ДОДАТОК А
Слайди презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Дослідження методів оптимізації продуктивності хмарних веб-додатків контейнеризованих у Kubernetes

Ворона Д. О., ІПЗм-23-1
Науковий керівник: проф. Галуза О. А.

software engineering

17 червня 2025

Рисунок А.1 – Слайд 1

Дослідження

У сучасних умовах зростання вимог до доступності та швидкодії веб-додатків, хмарні технології стали стандартом де-факто в розробці програмного забезпечення.
Однією з ключових платформ є **Kubernetes**, яка дозволяє автоматизувати процеси розгортання, масштабування та управління контейнеризованими додатками.

Напрям: оптимізація продуктивності хмарних веб-додатків, що працюють у середовищі Kubernetes.

Об’єкт дослідження: методи оптимізації хмарних веб-додатків контенеризованих у Kubernetes.

software engineering

Рисунок А.2 – Слайд 2

Огляд літератури (аналогів)

- Kubernetes and Docker Load Balancing: State-of-the-Art Techniques and Challenges (2023)
- Enhanced Visibility for Real-time Monitoring and Alerting in Kubernetes by Integrating Prometheus,Prometheus, Grafana, Loki, and Alerta (2024)
- Kubernetes in Microservices (2022)
- Optimizing Container Management with Kubernetes on Linux: Key Strategies and Common Obstacles (2024)
- Navigating the Landscape of Kubernetes Security Threats and Challenges (2024)
- Containerization in cloud computing: comparing Docker and Kubernetes for scalable web applications (2024)

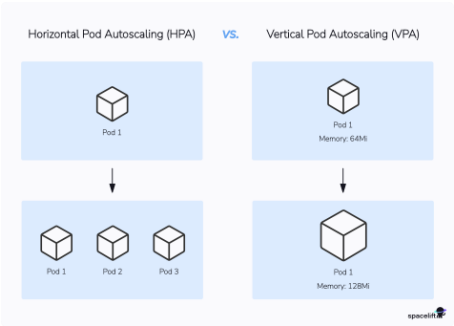
- Зосереджені на окремих аспектах, таких як балансування навантаження або захист від кіберзагроз;
- Бракує інтегрованого підходу, який охоплює всі етапи життєвого циклу додатків;
- Є проблеми з автоматичним масштабуванням і адаптацією до змін у робочих навантаженнях.

Рисунок А.3 – Слайд 3

Постановка задачі

- Контейнери запускаються з невизначеними або фіксованими ресурсами, що призводить до перевитрат або нестабільної роботи;
- Відсутнє автоматичне масштабування й адаптація до змін трафіку, що обмежує гнучкість і стійкість системи.

Тому необхідно знайти методи оптимізації, які забезпечать баланс між продуктивністю, доступністю та вартістю інфраструктури.



У результаті дослідження планується:

- Визначити найефективніші підходи до масштабування та оптимізації ресурсів (HPA, VPA, мікросервіси);
- Реалізувати порівняльний аналіз продуктивності системи до і після застосування методів оптимізації;
- Сформувати рекомендації щодо впровадження практик оптимізації у хмарних середовищах;
- Розробити прикладне рішення, яке продемонструє покращення стабільності, масштабованості та ефективності використання ресурсів на практиці.

Рисунок А.4 – Слайд 4

Методологія

Використані методи дослідження:

- Емпіричне моделювання навантаження
- Порівняльний аналіз
- Багатокритеріальна оцінка
- Аналіз наукової літератури
- Проектування програми для дослідження

