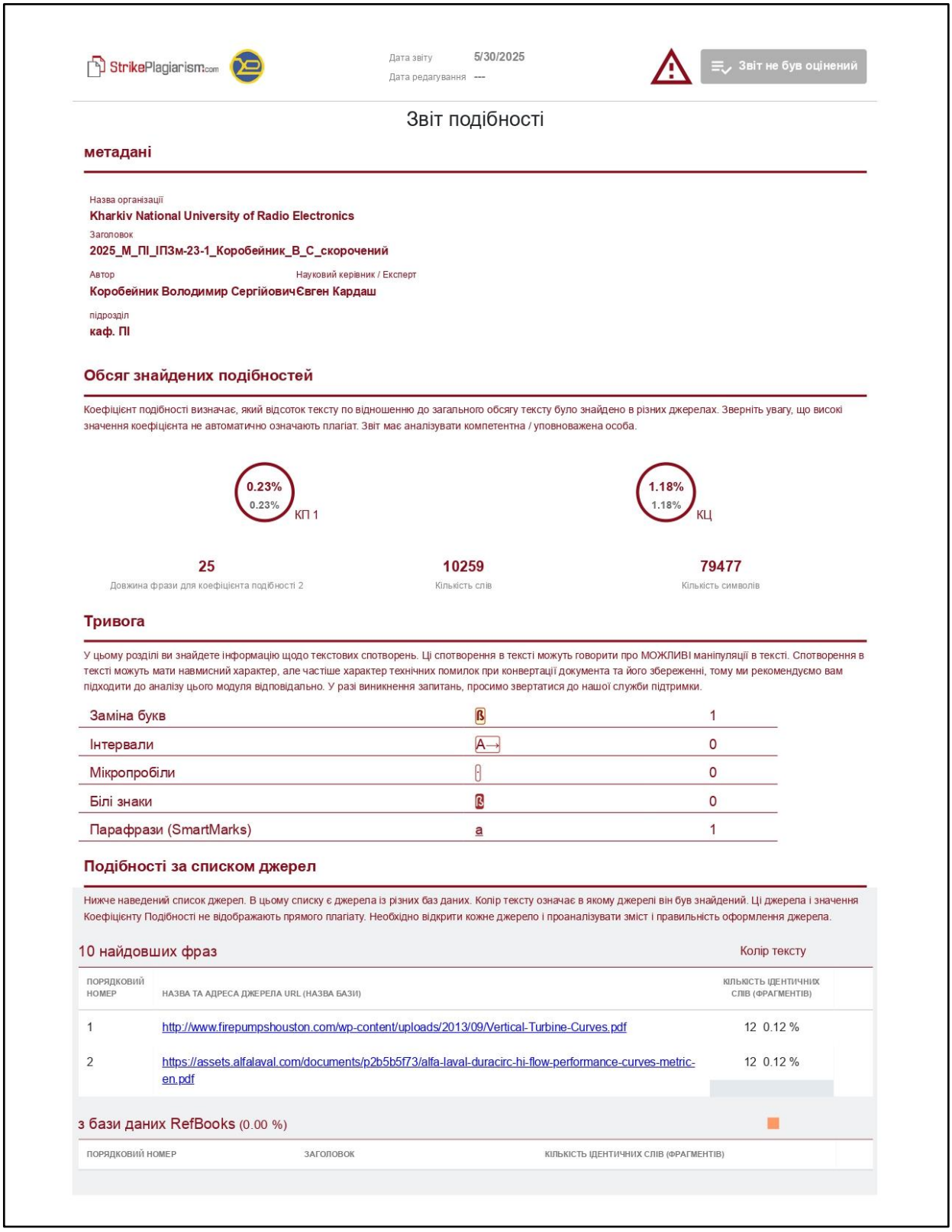
ДОДАТОК А

ЗВІТ РЕЗУЛЬТАТІВ ПЕРЕВІРКИ НА УНІКАЛЬНІСТЬ ТЕКСТУ В БАЗІ ХНУРЕ



StrikePlagiarism.com

Дата звіту 5/30/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_М_ПІ_ІПЗм-23-1_Коробейник_В_С_скорочений
Автор
Коробейник Володимир Сергійович
Науковий керівник / Експерт
Євген Кардаш
підрозділ
каф. ПІ

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.

