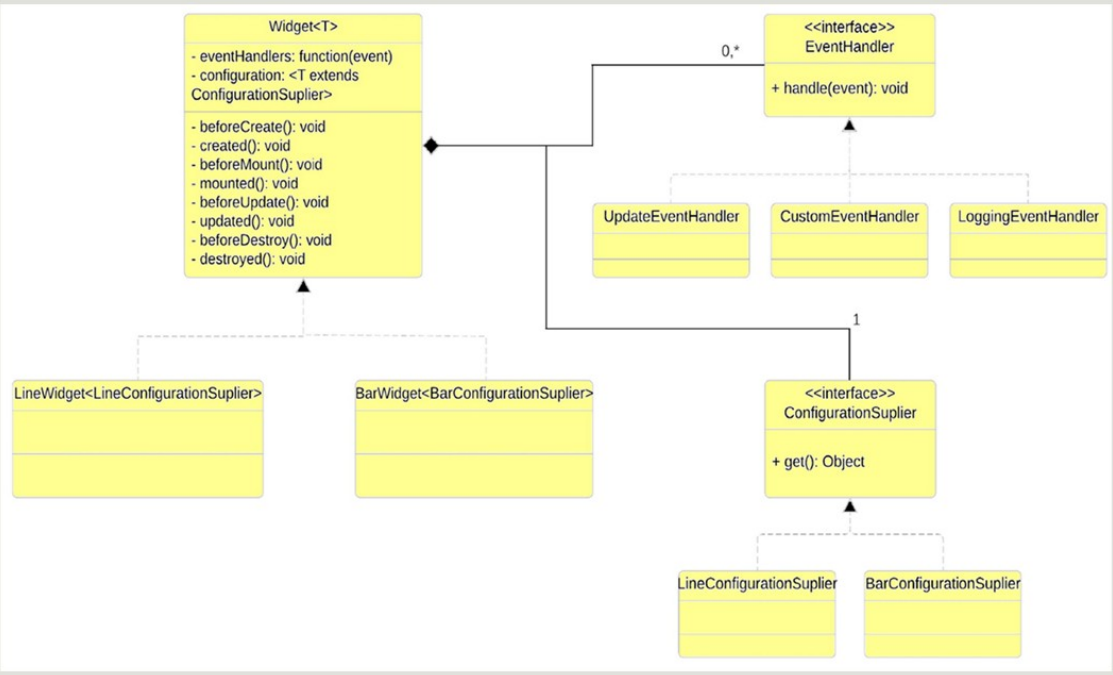
Приклад діаграми у бібліотеці Echarts

Схема життєвого циклу компоненту для візуалізації числових даних





Блок-схема алгоритму обробки подій компоненту через Vue.js



