УДК 004.4, 004.6, 004.7, 510.6 https://doi.org/10.17721/1812-5409.2023/1.13

О. П. Ількун, аспірант

Вебзастосунки як один із сучасних способів реалізації систем підтримки прийняття рішень

Київський національний університет імені Тараса Шевченка, 03022, м. Київ,
пр-т. Академіка Глушкова 4д,
e-mail: alexander.ilkun@gmail.com

O. P. Ilkun, PhD student

Web applications as one of the modern ways of implementing decision support systems

Taras Shevchenko National University of Kyiv, 03022, Kyiv,
4D, Academician Glushkov ave.,
e-mail: alexander.ilkun@gmail.com

Системи прийняття рішень мають широкий діапазон застосувань у різних сферах, включаючи, але не обмежуючись, комерційну діяльність, медицинські установи, освітні заклади, наукові дослідження, інженерію, а також урядові установи. Ці системи підтримують ефективну обробку інформації з різноманітних джерел, використовуючи множину методологічних підходів, що включає статистичний аналіз, аналіз даних, методи машинного навчання, оптимізацію та багато інших.

З огляду на вищезазначені особливості, виникає важливе питання розробки ефективних систем прийняття рішень, які відповідали б потребам сучасних організацій. Ця стаття розглядає актуальні підходи до розробки програмного забезпечення, в тому числі аналізує потенційну архітектуру для реалізації системи прийняття рішень, орієнтованої на діагностику пацієнтів з використанням нечіткої логіки.

Ключові слова: системи прийняття рішень, нечітка логіка, архітектура вебзастосунків, моделювання баз даних.

Decision support systems have a wide range of applications in various fields, including but not limited to commercial activities, medical institutions, educational institutions, scientific research, engineering, and government agencies. These systems support efficient processing of information from various sources using a variety of methodological approaches, including statistical analysis, data analysis, machine learning methods, optimization, and many others.

In view of the above-mentioned features, there is an important issue of developing effective decision support systems that would meet the needs of modern organizations. This article reviews current approaches to software development, including analyzing a potential architecture for implementing a decision support system focused on patient diagnosis using fuzzy logic.

Key Words: decision support systems, fuzzy logic, web application architecture, database modeling.

Вступ

В сучасному глобалізованому світі, де інформація є важливим активом, системи прийняття рішень відіграють критичну роль [1]. Ці системи допомагають в управлінні, аналізі та інтерпретації великих обсягів даних, що сприяє кращому розумінню бізнес-процесів та підвищенню ефективності управління.

Системи прийняття рішень використовуються в різних галузях діяльності, включаючи бізнес, медицину, освіту, науку, інженерію та урядові структури. Вони забезпечують обробку інформації з різних джерел і можуть використовувати різноманітні методики, включаючи статистичний аналіз, аналіз даних, машинне навчання, оптимізацію та інше.

Постає питання побудови ефективних систем прийняття рішень, що відповідають потребам сучасних організацій. В рамках статті проаналізовані сучасні підходи до розробки програмних застосунків, розглядається можлива архітектура реалізації системи прийняття рішень для діагностики захворювань пацієнтів [2] з використанням нечіткої логіки [3].

Огляд основних класів програмного забезпечення

До основних класів сучасного програмного

забезпечення можна віднести десктопні та веборієнтовані програми. Розглянемо кожен з цих класів програм окремо.

Десктопні програми - це програми, які розробляються для встановлення та використання на локальному комп'ютері або ноутбуці, що працює під операційною системою, такою як Windows, MacOS чи Linux. Класичний приклад десктопної програми - текстовий редактор, який дозволяє створювати та редагувати документи, що зберігаються на локальному диску. Інші приклади десктопних програм включають графічні редактори, програми для обробки відео та звуку, браузери, клієнти електронної пошти, різноманітні ігри, програми для налаштування системи, тощо.

