СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет ___інформаційних технологій та електроніки_____

Кафедра _____інформаційних технологій та програмування_____

Пояснювальна записка

до магістерської дипломної роботи

_____магістр_____
(освітньо-кваліфікаційний рівень)

на тему: ___Розробка системи підтримки прийняття рішень для вибору хмарних сервісів_____

Виконав: студент 2 курсу, групи ІСТ-23дм_
126 «Інформаційні системи та технології___
(шифр і назва спеціальності)
_____Борзикін В. І._____
(прізвище та ініціали)
Керівник __Дьомін М.К._____
(прізвище та ініціали)
Рецензент_____
(прізвище та ініціали)

Київ – 2024 року

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет інформаційних технологій та електроніки
Кафедра інформаційних технологій та програмування
Освітньо-кваліфікаційний рівень магістр
Спеціальність 126 «Інформаційні системи та технології»
(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТП
_____ д.т.н., доц. Захожай О.І.
(підпис)
« ___ » _____2024 р.

ЗАВДАННЯ

на магістерську дипломну роботу студенту
Борзикін Вадим Ігорович
(прізвище, ім'я, по батькові)

1.Тема роботи: «Розробка СППР для вибору хмарних сервісів»

керівник роботи к.т.н., доц. Дьомін Максим Костянтинович,
(вчене звання, науковий ступінь, прізвище, ім'я, по батькові)

затверджені наказом університету від «06» грудня 2024 року №361/15.15-С

2. Строк подання студентом роботи: 06 грудня 2024 р

3. Вихідні дані до роботи: матеріали науково-дослідної практики, науково-методична література; дані інтернет-мережі тощо.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

4.1 Вступ

4.2 Аналітичний огляд питання вибору хмарних сервісів (огляд публічних джерел інформації)

4.3 Основна частина, в якої висвітлити методи, які будуть використовуватися для реалізації проекту.

4.4 Практична частина – огляд технологій, які використовуються під час реалізації проекту.

4.4 Висновки

4.5 Перелік використаних джерел

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

_____

_____

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Розділ 1. Аналіз використання хмарних сервісів			
Розділ 2. Проєктування СППР для вибору хмарних сервісів			
Розділ 3. Розробка СППР для вибору хмарних сервісів			

7. Дата видачі завдання 20 жовтня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів написання випускної кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Одержання завдання на виконання роботи	20.10.2024	
2	Укладання і погодження з керівником плану і етапів виконання роботи	24.10.2024	
3	Узагальнення даних літературних джерел	28.10.2024	
4	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху виконання завдання	01.11.2024	
5	Аналіз технічних засобів та існуючих систем	07.11.2024	

6	Реалізація практичної частини завдання	24.11.2024	
7	Укладання, оформлення та погодження пояснювальної записки з керівником	05.12.2024	
8	Здача пояснювальної записки на кафедру	06.12.2024	
9	Здача пояснювальної записки на кафедру	06.12.2024	

Студент_____Борзикін В. І.
(підпис) (прізвище та ініціали)

Керівник роботи_____Дьомін М. К.
(підпис) (прізвище та ініціал

Анотація

Дипломна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, додатки і має 75 сторінок основного тексту, 9 рисунків, 10 таблиця, 4 сторінки додатків. Список використаних джерел містить 26 найменувань і займає 3 сторінки. Загальний обсяг роботи 82 стор.

Актуальність теми роботи. У сучасному вельми складному діловому світі організації повинні знайти інноваційні способи виділитися серед конкурентів, ставши більш спільними, віртуальними, точними, синхронними, адаптивними та гнучкими. Вони повинні мати можливість швидко реагувати на потреби та зміни ринку. Багато організацій помітили, що дані, якими вони володіють, і спосіб їх використання можуть відрізняти їх від інших. Дані та інформація стають основним активом для багатьох організацій. Окрім викликів, пов'язаних із швидким зростанням обсягу даних, світ також відкриває широкі можливості, оскільки він стає все більш цифровим, що дозволяє агрегувати й аналізувати дані в залежності від контексту. Відтак інформація та/або знання, отримані з цифрових записів, можуть полегшити роботу лікарів у точному діагностуванні та лікуванні захворювань, а також знизити витрати на медичне обслуговування для постачальників і пацієнтів, а отже, підвищити загальну якість та ефективність медичної допомоги.

Об'єкт дослідження – хмарні сервіси.

Предмет дослідження. Методи та інформаційні технології для вибору хмарних сервісів.

Методи дослідження. Теоретичною основою дослідження є загальнонауковий аналітичний метод, а також системний підхід і праці провідних вчених з проблем дослідження і оцінювання хмарних сервісів. Для практичного вирішення поставлених задач використовувалися такі методи: загальнонауковий аналітичний метод; методи теорії прийняття рішення для розробки алгоритмічного забезпечення для СППР; методи алгоритмічного програмування, для створення СППР.

Наукова новизна роботи полягає в розробці двоетапної методики вибору хмарного провайдера на основі теорії нечітких множин та метода TOPSIS. А також розробки системи критеріїв для оцінки хмарних сервісів.

Практичне значення одержаних результатів. Практична значущість роботи полягає у створенні СППР, яка може використовуватись невеликими підприємствами для вирішення питання міграції в хмару.

Abstracts

The thesis consists of an introduction, three chapters, general conclusions, a list of references, an appendix and has 75 pages of main text, 9 figures, 10 tables, and 4 appendix pages. The list of references includes 26 titles and occupies 3 pages. The total volume of the work is 82 pages.

Relevance of the topic of the work. In today's highly complex business world, organizations must find innovative ways to stand out from the competition by becoming more collaborative, virtual, accurate, synchronous, adaptive and flexible. They must be able to respond quickly to market needs and changes. Many organizations have noticed that the data they have and the way they use it can differentiate them from others. Data and information are becoming a major asset for many organizations. In addition to the challenges associated with the rapid growth of data, the world also opens up great opportunities as it becomes more digital, allowing data to be aggregated and analyzed depending on the context. Therefore, information and/or knowledge obtained from digital records can facilitate the work of doctors in accurately diagnosing and treating diseases, as well as reduce healthcare costs for providers and patients, and therefore increase the overall quality and efficiency of healthcare.

Object of research – cloud services.

The subject of the research is methods and software tools for protecting information systems.

Research methods. The theoretical basis of the study is the general scientific analytical method, as well as the systematic approach and works of leading scientists on the problems of research and evaluation of cloud services. For the practical solution of the tasks set, the following methods were used: general scientific analytical method; decision theory methods for developing algorithmic support for DSS; algorithmic programming methods for creating DSS.

The scientific novelty of the work lies in the development of a two-stage methodology for selecting a cloud provider based on fuzzy set theory and the TOPSIS method. As well as developing a system of criteria for evaluating cloud services.

Practical significance of the results obtained. The practical significance of the work lies in creating a DSS that can be used by small businesses to solve the issue of migration to the cloud.

ЗМІСТ

ВСТУП

Актуальність роботи. У сучасному вельми складному діловому світі організації повинні знайти інноваційні способи виділитися серед конкурентів, ставши більш спільними, віртуальними, точними, синхронними, адаптивними та гнучкими. Вони повинні мати можливість швидко реагувати на потреби та зміни ринку. Багато організацій помітили, що дані, якими вони володіють, і спосіб їх використання можуть відрізняти їх від інших. Дані та інформація стають основним активом для багатьох організацій.

Окрім викликів, пов’язаних із швидким зростанням обсягу даних, світ також відкриває широкі можливості, оскільки він стає все більш цифровим, що дозволяє агрегувати й аналізувати дані в залежності від контексту. Відтак інформація та/або знання, отримані з цифрових записів, можуть полегшити роботу лікарів у точному діагностуванні та лікуванні захворювань, а також знизити витрати на медичне обслуговування для постачальників і пацієнтів, а отже, підвищити загальну якість та ефективність медичної допомоги. Подібним чином можна отримати доступ до оцифрованих даних (інституційних і публічних – переважно в Інтернеті) і проаналізувати їх, щоб сприяти успішній боротьбі зі злочинністю.

Сервісно-орієнтоване мислення є однією з найбільш швидкозростаючих парадигм в інформаційних технологіях, що має відношення до багатьох інших дисциплін, таких як бухгалтерський облік, фінанси та операції.

Для багатьох компаній (особливо малих і середніх) модель обчислень, орієнтована на послуги з оплатою за використання (хмарні обчислення), коли хтось інший турбується про підтримку апаратного та програмного забезпечення, стає дуже привабливою. Хмарні обчислення відповідають таким парадигмам програмного забезпечення як послуги, інфраструктури як послуги, бази даних як послуги. Платформи хмарних обчислень, такі як Amazon Web Services, AWS, IBM Cloud, Microsoft 365, Slack, Google WorkSpace, тощо підтримують більше, ніж апаратне забезпечення, а також

надають клієнтам набір віртуальних машин, на яких можна встановити власне програмне забезпечення. Доступність ресурсів, як правило, еластична, з, здавалося б, нескінченною кількістю обчислювальних потужності та місця для зберігання, доступних за запитом, примушує організації до міграції в хмару. В той же час вибір відповідної моделі та конкретного провайдера для багатьох організацій залишається складною задачею, оскільки всі вони мають, як переваги, так і недоліки.

У сучасних умовах, коли компанії активно переходять на використання хмарних технологій, вибір правильного хмарного сервісу (IaaS, PaaS, SaaS) стає стратегічно важливим. Хмарні платформи визначають продуктивність, безпеку та вартість IT-інфраструктури. У цьому процесі Системи підтримки прийняття рішень (СППР) відіграють ключову роль, надаючи структуровані підходи до аналізу, оцінки та вибору найкращого варіанту.

Існує безліч хмарних платформ (AWS, Azure, GCP тощо), які пропонують схожі послуги, але з різними характеристиками, цінами та умовами. Ручний аналіз усіх факторів може зайняти багато часу і привести до суб’єктивних помилок. СППР допомагають автоматизувати процес порівняння.

Вибір неправильного сервісу може спричинити надмірні витрати або проблеми із масштабуванням у майбутньому. СППР допомагають врахувати ризики та зменшити ймовірність неправильного вибору. Тому задача розробки СППР для вибору хмарного сервісу є актуальною задачею.

Об’єкт дослідження – хмарні сервіси.

Предмет дослідження. Методи та інформаційні технології для вибору хмарних сервісів.

Мета роботи. Розробка системи підтримки прийняття рішень для визначення оптимального хмарного провайдера.

Основні задачі роботи. Для досягнення поставленої мети необхідно рішити такі задачі.