UML-діаграма класів для візуалізації числових даних

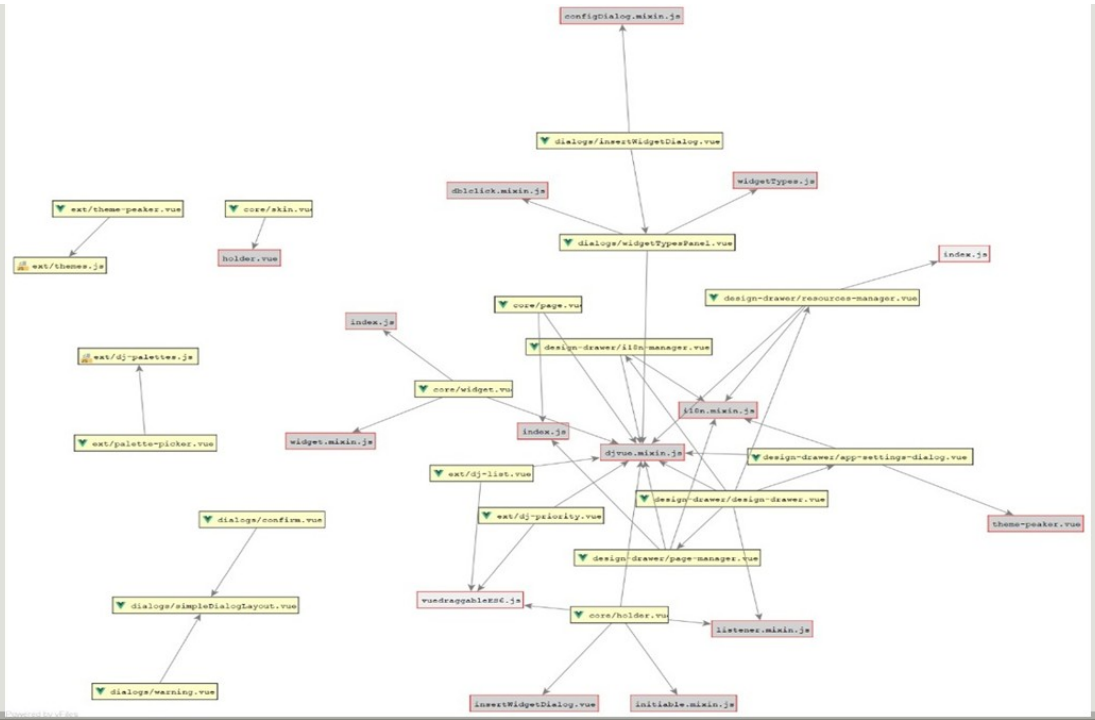


Схема зв’язку основних компонентів для віджетів

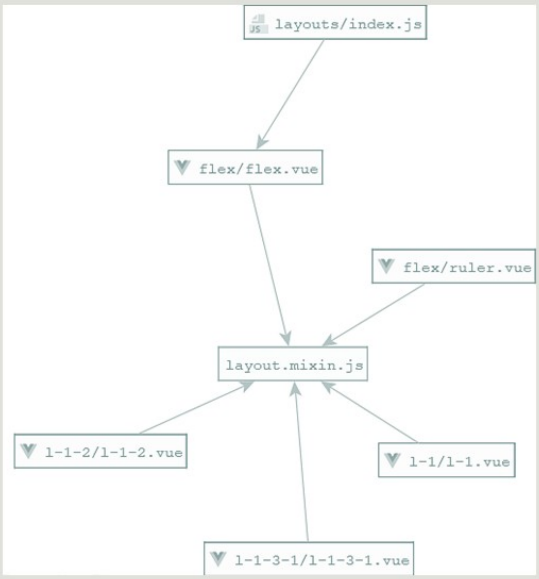
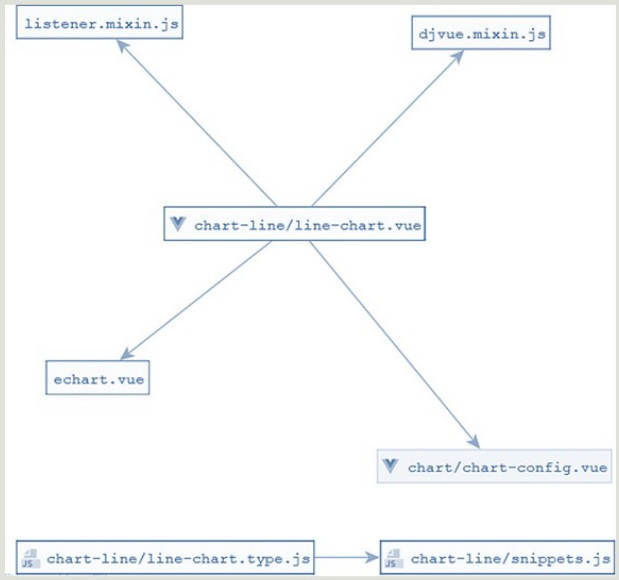