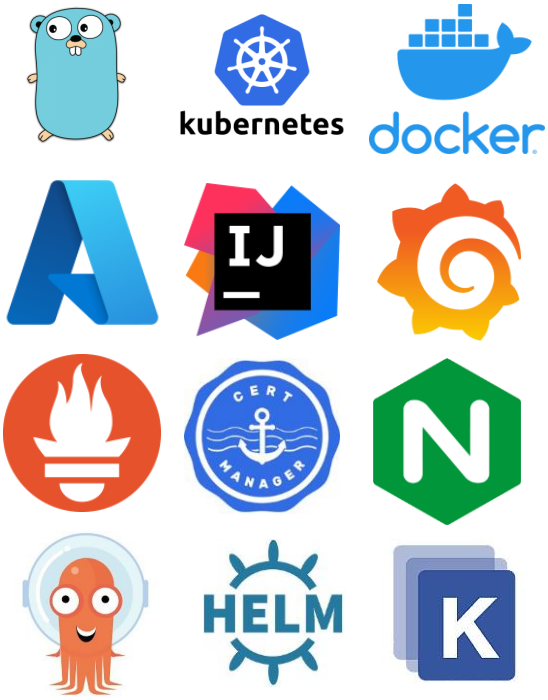


Рисунок А.5 – Слайд 5

Архітектура система для проведення експериментального дослідження

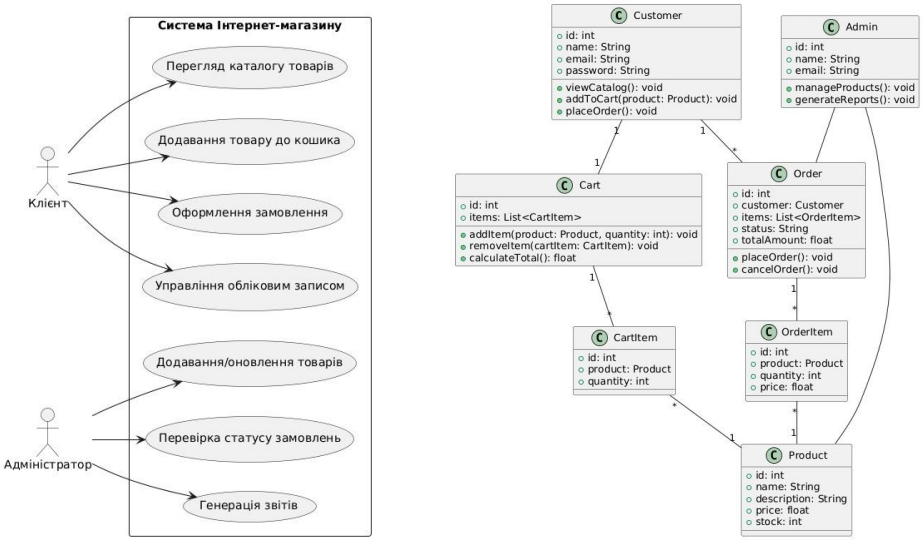


Рисунок А.6 – Слайд 6

Архітектура система для проведення експериментального дослідження

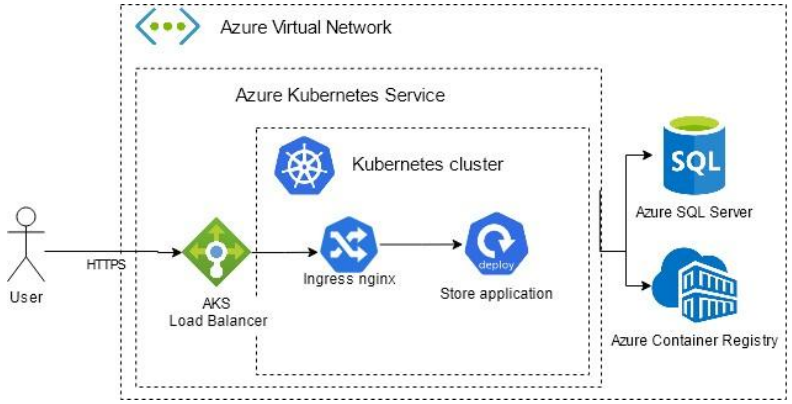


Рисунок А.7 – Слайд 7

Архітектура система для проведення експериментального дослідження

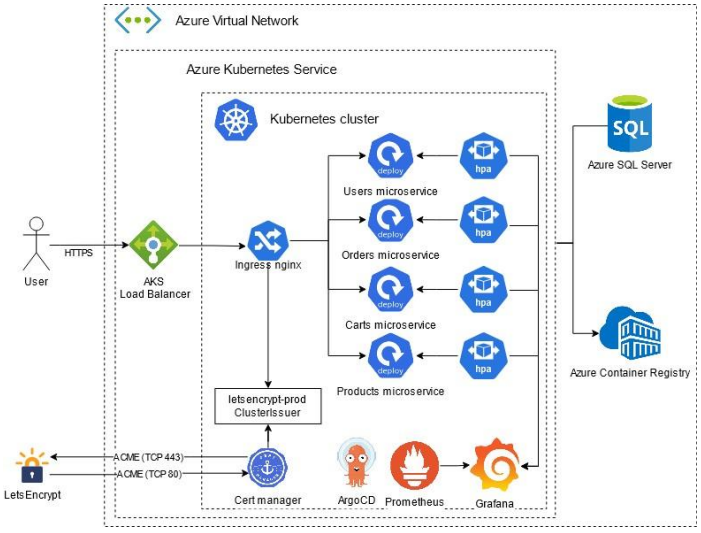


Рисунок А.8 – Слайд 8

Опис програмного забезпечення, що було використано у дослідженні

Опис розробки ПЗ:

1) Розробка монолітного додатку
2) Розробка HPA для монолітного додатку
3) Розробка VPA для монолітного додатку
4) Рефакторінг моноліту до формату мікросервісів

software engineering

Рисунок А.9 – Слайд 9

Зміст проведеного експерименту

Методи:
- Метод вагових коефіцієнтів для інтегральної оцінки моделей;
- Тестування продуктивності в середовищі Kubernetes на основі реального веб-додатку.