0.23% 0.23% КП 1

1.18% 1.18% КЦ

25	10259	79477
Довжина фрази для коефіцієнта подібності 2	Кількість слів	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	ß	1
Інтервали	A→	0
Мікропробіли	¶	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	1

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	http://www.firepumpshouston.com/wp-content/uploads/2013/09/Vertical-Turbine-Curves.pdf	12 0.12 %
2	https://assets.alfalaval.com/documents/p2b5b5f73/alfa-laval-duracirc-hi-flow-performance-curves-metric-en.pdf	12 0.12 %

з бази даних RefBooks (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

Рисунок А.1 – Звіт з перевірки роботи на унікальність (рисунок створено самостійно)

ДОДАТОК Б
СЛАЙДИ ПРЕЗЕНТАЦІЇ

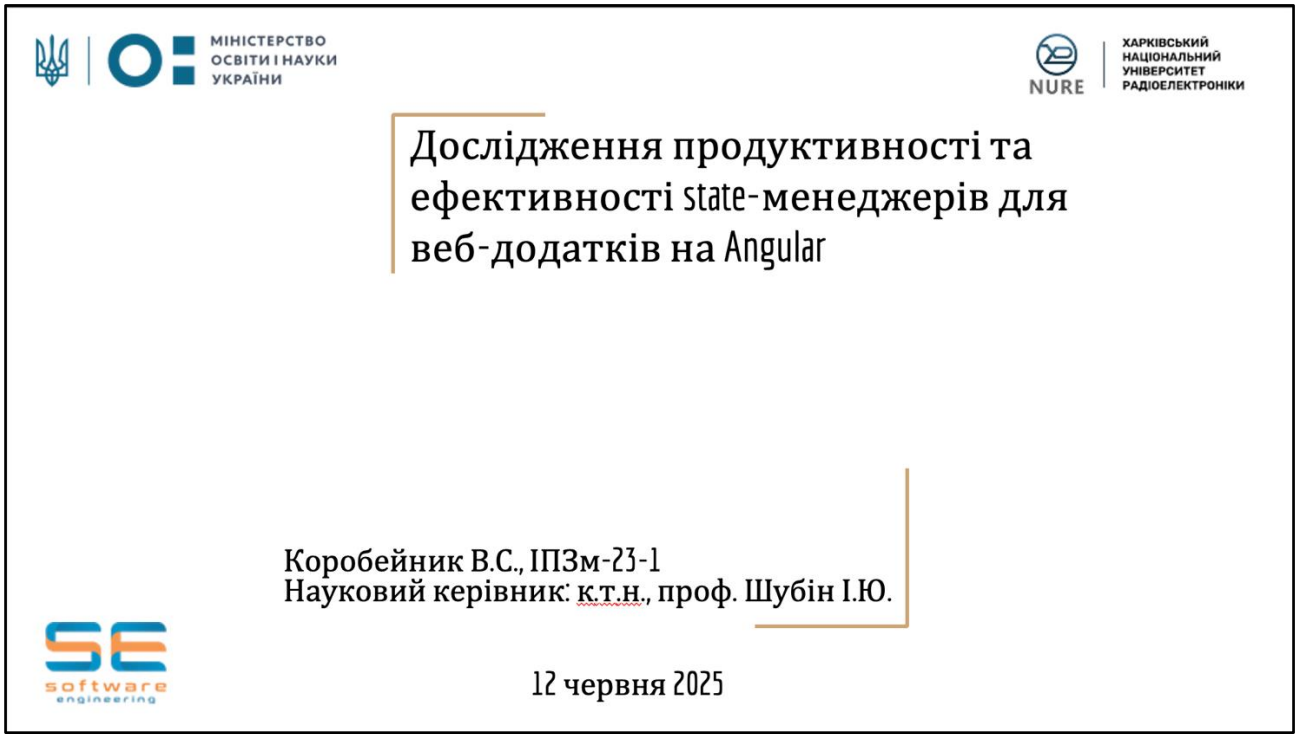


Рисунок Б.1 – Слайд 1 (рисунок створено самостійно)