1. Проаналізувати особливості хмарних сервісів та моделей подання хмарних послуг.
2. Визначити особливості використання СППР при розробці ІТ-рішень.
3. Визначити критерії, за якими можна організувати порівняння хмарних сервісів.
4. Розробити алгоритм визначення оптимальної моделі надання хмарних послуг і на її основі методику для отримання оптимального провайдера.
5. Реалізувати розроблену методику у вигляді окремого програмного продукту.

Методи дослідження. Теоретичною основою дослідження є загальнонауковий аналітичний метод, а також системний підхід і праці провідних вчених з проблем дослідження і оцінювання хмарних сервісів. Для практичного вирішення поставлених задач використовувалися такі методи: загальнонауковий аналітичний метод; методи теорії прийняття рішення для розробки алгоритмічного забезпечення для СППР; методи алгоритмічного програмування, для створення СППР.

Наукова новизна роботи полягає в розробці двохетапної методики вибору хмарного провайдера на основі теорії нечітких множин та метода TOPSIS. А також розробки системи критеріїв для оцінки хмарних сервісів.

Практична значущість роботи полягає у створенні СППР, яка може використовуватись невеликими підприємствами для вирішення питання міграції в хмару.

1 АНАЛІЗ ВИКОРИСТАННЯ ХМАРНИХ СЕРВІСІВ

1.1 Хмарні сервіси: види, особливості та принципи використання

Хмарні сервіси – це використання комп'ютерних ресурсів, які безпосередньо не знаходяться поруч з користувачем і не керуються ним безпосередньо для забезпечення обчислювальної потужності. Це спосіб надання комп'ютерних ресурсів: оренда апаратної інфраструктури, доступ до дорогих програм та програм через Інтернет за невелику суму.

Наразі хмари складаються з тисяч серверів, кожен з яких розміщений у центрах обробки даних (ЦОД). Кожна хмара забезпечує ресурсами десятки тисяч програм та сервісів, якими одночасно користуються мільйони людей.

Відповідно до Національного інституту стандартів і технологій (NIST), визначення хмарних обчислень – це модель повсюдного, зручного мережевого доступу за вимогою до спільного пулу конфігурованих обчислювальних ресурсів (наприклад, мереж, серверів, сховищ, програм і послуги), які можна швидко надати та вивільнити з мінімальними зусиллями керівництва або взаємодії постачальника послуг (рис.1.1) [1].

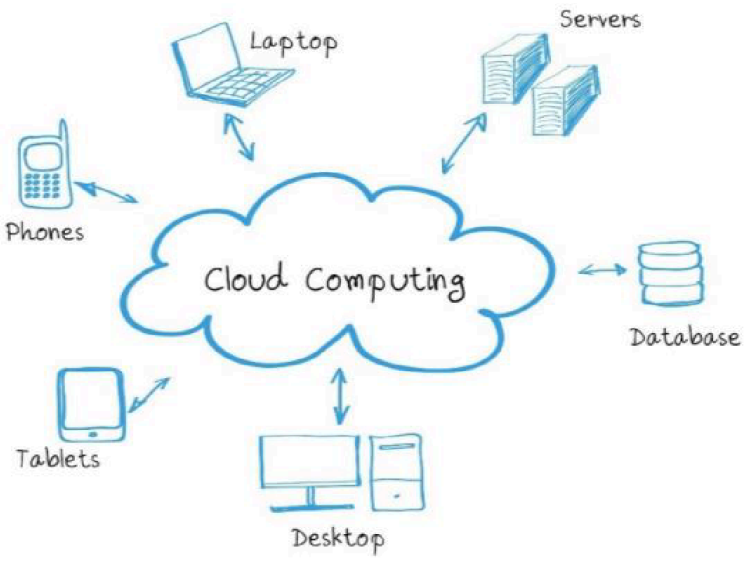

Рисунок 1.1 – Хмарні обчислення

Згідно з NIST, хмарна модель складається з п’яти основних характеристик, трьох моделей послуг хмарних обчислень і чотирьох моделей розгортання.

Наведемо п’ять характеристик хмарних обчислень:

1. Самообслуговування, тобто здатність легко й ефективно користуватися перевагами обчислювальних ресурсів (наприклад, пам’яті, обчисленнями тощо) та інших послуг, які клієнт споживає.

2. Широкий доступ до мережі – можливість використовувати широку мережу, до якої можуть отримати доступ кілька користувачів у багатьох регіонах.

3. Об’єднання ресурсів – групування ресурсів необхідне для ефективного надання обчислювальних ресурсів для різних потреб, використовуючи їх найбільш оптимізованим способом.

4. Еластичність – можливість змінювати розмір обчислювальних ресурсів (ЦП, пам’ять і сховище) вручну або автоматично відповідно до попиту на доступ до послуг, доступних у хмарі. Еластичність допомагає, наприклад, у періоди піку доступу.

5. Виміряна послуга, тобто можливість вимірювати та контролювати те, що споживається в хмарі. Таке вимірювання допомагає визначити достатність цих ресурсів, дає можливість налаштувати їх відповідно до попиту [2].

Можемо виділити чотири способи реалізації хмарних сервісів:

1. Приватна хмара – усі служби та обчислювальні ресурси (пам’ять, ЦП, мережі та сховище) знаходяться у власному приміщенні організації, і не використовуються спільно з іншими компаніями.

2. Публічна хмара – усі служби та обчислювальні ресурси доступні у середовищі віртуальних хмарних обчислень і до них можна отримати доступ через Інтернет. Найпоширенішими постачальниками публічних хмарних обчислень є: Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), Alibaba тощо.

3. Гібридна хмара – хмарна модель, яка складається з двох або більше типів хмарних обчислень (наприклад, дані та програми компанії знаходяться в приватній хмарі, а інші робочі навантаження – у публічній хмарі). У моделі дуже важливо, щоб компанія мала певний вид хорошого управління, оркестрування та переносимості між хмарами, щоб гарантувати, що інфраструктура пропонує ефективні ресурси для всіх своїх користувачів.

4. Багатохмарна стратегія – у той час як гібридні хмарні обчислення стосуються двох хмар одного типу, які співіснують, мультихмарна стратегія стосується двох або більше хмар, незалежно від типу (наприклад, компанія, яка використовує служби Azure та AWS).

Однак, будь-яка хмара повинна задовольняти п’яти характеристикам, наведеним вище.

Поява та розвиток хмарних обчислень призвело до створення кількох моделей хмарних сервісів. Моделі відрізняються принципом роботи, а також тим, що пропонують своїм користувачам. Яка модель підійде окремому користувачеві, залежить від того, яким із сервісів він хоче скористатися або на якому етапі розробки він знаходиться. Тобто, на якому рівні знаходиться їхня внутрішня структура інформаційної системи, сфери діяльності

Визначаючи моделі хмарних послуг, потрібно враховувати, за які рівні інфраструктури відповідає постачальник послуг, а які будуть у відповідальності користувача. Для кожної моделі хмарного сервісу повинні бути чітко визначені обов’язки кожної сторони щодо інфраструктури (рис.1.2).

Виділяють три основні моделі:

– Інфраструктура як послуга, або IaaS.
– Платформа як послуга, або PaaS.
– Програмне забезпечення як послуга, або SaaS.

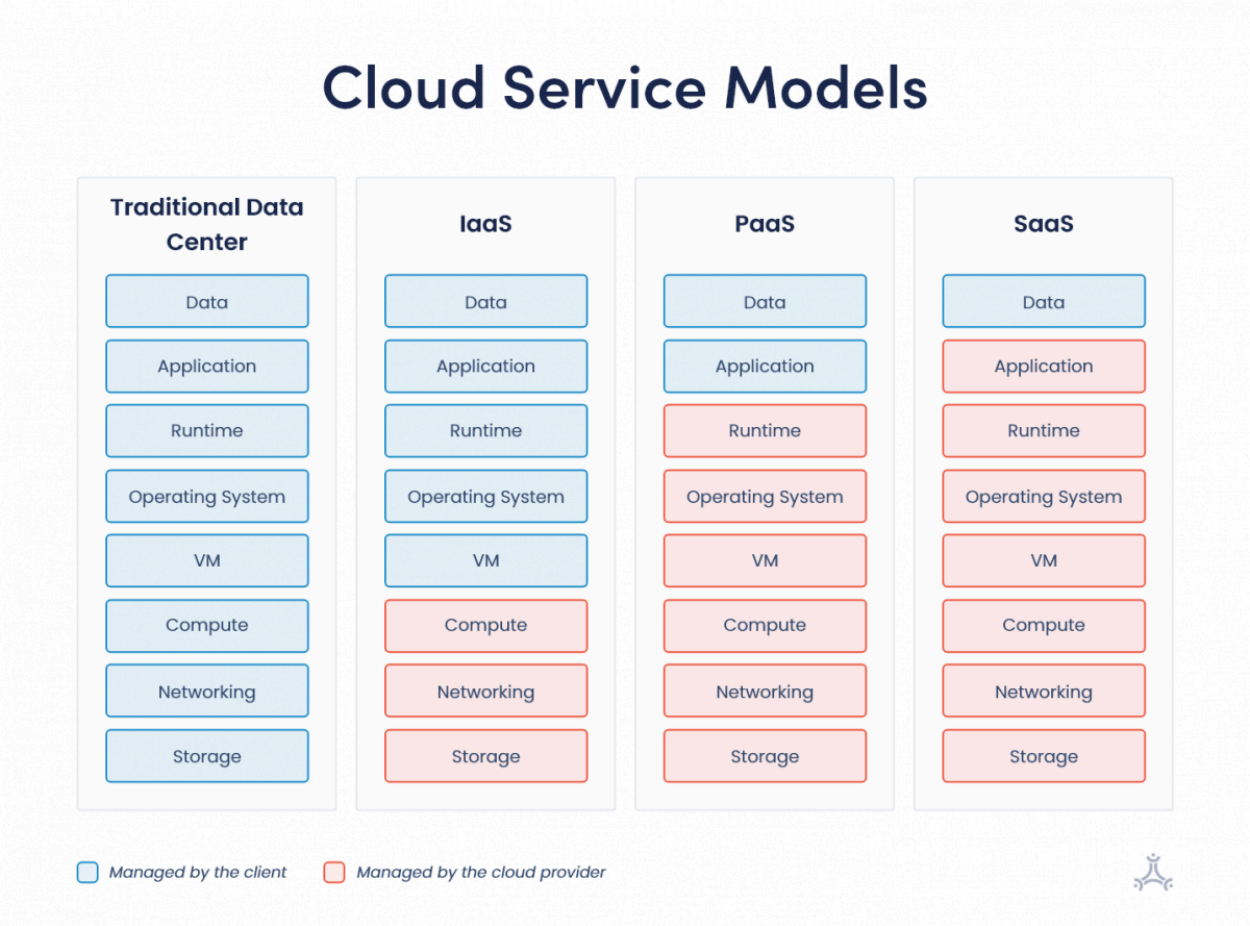

Рисунок 1.2 – Моделі хмарних сервісів [3]

Розглянемо більш детально перелічені моделі.