Вхідні дані:
- Веб-додаток, що обробляє HTTP-запити: /product, /order, /cart, /user
- Генерація навантаження за допомогою інструменту hey

Критерії:
- Масштабованість
- Надійність
- Гнучкість підтримки
- Вартість ресурсів
- Час розгортання

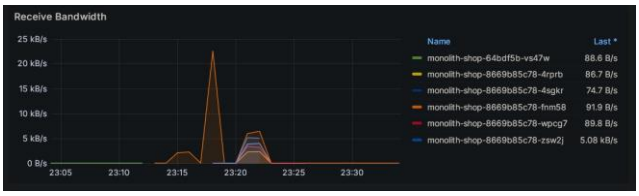
Послідовність:
1. Проведення навантажувального тесту
2. Збір метрик через Prometheus та Grafana
3. Розгортання додатку у варіантах: моноліт → HPA → VPA → мікросервіси
4. Побудова таблиць і порівняння результатів

Вимірювання:
- Метрики CPU, RAM, network bandwidth
- Кількість активних реплік під час стресу
- Час реакції та стабільність системи

software engineering

Рисунок А.10 – Слайд 10

Результати експерименту



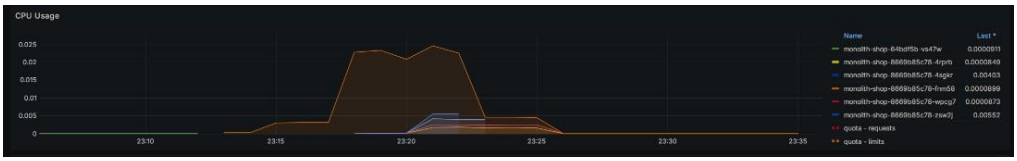
Horizontal Pod Autoscaler

Vertical Pod Autoscaler

Microservices

Рисунок А.11 – Слайд 11

Результати експерименту



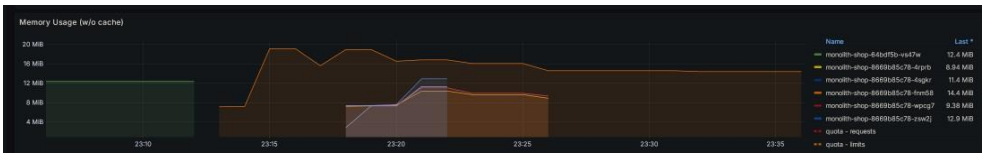
Horizontal Pod Autoscaler

Vertical Pod Autoscaler

Microservices

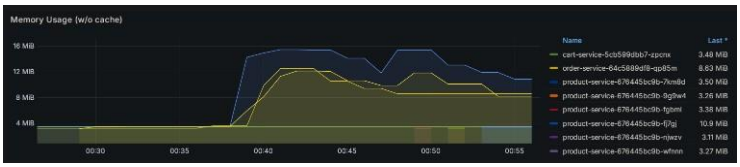
Рисунок А.12 – Слайд 12

Результати експерименту



Horizontal Pod Autoscaler

Vertical Pod Autoscaler



Microservices

Рисунок А.13 – Слайд 13

Аналіз отриманих результатів

Критерій	Ваговий коефіцієнт
Масштабованість	0.35
Надійність	0.25
Гнучкість підтримки	0.15
Вартість ресурсів	0.15
Час розгортання	0.10

Критерій	HPA	VPA	Microservices
Масштабованість	0.350	0.228	0.350
Надійність	0.200	0.213	0.238
Гнучкість підтримки	0.105	0.090	0.135
Вартість ресурсів	0.113	0.135	0.128
Час розгортання	0.085	0.080	0.075
Разом	0.853	0.746	**0.926**

Висновок:
Мікросервісна архітектура є найбільш збалансованим і перспективним методом, що поєднує масштабованість, надійність і підтримуваність.

Рисунок А.14 – Слайд 14

Публікація результатів

ХХІХ МІЖНАРОДНИЙ МОЛОДІЖНИЙ ФОРУМ «РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У ХХІ СТОЛІТТІ»

УДК 004.4:004.7

ДОСЛІДЖЕННЯ МЕТОДІВ ОПТИМІЗАЦІЇ ПРОДУКТИВНОСТІ ХМАРНИХ ВЕБ-ДОДАТКІВ КОНТЕЙНЕРИЗОВАНИХ У KUBERNETES

Ворона Д.О.
e-mail: dmytro.vorona1@nure.ua
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна

This work is devoted to the study of methods for optimizing the performance of cloud-based web applications containerized in Kubernetes. Modern cloud applications require high scalability, reliability, and efficiency, making optimization an essential aspect of development. During the research, existing optimization approaches were analyzed, monolithic web application was implemented and transformed into a microservices architecture. The study involved testing different scaling strategies, resource allocation techniques, and monitoring tools to measure the effectiveness of optimization methods.