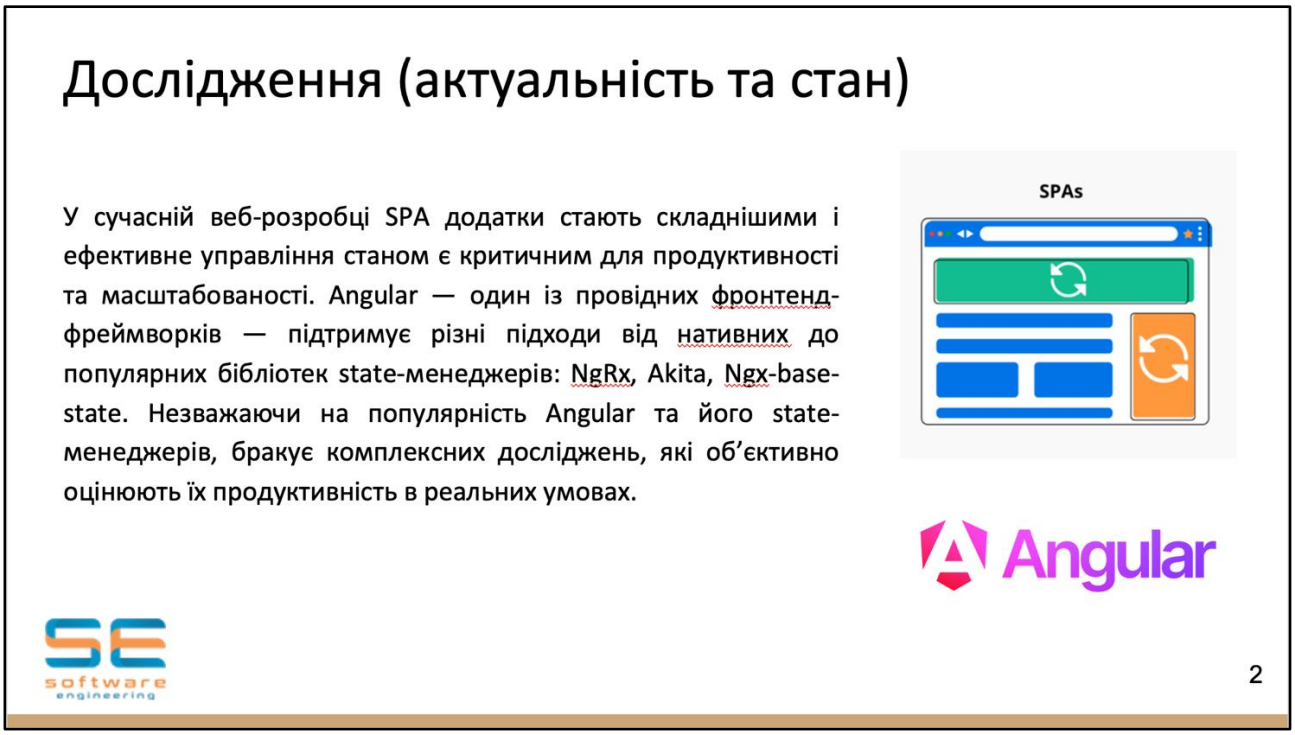


Рисунок Б.2 – Слайд 2 (рисунок створено самостійно)

Дослідження (напрям та об’єкт)

Напрям дослідження
Порівняльний аналіз продуктивності та ефективності state-менеджерів NgRx, Akita і Ngx-base-state в Angular-додатках. Фокус зроблено на оцінку практичних критеріїв вибору оптимального інструменту залежно від масштабу та складності проєкту.

Об’єкт дослідження
Процеси управління станом у веб-додатках, створених з використанням Angular

Предмет дослідження
Стейт-менеджери та їх вплив на продуктивність та ефективність роботи веб-додатків

software engineering

3

Рисунок Б.3 – Слайд 3 (рисунок створено самостійно)

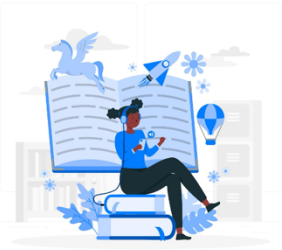
Огляд літератури (аналогів)

Основні джерела

За основу було взято офіційні документації Angular, NgRx, Akita, Ngx-base-state, книжку “RxJs In Action”, для вивчення попередніх досліджень сучасні статті від розробників — “Deep Comparison of State Management Solutions in Angular”, “State Management in Angular: Exploring Strategies”.

Виявлені прогалини

Більшість досліджень мають теоретичний характер та не включають тестування в реальних умовах, а також мають загальні висновки, які не підкріплені конкретними метриками чи прикладами. Маловідомі та нові рішення, зокрема Ngx-base-state, переважно ігноруються.



software engineering

4

Рисунок Б.4 – Слайд 4 (рисунок створено самостійно)

Рисунок Б.5 – Слайд 5 (рисунок створено самостійно)

Рисунок Б.6 – Слайд 6 (рисунок створено самостійно)

Методологія

Методи дослідження

У дослідженні використано теоретичний аналіз та експериментальний підхід для практичної частини. В теоретичному аналіз розглянуто архітектури, API та принципи роботи state-менеджерів а також особливості їх інтеграції. В практичній частині створено три тестових Angular-додатки зі ідентичним функціоналом, у кожному з яких реалізовано один із стейт-менеджерів — NgRx, Akita, Ngx-base-state. Проведено тестування та порівняльний аналіз за такими метриками, як швидкодія, споживання пам'яті, затримка рендеру та обсяг коду.

software engineering

7

Рисунок Б.7 – Слайд 7 (рисунок створено самостійно)

Інструменти та технології

- Клієнтська частина: Angular (v17+), TypeScript, RxJS
- Серверна частина: Node.js (NestJS) + PostgreSQL
- Тестування: Chrome DevTools, Lighthouse, Jasmine/Karma
- IDE: WebStorm, IntelliJ IDEA, Postman
- Розробка клієнтської частини велась у монорепозиторії Nx з бібліотекою спільних компонентів

software engineering

8

Рисунок Б.8 – Слайд 8 (рисунок створено самостійно)