Схема зв’язку шаблонів розміщення



Модель віджету для лінійчатих діаграм

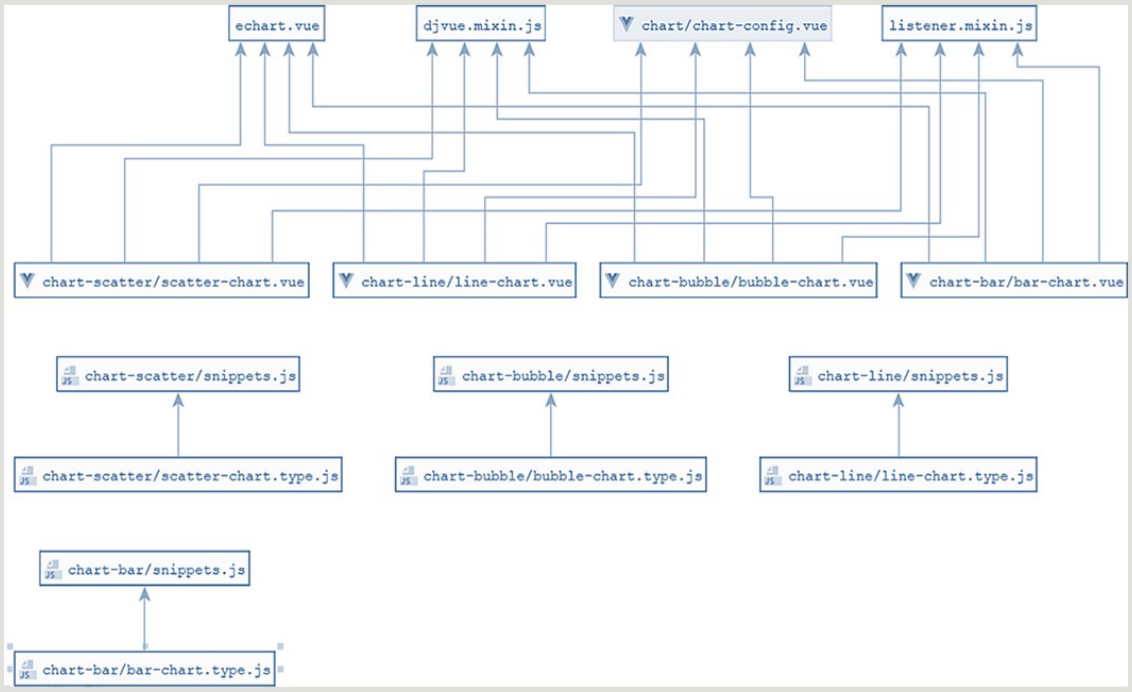
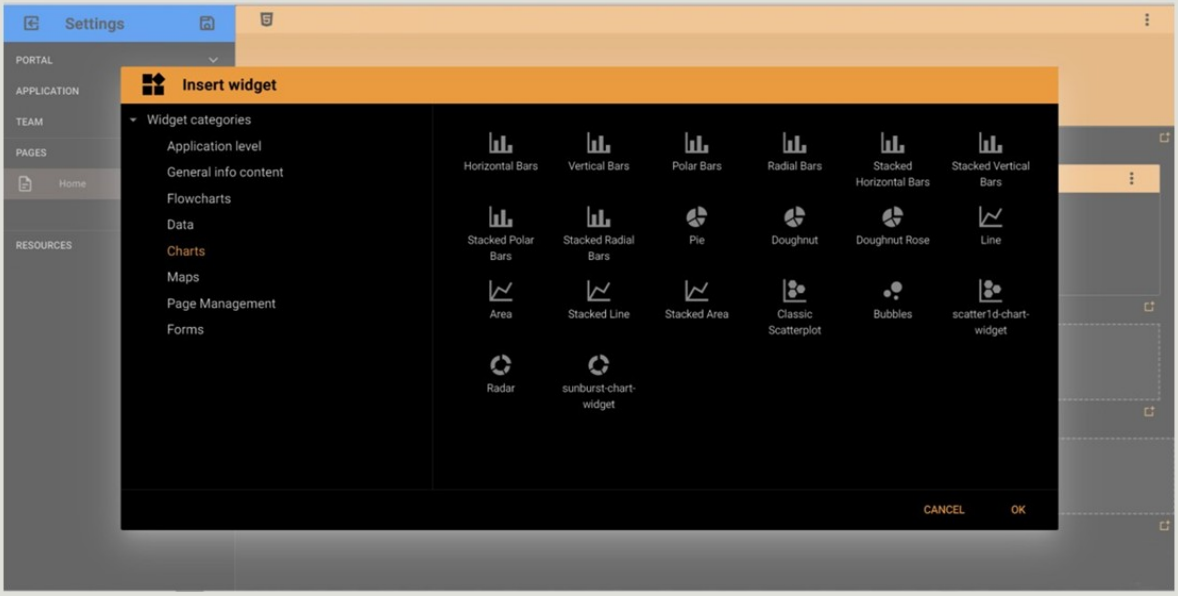