Сучасні хмарні веб-додатки потребують високої продуктивності, масштабованості та надійності. Kubernetes став основною платформою для оркестрації контейнеризованих додатків, але його ефективне використання вимагає оптимізації ресурсів. Саме тому актуальним є дослідження методів оптимізації продуктивності веб-додатків у Kubernetes.

Метою дослідження є аналіз існуючих методів оптимізації продуктивності хмарних веб-додатків у Kubernetes, а також розробка та впровадження підходів для підвищення ефективності використання ресурсів. Для досягнення цієї мети було проведено аналіз предметної галузі, огляд літератури та практичних рішень, розроблено та протестовано монолітний веб-додаток, який згодом було оптимізовано шляхом переходу до мікросервісної архітектури. Дослідження також включало використання Prometheus, Kube State Metrics та Cert Manager для моніторингу та безпеки.

Результати дослідження показали, що первинна монолітна архітектура додатка мала значні обмеження у масштабованості. Оптимізація через розбиття на мікросервіси (див. рис. 1) дозволила зменшити затримки, підвищити ефективність використання ресурсів та покращити надійність системи. Автоматизоване масштабування (HPA), балансування навантаження та моніторинг у режимі реального часу суттєво підвищили продуктивність веб-додатків.

Одним із ключових факторів успішної оптимізації є використання ефективних засобів моніторингу. Інтеграція Prometheus і Kube State Metrics дозволяє отримувати детальну інформацію про використання ресурсів та швидко реагувати на зміни в системі. Це дає змогу оперативно виявляти потенційні проблеми та оптимізувати конфігурацію сервісів без значного зниження продуктивності.

Рисунок 1 – Високорівнева архітектура мікросервісної інфраструктури

У сучасних веб-додатках контейнери широко використовуються для мікросервісної архітектури, де кожна функція або сервіс виконується в окремому контейнері. Це забезпечує гнучкість у масштабуванні та розподілі ресурсів, дозволяючи збільшувати або зменшувати кількість екземплярів конкретних мікросервісів залежно від навантаження. Оркестраційні платформи, такі як Kubernetes, надають можливості для автоматичного масштабування, балансування навантаження та управління відмовами, що робить контейнеризацію ефективною для створення високонавантажених веб-додатків.

Контейнеризація має свої виклики. Оптимальне управління ресурсами, наприклад CPU та пам'яттю, є критичним завданням, оскільки недостатнє виділення ресурсів може призвести до зниження продуктивності, тоді як надлишкове використання збільшує витрати. Важливим аспектом є також безпека контейнеризованих додатків, включаючи управління вразливостями у базових образах та налаштування правил доступу.

На основі отриманих результатів можна зробити висновок, що використання мікросервісної архітектури значно покращує продуктивність хмарних додатків. Це підтверджується вимірюваннями часу обробки запитів, середнього навантаження на процесор (CPU utilization) та затримки відповіді. Kubernetes надає ефективні інструменти для автоматизації, проте їх застосування вимагає глибокого налаштування та постійного моніторингу.

Список використаних джерел:

1. Kubernetes and Docker Load Balancing: State-of-the-Art Techniques and Challenges. URL: https://www.researchgate.net/publication/376593267_Kubernetes_and_Docker_Load_Balancing_State-of-the-Art_Techniques_and_Challenges (дата звернення: 04.03.2025).
2. Enhanced Visibility for Real-time Monitoring and Alerting in Kubernetes by Integrating Prometheus, Grafana, Loki, and Alerta. URL: https://www.researchgate.net/publication/381347060_Enhanced_Visibility_for_Real-time_Monitoring_and_Alerting_in_Kubernetes_by_Integrating_PrometheusPrometheus_Grafana_Loki_and_Alerta (дата звернення: 04.03.2025).
3. Kubernetes in Microservices. URL: https://www.researchgate.net/publication/365344228_Kubernetes_in_Microservices (дата звернення: 04.03.2025).

Рисунок А.15 – Слайд 15

Підсумки

Реалістичність та корисність отриманих результатів:

- Результати базуються на експериментальному тестуванні у справжньому середовищі Kubernetes;
- Усі методи оптимізації були реалізовані, протестовані та порівняні на прикладі веб-додатку;
- Результати мають практичну цінність — вони можуть бути безпосередньо використані розробниками та DevOps-фахівцями для покращення продуктивності хмарних систем.