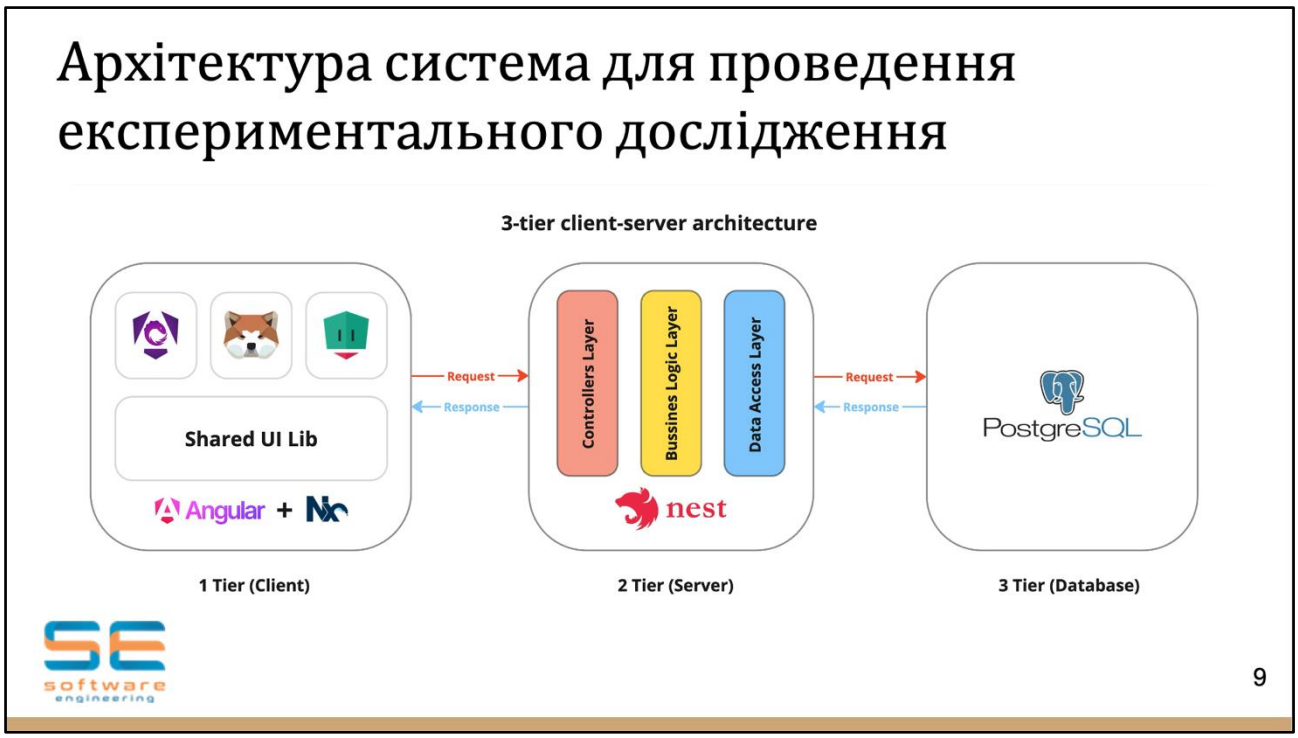


Рисунок Б.9 – Слайд 9 (рисунок створено самостійно)

Рисунок Б.10 – Слайд 10 (рисунок створено самостійно)

Рисунок Б.11 – Слайд 11 (рисунок створено самостійно)