Основна перевага десктопних програм полягає в тому, що вони працюють безпосередньо на комп'ютері, що дає можливість отримати більшу продуктивність та функціональність, ніж в вебпрограмах. Крім того, десктопні програми можуть працювати в автономному режимі, не потребуючи постійного з'єднання з Інтернетом, що робить їх більш зручними для користувачів, які працюють у зоні з обмеженим доступом до Інтернету або у віддалених місцях.

Однак десктопні програми мають свої недоліки, зокрема вони потребують встановлення та налаштування на кожному комп'ютері, на якому вони повинні працювати. Крім того, вони не можуть бути використані на інших пристроях без встановлення на кожному з них, що робить їх менш зручними для користувачів, які працюють з різних місць.

Вебзастосунки - це клас програмного забезпечення, які використовують вебтехнології для забезпечення доступу до різних функцій через браузер. Вони можуть бути доступні як локально на сервері, так і в мережі Інтернет.

Основні компоненти вебсистем включають вебсервер, базу даних та фронтенд. Вебсервер відповідає за обробку запитів від клієнтів та відправку відповіді. База даних використовується для зберігання даних, які використовуються в системі. Фронтенд - це один із можливих інтерфейсів користувача, який відображається в браузері та дозволяє користувачам взаємодіяти з системою. Крім інтерфейсу в браузері, універсальні протоколи передачі даних дозволяють реалізовувати клієнтську частину на широкому колі пристроїв – від звичайних комп'ютерів та смартфонів до вбудованих систем або суперкомп'ютерів.

Однією з головних переваг створення вебсистем є їх доступність з будь-якого місця, де є Інтернет-з'єднання. Користувач може отримати доступ до вебистеми зі свого персонального пристрою. В цьому випадку не потрібно встановлювати окремий додаток на кожен пристрій, що значно спрощує процес розгортання та підтримки системи.

Іншою важливою перевагою є можливість оновлення системи з використанням централізованих серверів. Користувачам не потрібно завантажувати нову версію системи на свій комп'ютер, оскільки оновлення відбуваються на сервері. Це зменшує ризик помилок та вразливостей, що можуть виникнути при локальному оновленні десктопної програми.

Також вебсистеми зазвичай мають більш простий інтерфейс, який дозволяє користувачам легко розуміти та використовувати систему без необхідності вивчення складних інструкцій. Більш того, вебсистеми можуть бути більш гнучкими та модульними, що дозволяє змінювати функціональність та додавати нові можливості в систему без необхідності встановлення нових версій на локальні комп'ютери.

Нарешті, створення вебсистем може бути більш ефективним з точки зору вартості, оскільки вони можуть бути створені з використанням відкритих технологій та безкоштовних програмних засобів. Це може знизити вартість розробки та підтримки системи, зокрема у порівнянні з десктопними додатками, які часто потребують дорогих ліцензій на програмне забезпечення та обладнання.

Отже, створення вебсистем має багато переваг порівняно з десктопними додатками. Вебсистеми є більш доступними, оскільки вони можуть бути запущені на будь-якому пристрої з доступом до Інтернету. Крім того, вебсистеми можуть бути оновлені централізовано на сервері, тим самим зменшуючи зусилля з оновлення на кожному пристрої окремо.

Крім того, вебсистеми є більш зручним і ефективним способом для створення програмного забезпечення, оскільки вони дозволяють знизити витрати на розробку, зберігання та підтримку програмного забезпечення. Відтак, вебсистеми стають все більш популярними у сучасному світі, де швидкість і доступність є клю-

човими факторами успіху.

Таким чином, найкращим класом, для реалізації сучасної програмної системи для підтримки прийняття рішень є вебсистема.

Вимоги до сучасних вебсистем

Сучасні вебсистеми мають високі апаратні та програмні вимоги, щоб відповідати вимогам користувачів. Для успішної роботи вебсистеми необхідне забезпечення достатньої швидкодії, безпеки, надійності та масштабованості.

Перш за все, необхідність масштабованості приводить до вимог до серверного обладнання, яке повинно забезпечувати високу продуктивність та надійність. Вебсистема повинна бути здатна працювати з великою кількістю користувачів та обробляти великі обсяги даних. Тому, серверне обладнання повинно мати достатньо ресурсів для забезпечення продуктивності та можливості масштабування.

