ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
(повне найменування вищого навчального закладу)
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
(повне найменування інституту, назва факультету (відділення))
КАФЕДРА ПРОГРАМНИХ ЗАСОБІВ І ТЕХНОЛОГІЙ
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до кваліфікаційної роботи магістра
(освітній рівень)

на тему: «Розробка мікросервісної архітектури веб-додатків на Node.js із застосуванням хмарних технологій та контейнеризації»

Виконав: студент групи 6ПР2
спеціальності
121 - «Інженерія програмного забезпечення»
(шифр і назва спеціальності)

Спринчан Андрій Костянтинович
(прізвище та ініціали)

Керівник д.т.н, проф. Жарікова М. В.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Хмельницький – 2025

Херсонський національний технічний університет
(повне найменування закладу вищої освіти)

Факультет, відділення Інформаційних технологій та дизайну
Кафедра Програмних засобів і технологій
Освітній рівень магістр
Спеціальність 121 – Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри
Програмних засобів і технологій
к.т.н. доц. Огнєва О. Є.
“___” _____ 2025 р.

З А В Д А Н Н Я
НА ВИПУСКНУ РОБОТУ СТУДЕНТУ

Спринчану Андрію Костянтиновичу
(прізвище, ім’я, по батькові)

Тема роботи «Розробка мікросервісної архітектури веб-додатків на Node.js із застосуванням хмарних технологій та контейнеризації»

керівник роботи д.т.н, проф. Жарікова М. В.,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від .. . .. . 2025 р. № -

1. Строк подання студентом роботи _____
2. Вихідні дані до роботи літературні та періодичні джерела, матеріали преддипломної практики, технічна документація інструментарію
3. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
   a Дослідження та аналіз предметної області
   b Аналіз вимог та розробка проектних специфікацій
   c Проектування програмного продукту
   d Розробка, тестування та впровадження програмного продукту
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
   _____
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 15.09.2025

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання роботи	Термін виконання етапів роботи	Примітки
1.	Отримання завдання	15.09.2025	Виконано
2.	Підбір літератури	28.09.2025	Виконано
3.	Аналіз предметної області	03.10.2024	Виконано
4.	Розробка та обґрунтування завдання	10.10.2025	Виконано
5.	Розробка концептуальної моделі	21.10.2025	Виконано
6.	Розробка алгоритму	25.10.2025	Виконано
7.	Проектування програми	01.11.2025	Виконано
8.	Розробка інтерфейсу програми	12.11.2025	Виконано
9.	Тестування програми	23.11.2025	Виконано
10.	Оформлення пояснювальної записки	25.11.2025	Виконано
11.	Захист кваліфікаційної роботи	22.12.2025	Виконано

Студент _____ Спринчан А. К.
(підпис) (прізвище та ініціали)

Керівник роботи _____ д.т.н, проф. Жарікова М. В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Кваліфікаційна робота магістра складається з 124 сторінок, 19 рисунків, 12 таблиць, 16 блок-схем та 29 джерел.

Мета роботи – розробити концепцію, архітектуру та програмну реалізацію прототипу хмарного веб-додатку на основі мікросервісної архітектури Node.js із використанням контейнеризації, оркестрації та хмарної інфраструктури, а також дослідити ефективність такого підходу для масштабованих та відмовостійких систем.

Об'єкт дослідження – методи, моделі та архітектурні підходи до проєктування та розробки веб-додатків у розподіленому хмарному середовищі.

Предмет дослідження – мікросервісна архітектура веб-додатку, побудована на Node.js з використанням Docker-контейнеризації, Kubernetes-оркестрації, брокера повідомлень RabbitMQ, API-шлюзу Kong та хмарних сервісів Google Cloud Platform

Методи дослідження – аналіз та узагальнення сучасних архітектурних підходів до побудови розподілених систем; методи моделювання мікросервісної інфраструктури; практичні методи контейнеризації та оркестрації (Docker, Kubernetes); експериментальні дослідження продуктивності й масштабованості сервісів; використання інструментів моніторингу та логування (Prometheus, Grafana, ELK-stack).

Результат роботи:

- розроблено архітектурну модель хмарного веб-додатку на основі мікросервісного підходу;

- створено набір функціональних мікросервісів на Node.js із використанням NestJS;

- реалізовано контейнеризацію сервісів у Docker та організовано їх розгортання в Kubernetes;

- налаштовано API-Gateway для управління зовнішнім трафіком і безпекою;

- забезпечено взаємодію сервісів через RabbitMQ та REST/gRPC-інтерфейси;

- виконано експериментальне дослідження відмовостійкості, масштабованості й продуктивності системи в хмарному середовищі;

- сформовано висновки щодо ефективності запропонованої архітектури та можливостей її подальшого розвитку.