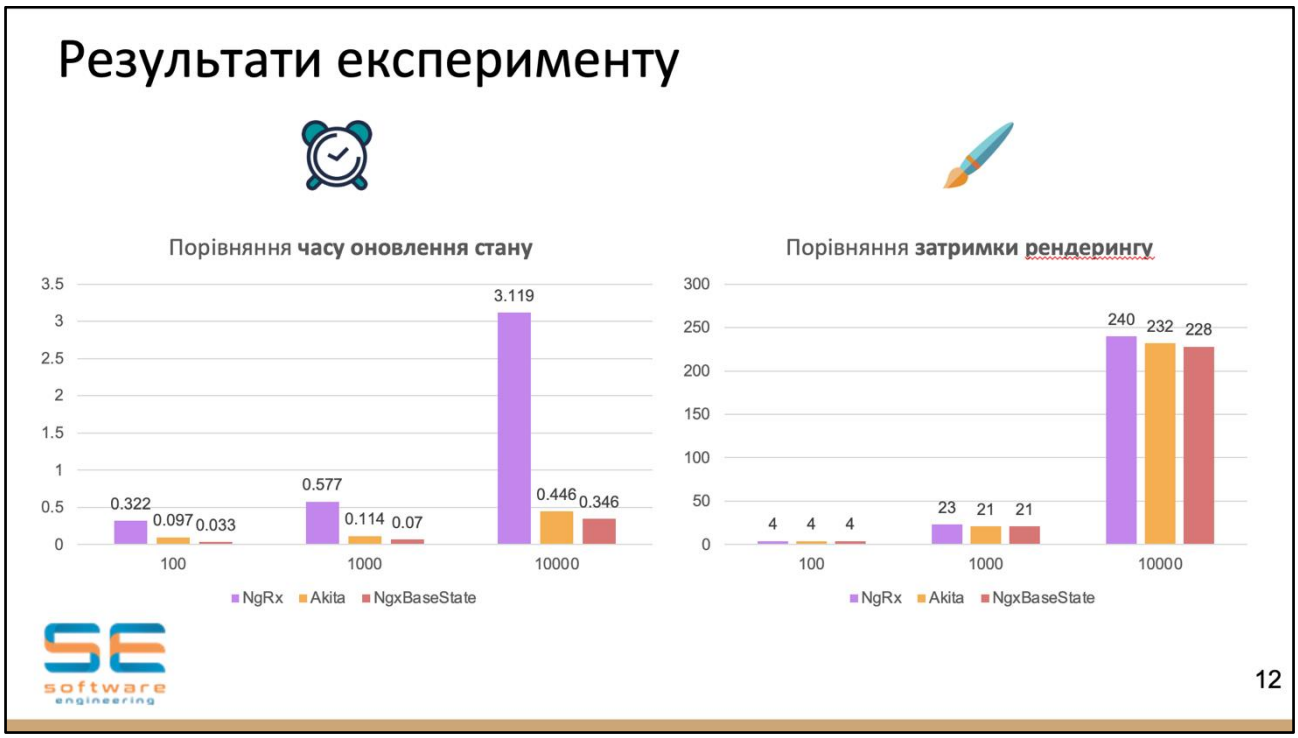


Рисунок Б.12 – Слайд 12 (рисунок створено самостійно)

Рисунок Б.13 – Слайд 13 (рисунок створено самостійно)

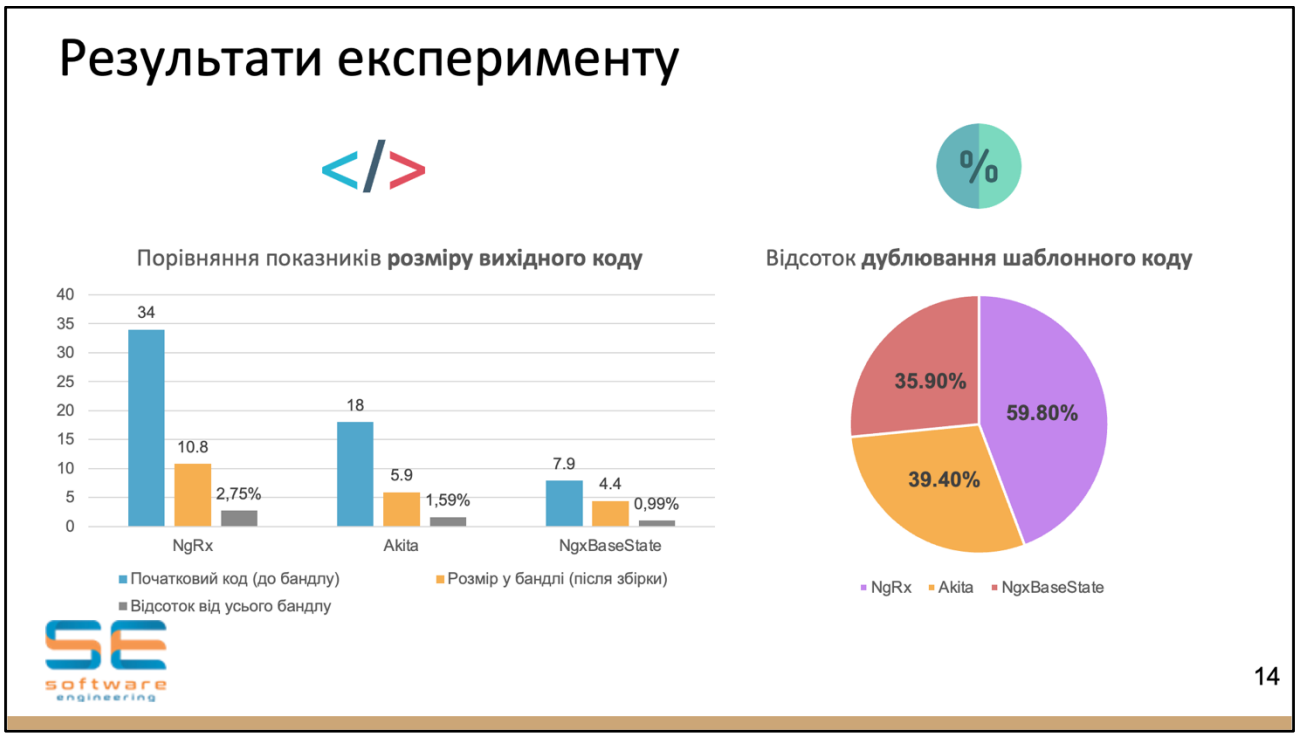


Рисунок Б.14 – Слайд 14 (рисунок створено самостійно)

Аналіз отриманих результатів

Метрика	Ngx-base-state	Akita	NgRx
Час оновлення стану	< 0.4 мс	0.4 – 0.7 мс	> 0.7 мс
Затримка рендерингу	< 230 мс	230 – 240 мс	> 240 мс
Використання пам'яті	< 170 МБ	170 – 172 МБ	> 172 МБ
Розмір коду (LoC)	< 300	300 – 450	> 800
Розмір у бандлі (КБ)	< 5 КБ	5 – 6 КБ	> 10 КБ
Відсоток дублювання шаблонного коду	< 38%	38 – 45%	> 50%

software engineering

15

Рисунок Б.15 – Слайд 15 (рисунок створено самостійно)

Аналіз отриманих результатів

Інтерпретація результатів

- NgRx - для великих проєктів, де важлива структурованість і масштабованість.
- Akita - баланс між гнучкістю та структурністю підходить для середніх проєктів.
- Ngx base state - для невеликих додатків з фокусом на швидкість розробки.