Друге, вимога до безпеки системи є дуже важливою, оскільки вебсистеми можуть зберігати велику кількість конфіденційної інформації, такої як особисті дані користувачів, банківські реквізити, інформацію про платежі тощо. Для забезпечення безпеки, вебсистема повинна мати вбудовану систему безпеки, яка забезпечує шифрування та автентифікацію користувачів, перевірку прав доступу, захист від атак та вірусів.

Третє, вимога до швидкодії вебсистеми є важливою, оскільки користувачі очікують, що система буде працювати швидко та ефективно. Швидкодія вебсистеми залежить від багатьох чинників, таких як швидкість серверного обладнання, кількість запитів до бази даних, оптимізація коду програмного забезпечення тощо.

Існує безліч високорівневих архітектур для вебзастосунків. До найбільш відомого типу можна віднести трирівневу архітектуру [4].

В рамках статті розглянемо розширену архітектуру програмного забезпечення вебзастосунку, яка враховує вимоги масштабованості, швидкодії та безпеки в контексті наступних основних рівнів:

1) Рівень інфраструктури
2) Презентаційний рівень
3) Рівень бізнес-логіки
4) Рівень моніторингу
5) Рівень зберігання даних

Рівень інфраструктури - це основа, на якій працює вся вебсистема. Він включає всі аспекти фізичної та віртуальної інфраструктури, необхідні для підтримки роботи вебсистеми.

Фізична інфраструктура може бути розміщена у власному дата-центрі або може бути орендована від постачальників хмарних сервісів. У сенсі управління інфраструктурою також потрібно чітко відрізняти *внутрішню інфраструктуру*, яка знаходиться у зоні відповідальності команди розробників, і *зовнішню інфраструктуру*, за яку відповідає зовнішній постачальник. Окремим важливим компонентом зовнішньою інфраструктури є так звана *мережа розповсюдження контенту*, що має на меті оптимізацію доправлення та розповсюдження відповідей вебзастосунку.

В якості конкретних технологій для реалізації інфраструктурного рівня для системи діагностики пацієнтів пропонується використовувати Hetzner Cloud [5], Kubernetes [6] та Cloudflare [7].

Презентаційний рівень - це перший рівень архітектури, з яким безпосередньо взаємодіє користувач. Його головною метою є надання користувачам зрозумілого, ефективного та приємного інтерфейсу для взаємодії з системою. Презентаційний рівень включає в себе елементи дизайну інтерфейсу, технології фронтенду та процеси, що забезпечують відображення даних та збір вводу від користувача.

До основних модулів презентаційного рівня відносяться *шлюз API*, що зв'язує кінцевих клієнтів з набором внутрішніх сервісів, а також *фронтенд сервіс*, який виконує логіку інтерфейсної частини вебзастосунку у браузері.

З метою побудови презентаційного рівня для системи діагностики стану здоров'я пацієнтів, пропонується використати такі технології як SvelteKit [8] та Spring Cloud Gateway [9].

Рівень бізнес-логіки - це серце будь-якої вебсистеми. Цей рівень містить в собі код, який виконує ключові функції системи, обробляє вхідні дані від презентаційного рівня та взаємодіє з рівнем зберігання даних для зберігання та отримання інформації.

Основними модулями рівня бізнес-логіки є *сервіс предметної області*, що оперує сутностями та функціоналом деякої предметної області

(в нашому випадку системою діагностики захворювань), а також *сервіс підтримки прийняття рішень*, який здійснює обчислення результату відповідно до деякої математичної моделі (у нашому випадку це нечітка логіка).

Для гарантування і виконання основних вимог системи діагностики пацієнтів пропонується застосувати технології та супутні бібілотеки мов програмування Java [10] та Python [11].

Рівень моніторингу - це критичний компонент архітектури будь-якої вебсистеми, який дозволяє командам відстежувати роботу системи в реальному часі та виявляти проблеми ще до того, як вони вплинуть на користувачів. Він також допомагає аналізувати ефективність системи та планувати майбутнє масштабування.