Можливий розвиток досліджень:

- Використання сервісної сітки (Service Mesh) для глибшого аналізу трафіку та надійності;
- Впровадження AI/ML-алгоритмів для прогнозування навантаження та автоматичного налаштування ресурсів;
- Оптимізація на рівні мережевих затримок, кешування та баз даних для повного покриття всіх вузьких місць у продуктивності.

Рисунок А.16 – Слайд 16

ДОДАТОК Б
Апробація результатів роботи

УДК 004.4:004.7

ДОСЛІДЖЕННЯ МЕТОДІВ ОПТИМІЗАЦІЇ ПРОДУКТИВНОСТІ ХМАРНИХ ВЕБ-ДОДАТКІВ КОНТЕЙНЕРИЗОВАНИХ У KUBERNETES

Ворона Д.О.
e-mail: dmytro.vorona1@nure.ua
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна

This work is devoted to the study of methods for optimizing the performance of cloud-based web applications containerized in Kubernetes. Modern cloud applications require high scalability, reliability, and efficiency, making optimization an essential aspect of development. During the research, existing optimization approaches were analyzed, monolithic web application was implemented and transformed into a microservices architecture. The study involved testing different scaling strategies, resource allocation techniques, and monitoring tools to measure the effectiveness of optimization methods.

Сучасні хмарні веб-додатки потребують високої продуктивності, масштабованості та надійності. Kubernetes став основною платформою для оркестрації контейнеризованих додатків, але його ефективне використання вимагає оптимізації ресурсів. Саме тому актуальним є дослідження методів оптимізації продуктивності веб-додатків у Kubernetes.

Метою дослідження є аналіз існуючих методів оптимізації продуктивності хмарних веб-додатків у Kubernetes, а також розробка та впровадження підходів для підвищення ефективності використання ресурсів. Для досягнення цієї мети було проведено аналіз предметної галузі, огляд літератури та практичних рішень, розроблено та протестовано монолітний веб-додаток, який згодом було оптимізовано шляхом переходу до мікросервісної архітектури. Дослідження також включало використання Prometheus, Kube State Metrics та Cert Manager для моніторингу та безпеки.

Результати дослідження показали, що первинна монолітна архітектура додатка мала значні обмеження у масштабованості. Оптимізація через розбиття на мікросервіси (див. рис. 1) дозволила зменшити затримки, підвищити ефективність використання ресурсів та покращити надійність системи. Автоматизоване масштабування (HPA), балансування навантаження та моніторинг у режимі реального часу суттєво підвищили продуктивність веб-додатків.

Одним із ключових факторів успішної оптимізації є використання ефективних засобів моніторингу. Інтеграція Prometheus і Kube State Metrics дозволяє отримувати детальну інформацію про використання ресурсів та швидко реагувати на зміни в системі. Це дає змогу оперативно виявляти потенційні проблеми та оптимізувати конфігурацію сервісів без значного зниження продуктивності.