Вплив на практику

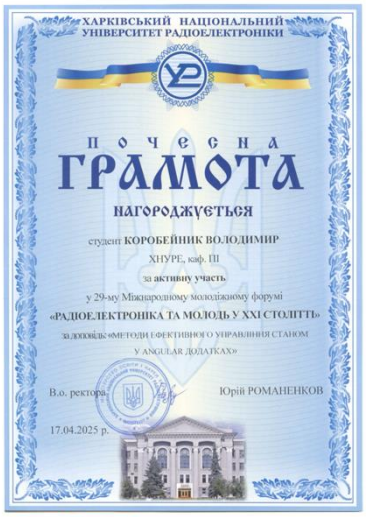
Отримані результати є орієнтиром для розробників Angular при виборі state-менеджера з урахуванням масштабу та складності проєкту, надаючи чіткі рекомендації для вибору інструменту.

software engineering

16

Рисунок Б.16 – Слайд 16 (рисунок створено самостійно)

Публікація результатів


ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

ПОЧЕСНА ГРАМОТА

НАГОРОДЖУЄТЬСЯ

студент КОРОБЕЙНИК ВОЛОДИМИР

ХНУРЕ, каф. ПІ

за активну участь

у 29-му Міжнародному молодіжному форумі

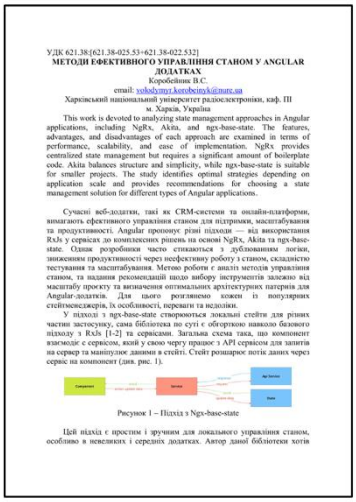
«РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ»

за доповідь «МЕТОДИ ЕФЕКТИВНОГО УПРАВЛІННЯ СТАНОМ У ANGULAR ДОДАТКАХ»

В.о. ректора Юрій РОМАНЕНКОВ

17.04.2025 р.

Результати дослідження були опубліковані на 29-му Міжнародному молодіжному форумі "Радіоелектроніка та молодь у XXI столітті", за результатами якого було отримано нагороду за активну участь і цікаву доповідь.


УДК 621.38:[621.38-025.53+621.38-022.532]

МЕТОДИ ЕФЕКТИВНОГО УПРАВЛІННЯ СТАНОМ У ANGULAR ДОДАТКАХ

Коробейник В.С.

email: volodymyr.korobeinyk@nure.ua

Харківський національний університет радіоелектроніки, каф. ПІ

м. Харків, Україна

This work is devoted to analyzing state management approaches in Angular applications, including NgRx, Akita, and ngx-base-state. The features, advantages, and disadvantages of each approach are examined in terms of performance, scalability, and ease of implementation. NgRx provides centralized state management but requires a significant amount of boilerplate code. Akita balances structure and simplicity, while ngx-base-state is suitable for smaller projects. The study identifies optimal strategies depending on application scale and provides recommendations for choosing a state management solution for different types of Angular applications.

Сучасні веб-додатки, такі як CRM-системи та онлайн-платформи, вимагають ефективного управління станом для підтримки, масштабування та продуктивності. Angular пропонує різні підходи — від використання RxJs у сервісах до комплексних рішень на основі NgRx, Akita та ngx-base-state. Однак розробники часто стикаються з дубльованими логіками, зниженням продуктивності через неефективну роботу з станом, складністю тестування та масштабування. Метою роботи є аналіз методів управління станом, та надання рекомендацій щодо вибору інструментів залежно від масштабу проєкту та визначення оптимальних архітектурних патернів для Angular-додатків. Для цього розглянемо кожен із популярних стейт-менеджерів, їх особливості, переваги та недоліки.

У підході з ngx-base-state створюються локальні стейти для різних частин застосунку, сама бібліотека по суті є обгорткою навколо базового підходу з RxJs [1-2] та сервісами. Загальна схема така, що компонент взаємодіє с сервісом, який у свою чергу працює з API сервісом для запитів на сервер та маніпулює даними в стейті. Стейт розширює потік даних через сервіс на компонент (див. рис. 1).