Рівень моніторингу складається з трьох основних модулів: *сервіс логування*, який дозволяє агрегувати та здійснювати пошук логів, *збірник метрик*, що допомагає відстежувати основні показники та стан процесів, та *візуалізатор метрик*, який спрощує обробку та сприйняття ключових метрик системи.

Для впровадження ефективного рівня моніторингу нашої системи пропонується використати наступні технології - ELK Stack [13], Prometheus [11] та Grafana [14].

Рівень зберігання даних - це надважливий компонент архітектури будь-якої вебсистеми. Цей рівень відповідає за зберігання, відновлення та управління даними, які використовуються системою. Надамо загальний опис спільних модулів цього рівня, який притаманний більшості систем.

При побудові високопродуктивної бази даних має сенс використовувати схему з *головним вузлом*, який може здійснювати будь-які операціїї на базі даних, та *вузол читання*, який оптимізований для операцій отримання інформації. Пропонується використати PostgreSQL [15].

Схематично всі рівні вебзастосунку можна представити діаграмою на Рис. 1.

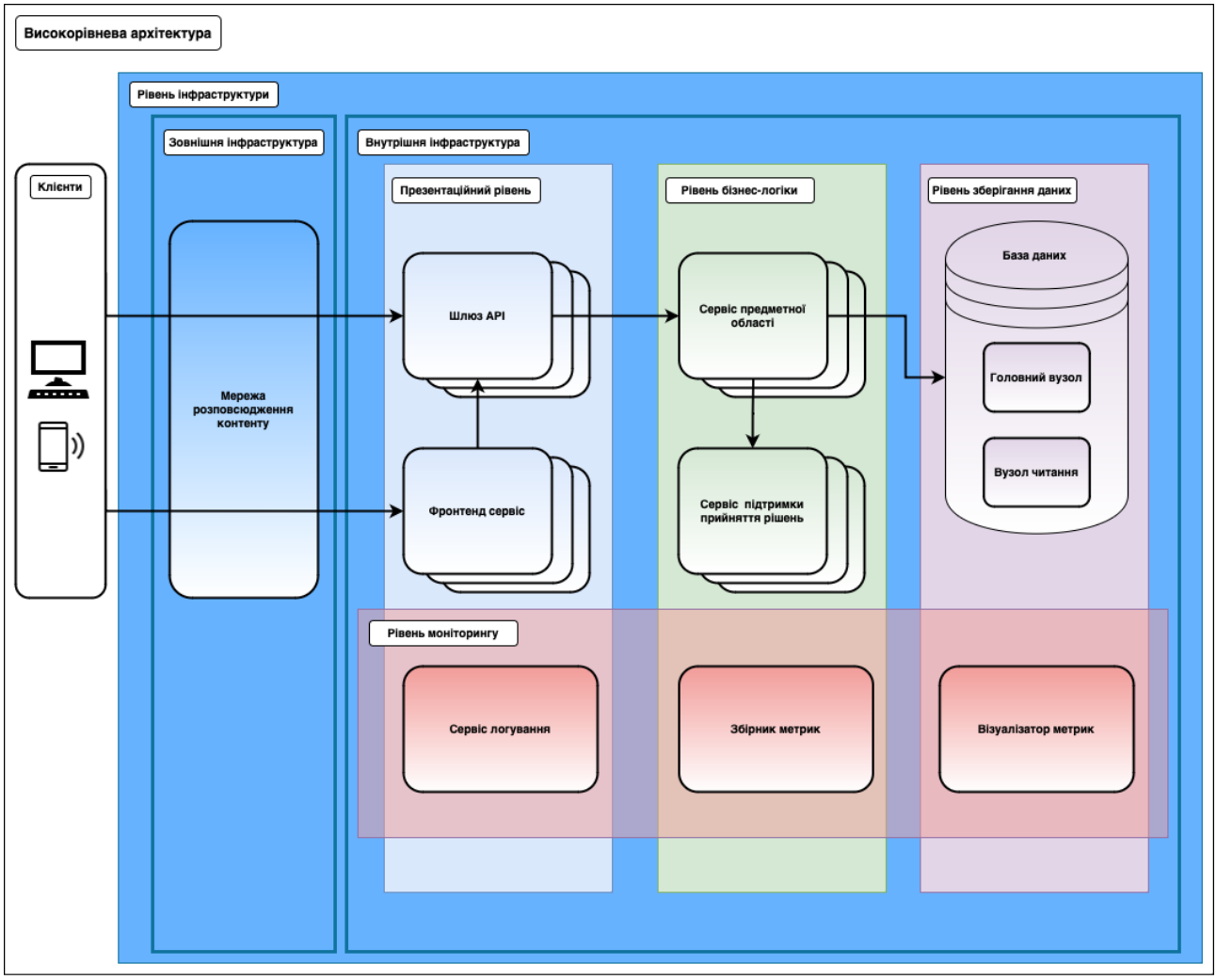

Рис. 1: Високорівнева архітектура вебзастосунка для системи підтримки прийняття рішень

Моделювання рівня зберігання даних відповідно до предметної області

Необхідно зазначити, що моделювання даних має суттєво предметно-орієнтований характер. Концептуальні сутності, які репрезентують структуру даних, які обробляються системою, суттєво залежать від специфіки предметної області та цілей самої системи.

Важливо підкреслити, що кількість, типи та взаємозв'язки сутностей предметної області можуть значно відрізнятися, коли ми переходимо від одного проекту до іншого. Це робить моделювання даних складним адаптивним процесом, який вимагає глибокого розуміння предметної області та потреб користувачів.

Беручи до уваги конкретні вимоги до системи діагностики медичних станів пацієнтів, пропонується наступна схема даних, представлена на Рис. 2.

Ця схема була розроблена з урахуванням унікальних потреб цієї системи, зокрема потреби в ефективній організації та обробці великих обсягів медичних даних. Вона відображає ключові сутності та їх взаємозв'язки в контексті цієї системи, забезпечуючи таким чином спрощення процесу обробки даних та підвищення загальної ефективності системи.