Рисунок Б.1 – Текст апробації 1

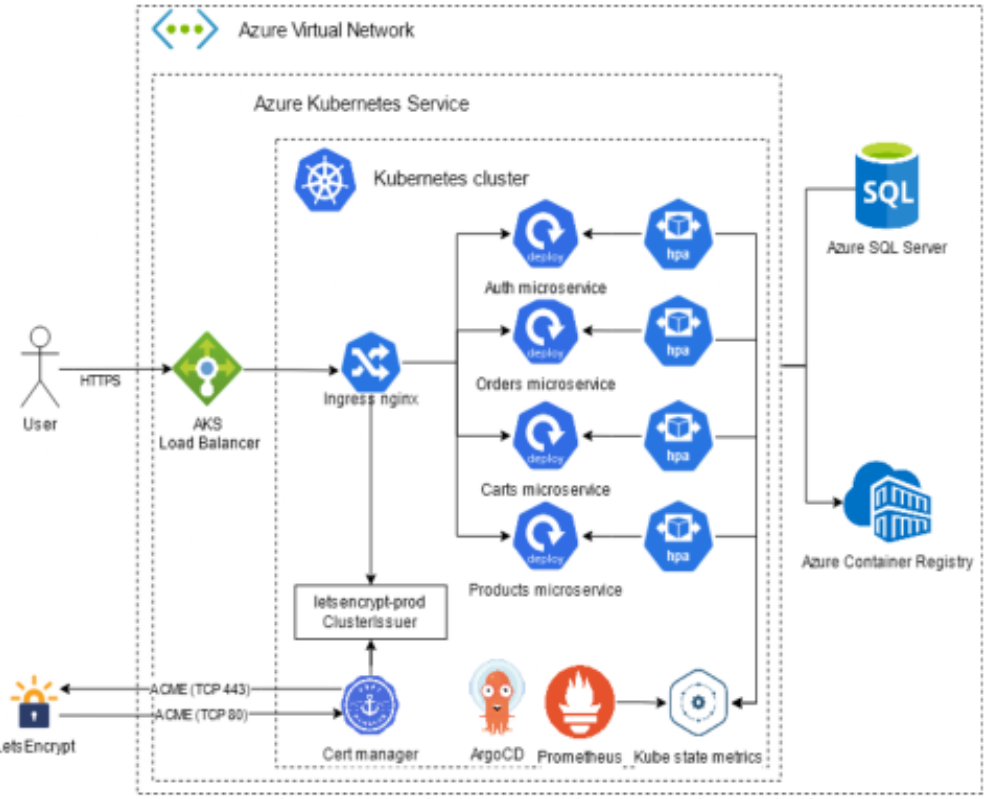


Рисунок 1 – Високорівнева архітектура мікросервісної інфраструктури

У сучасних веб-додатках контейнери широко використовуються для мікросервісної архітектури, де кожна функція або сервіс виконується в окремому контейнері. Це забезпечує гнучкість у масштабуванні та розподілі ресурсів, дозволяючи збільшувати або зменшувати кількість екземплярів конкретних мікросервісів залежно від навантаження. Оркестраційні платформи, такі як Kubernetes, надають можливості для автоматичного масштабування, балансування навантаження та управління відмовами, що робить контейнеризацію ефективною для створення високонавантажених веб-додатків.

Контейнеризація має свої виклики. Оптимальне управління ресурсами, наприклад CPU та пам'яттю, є критичним завданням, оскільки недостатнє виділення ресурсів може призвести до зниження продуктивності, тоді як надлишкове використання збільшує витрати. Важливим аспектом є також безпека контейнеризованих додатків, включаючи управління вразливостями у базових образах та налаштування правил доступу.

На основі отриманих результатів можна зробити висновок, що використання мікросервісної архітектури значно покращує продуктивність хмарних додатків. Це підтверджується вимірюваннями часу обробки запитів, середнього навантаження на процесор (CPU utilization) та

Рисунок Б.2 – Текст апробації 2

затримки відповіді. Kubernetes надає ефективні інструменти для автоматизації, проте їх застосування вимагає глибокого налаштування та постійного моніторингу.

Список використаних джерел:

1. Kubernetes and Docker Load Balancing: State-of-the-Art Techniques and Challenges. URL: https://www.researchgate.net/publication/376593267_Kubernetes_and_Docker_Load_Balancing_State-of-the-Art_Techniques_and_Challenges (дата звернення: 04.03.2025).