Схема зв’язку віджетів

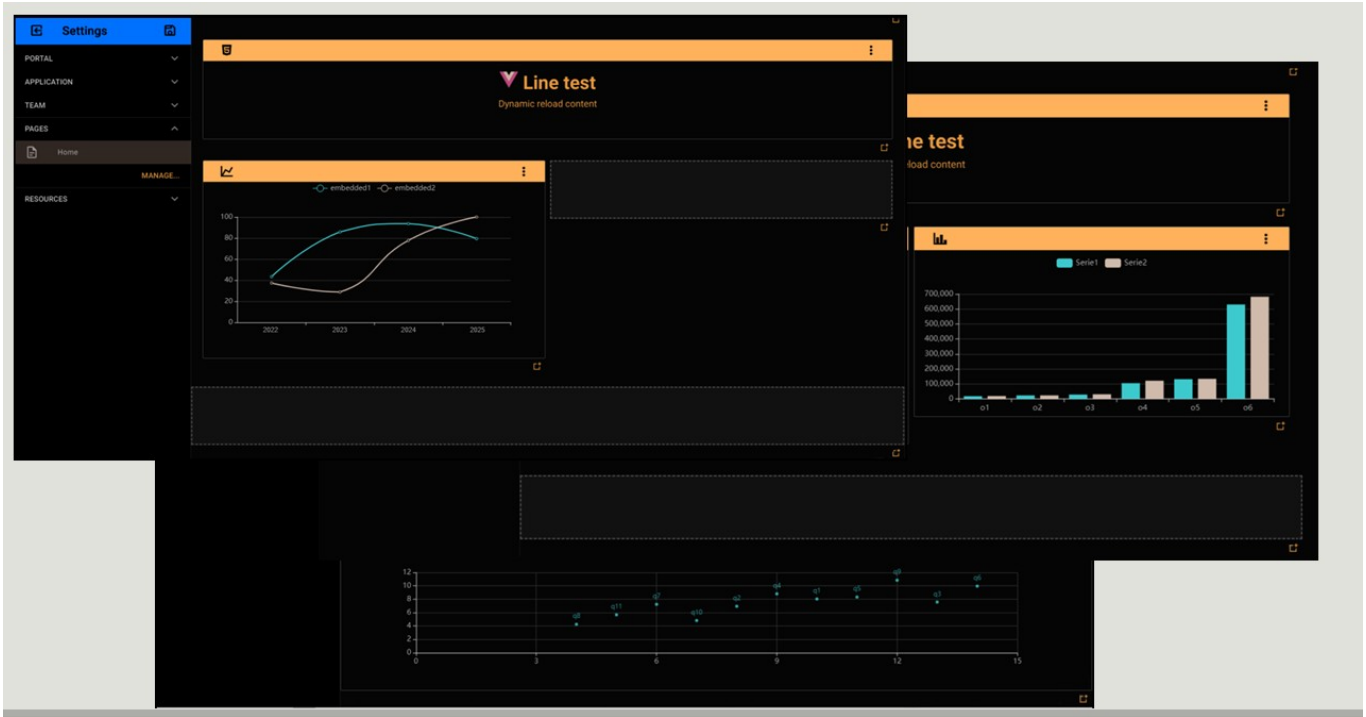
```
PASS  app/actions/__tests__/activity_list.spec.js
PASS  app/actions/__tests__/messaging.spec.js
PASS  app/actions/__tests__/authentication.spec.js
PASS  app/actions/__tests__/payment.spec.js
PASS  app/actions/__tests__/fellow.spec.js
PASS  app/actions/__tests__/workflow.spec.js
PASS  app/actions/__tests__/environment.spec.js
PASS  app/actions/__tests__/loading.spec.js
PASS  app/reducers/mixpanel/__tests__/mixpanel.spec.js
PASS  app/reducers/loading/__tests__/loading.spec.js
PASS  app/components/__tests__/messaging.spec.js
PASS  app/components/__tests__/workflow.spec.js
PASS  app/components/custom/carousel/__tests__/carousel.spec.js (9.183s)
PASS  app/components/custom/v2/task_container/__tests__/task_container.spec.js (9.251s)
PASS  app/components/custom/v2/navigation_bar/__tests__/home_nav_bar.spec.js (10.412s)
PASS  app/components/custom/v2/activity_list/__tests__/task_row.spec.js (10.017s)
PASS  app/components/custom/v2/navigation_bar/__tests__/back_nav_bar.spec.js (10.745s)
PASS  app/components/custom/v2/navigation_bar/__tests__/cancel_icon.spec.js (9.212s)
PASS  app/components/custom/headers/__tests__/edit_nav_bar.spec.js (9.852s)
PASS  app/components/custom/headers/__tests__/modal_close_icon.spec.js (9.623s)
PASS  app/components/custom/headers/__tests__/modal_header.spec.js (8.961s)
PASS  app/components/custom/headers/__tests__/save_icon.spec.js (9.784s)
PASS  app/components/authentication/__tests__/password.spec.js
PASS  app/components/authentication/__tests__/enter_login.spec.js (11.327s)

Test Suites: 22 passed, 22 total
Tests:       229 passed, 229 total
Snapshots:   229 passed, 229 total
Time:        14.562s
Ran all test suites.
```