У моделі обслуговування інфраструктури як послуги або IaaS постачальник хмарних послуг відповідає за обчислювальну інфраструктуру. Однак компанія відповідає за керування віртуальними машинами, налаштування мереж і вибір операційної системи.

IaaS схожий на управління та обслуговування серверів у приватному центрі обробки даних. Тим не менш, компанії не потрібно "чіпати" сервери або турбуватися про те, щоб адміністратори присвячували години створенню всієї IT-інфраструктури. Тим не менш, компанії потрібні співробітники, які знають, як керувати інфраструктурними послугами, оскільки IaaS не знімає всіх обов’язків з клієнта.

Переваги рішень IaaS опишемо наступним чином.

– Знижує вартість капітальних витрат.

– Користувачі платять за послуги, які їм потрібні.

– Доступ до IT-ресурсів корпоративного рівня та інфраструктури.

– Користувачі можуть збільшувати та зменшувати масштаб ресурсів на основі своїх вимог у будь-який час.

Важливо пам’ятати, що, оскільки інфраструктура знаходиться в руках постачальника хмарних послуг, то в компанії можуть виникнути можливі простої у системі, який залишає веб-сайти клієнта та інші додатки недоступними.

Деякі з великих компаній, що пропонують інфраструктуру як послугу, включають Rackspace Cloud Servers, Google, Amazon EC2, IBM і Verizon

У платформі як послуга або PaaS клієнт несе відповідальність за додаток і програмне забезпечення, і йому не потрібно турбуватися про керування віртуальною машиною чи операційною системою. Постачальник послуг у цьому випадку пропонує клієнтам різні варіанти використанню своєї віртуальної машини.

Хоча клієнт відповідає за меншу кількість рівнів інфраструктури, його вибір також обмежений, оскільки він залежить від того, що може запропонувати постачальник послуг.

Деякі програми працюють краще на одному типі сервера та конфігурації, тому клієнт повинен знати, на якій службі PaaS найкраще розмістити його програму.

У рішень PaaS є значні переваги. По-перше, робота в середовищах PaaS створюється сильна спільнота підтримки, яка може допомогти команді розробників.

По-друге, провайдер PaaS займається всіма оновленнями, виправленнями та регулярним обслуговуванням програмного забезпечення, знімаючи відповідальність з клієнтів.

Оскільки не потрібно робити попередні інвестиції в апаратне та програмне забезпечення, зменшується вартість IT-інфраструктури.

Нарешті, PaaS-рішення забезпечують спрощене розгортання. Команда розробників може зосередитися на розробці хмарної програми, не турбуючись про інфраструктуру тестування та розгортання. Прикладом PaaS є Cloud Run від Google Cloud Platform (GCP). Клієнт може створювати та розгортати додатки різними мовами програмування, але лише сумісними з сервісом: Go, Python, Java, Node.js та NET.

Послуги програмного забезпечення як послуги (SaaS) пропонуються кінцевим користувачам, як правило, через браузер, повним чином: користувачеві не потрібно турбуватися про запуск програми у своєму середовищі, і йому не потрібно турбуватися про фізичні пристрої для зберігання та обчислювальних ресурсів [3].

Постачальник послуг виконує всі оновлення програм, виправляє помилки та технічне обслуговування. Користувачеві не потрібно турбуватися про ці дії з обслуговування.

Одним із прикладів SaaS є Slack, де у клієнта є вся інфраструктура, яку пропонує компанія. Користувача хвилює лише розміщення своїх даних та інформація в програмному забезпеченні чи додатку.

Невеликі компанії та стартапи можуть отримати вигоду від використання послуг SaaS, оскільки інвестиції в інфраструктуру та створення власного центру обробки даних (навіть якщо він невеликий) не є рентабельними.

Amazon, наприклад, має програму, яка пропонує кредити власникам малого бізнесу та засновникам стартапів. Іншими приклидами SaaS є Gmail і Salesforce.

Обмеження SaaS. Оскільки замовник використовує готове програмне забезпечення, він не контролює його функціональність. Вибираючи найкраще рішення SaaS, важливо враховувати потреби бізнес-додатків і обмеження постачальника послуг. Іншим обмеженням є контроль, який компанія залишає в руках постачальника послуг.

1.2 Аналіз використання СППР для підтримки ІТ-рішень

Система підтримки прийняття рішень (СППР) – це комп’ютерна інформаційна система, яка підтримує діяльність з прийняття бізнес- або організаційних рішень. СППР допомагають менеджерам та іншим особам, які приймають рішення, збирати корисну інформацію з необроблених даних, документів, особистих знань і бізнес-моделей для виявлення та вирішення проблем і прийняття рішень.

СППР – це інтерактивні програмні системи, які допомагають користувачам аналізувати великі обсяги інформації та отримувати доступ до даних із різних джерел для підтримки напівструктурованого та неструктурованого прийняття рішень. Вони забезпечують інтерактивний інтерфейс, який дозволяє користувачам маніпулювати даними, коригувати параметри та досліджувати різні сценарії. Основна мета СППР – допомогти особам, які приймають рішення, зробити обґрунтований вибір шляхом надання відповідних даних, аналізу та розуміння [4].

СППР має такі властивості. Перелічимо їх.

1. СППР надають інтерактивний інтерфейс, який дозволяє користувачам маніпулювати даними, коригувати параметри та досліджувати різні сценарії. Інтерактивність необхідна для перевірки різних гіпотез і розуміння потенційних результатів різних рішень.

2. Основна мета СППР – допомогти особам, які приймають рішення, зробити обґрунтований вибір. Підтримка надається за допомогою відповідних даних, аналізів і розуміння, які допомагають оцінити різні варіанти та вибрати найкращий курс дій.

3. СППР включають різні моделі, алгоритми та аналітичні інструменти для обробки даних і створення ідей. Інструменти містять прогнози, моделювання та оптимізаційні рішення, які допомагають зрозуміти складні ситуації та приймати кращі рішення.

4. DSS пропонують інтуїтивно зрозумілі та прості у використанні інтерфейси, що робить їх доступними для нетехнічних користувачів. Така зручність гарантує, що особи, які приймають рішення, можуть ефективно взаємодіяти з системою та використовувати її можливості, не потребуючи великих технічних знань.

Теоретично СППР використовується в різних сферах знань від організації до управління різними технологічними процесами.

Структура СППР складається з трьох основних компонентів:

1. Система управління моделями зберігає моделі, які менеджери можуть використовувати для прийняття рішень. Моделі використовуються для прийняття рішень щодо різних аспектів діяльності організації.

2. Інтерфейс користувача містить інструменти, які допомагають кінцевому користувачеві СППР орієнтуватися в системі.

3. База знань включає інформацію з внутрішніх джерел (інформація, зібрана в системі процесу транзакцій) і зовнішніх джерел (газети та онлайн-бази даних) [4].

Також існує декілька типів систем підтримки прийняття рішень.

СППР на основі управління комунікаціями дозволяє компаніям підтримувати завдання, над виконанням яких потрібно працювати більше ніж одній людині. Вона може містити такі інтегровані інструменти, як Microsoft SharePoint Workspace і Google Docs.

СППР на основі моделі дозволяє отримати доступ до фінансових, організаційних і статистичних моделей і керувати ними. Дані збираються, а параметри визначаються на основі інформації, наданої користувачами. Інформація створюється в моделі прийняття рішень для аналізу ситуацій.

СППР на основі знань надає фактичні та спеціалізовані рішення для ситуацій, використовуючи збережені факти, процедури, правила або інтерактивні структури прийняття рішень, такі як блок-схеми.

СППР, що управляється документами, керує неструктурованою інформацією в різних електронних форматах.

Якщо говорити про переваги використання СППР, то для організацій вони проявляються наступним чином.

Система підтримки прийняття рішень підвищує швидкість і ефективність прийняття рішень, що реалізується завдяки тому, що СППР може збирає та аналізує дані в реальному часі. Це сприяє навчанню всередині організації, оскільки необхідно розвинути спеціальні навички для впровадження та використання СППР в організації.

Додатково система автоматизує монотонні управлінські процеси, а значить, більше часу керівника може витрачатися на прийняття рішень. Власне це покращує міжособистісне спілкування всередині організації.

Системи підтримки прийняття рішень також має і певні недоліки:

Витрати на розробку та впровадження СППР є досить значними капіталовкладеннями, що робить такі системи менш доступним для невеликих організацій.

Компанія залежить від СППР, оскільки вона інтегрована в щоденні процеси прийняття рішень для підвищення ефективності та швидкості. Однак менеджери схильні занадто сильно покладатися на систему, що позбавляє суб’єктивного аспекту прийняття рішень.

З іншого боку, використання СППР призводить до перевантаження інформацією, оскільки інформаційна система має тенденцію розглядати всі аспекти проблеми. Це створює дилему для кінцевих користувачів, оскільки вони мають кілька варіантів вибору.

Системи підтримки прийняття рішень (СППР) стали невід’ємною частиною сучасного управління, особливо в сфері інформаційних технологій (IT). Вони надають можливість оптимізувати процеси прийняття рішень, зменшувати ризики та забезпечувати більшу точність і ефективність.

IT-галузь відзначається швидкими змінами, високим рівнем невизначеності та необхідністю прийняття рішень у стислі терміни. СППР відіграють важливу роль у:

1. Аналізі великих обсягів даних. СППР дозволяють обробляти великі масиви інформації за допомогою інструментів бізнес-аналітики (BI) та методів машинного навчання. Це забезпечує аналітикам можливість швидко отримувати певні прогнози для ухвалення рішень.

2. Оптимізації ресурсів. СППР допомагають раціонально розподіляти ресурси (сервери, мережі, хмарні потужності) на основі аналізу попиту та прогнозів.

3. Виборі інноваційних рішень. СППР дозволяють оцінювати ефективність нових технологій, таких як впровадження штучного інтелекту чи блокчейн, на основі критеріїв вартості, ризиків тощо.