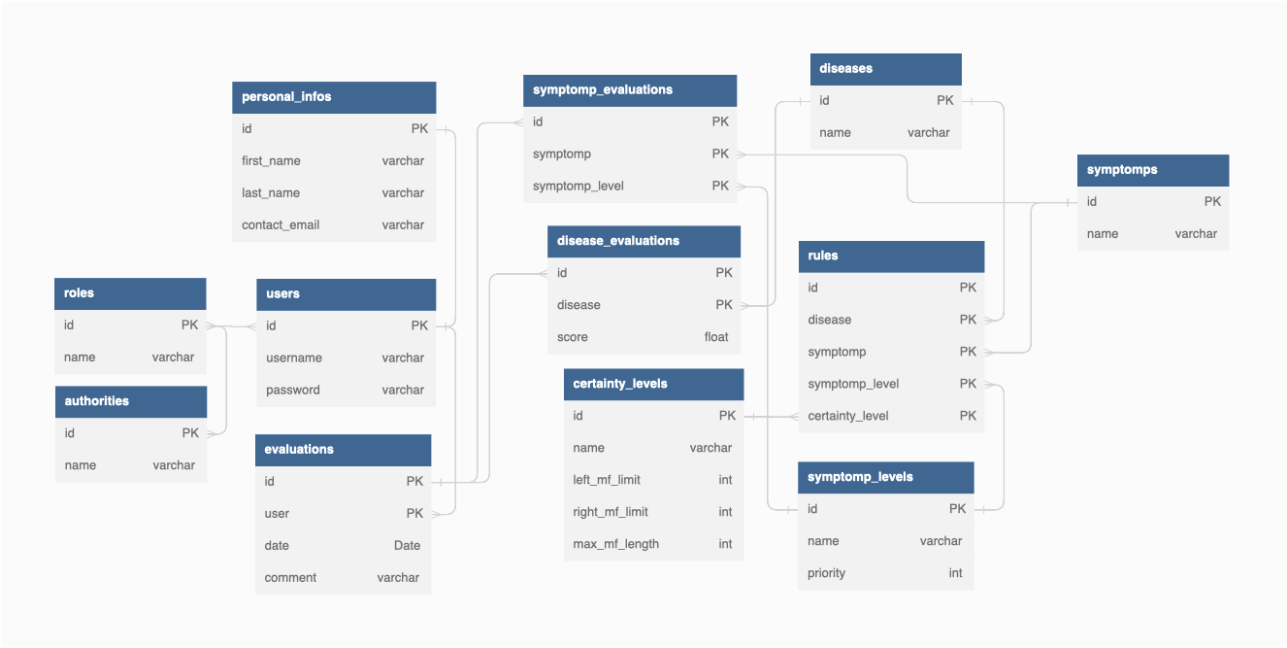

Рис. 2: База даних для системи діагностики захворювань пацієнтів

Серед усіх сутностей у цій схемі даних особливої уваги заслуговує сутність *rules*, яка відповідає за правила логічного виведення нечітких множин.

Наприклад, найбільш спрощена нечітка система логічного виведення може бути представлена у наступній формі вхідних даних, умов та вихідних даних [2]:

$$\text{вхід } (x)$$

$$\text{якщо } x \in A, \text{ то } y \in B$$

$$\text{вихід } (y)$$

де A, B - нечіткі множини, а x, y є лінгвістичними змінними.

Висновки

Системи прийняття рішень відіграють визначальну роль в широкому спектрі промислових, академічних, державних та соціальних сфер. Одна з ключових характеристик цих систем полягає у їхній здатності ефективно обробляти великі обсяги інформації за допомогою широкого діапазону методологічних технік. Ці техніки включають статистичний аналіз, аналіз даних, машинне навчання, оптимізацію, нечіткі множини та багато інших.

У світлі вищенаведеного, проблема розробки високоефективних систем прийняття рішень, які відповідали б потребам сучасних організацій, є особливо актуальною. У цій статті було розглянуто найновіші методи розроб-

ки програмного забезпечення та запропоновано потенційну архітектуру для реалізації систем прийняття рішень, яка може бути адаптована до широкого спектра задач.

Було продемонстровано застосування цієї архітектури на прикладі конкретної системи підтримки прийняття рішень, розробленої для діагностики медичних станів пацієнтів.

Додатково, було розглянуто виклики, пов'язані з моделюванням даних у великомасштабних інформаційних системах, зокрема з урахуванням специфіки предметної області. Було акцентовно увагу на тому, що кількість та типи сутностей предметної області можуть значно відрізнятися залежно від специфіки задачі.

В основній частині статті, було висвітлено ряд сучасних технологій, які мають потенціал підтримувати вимоги до масштабованості, безпеки і швидкодії, необхідні для ефективної реалізації вебзастосунка з запропонованою архітектурою

Розвиток та удосконалення систем підтримки прийняття рішень не лише збільшує ефективність діяльності організацій, але також створює нові методи для суттєвого прогресу. Значна кількість сучасних індустрій уже залежать від використання аналітики даних, що робить системи підтримки прийняття рішень важливими компонентами успіху, заслуговуючого подальшого вивчення.

Список використаних джерел

1. *Filip, F.G.* Decision Support Systems / Filip, F.G., Zamfirescu, CB., Ciurea, C. — Cham: Springer, 2017. — P. 31–69.

2. *Саввакін В.Д.* Діагностична система на основі нечітких знань / В.Д. Саввакін, О.І. Провотар // Комп'ютерна математика — Київ: Інститут кібернетики імені В.М. Глушкова НАН України, 2019. — С. 56-63.

3. *Провотар А.И.* О некоторых подходах к вычислению неопределенностей / А.И. Провотар, А.В. Лапко // Проблеми програмування — Київ: Інститут програмних систем НАН України, 2010. — С. 22-27.