Рисунок 1 – Підхід з Ngx-base-state

Цей підхід є простим і зручним для локального управління станом, особливо в невеликих і середніх додатках. Автор даної бібліотеки хотів

software engineering

17

Рисунок Б.17 – Слайд 17 (рисунок створено самостійно)

Підсумки

Реалістичність та корисність

Отримані результати базуються на практичному тестуванні в умовах, наближених до реальних у сучасних веб-додатках. Порівняння охоплює як технічні метрики, так і зручність впровадження, що робить результати практично корисними для розробників.

Можливий розвиток досліджень

Подальші дослідження можуть включати порівняння з іншими стейт-менеджерами, зокрема менш відомими рішеннями, а також аналіз сучасного підходу Angular Signals. Перспективним є дослідження ефективності state management у microfrontend-архітектурі, SSR-додатках, PWA та мобільних застосунках.

software engineering

18

Рисунок Б.18 – Слайд 18 (рисунок створено самостійно)

ДОДАТОК В

АПРОБАЦІЯ РЕЗУЛЬТАТІВ РОБОТИ

Рисунок В.1 – Грамота переможця 29-го Молодіжного Форума (рисунок створено самостійно)

УДК 621.38:[621.38-025.53+621.38-022.532]

МЕТОДИ ЕФЕКТИВНОГО УПРАВЛІННЯ СТАНОМ У ANGULAR ДОДАТКАХ

Коробейник В.С.
email: volodymyr.korobeinyk@nure.ua
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна

This work is devoted to analyzing state management approaches in Angular applications, including NgRx, Akita, and ngx-base-state. The features, advantages, and disadvantages of each approach are examined in terms of performance, scalability, and ease of implementation. NgRx provides centralized state management but requires a significant amount of boilerplate code. Akita balances structure and simplicity, while ngx-base-state is suitable for smaller projects. The study identifies optimal strategies depending on application scale and provides recommendations for choosing a state management solution for different types of Angular applications.

Сучасні веб-додатки, такі як CRM-системи та онлайн-платформи, вимагають ефективного управління станом для підтримки, масштабування та продуктивності. Angular пропонує різні підходи — від використання RxJs у сервісах до комплексних рішень на основі NgRx, Akita та ngx-base-state. Однак розробники часто стикаються з дублюванням логіки, зниженням продуктивності через неефективну роботу з станом, складністю тестування та масштабування. Метою роботи є аналіз методів управління станом, та надання рекомендацій щодо вибору інструментів залежно від масштабу проєкту та визначення оптимальних архітектурних патернів для Angular-додатків. Для цього розглянемо кожен із популярних стейтменеджерів, їх особливості, переваги та недоліки.

У підході з ngx-base-state створюються локальні стейти для різних частин застосунку, сама бібліотека по суті є обгорткою навколо базового підходу з RxJs [1-2] та сервісами. Загальна схема така, що компонент взаємодіє с сервісом, який у свою чергу працює з API сервісом для запитів на сервер та маніпулює даними в стейті. Стейт розшарює потік даних через сервіс на компонент (див. рис. 1).

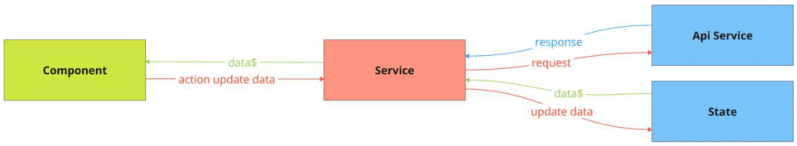


Рисунок 1 – Підхід з Ngx-base-state

Цей підхід є простим і зручним для локального управління станом, особливо в невеликих і середніх додатках. Автор даної бібліотеки хотів

Рисунок В.2 – Перша сторінка матеріалу (рисунок створено самостійно)

створити просте рішення для гнучкого керування станом, протилежне NgRx з великою кількістю шаблонного коду. Серед його переваг — легкість впровадження, реактивність завдяки RxJs і можливість ізоляції стану в сервісах, що сприяє зручній організації коду. Водночас для великих додатків із складною логікою цей підхід може виявитися менш ефективним, оскільки може виникнути проблема із підтримкою великої кількості сервісів та дублювання станів, що ускладнює масштабування.

У підході з Akita стан зберігається в Entity Store [3], що забезпечує централізоване управління та реактивність завдяки RxJs. Akita використовує модельно-орієнтований підхід, де стан представлений у вигляді окремих моделей, а доступ до нього здійснюється через Query. Оновлення стану відбувається на рівні сервісів, які взаємодіять з store та API сервісами, що мінімізує шаблонний код (див. рис. 2).