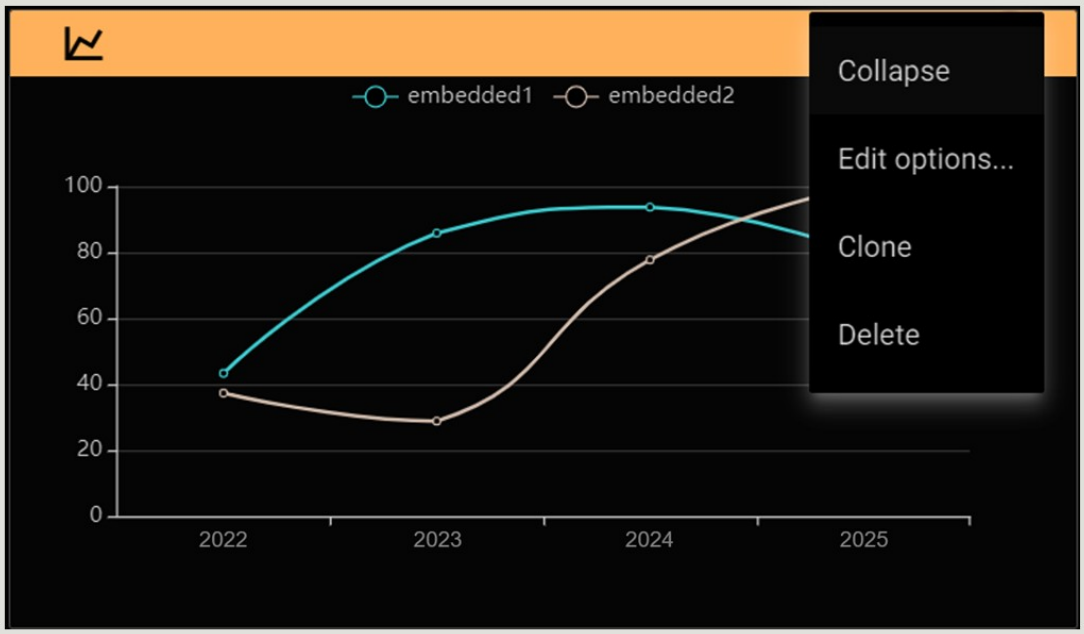
Тестовий звіт з використанням фреймворку Jest



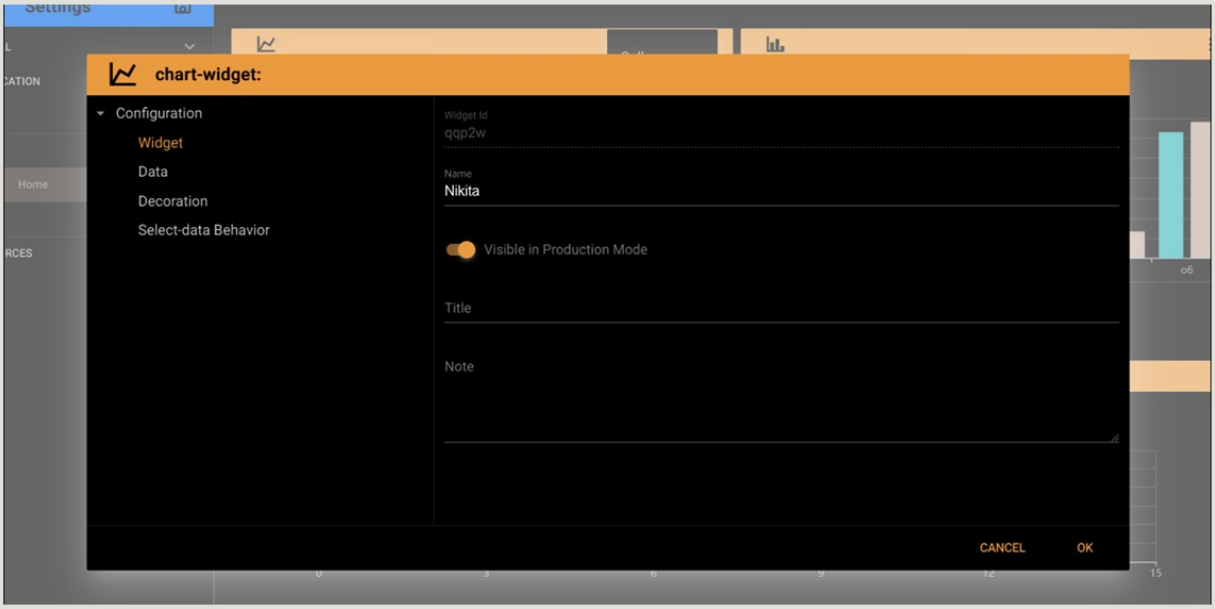
Вигляд головного екрану та вікна обирання віджетів



