5. Пристемський, О. Вплив сучасних інформаційних технологій на ефективність управління підприємством [Електрон. ресурс] / О. Пристемський, А. Пашинний // Херсонський державний аграрно-економічний університет, 2023. – Режим доступу : https://dspace.ksaeu.kherson.ua/handle/123456789/8248.
6. Дриньов, Д. Штучний інтелект в процесі прийняття та реалізації управлінських рішень [Електрон. ресурс] / Д. Дриньов, К. Войтех, Р. Тимошенко // Таврійський науковий вісник. Серія : Економіка. – 2023. – Вип. 18. – Режим доступу : https://doi.org/10.32782/2708-0366/2023.18.7.
7. Employee/HR Dataset (All in One) [Electronic resource] // KaggleDatasets. – Access mode : https://www.kaggle.com/datasets/ravindrasinghrana/employeedataset.
8. Montgomery, D. Introduction to linear regression analysis [Text] / D. Montgomery, G. Vinind, E. Peck. – 6th ed. – Hoboken, NJ : Wiley, 2021. – 704 p.
9. Theobad, O. Machine learning and AI for absolute beginners: decision trees [Text] / O. Theobad. – O'Reilly Media, 2024.
10. Salame, N. The decision tree algorithm use in supervised machine learning [Electronic resource] / N. Salame // IJSREM. – Access mode : https://www.researchgate.net/publication/372409344_The_Decision_Tree_Algorithm_Use_in_Supervised_Machine_Learning.
11. Rose, M. Survey of random forest pruning techniques [Electronic resource] / M. Rose, H. Hassen // Computer Science & Information Techology. – 2019. – Vol. 9, № 18. – Access mode : https://aircconline.com/csit/papers/vol9/csit91808.pdf.
12. DataFrame.to_csv [Electronic resource] // Pandas. – Access mode : https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_csv.html.

УДК 005.353

Є. Д. Чемерис[1], Л.В. Кабак[1], Варех Н.В.[2]
[1]Національний технічний університет «Дніпровська політехніка», Дніпро, Україна
[2]Університет «Метінвест політехніка», м. Запоріжжя, Україна.

СТВОРЕННЯ СИСТЕМИ УПРАВЛІННЯ ЕЛЕКТРОННИМИ ЧЕРГАМИ З ПРОГНОЗУВАННЯМ ЧАСУ ОЧІКУВАННЯ НА БАЗІ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ В KUBERNETES

Анотація. У роботі розглянуто підхід до створення системи управління електронними чергами використовуючи середовище Kubernetes. Розроблена архітектура базується на мікросервісній моделі та використовує прогнозування часу очікування користувачів для динамічного масштабування сервісів.

Ключові слова: *електронна черга, Kubernetes, мікросервісна архітектура, Go, оператор, CRD, прогнозування часу очікування, автоматичне масштабування.*

Вступ. Для реалізації сучасних, навантажених інформаційних систем необхідно використовувати ефективні механізми керування запитами і користувачами. Електронні черги є частим інструментом взаємодії користувача і системи обслуговування, проте їх ефективність залежить від можливості прогнозування часу очікування та динамічного розподілу навантаження між

компонентами. Традиційні підходи до побудови черг мають обмеження щодо масштабованості й стійкості. Тому актуальним є створення системи управління електронними чергами на базі мікросервісної архітектури в Kubernetes [1], яка динамічно реагує на зміну кількості запитів за допомогою прогнозування та автоматичного масштабування.

Постановка задачі. Метою роботи є розробка системи управління електронними чергами з прогнозуванням часу очікування користувачів у середовищі Kubernetes. Для досягнення мети необхідно вирішити такі завдання:

- Розробити мікросервісну архітектуру, яка забезпечує масштабованість компонентів системи.
- Реалізувати модель прогнозування часу очікування та експериментально дослідити її точність.
- Створити Kubernetes-оператор і власний тип ресурсу (CRD QueuePolicy), що дозволяє описувати правила роботи черги та автоматично узгоджувати їх зі станом кластера.
- Забезпечити автоматичне масштабування компонентів системи на основі прогнозованого часу очікування користувачів за допомогою HPA/KEDA[2].
- Провести експериментальну оцінку ефективності запропонованого рішення за показниками середнього часу очікування (AWT) у порівнянні із більш традиційними системами.