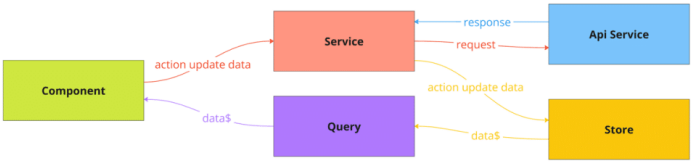


Рисунок 2 – Підхід з Akita

Завдяки вбудованому кешуванню та простому API Akita дозволяє легко керувати станом у середніх і великих додатках. Перевагами є менша складність у порівнянні з NgRx, легше впровадження та підтримка реактивного оновлення. Проте Akita не має підтримки сигналів, та має менш активну спільноту та посередню документацію, що може ускладнити довгострокову підтримку в проєктах.

У підході з NgRx стан зберігається в глобальному Store, який є централізованим місцем для управління станом додатка. Для передачі та оновлення даних використовуються actions, reducers, selectors [4], а для асинхронних операцій такі як запити до API використовуються effects (див. рис. 3).

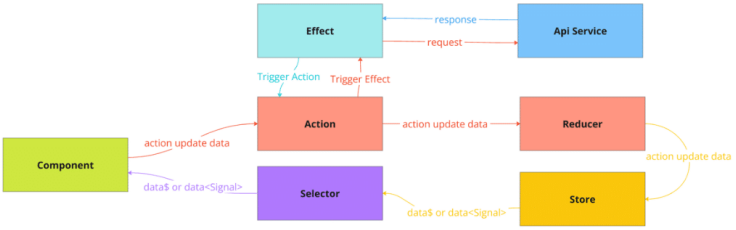


Рисунок 3 – Підхід з NgRx

Цей підхід базується на шаблоні Redux і забезпечує чітке відокремлення бізнес-логіки від компонентів. Завдяки централізованому

Рисунок В.3 – Друга сторінка матеріалу (рисунок створено самостійно)

управлінню станом та строгій архітектурі NgRx полегшує масштабування додатка, роблячи його більш структурованим і передбачуваним. Також у NgRx є підтримка сигналів, що говорить про актуальність інструменту, в цілому він має велике ком'юніті та часто використовується для розробки. Однак NgRx може бути складним для освоєння через складність архітектури. Також він потребує написання значної кількості шаблонного коду та необхідність використання додаткових бібліотек. У невеликих проектах використання NgRx може спричинити зниження продуктивності через зайві абстракції та накладні витрати.

Порівняння підходів до управління станом в Angular додатках показало, що Ngx-base-state поєднує простоту та реактивність, але його може не вистачати для складних систем. У великих проєктах його використання може призвести до створення великої кількості сервісів та стейтів без чіткої організації, що ускладнює підтримку та масштабування додатка. Akita збалансована між структурованістю та простотою, зменшує обсяг шаблонного коду порівняно з NgRx і добре підходить для середніх і великих проєктів, хоча її популярність і підтримка обмежені. NgRx забезпечує централізоване управління станом, підходить для великих додатків, але потребує більше ресурсів на освоєння й створює багато шаблонного коду, що може ускладнювати розробку.

Для малих проєктів доцільно використовувати локальне управління станом із RxJs, Signal або Ngx-base-state, оскільки ці підходи дозволяють швидко організувати управління даними без зайвої складності. Середні проєкти виграють від поєднання локального і глобального станів, де Akita може бути зручною альтернативою NgRx завдяки простішій інтеграції та меншій кількості шаблонного коду. У великих системах оптимальним вибором є NgRx із модульним підходом, lazy loading і кешуванням [5], що допомагає ефективно працювати з даними та зменшувати навантаження на додаток. При виборі підходу потрібно враховувати масштаб та особливості проєкту, а також вимог до продуктивності та підтримки додатку.

Список використаних джерел:

1. Reactive Programming with RxJS. URL: https://rxjs.dev/guide/overview (дата звернення: 25.02.2025).

2. Ngx-base-state Documentation. URL: https://github.com/ngx-base-state (дата звернення: 25.02.2025).

3. Akita Documentation. URL: https://opensource.salesforce.com/akita/ (дата звернення: 25.02.2025).

4. NgRx Documentation. URL: https://ngrx.io/guide/store (дата звернення: 25.02.2025).

5. Angular Best Practices. URL: https://v17.angular.io/guide/lazy-loading-ngmodules (дата звернення: 25.02.2025).

Рисунок В.4 – Третя сторінка матеріалу (рисунок створено самостійно)

ДОДАТОК Г
ЕКСПЕРТНИЙ ВИСНОВОК РЕЗУЛЬТАТІВ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА ВІДПОВІДНІСТЬ ОФОРМЛЕННЯ ВИМОГАМ ДСТУ 3008:2015

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗм-23-1 (група)

Коробейник Володимир Сергійович
(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.4 Нумерація розділів, підрозділів, пунктів, підпунктів	
	7.5 Рисунки	
	7.6 Таблиці	
	7.7 Переліки	
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні познаки	
	7.15 Додатки	

зауважень немає

Експерт _____ (підпис) Олена ОЛІЙНИК (прізвище, ініціали)

01.06.2025

Рисунок Г.1 –Результати перевірки на нормоконтроль (рисунок створено самостійно)