4. Розв’язанні стратегічних завдань. Ухвалення рішень щодо довгострокового розвитку IT-інфраструктури, вибору архітектурних рішень або інтеграції з іншими системами часто базується на результатах моделювання у СППР [3, 4].

Основні функції СППР у підтримці IT-рішень

1. Моделювання сценаріїв. СППР дозволяють створювати й аналізувати різні сценарії, враховуючи змінні зовнішнього середовища та внутрішні параметри. Наприклад, у разі масштабування інфраструктури моделюються витрати, час реалізації та потенційні ризики.

2. Інтеграція з системами даних. Інструменти СППР інтегруються з базами даних, хмарними сервісами та програмами моніторингу для забезпечення актуальних даних для аналізу.

3. Візуалізація даних. Діаграми, графіки та інтерактивні інтерфейси сприяють кращому розумінню складної інформації.

4. Автоматизація рутинних процесів. СППР можуть автоматично генерувати звіти, надавати рекомендації або виявляти проблеми, які потребують уваги.

Переваги використання СППР для IT-рішень включають високу швидкість та точність, оскільки ці системи зменшують час на аналіз інформації і забезпечують більшу точність завдяки застосуванню об’єктивних

алгоритмів. Вони сприяють підтримці командної роботи, надаючи можливості для спільної участі декількох користувачів у процесі прийняття рішень. Також СППР допомагають прогнозувати ризики, виявляючи потенційні загрози на основі аналізу минулих даних і тенденцій. Окрім цього, системи відзначаються гнучкістю, що дозволяє їм легко адаптуватися до специфічних потреб організації або проєкту.

Значна частина останніх досліджень пов'язана з рішенням про впровадження хмарних сервісів. В частині робіт пропонується використовувати різні методи для прийняття рішень. В роботі [4] пропонується здійснювати вибір сервісу на основі відгуків користувачів. Однак ця модель охоплює суб'єктивну оцінку клієнтів і оцінку сторонньої організації, що може викликати певні незручності при її використанні. В роботі [5] використовується метод аналізу ієрархій, як основа для побудови СППР для вибору хмарних сервісів. Багатокритеріальні методи розглядаються в роботі [6] і пропонується їх використовувати на основі сценарію для вибору IaaS.

Автори роботи [7] запропонували систему, яка дозволяє адаптивно коригувати виділені ресурси в умовах хмарних сервісів. Система використовує онлайн-модель прогнозування попиту на ресурси для визначення короткострокових потреб у ресурсах. Запропонований підхід використовує спеціальну програму всередині кожної системи прийняття рішень і підтримки, що може бути неприйнятним для багатьох систем і платформ хмарних сервісів.

В роботі [8] запропонована структурі ієрархії управління на основі вимог до розміщення програм і послуг. Запропонована ієрархічна структура автоматично створює та підтримує ієрархію, яка потім може бути використана керуючою програмою. Таким чином, запропонована структура використовує поділ на групи при управлінні дочірніми вузлами, визначеними ієрархією. Ієрархія вузлів побудована та керована на вершині фізичної мережі хмарних сервісів.

У роботах [9,10] автори обговорюють спроби розробки методів, алгоритмів і СППР. Дослідники в основному зосереджені на сферах надання, розподілу, планування потужностей, порівняння та масштабування ресурсів хмарних сервісів. У той же час вони описують багато систем із програмно-визначеними технологіями та програмними платформами для надання та моніторингу ресурсів, включаючи точні методи, механізми прогнозування навантаження тощо.

У роботі [10] запропоновано концепцію створення технологій глобального обміну інформацією в хмарі та ринково орієнтовану архітектуру розподілу ресурсів у хмарах. Ці технології базуються на необхідності зближення конкуруючих IT-парадигм. Для успішного впровадження хмарних обчислень автори представили різні варіанти вирішення проблеми вибору відповідних хмарних інструментів, щоб розкрити наявний потенціал на практиці.

Загалом можемо зробити висновок, що існує обмежена кількість досліджень, які пропонують СППР для вибору хмарних служб. Аналіз існуючих досліджень показує, що вони пропонують рішення з використанням максимум двох моделей рішень, а їхні структури не можна настроювати відповідно до конкретних вимог підприємства.

Системи підтримки прийняття рішень є ефективним інструментом для вибору оптимального хмарного сервісу, враховуючи складність багатокритеріального аналізу. Вони дозволяють оцінювати альтернативи за такими критеріями, як продуктивність, вартість, рівень безпеки, масштабованість та доступність технічної підтримки. Завдяки автоматизації процесу аналізу СППР зменшують суб’єктивність і підвищують точність вибору. Ці системи можуть інтегрувати експертні оцінки та вагові коефіцієнти для налаштування рекомендацій відповідно до потреб конкретної організації. Вибір між моделями IaaS, PaaS чи SaaS також може бути оптимізований за допомогою аналізу, що включає специфічні вимоги до інфраструктури. Інтерфейс СППР забезпечує зручність у використанні та

візуалізацію порівняльних характеристик доступних провайдерів. У підсумку, використання СППР сприяє економії часу, зниженню ризиків і прийняттю обґрунтованих рішень для впровадження хмарних технологій.

1.3 Постановка задачі

У сучасному бізнес-середовищі організації часто стикаються з необхідністю вибору хмарного сервісу, який найбільше відповідає їхнім потребам. Це рішення є багатокритеріальним, оскільки залежить від таких параметрів, як вартість, продуктивність, масштабованість, рівень безпеки, технічна підтримка, географічне розташування серверів тощо. Відсутність автоматизованих інструментів для комплексного аналізу цих критеріїв ускладнює вибір і може призвести до неправильного рішення, яке негативно вплине на ефективність бізнесу.

Проведений аналіз показав, що така задача часто розв’язується як вузьке рішення, направлене на підтримання вибору конкретної технології.

Тому, щоб забезпечити більш комплексне рішення, необхідно розробити СППР, що можна налаштовувати, для вибору найбільш прийнятних хмарних сервісів. Система також призначена для користувачів з обмеженими знаннями щодо хмарних сервісів та підходів до прийняття рішень.

Метою розробки є створення СППР, яка допоможе користувачам вибрати оптимальний хмарний сервіс, враховуючи їхні вимоги та обмеження. Система повинна бути здатною аналізувати альтернативи на основі багатьох критеріїв, забезпечуючи прозорість і обґрунтованість прийнятих рішень.

Система повинна використовувати методи прийняття рішень, які враховують багатокритеріальність задачі, певну суб’єктивність при оцінюванні конкретних платформ.

Система повинна включати набір критеріїв, які мають вирішальний вплив на вибір хмарного сервісу. А методика оцінювання альтернатив за цими критеріями повинна здійснюватися з використанням методів

багатокритеріального прийняття рішень (нечіткої логіки – для вибору моделі хмарного сервісу, TOPSIS – для вибору конкретного сервісу).

В системі необхідно передбачити можливість реалізувати механізм інтеграції експертних оцінок для встановлення ваг критеріїв.

Також користувач повинен мати можливість управляти критеріями, щоб забезпечити дотримання специфічних вимог.

Реалізувати зручний інтерфейс для візуалізації результатів аналізу, рекомендацій і прозорості вибору.

Система повинна бути орієнтована на невеликий та середній бізнес.

Обсяг даних про хмарні сервіси обмежується публічно доступною інформацією.

Критерії вибору обмежуються загальновизнаними характеристиками, такими як вартість, надійність, масштабованість, продуктивність і рівень підтримки.

Результатом розробки повинна стати СППР, яка здатна ідентифікувати та рекомендувати оптимальну модель хмарного сервісу (IaaS, PaaS, SaaS) і конкретного провайдера на основі визначених критеріїв.

Система забезпечить підвищення швидкості, точності й обґрунтованості прийняття рішень, а користувач зможе адаптувати її під свої потреби.

1.4 Висновки до розділу 1

Підприємства впроваджують технології хмарних обчислень, які надають різноманітні можливості, такі як масштабованість, гнучкість і доступність за вимогою. Хмарні сервіси забезпечують фінансові переваги, включаючи зниження витрат на існуючі програми, а також доступність інноваційних ІТ за прийнятною експлуатаційною вартістю. Серед основних рушійних сил хмарних сервісів економічність і спрощення доставки та експлуатації програмного забезпечення. Завдяки пропонованим перевагам,

таким як конкурентні переваги, значна економія коштів і вдосконалені бізнес-процеси, хмарні сервіси є привабливою пропозицією для багатьох малих і середніх підприємств.

Використання СППР для підтримки ІТ-рішень, зокрема, у виборі хмарних сервісів, надає значні переваги в умовах сучасного динамічного ринку. Завдяки своїй здатності обробляти дані, моделювати сценарії та автоматизувати аналіз, ці системи є важливим інструментом для управління ІТ-проєктами та оптимізації бізнес-процесів. Однак для максимального ефекту необхідно враховувати виклики, пов’язані з їх впровадженням і використанням.

2 ПРОЄКТУВАННЯ СППР ДЛЯ ВИБОРУ ХМАРНИХ СЕРВІСІВ

2.1 Аналіз критеріїв роботи хмарних сервісів

Аналіз показує, що в літературі зустрічається різний набір характеристик хмарного сервісу. Оскільки пропонується СППР саме з точки зору користувача хмарних сервісів, то пропонується вибір характеристик здійснювати на основі індексу вимірювання послуг (SMI) [12], який розроблено Консорціумом ініціативи вимірювання хмарних послуг (CSMIC), щоб допомогти організаціям вимірювати хмарні послуги на основі їхніх вимог. SMI містить понад 40 характеристик, однак це досить багато для користувачів даної системи, тому пропонується використати основні характеристики, за допомогою яких можуть провести оцінку як експерти, так і звичайні користувачі.

Перш за все, для визначення моделі пропонується використовувати такі показники:

– вартість,
– рівень контролю,
– безпека.

Вартість визначає кількість грошей, витрачених на послугу. Загалом, вона може характеризуватися поточними витратами: типові моделі ціноутворення постачальників хмарних послуг базуються на ресурсах, необхідних клієнту. Ресурси можуть бути, у випадку PaaS/IaaS, використаною інфраструктурою (наприклад, процесор, оперативна пам’ять або пам’ять) або, у випадку SaaS, кількістю користувачів, які користуються послугою. Існують також інші схеми ціноутворення, і постачальники часто мають калькулятор цін або принаймні таблицю цін, де користувач може оцінити місячні витрати.

Відповідними також є можливості оплати (наприклад, щомісяця або щороку, автоматизація платежів), а також чи правила оплати є прозорими, масштабованими та передбачуваними [13].