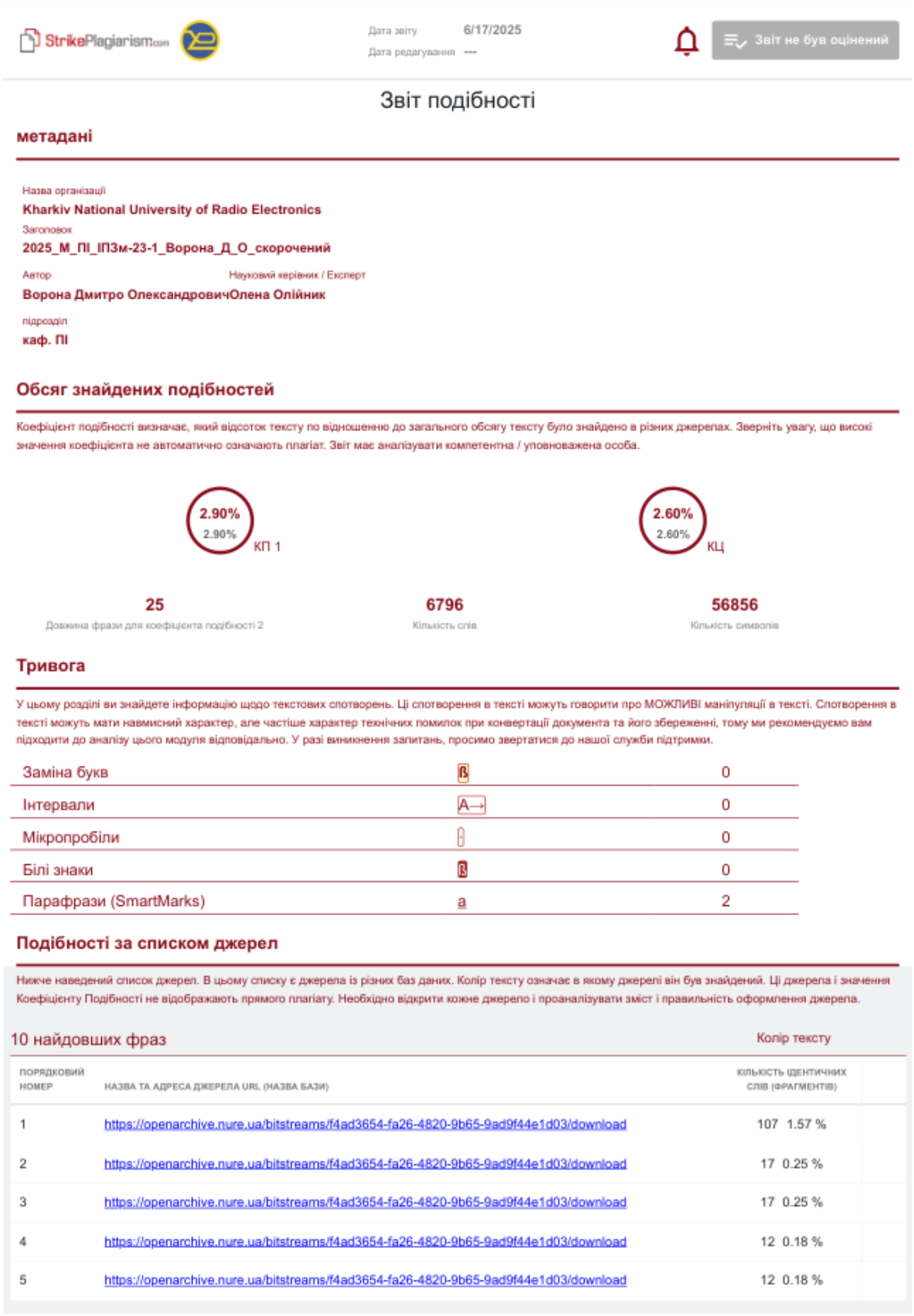
2. Enhanced Visibility for Real-time Monitoring and Alerting in Kubernetes by Integrating Prometheus, Grafana, Loki, and Alerta. URL: https://www.researchgate.net/publication/381347090_Enhanced_Visibility_for_Real-time_Monitoring_and_Alerting_in_Kubernetes_by_Integrating_PrometheusPrometheus_Grafana_Loki_and_Alerta (дата звернення: 04.03.2025).

3. Kubernetes in Microservices. URL: https://www.researchgate.net/publication/365344228_Kubernetes_in_Microservices (дата звернення: 04.03.2025).

Рисунок Б.3 – Текст апробації 3

ДОДАТОК В

Звіт з результатами перевірки на унікальність тексту в базі ХНУРЕ



StrikePlagiarism.com

Дата звіту 6/17/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_М_ПІ_ІПЗм-23-1_Ворона_Д_О_скорочений
Автор
Ворона Дмитро Олександрович
Науковий керівник / Експерт
Олена Олійник
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.

2.90% 2.90% КП 1

2.60% 2.60% КЦ

25 Довжина фрази для коефіцієнта подібності 2

6796 Кількість слів

56856 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		0
Інтервали		0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)	a	2