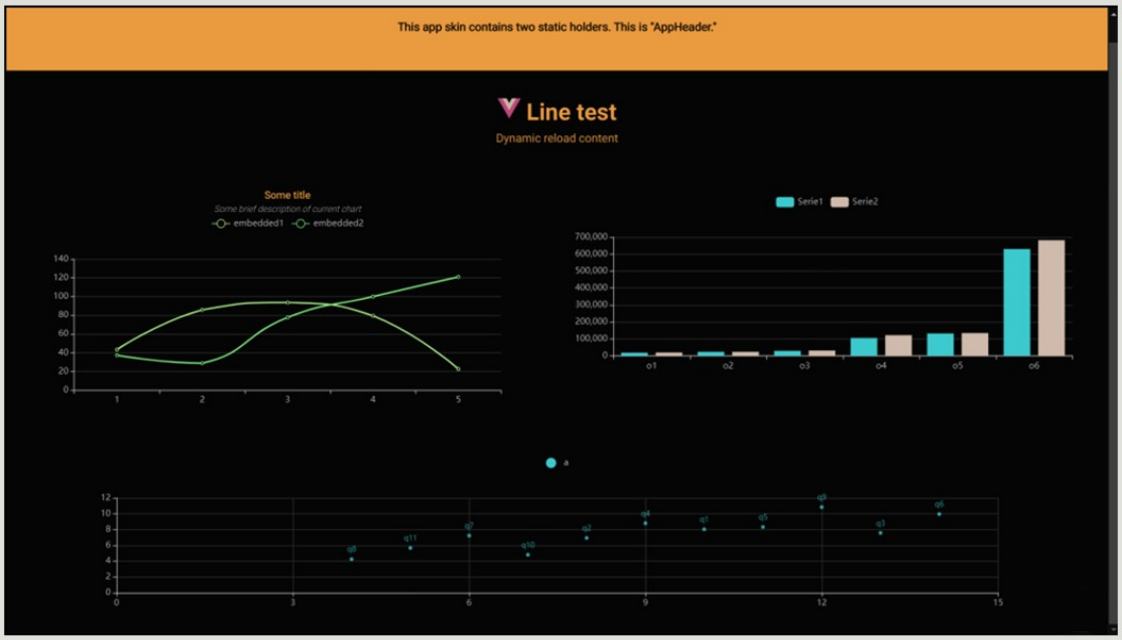
Створення лінійчатої, стовпчастої діаграми, діаграми розкиду



Управління віджетом з спливаючого меню



Екран конфігурування властивостей віджету



Інтерфейс для відображення даних у робочому режимі

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Транського Микити Олексійовича

(прізвище, ім`я та по батькові)

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Комп'ютерна графіка і Web-дизайн»*

Керівник дипломного проекту (роботи) *Закроєв Юрій Михайлович*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка набору віджетів для візуалізації числових даних у вигляді діаграм*

Обсяг розрахунково-пояснювальної записки *80* сторінок

Обсяг графічної (презентаційної) частини *16* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту завданню

Представлений дипломний проект відповідає затвердженій темі та виконаний відповідно технічному завданню. Дипломний проект присвячено огляду та аналізу засобів візуалізації даних, розробці алгоритмічного та програмного забезпечення, що реалізує візуалізацію даних у вигляді діаграм, і складається з пояснювальної записки та мультимедійної презентації

б) характеристика виконання кожного розділу дипломного проекту

Пояснювальна записка складається з основного розділу (Огляд методів візуального представлення інформації; Аналітичний огляд засобів візуалізації числових даних; Розробка моделі життєвого циклу компоненту; Розробка UML-діаграми класів; Розробка структури застосунку; Реалізація застосунку; Встановлення та тестування розробленого набору для візуалізації числових даних), економічного розділу, розділу охорони праці та додатків. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту

Графічна частина складається з 16 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять структурні, та функціональні схеми, діаграми, скріншоти, блок-схеми алгоритмів, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання пояснювальної записки відмінна, розробку виконано у повному обсязі.

г) перелік позитивних якостей дипломного проекту *Розроблені віджети можуть бути вбудованими у web-застосунки, передбачають різні види візуалізації числових даних (лінійчатої, стовпчастої діаграми та діаграми розкиду). Використано актуальні засоби розробки JavaScript та Vue.js*

д) основні недоліки дипломного проекту *На аркуші змісту пояснювальної записки не позначено підрозділи у розділі 3 (розділ охорони праці та техніки безпеки);*
Бажано було б передбачити не тільки англомовний, але і україномовний інтерфейс у віджетах

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові рецензента *к.т.н. Рудніченко Микола Дмитрович*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис

«*20*» *червня* *2025 р.*

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Транського Микити Олексійовича

(прізвище, ім`я та по батькові)

Спеціальність: *123 «Комп'ютерна інженерія»*

Освітня програма: *«Комп'ютерна графіка і Web-дизайн»*

Тема дипломного проекту: *Розробка набору віджетів для візуалізації числових даних у вигляді діаграм*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка до дипломного проекту містить 80 сторінок. У пояснювальній записці виконано огляд та аналіз засобів візуалізації даних, розроблено алгоритмічне та програмне забезпечення, що реалізує візуалізацію даних у вигляді діаграм. Графічна частина складається з 16 слайдів, оформлених у вигляді презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та слайдів добра.*

б) самостійність роботи над проектом: *Протягом виконання дипломного проекту здобувач освіти Транський Микита поступово та послідовно виконував всі етапи, проявив ініціативу в створенні загальної концепції та реалізації роботи. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Транський Микита під час роботи над дипломним проектом вивчив достатньо багато літературних та інтернет-джерел за даною тематикою.*

Вважаю, що теоретична підготовка дипломника достатня і він готовий до захисту проекту.

