Shop

Специфікація вимог до програмного забезпечення

1.0

15.06.2025

Ворона Дмитро

1. **Історія версій**

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Опис** | **Автор** | **Коментар** |
| **15.06.2025** | **Версія 1.0** | **Ворона Дмитро** | **Створення документу** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**ЗМІСТ**

[**1.** **Вступ** 4](#_Toc201540817)

[**2. Загальний опис** 5](#_Toc201540818)

[**2.1. Опис продукту** 5](#_Toc201540819)

[**2.2. Функції продукту** 5](#_Toc201540820)

[**2.3. Характеристики користувачів** 5](#_Toc201540821)

[2.4. Загальні обмеження 6](#_Toc201540822)

[**3. Вимоги до системи** 7](#_Toc201540823)

[**3.1. Функціональні вимоги** 7](#_Toc201540824)

[**3.2. Нефункціональні вимоги** 7](#_Toc201540825)

[**3.3. Вимоги до середовища** 8](#_Toc201540826)

# **Вступ**

У сучасних умовах розвитку інформаційних технологій хмарні обчислення стали невід'ємною частиною побудови масштабованих і доступних веб-додатків. Зростання обсягів даних, збільшення кількості користувачів та вимоги до безперервної доступності зумовлюють необхідність розробки ефективних підходів до оптимізації продуктивності таких систем. Особливу роль у цьому процесі відіграє Kubernetes — платформа оркестрації контейнерів, яка забезпечує автоматизоване розгортання, масштабування та моніторинг сервісів у хмарному середовищі.

Об’єктом дослідження є хмарні веб-додатки, які працюють у контейнеризованому середовищі на базі Kubernetes. Предметом дослідження є методи оптимізації їх продуктивності, включаючи масштабування, профілювання, моніторинг і архітектурні підходи до організації додатків.

Метою кваліфікаційної роботи є розробка, експериментальне впровадження та оцінка ефективності методів оптимізації хмарних веб-додатків, контейнеризованих у Kubernetes. Для досягнення цієї мети було поставлено ряд завдань: аналіз існуючих підходів до оптимізації, реалізація прототипу монолітного додатку, його поступова трансформація у мікросервісну архітектуру, впровадження горизонтального та вертикального автоскейлінгу, а також налаштування систем моніторингу на основі Grafana, Prometheus та інших інструментів.

Наукова новизна роботи полягає у практичному порівнянні методів масштабування та архітектурних рішень у контексті їх впливу на продуктивність веб-додатків. Уперше було реалізовано єдину тестову платформу, яка дає змогу автоматизовано здійснювати навантажувальне тестування різних конфігурацій та аналізувати результати за допомогою метрик продуктивності.

Практичне значення полягає в тому, що результати дослідження можуть бути використані при проєктуванні реальних веб-систем для забезпечення стабільності та продуктивності у хмарному середовищі, особливо у проєктах, що використовують Kubernetes як основну платформу для розгортання.

# **2. Загальний опис**

## **2.1. Опис продукту**

Програмний продукт є хмарним веб-додатком, побудованим за мікросервісною архітектурою та оркестрованим за допомогою платформи Kubernetes. Система призначена для демонстрації та дослідження методів оптимізації продуктивності контейнеризованих веб-додатків. Вона включає модулі для управління товарами, користувачами, кошиком і замовленнями, кожен з яких реалізований як окремий мікросервіс. Додаток підтримує горизонтальне та вертикальне масштабування через HPA та VPA, а також централізований моніторинг за допомогою Prometheus і Grafana.

## **2.2. Функції продукту**

Основні функції продукту включають:

* Робота в режимі моноліту або мікросервісів (перемикається через конфігурацію).
* CRUD-операції над користувачами, товарами, замовленнями та кошиками.
* Автоматичне масштабування мікросервісів залежно від навантаження.
* Збір та візуалізація метрик продуктивності системи.
* Проведення навантажувального тестування за допомогою hey.

## **2.3. Характеристики користувачів**

Продукт орієнтований на наступні категорії користувачів:

* **DevOps-інженери** — для тестування оптимізації масштабування в Kubernetes.
* **Розробники** — для демонстрації архітектурних рішень (перехід від моноліту до мікросервісів).
* **Дослідники** — для збору метрик продуктивності у хмарних середовищах.
* **Адміністратори систем** — для контролю навантаження, стабільності та розгортання.

## **2.4. Загальні обмеження**

* Система вимагає наявності Kubernetes-кластеру та правильно налаштованих ingress-контролерів.
* Сертифікати TLS мають бути створені і оновлювані через cert-manager (наприклад, через Let's Encrypt).
* Продуктивність та надійність залежать від обчислювальних ресурсів, виділених на кластер.
* Тестування проводиться в умовах Azure Kubernetes Service (AKS), що може впливати на узагальнення результатів.

**3. Вимоги до системи**

**3.1. Функціональні вимоги**

1. **Реєстрація користувача**  
   Система повинна забезпечувати можливість реєстрації нового користувача із збереженням його даних у базі даних.
2. **Аутентифікація та авторизація**  
   Система повинна надавати механізми входу та виходу користувача, а також обмеження доступу до частини функціоналу.
3. **Управління товарами**  
   Повинен бути реалізований CRUD-функціонал (створення, перегляд, оновлення, видалення) для товарів.
4. **Оформлення замовлень**  
   Користувач повинен мати змогу оформити замовлення на товари із вказаною кількістю.
5. **Робота з кошиком**  
   Користувач повинен мати змогу додавати товари до кошика, переглядати його вміст, змінювати кількість товарів або видаляти позиції.
6. **Моніторинг і збір метрик**  
   Система повинна експортувати метрики продуктивності (CPU, памʼять, пропускна здатність) до Prometheus і надавати їх через /metrics.
7. **Перемикання режимів архітектури**  
   Система повинна підтримувати можливість перемикання між монолітною та мікросервісною архітектурою шляхом конфігурації.
8. **Горизонтальне та вертикальне масштабування**  
   Платформа має підтримувати HPA та VPA для основних сервісів відповідно до навантаження.

**3.2. Нефункціональні вимоги**

1. **Продуктивність**  
   Система повинна витримувати навантаження не менше 1000 запитів на хвилину без деградації основних функцій.
2. **Масштабованість**  
   Сервіси повинні автоматично масштабуватись відповідно до використання CPU або памʼяті.
3. **Надійність**  
   У разі виходу з ладу одного з мікросервісів, інші мають залишатись доступними (fault-tolerant).
4. **Безпека**  
   Дані користувачів повинні зберігатись у зашифрованому вигляді. Доступ до API має бути обмежено через авторизацію.
5. **Збір та збереження логів**  
   Усі ключові події в системі повинні логуватись та бути доступними для подальшого аналізу.
6. **Розгортання**  
   Додаток повинен мати автоматизовану систему розгортання у Kubernetes-кластері з використанням Helm та ArgoCD.

**3.3. Вимоги до середовища**

* **Операційна система**: Kubernetes (Linux-based nodes)
* **Контейнеризація**: Docker
* **Оркестрація**: Kubernetes (версія 1.25+)
* **CI/CD**: ArgoCD (автоматичне розгортання)
* **Моніторинг**: Prometheus + Grafana
* **База даних**: Azure SQL або PostgreSQL
* **Інструменти навантаження**: hey