Новизна роботи:

- запропоновано комплексну архітектурну модель хмарної мікросервісної системи, що поєднує контейнеризацію, оркестрацію та автоматичне масштабування;

- розроблено оригінальний прототип розподіленого веб-додатку з використанням Node.js та Kubernetes як базової платформи хмарної інфраструктури;

- досліджено поведінку мікросервісів під навантаженням у динамічному середовищі, включаючи авто-масштабування та відмовостійкість;

- показано ефективність використання брокера повідомлень RabbitMQ для забезпечення асинхронної взаємодії між сервісами у складних хмарних системах.

Ключові слова: *мікросервісна архітектура, Node.js, Docker, Kubernetes, хмарні технології, GCP, API-Gateway, RabbitMQ, контейнеризація, масштабування, оркестрація, веб-додаток.*

АНОТАЦІЯ

Магістерська кваліфікаційна робота присвячена дослідженню, проектуванню та розробці архітектури мікросервісів для веб-застосунків з використанням Node.js, контейнеризації Docker, оркестрування Kubernetes та хмарних сервісів Google Cloud Platform. У роботі обґрунтовується актуальність мікросервісного підходу до побудови масштабованих, гнучких та стійких до відмови інформаційних систем у сучасному мережному середовищі.

Наведено порівняльний аналіз монолітної та мікросервісної архітектур, виявлено ключові принципи побудови розподілених хмарних систем та особливості управління їх життєвим циклом. В роботі розроблена архітектура унікального веб-додатку, побудованого з використанням моделі мікросервісів, та реалізована з використанням Node.js, включаючи брокер повідомлень RabbitMQ, шлюз Kong API, сховища даних PostgreSQL та MongoDB, а також інструменти моніторингу та журналування..

Особлива увага приділяється контейнеризації програмних компонентів, автоматизації розгортання, відмовостійкості, горизонтальній масштабованості та процесам CI/CD. Проведено експериментальне дослідження ефективності роботи системи у хмарному середовищі, яке дозволило виявити переваги використання Kubernetes для керування мікросервісною архітектурою.

Результатом роботи є побудова комплексної концепції, архітектури та програмної реалізації розподіленого хмарного веб-додатку, а також практичних рекомендацій щодо впровадження мікросервісних підходів у розробці сучасних інформаційних систем.

ABSTRACT

The master's thesis is devoted to the research, design, and development of a microservice architecture for web applications using Node.js, Docker containerization, Kubernetes orchestration, and Google Cloud Platform services. The study substantiates the relevance of the microservice approach for building scalable, flexible, and fault-tolerant information systems in modern network environments.

A comparative analysis of monolithic and microservice architectures is presented, key principles of constructing distributed cloud systems are identified, and the specific aspects of managing their life cycle are examined. The thesis introduces the architecture of an original web application built using the microservice model and implemented with Node.js, including the RabbitMQ message broker, the Kong API gateway, PostgreSQL and MongoDB data storage systems, as well as monitoring and logging tools.

Special attention is given to the containerization of software components, deployment automation, fault tolerance, horizontal scalability, and CI/CD processes. An experimental evaluation of the system’s performance in a cloud environment is conducted, demonstrating the advantages of using Kubernetes for managing a microservice architecture.

The results of the work include the development of a comprehensive concept, architecture, and software implementation of a distributed cloud-based web application, along with practical recommendations for applying microservice approaches in the development of modern information systems

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API – прикладний програмний інтерфейс, що визначає набір методів і правил, які дозволяють різним програмним компонентам взаємодіяти між собою. Використовується як основний механізм обміну даними між мікросервісами.

CI/CD – процеси безперервної інтеграції та безперервного розгортання, що забезпечують автоматичну збірку, тестування, перевірку та доставку змін у програмному забезпеченні. Дозволяють пришвидшити розробку та підвищити стабільність релізів.

CPU – центральний процесор, основний обчислювальний модуль, що виконує інструкції програм. У контексті Kubernetes використовується як ключова метрика для автоматичного масштабування навантаження.

CRUD – базовий набір операцій над даними: створення, читання, оновлення та видалення. Застосовується при побудові моделей даних і реалізації API у мікросервісній архітектурі.