г) вміння розв'язувати виробничі та конструкторські питання *Під час виконання дипломного проекту здобувач освіти Транський Микита показав вміння організовано працювати над поставленим завданням, застосовувати знання у галузі Web-дизайну, графіки та програмування, розробляти, встановлювати та налаштовувати спеціалізоване програмне забезпечення, користуватись сучасними засобами розробки, такими як JavaScript, Vue.js, ECharts*

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові керівника дипломного проекту *Закроєв Юрій Михайлович*

Місце роботи і посада керівника дипломного проекту *директор ТОВ «Біг ВОШ»*

Підпис

«16» 06 2025 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Транський М.О.,
здобувач освіти гр. 4КГ-08, та

Закроєв Ю.М.,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка набору віджетів для візуалізації числових даних у вигляді діаграм» (автор роботи – Транський М.О., керівник роботи – Закроєв Ю.М.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Транський М.О. /

Керівник _____ / Закроєв Ю.М. /

«16» червня 2025 р.

ДОВІДКА

циклової комісії КТ та ПІ
про допуск до захисту дипломного проєкту
здобувача (здобувачки) освіти IV курсу
відділення комп'ютерних систем групи 4КГ-08

Транського Микити Олексійовича

на тему *Розробка набору віджетів для візуалізації числових даних у вигляді діаграм*

Висновок відповідальної особи за проведення нормоконтролю:
пояснювальна записка до дипломного проєкту виконана з несуттєвими порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування

(підпис) *16.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 16.05.2025 р. значення коефіцієнту подібності в роботі становить 26,13%, коефіцієнт цитування – 5,64%.*

(підпис) *16.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) дипломного проєкту

здобувача (здобувачки) освіти *Транського М.О.* (П.І.Б.)

проведена «*16*» *червня* 20*25* р.

Висновки *Пояснювальна записка до дипломного проєкту виконана у повному обсязі. Випускна кваліфікаційна робота (дипломний проєкт) відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Голова ЦК КТ та ПІ (підпис) *Кривченко Ю.В.* (П.І.Б.)

StrikePlagiarism.com

Дата звіту 5/16/2025
Дата редагування 5/19/2025

Документ прийнятий

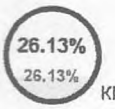
Звіт подібності

метадані

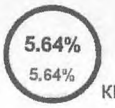
Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Розробка набору віджетів для візуалізації числових даних у вигляді діаграм
Автор
Транський Микита Олексійович
Науковий керівник / Експерт
Закроєв Юрій Михайлович
підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КЦ

25
Довжина фрази для коефіцієнта подібності 2

14092
Кількість слів

121657
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		12
Інтервали		0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)		146

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Бібліотека віджетів для візуалізації статистичних даних [redacted] 3/15/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	79 0.56 %
2	Бібліотека віджетів для візуалізації ієрархічних даних [redacted] 3/15/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	74 0.53 %

3	https://card-file.ontu.edu.ua/bitstreams/34a6756b-592f-4b77-a805-183aa03a6a26/download	56 0.40 %
4	Бібліотека віджетів для візуалізації статистичних даних [redacted] 3/15/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	55 0.39 %
5	https://card-file.ontu.edu.ua/bitstreams/341a820e-d025-42f3-b7dc-27e831d6c66f/download	54 0.38 %
6	https://card-file.ontu.edu.ua/server/api/core/bitstreams/8da72e29-656f-4ee4-9b22-716dedf53ff5/content	47 0.33 %
7	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	44 0.31 %
8	Бібліотека віджетів для візуалізації ієрархічних даних [redacted] 3/15/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	43 0.31 %
9	Бібліотека віджетів для візуалізації статистичних даних [redacted] 3/15/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	43 0.31 %
10	Бібліотека віджетів для візуалізації статистичних даних [redacted] 3/15/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	43 0.31 %