Основний зміст роботи. Метою дослідження є створення керованої системи електронних черг, здатної динамічно реагувати на зміну навантаження. Для цього використано підхід мікросервісної архітектури в поєднанні з можливостями середовища Kubernetes, що дозволяє забезпечити незалежність компонентів, їх оновлення, масштабування та стійкість до збоїв завдяки можливості реалізувати різні стратегії розгортання сервісів.

З архітектурної точки зору система складається з трьох логічних рівнів: прикладного, інфраструктурного та керуючого.

Прикладний рівень (Queue, Predictor, API Gateway): обробляє створення, виклик та завершення квитків, а також прогнозує час очікування користувачів у черзі.

Queue Service – це основний компонент, який, використовуючи ефективну мову Golang[3], реалізує логіку управління електронними чергами. Predictor Service – аналітичний модуль, який здійснює прогнозування середнього часу очікування користувачів у черзі.

Інфраструктурний рівень (Kubernetes, KEDA): здійснює моніторинг стану сервісів і автоматичне масштабування залежно від навантаження.

Kubernetes Cluster (рис. 1) виступає платформою для розгортання мікросервісів, забезпечуючи ізоляцію виконання коду, автоматичне відновлення сервісів і керування ресурсами[4]. KEDA (Kubernetes Event Driven Autoscaler) забезпечує автоматичне масштабування подів сервісів Queue та Predictor залежно від навантаження[5]. Argo CD реалізує GitOps-підхід до розгортання: усі компоненти системи описані у вигляді Helm-чартів, і зміни в репозиторії автоматично синхронізуються з кластером.