4. *Tie J.* Study on application model of three-tiered architecture /J. Tie, J. Jin, X. Wang // 2011 Second International Conference on Mechanic Automation and Control Engineering, 2011, Inner Mongolia, China. — 2011. — P. 7715-7718.

5. Hetzner Cloud Docs [Електронний ресурс] – Режим доступу до ресурсу: https://docs.hetzner.com/cloud/.

6. Kubernetes Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://kubernetes.io/docs/home/.

7. Cloudflare Docs [Електронний ресурс] – Режим доступу до ресурсу: https://developers.cloudflare.com.

8. Docs SvelteKit [Електронний ресурс] – Режим доступу до ресурсу: https://kit.svelte.dev/docs/introduction.

9. Spring Cloud Gateway reference [Електронний ресурс] – Режим доступу до ресурсу: https://docs.spring.io/spring-cloud-gateway/docs/current/reference/html/.

10. Java Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/en/java/.

11. Python 3.10 documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/.

12. ELK Stack documentation [Електронний ресурс] – Режим доступу до ресурсу: https://elastic-stack.readthedocs.io/en/latest/.

13. Prometheus Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://prometheus.io/docs/introduction/.

14. Documentation Grafana Labs [Електронний ресурс] – Режим доступу до ресурсу: https://grafana.com/docs/.

15. PostgreSQL Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.postgresql.org/docs/.

References

1. FILIP, F., ZAMFIRESCU, CB. and CIUREA, C. (2017) *Decision Support Systems*. Cham: Springer.

2. SAVVAKIN, V., PROVOTAR O. (2019) Diahnostychna systema na osnovi nechitkykh znan. Kompiuterna matematyka. — pp. 56-63.

3. PROVOTAR, A., LAPKO, A. (2010) O nekotoryih podhodah k vyichisleniyu neopredelennostey. Problemy prohramuvannia. — pp. 22-27.

4. TIE, J., JIN, J., WANG X. (2011) Study on application model of three-tiered architecture. In *2011 Second International Conference on Mechanic Automation and Control Engineering*. Inner Mongolia, China. — 2011. — pp. 7715-7718.

5. HETZNER. (2023) *Hetzner Cloud Docs* [Online] Available from: https://docs.hetzner.com/cloud/. [Accessed: 8th of January 2023].

6. KUBERNETES. (2023) *Kubernetes Documentation* [Online] Available from: https://kubernetes.io/docs/home/. [Accessed: 8th of January 2023].

7. CLOUDFLARE. (2023) Cloudflare Docs [Online] Available from: https://developers.cloudflare.com. [Accessed: 8th of January 2023].

8. SVELTEKIT. (2023) Docs SvelteKit [Online] Available from: https://kit.svelte.dev/docs/introduction. [Accessed: 8th of January 2023].

9. SPRING. (2023) Spring Cloud Gateway reference [Online] Available from: https://docs.spring.io/spring-cloud-gateway/docs/current/reference/html/. [Accessed: 8th of January 2023].

10. ORACLE. (2023) Java Documentation [Online] Available from: https://docs.oracle.com/en/java/. [Accessed: 8th of January 2023].

11. PYTHON. (2023) Python 3.10 documentation [Online] Available from: https://docs.python.org/3/. [Accessed: 8th of January 2023].

12. ELK. (2023) ELK Stack documentation [Online] Available from: https://elastic-stack.readthedocs.io/en/latest/. [Accessed: 8th of January 2023].

13. PROMETHEUS. (2023) Prometheus Documentation [Online] Available from: https://prometheus.io/docs/introduction/. [Accessed: 8th of January 2023].

14. GRAFANA. (2023) Documentation Grafana Labs [Online] Available from: https://grafana.com/docs/. [Accessed: 8th of January 2023].

15. POSTGRESQL. (2023) PostgreSQL Documentation [Online] Available from: https://www.postgresql.org/docs/. [Accessed: 8th of January 2023].

Надійшла до редколегії 21.03.2023