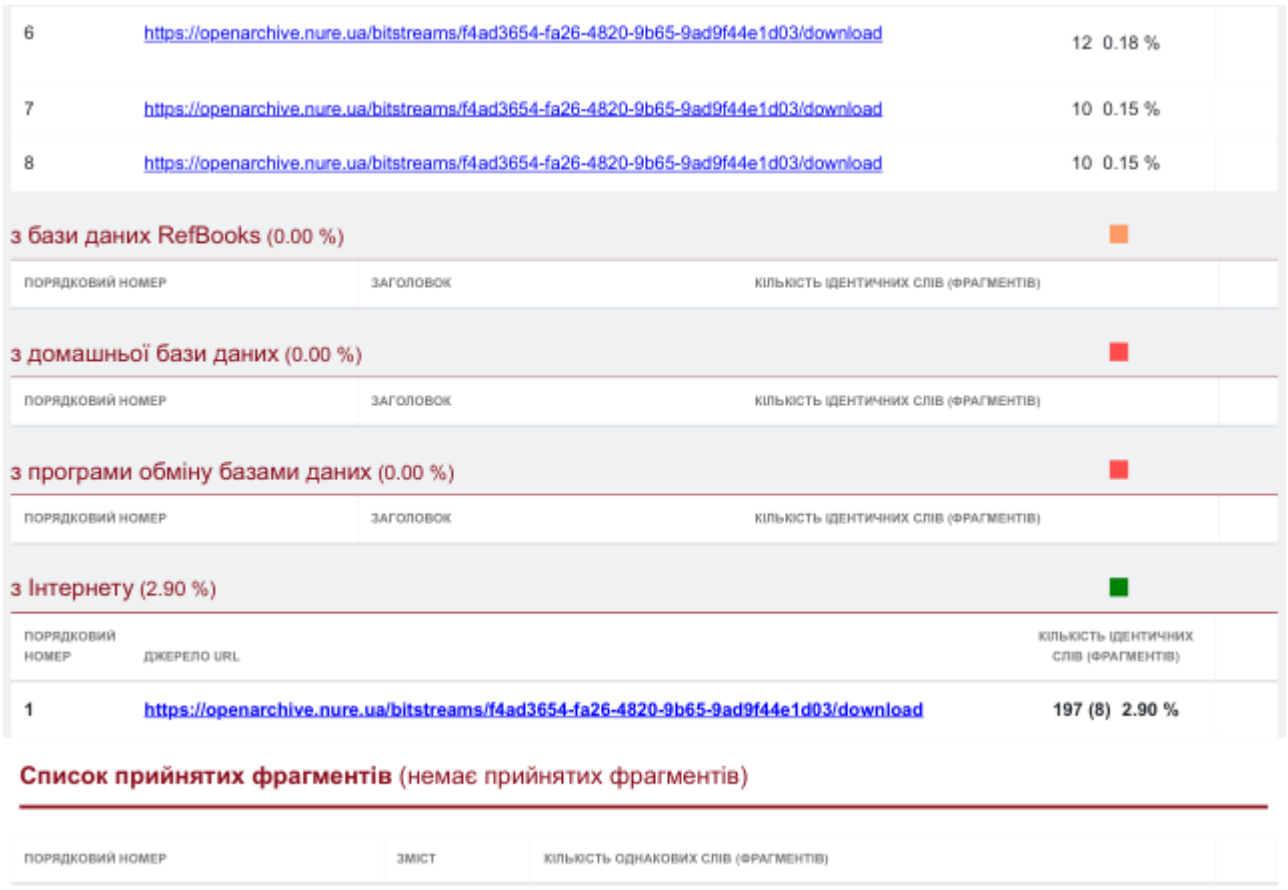
Подібності за списком джерел

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	https://openarchive.nure.ua/bitstreams/f4ad3654-fa26-4820-9b65-9ad9f44e1d03/download	107 1.57 %
2	https://openarchive.nure.ua/bitstreams/f4ad3654-fa26-4820-9b65-9ad9f44e1d03/download	17 0.25 %
3	https://openarchive.nure.ua/bitstreams/f4ad3654-fa26-4820-9b65-9ad9f44e1d03/download	17 0.25 %
4	https://openarchive.nure.ua/bitstreams/f4ad3654-fa26-4820-9b65-9ad9f44e1d03/download	12 0.18 %
5	https://openarchive.nure.ua/bitstreams/f4ad3654-fa26-4820-9b65-9ad9f44e1d03/download	12 0.18 %

Рисунок В.1 – Результат перевірки 1



6	https://openarchive.nure.ua/bitstreams/f4ad3654-fa26-4820-9b65-9ad9f44e1d03/download	12	0.18 %
7	https://openarchive.nure.ua/bitstreams/f4ad3654-fa26-4820-9b65-9ad9f44e1d03/download	10	0.15 %
8	https://openarchive.nure.ua/bitstreams/f4ad3654-fa26-4820-9b65-9ad9f44e1d03/download	10	0.15 %

з бази даних RefBooks (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з домашньої бази даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з програми обміну базами даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з Інтернету (2.90 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://openarchive.nure.ua/bitstreams/f4ad3654-fa26-4820-9b65-9ad9f44e1d03/download	197 (8) 2.90 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

Рисунок В.2 – Результат перевірки 2

ДОДАТОК Г

Експертний висновок результатів перевірки кваліфікаційної роботи

Експертний висновок результатів перевірки кваліфікаційної роботи

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

Дмитро ВОРОНА

(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	
	7.5 Рисунки	
	7.6 Таблиці	
	7.7 Переліки	
7.7.2	Якщо подають переліки одного рівня підпорядкованості, на які у звіті немає посилань, то перед кожним із переліків ставлять знак «тире». Якщо у звіті є посилання на переліки, підпорядкованість позначають малими літерами української абетки, далі — арабськими цифрами, далі — через знаки «тире». Після цифри або літери певної позиції переліку ставлять круглу дужку.	18
	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	

Експерт _____ (підпис) Вадим НЕЧВОЛОД (прізвище, ініціали)

Робота з перевірки оформлення пояснювальної записки кваліфікаційної роботи на нормоконтроль виконана у програмі Word Microsoft 365. Версія 2504 (збірка 18730.20220)

20.06.2025

Рисунок Г.1 - Результат