Також при оцінюванні даного показника слід враховувати вартість придбання та переходу: протягом періоду налаштування та розгортання сервісу можуть бути певні початкові витрати. Ці витрати виникають через те, що постачальник стягує плату за налаштування послуги або через те, що є певні витрати під час розробки, міграції та розгортання послуги. Крім того, витрати можуть виникнути через необхідну реструктуризацію інфраструктури або людських ресурсів. Крім того, приховані або неочевидні витрати, такі як витрати на навчання співробітників, можуть мати великий вплив на загальні початкові витрати, враховуючи, що як найм інструкторів, так і втрата робочого часу співробітників.

Пристосованість сервісу пов’язана з можливостями користувача швидко змінити напрямок, стратегію чи тактику за допомогою конкретної послуги. Відповідно характеризується масштабованістю: особливо для хмарних служб масштабованість є центральним атрибутом, оскільки це один із аргументів продажу, що ресурси можна легко масштабувати відповідно до вимог користувача. Сервіс повинен мати зручний спосіб здійснити це [12].

Також важливим є портативність, тому що існує кілька сценаріїв, коли необхідно перенести послугу від одного постачальника до іншого (юридичні причини, ціни чи нові вимоги). Щоб утримувати таку послугу переміщення в межах певної ціни та терміну, має бути гарантована мобільність. Портативність гарантує відсутність прив’язки до постачальника, що є великою проблемою для багатьох постачальників хмарних послуг.

Також важливим є розширюваність, тобто можливість додавати нові функції, оскільки змінились бізнес-вимоги змінилися.

Також очевидною вимогою до моделі є безпека та конфіденційність Ця категорія містить атрибути, пов’язані з ефективністю служби постачальника хмарних послуг і контролю доступу до даних. Служба повинна містити заходи для забезпечення конфіденційності даних. Це можна забезпечити за допомогою зашифрованих з’єднань і складних механізмів контролю доступу, що включає, наприклад, багатоетапний процес входу в систему. За допомогою

таких заходів гарантується, що дані можуть використовуватися та змінюватися лише персоналом із відповідними повноваженнями [12].

Крадіжка та втрата даних зазвичай призводять до фінансової чи репутаційної шкоди. Або тому, що втрачену роботу потрібно повторити, або тому, що важливі комерційні таємниці розголошуються. Таким чином, постачальник послуг повинен забезпечити високий рівень конфіденційності даних і складні методи запобігання втраті даних [13].

Різні співвідношення зазначених характеристик дають змогу користувачу визначитися з моделлю хмарного сервісу.

Для вибору конкретного провайдера пропонується розглянути такі критерії:

– продуктивність,
– функціональність,
– наявність підтримки,
– вартість.

Розглянемо їх більш детально.

Продуктивність пов’язана з можливостями та функціями, які надає служба. Вона може характеризуватися часом відповіді служби. Низький час відгуку гарантує плавну взаємодію з сервісом, що важливо для задоволення користувача. Географічне розташування хмарної служби та її серверів часто має великий вплив, тому службу слід розташовувати не надто далеко від своїх користувачів.

Кожна послуга надає певну функціональність, і клієнт повинен вивчити діапазон доступних функцій і можливостей. Чим більше функцій має служба, тим більша ймовірність вирішення поточних і майбутніх завдань. Функціональність може, залежно від послуги, включати різні атрибути.

Наявність підтримки є важливою характеристикою. Вона характеризує властивості організації-постачальника. Більшість хмарних послуг є предметом контракту та/або угоди про рівень обслуговування (SLA). Такий контракт або угода містить важливу інформацію про здібності та ставлення

постачальника. Центральними є гарантії наявності та виконання, а також компенсація, якщо така гарантія не може бути виконана. Таким чином, питання полягає в тому, чи договір/угода є достатнім для керування хмарною службою, а також зменшує ризик збою служби на прийнятному рівні. Крім того, договір може містити важливі положення про права на дані клієнтів, ліцензії на програмне забезпечення та інтелектуальну власність.

Постачальники відрізняються за тим, як вони взаємодіють зі своїми клієнтами, і оскільки взаємодія відбувається регулярно (наприклад, щомісячний процес виставлення рахунків), дуже важливо, щоб ця взаємодія відбувалася таким чином, щоб клієнту було легко виконувати свої дії. Важливими є стабільність і прозорість бізнесу провайдера, щоб клієнт міг планувати заздалегідь і не зіткнутися з несподіванками, а також простота всього процесу виставлення рахунків. Крім того, комунікація також є центральною точкою. У кращому випадку з провайдером можна зв’язатися по телефону, надати грамотну і професійну допомогу [8].

Сертифікати гарантують, що постачальник відповідає певним стандартам і вказівкам. Оскільки хмарні служби порівняно з традиційною IT-інфраструктурою є менш прозорими щодо всіх методів і механізмів у фоновому режимі, сертифікати можна використовувати як відправну точку для оцінки прихованих аспектів служби.

Визначені характеристики хмарних сервісів мають велике значення для здійснення впливу на те, як підприємства будуть використовувати дані сервіси та на скільки вони будуть задоволені можливістю міграції в хмару. Ці ключові характеристики надають користувачам визначити найкращий сервіс з точки зору гнучкості, ефективності та економічності.

2.2 Розробка моделі вибору хмарних сервісів

Зазвичай прийняття правильних рішень є складною задачею, оскільки кількість альтернатив є значною, а кожне рішення має ряд переваг і недоліків. Враховуючи складність прийняття відповідних рішень і зіткнення з різними результатами, моделі прийняття рішень полегшують процес вибору між різними доступними варіантами [14]. Моделі прийняття рішень забезпечують рамки та рекомендації, щоб зробити найкращий вибір і краще керувати процесом прийняття рішень. Вони є інструментами, які можна використовувати для прийняття ефективних рішень у випадках, коли прийняття рішення може бути складним.

Різні моделі прийняття рішень забезпечують основу для аналізу ситуації, розгляду ймовірних рішень і, зрештою, ведуть до обґрунтованого рішення. Вони також пропонують ряд підходів до прийняття ефективних рішень на основі контексту рішення та альтернатив, які на нього впливають. Використання моделей прийняття рішень для прийняття структурованих рішень значною мірою полегшує прийняття відповідних рішень і може зменшити ймовірність невдалих рішень.

Моделі прийняття рішень можна поділити різним способом. По-перше, процес прийняття рішень можна розділити на дві моделі: нормативну та описову.

Нормативні моделі прийняття рішень – це теоретичні моделі, які допомагають менеджерам робити вибір шляхом вивчення співвідношень в системі. Ці моделі включають рішення, прийняті раціональними особами, враховуючи максимальну корисність для отримання оптимального варіанту за будь-яких невизначених обставин, які можуть вплинути на процес прийняття рішень. Таким чином, нормативні моделі базуються на розгляді цілей особи, яка приймає рішення, і ймовірних результатів, щоб прийняти найкраще можливе рішення з доступних варіантів. Таким чином, процес прийняття рішень базується на стандартах і нормах і керується конкретними правилами

та вказівками для прийняття найкращого з можливих рішень. Іншими словами, нормативні моделі прийняття рішень просто посилаються на той факт, що найкращий вибір – це той, який призводить до найкращого результату. Такі моделі зазвичай використовують присвоєні числові значення параметрам, щоб зробити процес прийняття рішень раціональним.

Другим підходом до прийняття рішень є описовий підхід. Він вивчає, як люди використовують свій досвід для прийняття рішень у реальних ситуаціях. Увага зосереджується на трьох елементах, що впливають на прийняття рішень: характеристики, пов’язані з особою, яка приймає рішення, зокрема знання та досвід; чинники, пов’язані з роботою, такі як рівень її складності; а також фактори навколишнього середовища. Ці моделі базуються на аналізі того, як саме люди приймають рішення у реальних ситуаціях. Зазвичай людям рідко вдається слідувати раціональному процесу (як у нормативних моделях), оскільки прийняття рішень часто відбувається несвідомо і базується на попередньому досвіді. Тому, замість використання числових значень, процес прийняття рішень ґрунтується на розповідях про можливі наслідки. На прийняття рішень впливають особисті ідентичності та пов’язані з ними соціальні очікування. У цьому контексті серед різних сценаріїв майбутніх подій найбільш узгоджений сценарій впливає на остаточний вибір [14].

Проблема прийняття рішень розглядає єдину сукупну міру або критерій, наприклад вартість. Альтернатива з найкращим значенням (з огляду на критерій) визначається як остаточне рішення. Це класична задача оптимізації. Наприклад, якщо критерієм є мінімальна вартість, рішенням є альтернатива з найнижчою вартістю. Для пошуку рішення в цих випадках можна використовувати різноманітні методи оптимізації на основі функціонального опису та форми проблеми, такі як дискретна оптимізація, лінійне та нелінійне програмування. У всіх випадках критерієм є цільова функція, а вимогами до альтернатив є обмеження задачі оптимізації [15].

Багатокритеріальні проблеми прийняття рішень – це методи, які використовуються для вирішення проблем прийняття рішень за більш ніж одним критерієм. Коли є різноманітні фактори, які слід враховувати, прийняття рішень ускладнюється [16]. З іншого боку, слід враховувати кілька факторів. Багатокритеріальне прийняття рішень відноситься до складних ситуацій, які передбачають різні варіанти вибору для прийняття рішення.

Даний інструмент використовується у випадках, коли є різні альтернативи для розгляду. Кожен якісний або кількісний критерій аналізується, щоб визначити, чи є він корисним чи небажаним для результату.

У цих випадках особа, яка приймає рішення, може зіткнутися з двома загальними та основними підмножинами:

• Методи багатоцільового прийняття рішень (MODM): у цих методах атрибути та цілі неявні, а можливі варіанти – нескінченні/нечіткі.

• Багатоатрибутні методи прийняття рішень: це проблеми з кінцевими можливими відповідями, чіткими цілями та атрибутами.

У процесі прийняття рішень використовуються різноманітні методи, що забезпечують обґрунтований вибір оптимального варіанта серед альтернатив.

Для рішення задач багатокритеріальної оптимізації широко використовується метод аналізу ієрархій (MAI) через його простоту, зручність використання та значну адаптивність [16]. Підхід ґрунтується на ідеї, що оцінки повинні бути послідовними, а також випливає з ідеї, що суперечливі оцінки мають тенденцію відбуватися між варіантами, які, здається, мають мінімальне значення для менеджера, який приймає рішення.

Метод дозволяє структурувати проблему як ієрархію з кількох рівнів: мета, критерії та альтернативи. Основна перевага методу — врахування суб’єктивних оцінок через попарне порівняння критеріїв.