з домашньої бази даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з програми обміну базами даних (12.60 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Бібліотека віджетів для візуалізації статистичних даних [redacted] 3/15/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**1059 (69) 7.51 %**
2	Бібліотека віджетів для візуалізації ієрархічних даних [redacted] 3/15/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**690 (44) 4.90 %**
3	ФКПІ_2024_121_Кучміи_Р.В [redacted] 7/11/2024 Ukrainian national aviation university (Ukrainian national aviation university)	12 (1) 0.09 %
4	ВКР_Зуб [redacted] 11/28/2024 State University of Trade and Economics (Кафедра комп'ютерних наук та інформаційних систем)	10 (1) 0.07 %
5	Організація процесу відновлення технічного стану колінчастих валів дизелів з удосконаленням процесу наплавлення зношених поверхонь [redacted] 6/17/2024 Zhytomyr Agricultural Technical Professional College (Zhytomyr Agricultural Technical Professional College)	5 (1) 0.04 %

з Інтернету (13.53 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

1	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	**834 (63) 5.92 %**
2	https://card-file.ontu.edu.ua/server/api/core/bitstreams/4bb7255e-46d4-4349-9726-9698476da02d/content	**167 (11) 1.19 %**
3	https://card-file.ontu.edu.ua/bitstreams/c58b0ff5-46e0-49f8-8cbe-65c32256665d/download	**89 (5) 0.63 %**
4	https://card-file.ontu.edu.ua/bitstreams/34a6756b-592f-4b77-a805-183aa03a6a26/download	**73 (2) 0.52 %**
5	https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content	**64 (2) 0.45 %**
6	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	**59 (3) 0.42 %**
7	https://card-file.ontu.edu.ua/server/api/core/bitstreams/8da72e29-656f-4ee4-9b22-716dedf53ff5/content	**58 (3) 0.41 %**
8	https://card-file.ontu.edu.ua/bitstreams/0e6c3361-ffbf-4469-86a1-fe84a1fe21cd/download	**55 (4) 0.39 %**
9	https://card-file.ontu.edu.ua/bitstreams/341a820e-d025-42f3-b7dc-27e831d6c66f/download	**54 (1) 0.38 %**
10	https://card-file.ontu.edu.ua/bitstreams/538ada8a-2c79-4b1e-b7d2-b0c97f68bc1c/download	53 (4) 0.38 %
11	https://card-file.ontu.edu.ua/bitstreams/e4afae26-0a7e-4a4d-afc2-94341838de2a/download	53 (6) 0.38 %
12	https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download	**49 (2) 0.35 %**
13	https://card-file.ontu.edu.ua/server/api/core/bitstreams/44c16132-5f53-48e2-b6c0-61e9a2f0fd75/content	45 (3) 0.32 %
14	https://card-file.ontu.edu.ua/bitstreams/2b75599e-e1ac-412d-bf09-10d2eb49022f/download	37 (4) 0.26 %
15	https://card-file.ontu.edu.ua/bitstreams/82a6d375-2b69-4233-b80f-fbfd149b7747/download	**33 (1) 0.23 %**
16	https://card-file.ontu.edu.ua/bitstreams/bbaf3f38-16a8-4070-bead-5562769b7c71/download	28 (2) 0.20 %
17	https://card-file.ontu.edu.ua/server/api/core/bitstreams/3302e08a-9549-43ba-8861-728bff7dc7ff/content	21 (3) 0.15 %
18	https://card-file.ontu.edu.ua/server/api/core/bitstreams/6eb6bf1c-5813-45e6-93c5-25539b4709d3/content	17 (2) 0.12 %
19	https://card-file.ontu.edu.ua/bitstreams/72fa1396-889f-4082-af7d-898b6ac28dd4/download	17 (1) 0.12 %
20	http://um.co.ua/6/6-9/6-99690.html	17 (1) 0.12 %
21	https://studfile.net/preview/7305433/page:2/	15 (1) 0.11 %
22	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download	14 (1) 0.10 %
23	https://blog.skillfactory.ru/instrumenti-dlja-vizualizacii-dannyh/	11 (1) 0.08 %
24	https://card-file.ontu.edu.ua/bitstreams/158e44b0-583e-4b2d-b758-6b86979e33bb/download	9 (1) 0.06 %
25	https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download	8 (1) 0.06 %
26	http://dspace.wunu.edu.ua/bitstream/316497/39098/1/%D0%9F%D0%BE%D0%B3%D0%BE%D1%80%D0%B5%D1%86_%D0%BA%D0%B8%D0%B9_%D0%94%D0%9F_2019.pdf	6 (1) 0.04 %
27	https://elartu.tntu.edu.ua/bitstream/lib/45521/1/Zabuysky_V.pdf	5 (1) 0.04 %
28	https://card-file.ontu.edu.ua/bitstreams/bbed74c8-2ea7-44c5-8d00-0fe3fd9790ee/download	5 (1) 0.04 %
29	https://studfile.net/preview/5082963/page:7/	5 (1) 0.04 %
30	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	5 (1) 0.04 %

Список прийнятих фрагментів (немає прийнятих фрагментів)