DB – база даних, структуроване сховище інформації, що використовується окремими мікросервісами. Кожен сервіс у CloudMicroCare має власну незалежну DB згідно з принципами розподілених систем.

DNS – система доменних імен, що перетворює текстові адреси у мережеві IP. У Kubernetes використовується для внутрішнього сервіс-дискавері між подами та сервісами.

GCP – хмарна платформа Google Cloud Platform, що надає інфраструктурні та платформенні сервіси для розгортання застосунків. Використовується як середовище виконання CloudMicroCare.

GKE – Google Kubernetes Engine, керований сервіс Kubernetes у GCP, що забезпечує автоматичне масштабування, оновлення та контроль за роботою контейнерних застосунків.

GCR – Google Container Registry, сховище Docker-образів, яке використовується CI/CD-процесом для зберігання та доставки контейнерів до кластеру GKE.

HPA – Horizontal Pod Autoscaler, механізм автоматичного горизонтального масштабування подів у Kubernetes. Реагує на зміну навантаження, збільшуючи або зменшуючи кількість реплік сервісу.

HTTP – протокол взаємодії клієнт–сервер, що застосовується для передачі запитів між фронтендом, API та іншими компонентами. Є базовим транспортним механізмом у REST-архітектурі.

HTTPS – захищена версія HTTP, що використовує TLS для шифрування даних. Забезпечує конфіденційність та цілісність комунікацій між сервісами.

JSON – текстовий формат обміну даними, що використовується в API, повідомленнях та логах. Відзначається простотою, читабельністю та широкою підтримкою в інструментах JavaScript.

JWT – токен авторизації у форматі JSON Web Token, який використовується для підтвердження автентичності користувачів і сервісів. Забезпечує безпечний доступ до API.

K8s – скорочена назва Kubernetes, системи оркестрації контейнерів, що керує життєвим циклом мікросервісів, їх масштабуванням, оновленням і відмовостійкістю.

KMS – сервіс керування криптографічними ключами, що використовується для захисту секретів і шифрування даних. У GCP інтегрується із Kubernetes Secrets.

LTS – тривала підтримка (Long-Term Support), що гарантує стабільність і безпеку версії програмного забезпечення. Важливо при виборі Node.js або баз даних.

MQ – черга повідомлень, механізм асинхронної взаємодії між сервісами. У CloudMicroCare використовується RabbitMQ для передачі подій та зниження зв’язності мікросервісів.

PVC – запит на постійний том (Persistent Volume Claim), що дозволяє подам Kubernetes використовувати довготривале сховище. Застосовується для зберігання даних RabbitMQ, БД та логів.

RAM – оперативна пам’ять, яка використовується для виконання застосунків і контейнерів. Регулюється через Kubernetes Resources та HPA.

RBAC – модель контролю доступу на основі ролей, яка визначає, які дії дозволені користувачам чи сервісам. Забезпечує високий рівень безпеки в Kubernetes та API.

REST – архітектурний стиль побудови веб-сервісів, що використовує HTTP-методи та уніфіковані URI. Є основою API CloudMicroCare.

SLA – угода про рівень сервісу, що визначає гарантовані показники доступності й швидкодії. Враховується при проєктуванні хмарних систем.

TLS – протокол захисту даних, що забезпечує шифрування мережевих комунікацій. Використовується для безпечної взаємодії між сервісами.

URL – уніфікований локатор ресурсу, що визначає адресу сторінок, API та сервісів у мережі.

VM – віртуальна машина, логічний обчислювальний ресурс у хмарі. Може бути базовим рівнем інфраструктури під Kubernetes-кластери.

YAML – формат структурованих конфігураційних файлів, що використовується в Kubernetes manifests, Helm charts та CI/CD-пайплайнах.

ВСТУП

У сучасних умовах швидкого розвитку цифрових технологій, веб-додатки стають ключовими компонентами інформаційної інфраструктури державних, комерційних і наукових організацій. Сильна конкуренція на ринку програмного забезпечення, постійне зростання обсягів даних і підвищення вимог до доступності сервісів вимагають архітектур, що забезпечують масштабованість, надійність, гнучкість і відмовостійкість інформаційних систем. Традиційні монолітні моделі все частіше не справляються з новими викликами, оскільки ускладнюють модернізацію, унеможливлюють швидке оновлення і створюють значні ризики при збільшенні навантаження

Одним з найбільш ефективних підходів до вирішення цих проблем є мікросервісна архітектура, в якій складний веб-додаток розділяється на незалежні модулі, кожен з яких виконує окрему функцію. Такий підхід значно полегшує розробку, спрощує масштабування, підвищує надійність і дозволяє незалежно розгортати компоненти системи. Однак для повноцінного використання переваг мікросервісної моделі необхідні сучасні технології контейнеризації, координації та хмарної інфраструктури.