Мета ставиться на верхній рівень ієрархії, нижче розташовуються критерії та підкритерії, а на самому низькому рівні – альтернативи, які потрібно оцінити. Порівняння критеріїв і альтернатив: Для кожної пари елементів на одному рівні ієрархії експерт або група експертів проводять

парні порівняння, оцінюючи важливість або перевагу одного елемента перед іншим. Оцінка проводиться за шкалою від 1 до 9, де 1 означає рівність, а 9 – повну перевагу одного елемента. Після оцінки кожної пари елементів будується матриця порівнянь для кожного рівня ієрархії. Матриця містить значення, що відображають відносну важливість кожного елемента щодо іншого.

З матриці порівнянь розраховуються власні значення для кожного елемента, що дозволяє визначити вагові коефіцієнти для критеріїв, підкритеріїв і альтернатив. Формула для розрахунку ваг критеріїв:

$$w_i = \frac{\sum_{j=1}^{n} a_{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}} \qquad (2.1)$$

Метод дозволяє чітко структурувати складні проблеми і визначити важливість різних критеріїв, його можна використовувати як для великих, так і для малих проблем. Всі етапи оцінки та прийняття рішень є зрозумілими і відкритими для всіх учасників процесу.

В той же час метод залежить від експертних оцінок, які можуть бути суб’єктивними. Зі збільшенням числа альтернатив або критеріїв кількість парних порівнянь зростає експоненційно, що може ускладнити процес.

DEA (Data Envelopment Analysis) – це метод аналізу ефективності, який використовується для оцінки продуктивності одиниць, які здійснюють подібні операції, таких як підприємства, організації чи інші об’єкти, порівнюючи їх відносно певних критеріїв. DEA є основою для вимірювання ефективності в умовах, коли існує кілька входів і виходів, і не можна безпосередньо порівняти ці величини між різними одиницями.

DEA базується на лінійному програмуванні та аналізує співвідношення між входами та виходами для визначення "ефективності" одиниць. Він не потребує заздалегідь визначених параметрів або функцій, що описують

процеси виробництва. Натомість, DEA використовує емпіричні дані для побудови "фронтиру ефективності" – оптимального рівня продуктивності, на якому базуються порівняння.

У контексті DEA одиниці порівняння (DMUs) можуть бути підприємствами, відділами компаній або навіть окремими людьми, які виконують однакові функції. Вони мають певну кількість входів (ресурси, які використовуються в процесі) та виходів (результати, які досягаються).

Для порівняння одиниць ефективності DEA формує групи однорідних одиниць, звані "peer groups". Групи складаються з одиниць, що виконують подібні функції. Всі одиниці, що належать до певної групи, оцінюються на основі своїх відносних характеристик.

Виробнича межа (Production Frontier) – це оптимальний рівень ефективності, на якому одиниці можуть працювати, використовуючи свої ресурси. DEA будує цю межу на основі даних про входи та виходи всіх одиниць у вибірці. Всі одиниці, які знаходяться на цій межі, вважаються максимально ефективними.

Одиниці, які знаходяться на виробничій межі, оцінюються як ефективні, тоді як одиниці, що знаходяться під межою, вважаються неефективними. DEA визначає, наскільки кожна неефективна одиниця відстає від оптимального рівня ефективності. Порівняння одиниць за допомогою DEA дозволяє з’ясувати, які саме фактори впливають на продуктивність. Одиниці, які не досягають високої ефективності, можуть використовувати це порівняння для визначення шляхів покращення своїх операцій [14,18].

DEA не потребує чітко визначених функцій для опису процесів перетворення входів в виходи. Метод дозволяє одночасно враховувати кілька входів і виходів, що робить його потужним інструментом для багатокритеріального аналізу. Він дозволяє знаходити одиниці, які працюють на оптимальному рівні, і використовувати їх як еталон для інших.

Потрібно зазначити, що результати DEA можуть бути дуже чутливими до якості та точності вхідних даних. DEA порівнює одиниці лише на основі даних, без врахування факторів, що можуть бути важливими, але не були включені в модель. Якщо кількість одиниць порівняння дуже велика, процес аналізу може стати складним і вимагати великих обчислювальних ресурсів.

TOPSIS є корисним методом для вирішення реальних проблем багатокритеріального прийняття рішень. Основний принцип TOPSIS досить простий. Він заснований на ідеї зміщеної оптимальної точки. Метод визначає альтернативу, що є найкращою за відстанню до ідеального рішення. Алгоритм включає нормалізацію даних, зважування критеріїв, обчислення ідеальних та антиідеальних рішень і визначення відстаней.

ELECTRE та PROMETHEE використовуються для побудови часткового ранжування альтернатив. Вони враховують порогові значення критеріїв, що дозволяє моделювати складні системи.

Методи нечіткої логіки (fuzzy logic) використовуються для моделювання складних систем, де традиційні точні методи аналізу не підходять через невизначеність, неповноту або нечіткість вхідних даних. Нечітка логіка дозволяє враховувати нечіткі (нечітко визначені) значення, такі як "високий", "низький", "теплий", "холодний", і використовує їх для прийняття рішень або управління [18,19].

Ключовою ідеєю методів є розширення поняття множини так, що об’єкт може частково належати до множини. Кожному елементу присвоюється ступінь належності, значення якого знаходиться в діапазоні [0, 1].

Функції належності визначають, наскільки кожен елемент входить до нечіткої множини. Типові функції належності включають трикутні, трапецієподібні, гаусові та інші варіанти залежності.

Величини описуються словами (наприклад, "низький тиск", "висока температура"), які переводяться в нечіткі множини для обчислень.

Нечіткі системи використовують набір правил, наприклад: Якщо температура "висока", то швидкість вентилятора "швидка". Якщо температура "низька", то швидкість вентилятора "повільна".

Нечітке моделювання використовується для створення моделей, що описують поведінку систем із використанням лінгвістичних змінних і функцій належності. Наприклад, моделювання клімату, економічних процесів або поведінки клієнтів.

Нечітке управління застосовується для автоматичного управління складними системами (наприклад, системи клімат-контролю, робототехніка). Метод базується на нечітких правилах і прийнятті рішень у реальному часі. Процес, у якому нечіткі правила використовуються для отримання результатів на основі вхідних значень, називається нечітким виведенням.

Існують два основних підходи до нечіткого виведення. Мамдані-виведення використовується для керуючих систем. Результатом є нечітка множина, яку зазвичай дефазифікують для отримання чітких значень.

Виведення Такагі-Сугено (T-S) застосовується в моделях із математичними виразами для висновків. Результат – числове значення.

Методи нечіткої логіки дозволяють працювати з неповними та нечіткими даними. Вони є інтуїтивно зрозумілими, оскільки використовують лінгвістичні змінні. Методи застосовуються до нелінійних і складних систем.

Однак методи вимагають ретельного налаштування функцій належності та правил та є чутливими до кількості та якості вхідних даних.

Досить часто використовуються евристичні методи, які базуються на інтуїтивних та емпіричних правилах. Їх перевага – швидкість прийняття рішень, проте вони не гарантують оптимального результату. Серед них можна виділити метод сценаріїв, за допомогою якого досліджують різні сценарії розвитку подій, що дозволяє оцінити можливі наслідки кожного варіанту. SWOT-аналіз є оцінкою сильних сторін (Strengths), слабких сторін (Weaknesses), можливостей (Opportunities) та загроз (Threats). Експертні

оцінки грунтуються на використанні думок спеціалістів для отримання обґрунтованих оцінок.

Кожен з розглянутих методів має свої переваги та недоліки. Їх вибір залежить від специфіки задачі, доступних ресурсів та рівня невизначеності даних. У багатьох випадках ефективним є поєднання кількох підходів для отримання найбільш обґрунтованого рішення.

Тому для визначення хмарного сервісу пропонується використовувати поєднання двох методів: для вибору моделі надання послуг – метод нечіткої логіки, для вибору конкретного сервісу – TOPSIS.

Рішення щодо вибору моделі хмарного сервісу (IaaS, PaaS, SaaS) залежить від кількох критеріїв, таких як витрати, рівень контролю та безпека. Оскільки ці критерії є якісними та можуть бути суб’єктивно оцінені, застосування класичних кількісних методів не завжди доцільне. Натомість алгоритм нечіткої логіки дозволяє працювати з неточними даними. Критерії, такі як рівень безпеки чи контроль, важко оцінити чисельно. Нечітка логіка дозволяє використовувати лінгвістичні змінні, наприклад, "низький", "середній", "високий". Правила нечіткої логіки імітують спосіб мислення експертів, які приймають рішення на основі сукупності факторів. Система легко налаштовується для врахування нових критеріїв чи змін у важливості існуючих параметрів. Користувачам легко інтерпретувати результати завдяки прозорості правил. Таким чином, нечітка логіка забезпечує ефективне прийняття рішень у випадках, коли критерії є якісними, а дані – неповними або нечіткими [20].

Після визначення моделі хмарного сервісу необхідно обрати найкращий сервіс серед доступних альтернатив, враховуючи такі кількісні критерії, як продуктивність, вартість, функціональність та наявність підтримки. У цьому випадку алгоритм TOPSIS є найбільш підходящим, оскільки він забезпечує збалансований підхід до оцінки. TOPSIS дозволяє врахувати як переваги (максимізацію критеріїв, наприклад, продуктивності), так і недоліки (мінімізацію, наприклад, витрат). Алгоритм базується на обчисленні

відстаней до ідеального (найкращого) та антиідеального (найгіршого) рішень, що легко реалізується. TOPSIS дозволяє призначати ваги критеріям залежно від їхньої важливості, що підвищує точність оцінки. Метод генерує повний ранг альтернатив, що спрощує остаточний вибір.

TOPSIS успішно використовується в задачах, де необхідно оцінити варіанти з багатьма критеріями, наприклад, у виборі постачальників, аналізі продуктивності систем чи управлінні ресурсами.

Перейдемо до опису нечіткої моделі.

Нечіткий підхід забезпечує моделювання експертного процесу прийняття рішень, дозволяючи працювати з нечіткими або неповними даними. Для задачі вибору моделі хмарного сервісу (IaaS, PaaS, SaaS) були визначені наступні лінгвістичні змінні та функції приналежності.

Визначимо лінгвістичні змінні.

Вхідні змінні

1. Витрати (cost) описують фінансові витрати, пов’язані з використанням моделі хмарного сервісу.
2. Рівень контролю (control) визначає можливість користувача налаштовувати або керувати сервісом.
3. Безпека (security) характеризує рівень захисту даних і процесів у хмарному середовищі.