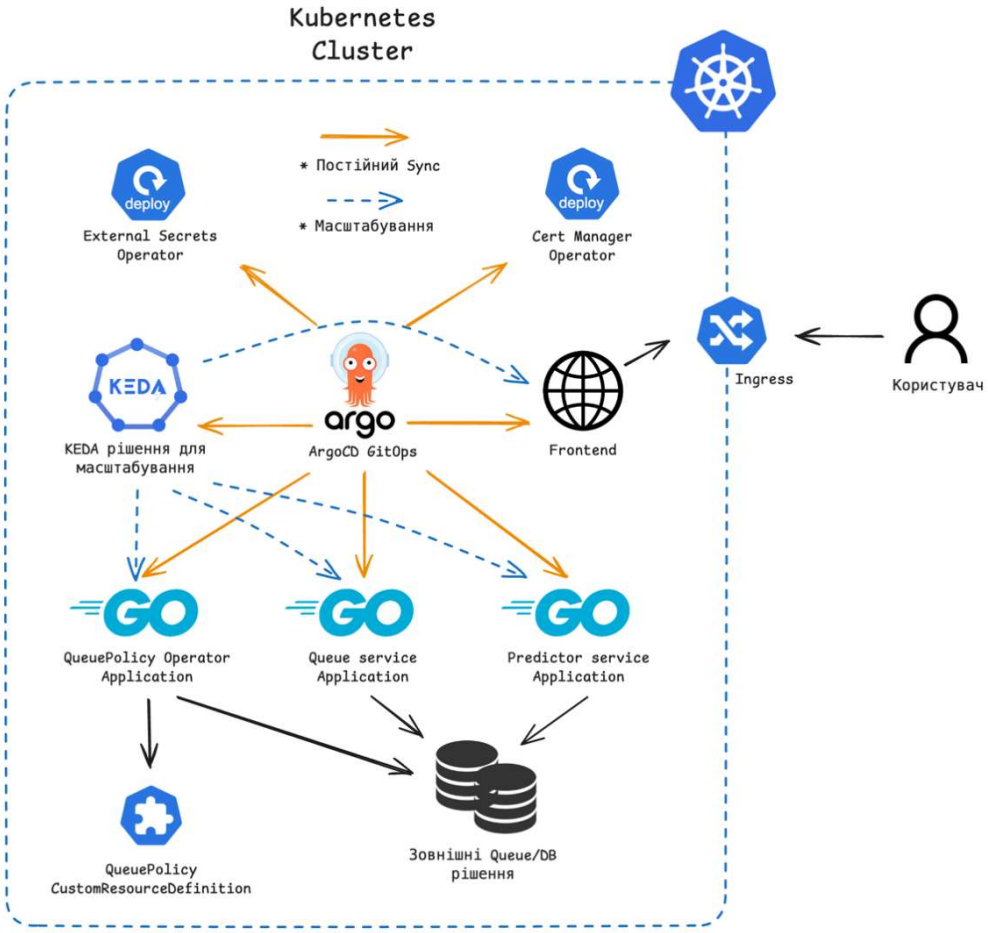

Рис. 1. Схема архітектури Kubernetes кластера

Керуючий рівень (QueuePolicy Operator): реалізує контроль політик черг у вигляді CRD, створює ресурси HPA та ServiceMonitor, а також узгоджує реальний стан системи з очікуваним.

Для контролю реалізовано власний Kubernetes-оператор що реалізує Custom Resource Definition (CRD) під назвою QueuePolicy. QueuePolicy — описує правила обслуговування, пріоритети, допустимий час очікування та порогові значення для масштабування. Оператор періодично зчитує ці об’єкти, порівнює їх із реальним станом системи та вносить зміни: створює або оновлює ресурси HPA, ServiceMonitor і ConfigMap.

Описана система реалізує наступний алгоритм роботи:

1. Користувач створює заявку через веб-портал.
2. Запит потрапляє до сервісу Queue, де створюється подія ticket.created , щоб усі інші сервіси могли реагувати асинхронно.
3. Predictor отримує ці події, оновлює статистику та обчислює новий прогноз середнього часу очікування.
4. KEDA порівнює значення метрики з порогом і у випадку перевищення збільшує кількість подів сервісу Queue або Predictor.
5. При зниженні навантаження кількість реплік автоматично скорочується.

Наукова новизна. У роботі запропоновано новий підхід до управління електронними чергами, який поєднує модель прогнозування часу очікування з автоматичним масштабуванням сервісів у Kubernetes. На відміну від звичайних систем, що працюють за фіксованими налаштуваннями, розроблене рішення самостійно адаптується до зміни навантаження. Такий підхід дозволяє зменшити середній час очікування користувачів і забезпечити стабільну роботу системи навіть при різких піках навантаження.

Реалізовано Kubernetes-оператор, що приймає політики обслуговування у вигляді CRD і автоматично підтримує працездатність системи.

Висновки. Запропонована система дозволяє ефективно керувати електронними чергами за допомогою Kubernetes-оператора, який забезпечує автоматичне масштабування сервісів відповідно до прогнозованого часу очікування користувачів. Такий підхід може бути використаний для підвищення ефективності сервісів у електронних системах і хмарних рішеннях, що базуються на інфраструктурі Kubernetes.

ПЕРЕЛІК ПОСИЛАНЬ

1. Burns B., Grant B., Oppenheimer D., Brewer E., Wilkes J. Borg, Omega, and Kubernetes: Lessons Learned from Three Container-Management Systems, 2016 — 25 ст.
2. Google Cloud. Horizontal Pod Autoscaling in Kubernetes. – [Електронний ресурс] Режим доступу: https://cloud.google.com/kubernetes-engine/docs/concepts/horizontalpodautoscaler
3. Go Documentation. Effective Go. – [Електронний ресурс] Режим доступу: https://go.dev/doc/effective_go
4. Hightower K., Burns B., Beda J. Kubernetes: Up and Running. 3rd edition. O'Reilly Media, 2023. — 342 ст.
5. KEDA: Kubernetes Event-Driven Autoscaling. – [Електронний ресурс] Режим доступу: https://keda.sh/docs/2.18/setupscaler/