Широке впровадження технологій Docker та Kubernetes стало важливим кроком у розвитку мікросервісних систем. Docker забезпечує стандартизоване та ізольоване середовище виконання програмного забезпечення, а Kubernetes — механізми автоматичного масштабування, самовідновлення, балансування навантаження та централізованого управління конфігураціями. У комплексі з хмарними сервісами, такими як Google Cloud Platform, вони створюють основу для формування високонадійних та ефективних веб-платформ нового покоління.

У межах цієї магістерської роботи розглядається побудова веб-додатку на основі мікросервісної архітектури з використанням технологій Node.js, Docker, Kubernetes та хмарних сервісів. Розроблена система демонструє можливість реалізації масштабованого, розподіленого та відмовостійкого застосунку, який

може адаптуватися до змін навантаження, розширюватися функціонально та забезпечувати стабільну роботу в умовах динамічного середовища.

Мета і задачі дослідження. Метою дослідження є розробка та впровадження мікросервісної архітектури веб-додатку на основі Node.js із використанням хмарних технологій та контейнеризації, а також аналіз ефективності такого підходу для побудови сучасних високонавантажених систем.

Для досягнення поставленої мети необхідно виконати такі основні завдання:

- проаналізувати сучасні архітектурні підходи до побудови веб-застосунків та порівняти монолітні та мікросервісні моделі;
- дослідити технології контейнеризації та оркестрації, зокрема Docker та Kubernetes;
- розробити архітектурну модель веб-додатку за принципами мікросервісної архітектури;
- створити набір функціональних мікросервісів на основі Node.js;
- забезпечити інтеграцію сервісів із брокером повідомлень RabbitMQ та API-шлюзом Kong;
- реалізувати розгортання системи в хмарному середовищі Google Cloud Platform;
- розробити систему моніторингу, логування та CI/CD-процеси;
- провести експериментальне дослідження продуктивності, стійкості та масштабованості розробленої системи.

Об’єкт дослідження. Процеси, методи та технології побудови розподілених веб-додатків у хмарному середовищі.

Предмет дослідження. Мікросервісна архітектура веб-додатку, реалізована на Node.js з використанням контейнеризації Docker, оркестрації Kubernetes, брокера повідомлень RabbitMQ, API-шлюзу Kong та хмарних сервісів GCP.

Наукова новизна одержаних результатів. Наукова новизна роботи полягає в комплексному підході до побудови хмарної мікросервісної системи та в експериментальному дослідженні ефективності її масштабування й відмовостійкості.

Зокрема:

- розроблено узагальнену архітектурну модель мікросервісної системи з використанням технологій контейнеризації та оркестрації;
- проаналізовано поведінку мікросервісів за різних режимів навантаження у хмарному середовищі;
- запропоновано оптимальні підходи до організації міжсервісної комунікації в умовах динамічного масштабування;
- обґрунтовано переваги використання API-шлюзу Kong для забезпечення безпеки та управління зовнішнім трафіком у мікросервісних системах;
- визначено закономірності оптимізації роботи системи за допомогою CI/CD, моніторингу та автоматизованих механізмів керування інфраструктурою.

Практичне значення одержаних результатів. Практичне значення полягає у створенні працездатного прототипу хмарної мікросервісної системи, придатного для використання як основи для реальних бізнес-рішень. Результати роботи дозволяють:

- впроваджувати масштабовані веб-додатки з високим рівнем надійності та доступності;
- скоротити витрати на підтримку та оновлення програмного забезпечення;
- підвищити стійкість системи до збоїв за рахунок контейнеризації та оркестрації;
- автоматизувати процеси розгортання та супроводу застосунків;
- удосконалити процеси інтеграції сервісів у хмарну інфраструктуру.

Теоретичне значення одержаних результатів. Теоретичне значення роботи полягає у розвитку методологічних засад побудови мікросервісних систем та узагальненні практик використання хмарних технологій при створенні розподілених веб-додатків.

Отримані результати можуть слугувати основою для:

- подальших наукових досліджень у сфері мікросервісної архітектури;
- удосконалення моделей взаємодії між сервісами в умовах динамічної хмарної інфраструктури;
- розробки навчальних матеріалів та методичних рекомендацій з побудови сучасних веб-додатків.