Для кожної змінної визначено три лінгвістичні терми: низький (low), середній (medium) і високий (high).

Вихідна змінна

Модель (model) визначає, яка модель хмарного сервісу є найкращою відповідно до вхідних критеріїв.

Перейдемо до опису лінгвістичних термів та функцій приналежності.

Для крайніх значень вхідних змінних будемо використовувати трапецієвидні функції приналежності. Вони добре моделюють нечіткі межі, дозволяючи включати більш широкі діапазони значень у певний терм. Значення функції змінюються поступово, що відображає реальні сценарії,

коли оцінка параметра змінюється без різких стрибків. Функція має достатньо простий вигляд.

Для середніх значень пропонується використовувати трикутну функцію, яка ефективно описує терми, які мають чітко визначене "середнє" значення. Трикутні функції легко визначити, вони вимагають лише трьох параметрів (нижня межа, центр і верхня межа). Середній терм найбільш точно визначається трикутною функцією, оскільки цей тип функції фокусується на одному центральному значенні.

Для вихідної змінної використовується функції трикутної форми. Для кожної з моделей (IaaS, PaaS, SaaS) визначений чіткий діапазон значень, який зручно описувати трикутною функцією. Найвища ступінь приналежності досягається в центральному значенні кожної моделі, що відображає її оптимальну область застосування [20].

Опишемо конкретні параметри кожної функції для кожної змінної.

Змінна "Витрати". Її функції приналежності (рис.2.1):

low: трапецієвидна функція (0,0,2,4)

medium: трикутна функція (2,5,8)

high: трапецієвидна функція (6,8,10,10)

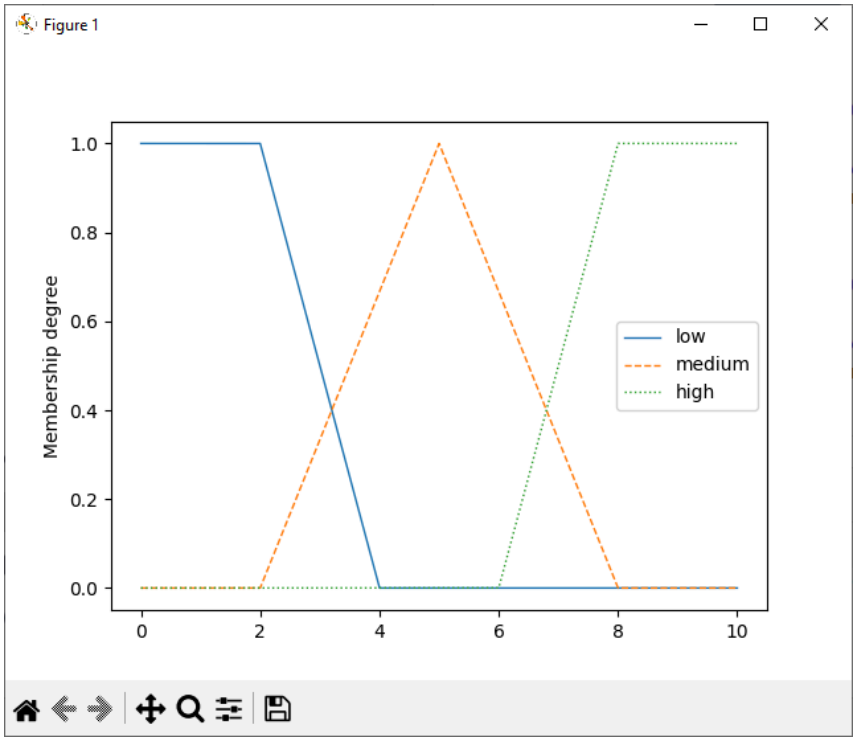

Рисунок 2.1 – Функції приналежності змінної "витрати"

Змінна "Рівень контролю". Її функції приналежності (рис.2.2):

low: трапецієвидна функція (0,0,2,4)

medium: трикутна функція (3,5,7)

high: трапецієвидна функція (6,8,10,10)

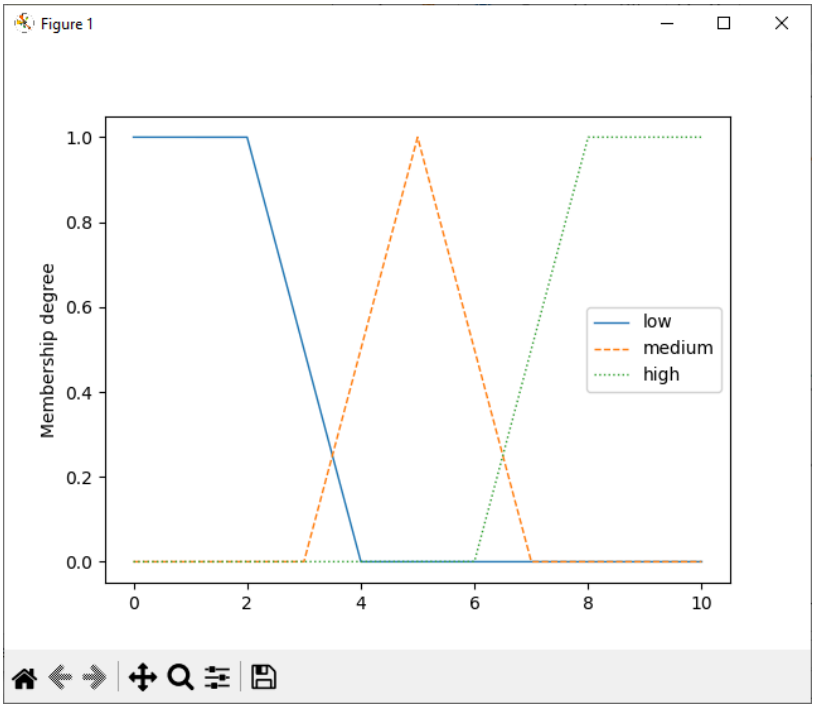

Рисунок 2.2 – Функції приналежності змінної "Рівень контролю"

Змінна "Безпека". Її функції приналежності (рис.2.3):

low: трапецієвидна функція (0,0,2,4)

medium: трикутна функція (3,5,7)

high: трапецієвидна функція (6,8,10,10)

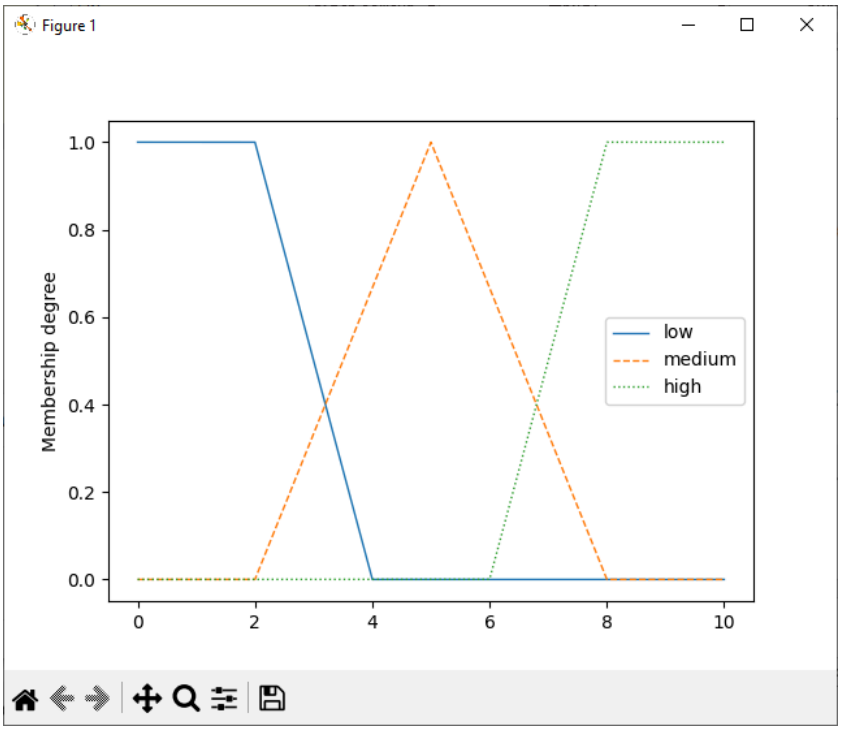

Рисунок 2.3 – Функції приналежності змінної "Безпека"

Змінна "Модель". Її функції приналежності (рис.2.4):

IaaS: трикутна функція (0,20,40)

PaaS: трикутна функція (30,50,70)

SaaS: трикутна функція (60,80,100)

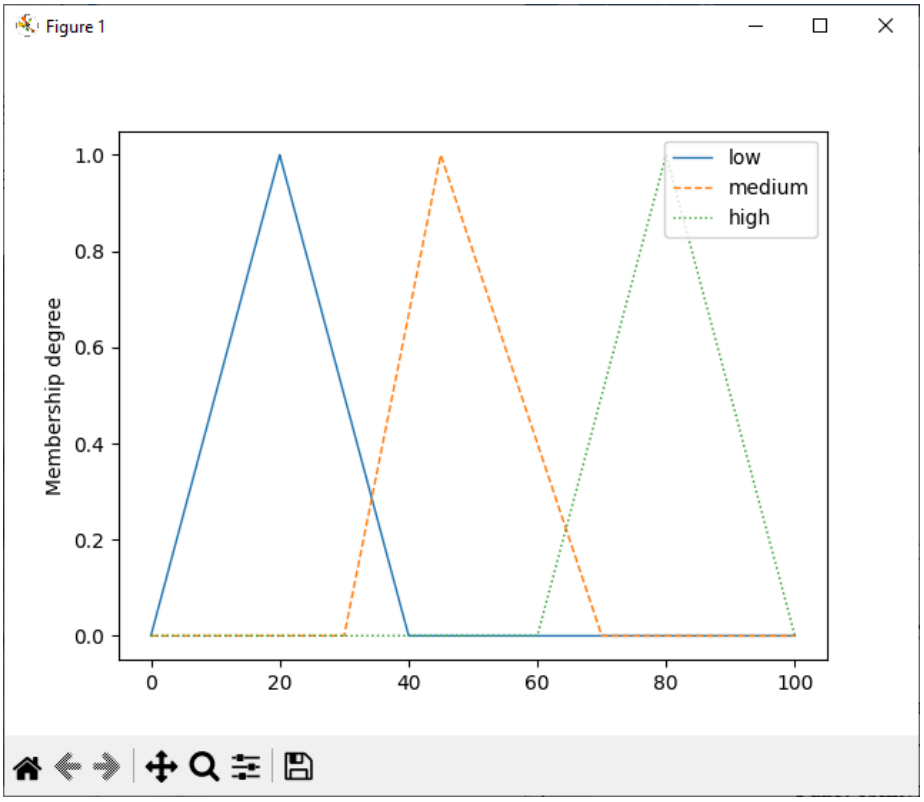

Рисунок 2.4 – Функції приналежності змінної "Модель"

Для визначення вихідної змінної (model) використовується набір нечітких правил, наприклад:

Для кожної можливої комбінації цих рівнів створюється правило.

Логіка правил

IaaS: Вибирається при низьких витратах і високому рівні контролю.

PaaS: Підходить для середніх значень усіх параметрів або якщо комбінація не підходить до інших моделей.

SaaS: Підходить для високих витрат і низького рівня контролю.

На основі нечітких правил система оцінює значення вихідної змінної (model) на основі нечіткого виводу Мамдані. Після дефазіфікації результатом буде Це дозволяє отримати чітке числове значення, яке визначає найбільш відповідну модель хмарного сервісу.

Розглянемо метод TOPSIS.

Нехай є *m* альтернатив і *n* критеріїв. Матриця рішень має вигляд:

$$X = \begin{bmatrix} x_{11} & x_{m1} & x_{12} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

Кожен елемент x_{ij} – це значення альтернативи i за критерієм j.

Першим кроком є нормалізація матриці X. Нормалізація здійснюється за формулами:

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}} \tag{2.2}$$

де r_{ij} – нормалізоване значення.

Після нормалізації кожен критерій множиться на вагу w_j (вага критерію визначається за допомогою експертної оцінки або іншого методу):

$$v_{ij} = r_{ij} \bullet w_j \tag{2.3}$$

Отримуємо зважену нормалізовану матрицю V

$$X = \begin{bmatrix} v_{11} & v_{m1} & v_{12} & \cdots & v_{1n} \\ \vdots & \ddots & \vdots & v_{m2} & \cdots & v_{mn} \end{bmatrix}$$

На наступному кроці здійснюється визначення ідеального та антиідеального рішень

Ідеальне рішення A^+

$$A^+ = \left\{v_1^+, v_2^+, \ldots, v_n^+\right\},\ v_j^+ = \{\left(v_{ij}\right),\ j\in\text{критерії вигоди}\ \left(v_{ij}\right),\ j\in\text{критерії витрат}$$

Антиідеальне рішення A^-

$$A^- = \left\{v_1^-, v_2^-, \ldots, v_n^-\right\},\ v_j^- = \{\left(v_{ij}\right),\ j\in\text{критерії вигоди}\ \left(v_{ij}\right),\ j\in\text{критерії витрат}$$

Обчислюються відстані до ідеального та антиідеального рішень

Відстань кожної альтернативи до ідеального рішення S_i^+ та антиідеального S_i^- обчислюється за формулою Евклідової відстані:

$$S_i^+ = \sqrt{\sum_{j=1}^{n} (v_{ij} - v_j^+)^2}$$

$$S_i^- = \sqrt{\sum_{j=1}^{n} (v_{ij} - v_j^-)^2}$$

Обчислюються коефіцієнта близькості до ідеального рішення.

Коефіцієнт близькості C_i для кожної альтернативи визначається як:

$$C_i = \frac{S_i^-}{S_i^+ + S_i^-} \quad (2.4)$$

де $C_i \in [0,1]$ чим ближче до 1, тим краща альтернатива.

Усі альтернативи сортуються за значенням C_i у спадному порядку. Альтернатива з найбільшим C_i вважається найкращою.

Отже, запропонований метод визначення хмарного сервісу можна представити у вигляді такої послідовності кроків

Етап 1 Вибір моделі хмарного сервісу (IaaS, PaaS, SaaS)

1.1. Визначення критеріїв вибору

Основні критерії, за якими можна оцінити моделі хмарних сервісів:

Витрати – очікуваний рівень витрат.

Рівень контролю – наскільки важливим є гнучке налаштування ресурсів.

Безпека – безпека та конфіденційність даних.

1.2. Визначення найкращого рішення

Створюються нечіткі змінні.

Створюються правила.

На основі правил розраховується значення результуючої змінної.

Проводиться дефаззифікація.

Визначається оптимальна модель (IaaS, PaaS, SaaS).

Етап 2 Вибір конкретного хмарного провайдера

2.1. Визначення релевантних критеріїв

Наприклад – продуктивність, функціональність, наявність підтримки, вартість

2.2. Використання TOPSIS для оцінки провайдерів

Нормалізація матриці критеріїв.

Обчислення зважених значень для кожного критерію.

Визначення ідеального та антиідеального рішення.

Розрахунок відстаней до ідеального та антиідеального рішень.

Ранжування провайдерів за показниками близькості

Результатом використання запропонованого методу стане конкретний провайдер, який надає хмарні послуги.

2.3 Розробка концептуального проєкту СППР

На основі запропонованого методу перейдемо до створення проекту СППР з вибору хмарного сервісу. Для розробки проекту буде використовуватись об’єктно-орієнтований підхід.

Об’єктно-орієнтоване проєктування (ООП) – це підхід до проєктування програмного забезпечення, який базується на концепції об’єктів, їх взаємодії та ієрархії. Об’єкти об’єднують дані та методи для їхньої обробки, забезпечуючи модульність, масштабованість та повторне використання коду.

Мовою об’єктно-орієнтованого проєктування виступає UML. UML (Unified Modeling Language, Уніфікована мова моделювання) — це стандартна мова для графічного моделювання, яка використовується для опису, проєктування та документування систем програмного забезпечення, бізнес-процесів або будь-яких інших систем. UML є основним інструментом

для створення візуальних моделей, які допомагають зрозуміти структуру і поведінку системи на всіх етапах її розробки [21].

За допомогою діаграми UML опишемо поведінку системи. Система має одного актора, який користується системо для визначення хмарного сервісу. Сценарії використання наведені на рис.2.5.

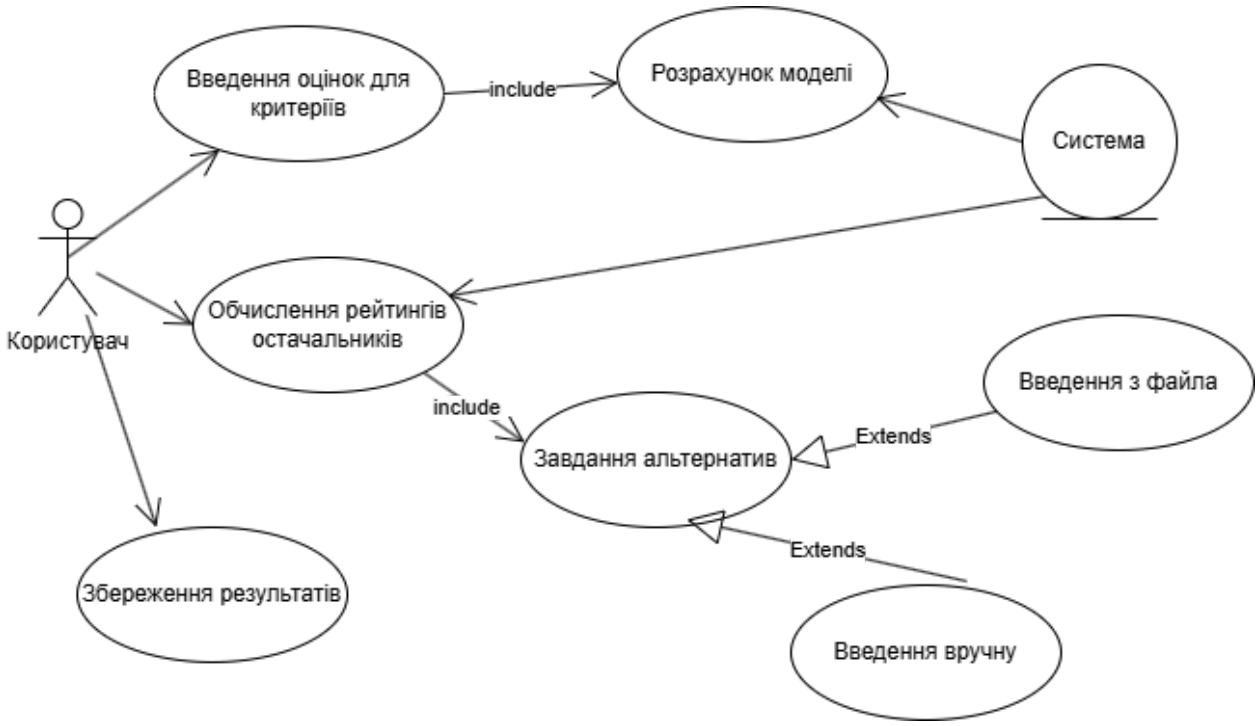

Рисунок 2.6 – Діаграма використання для розроблювальної СППР

Отже, сценаріями використання розроблювальної СППР є:

1. Користувач вводить оцінки для критеріїв (витрати, рівень контролю, безпека).
2. Система обирає модель хмарного сервісу (IaaS, PaaS або SaaS).
3. Користувач додає альтернативи вручну або завантажує їх із JSON-файлу.
4. Визначаються ваги критеріїв.
5. Система обчислює рейтинги постачальників і відображає результати.
6. Результати зберігаються у файл для подальшого аналізу.

Для основних сценаріїв використання системи створимо розробимо діаграми, щоб деталізувати роботу системи.

Діаграма активностей для вибору моделі хмарного сервісу описує процес послідовних дій користувача та системи, які ведуть до визначення найкращої моделі хмарного сервісу (IaaS, PaaS, SaaS) на основі вхідних параметрів (рис.2.6).

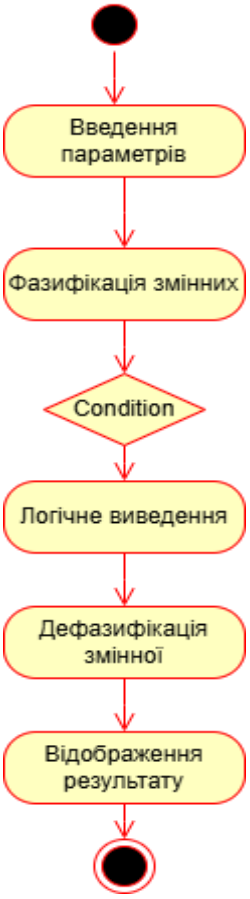

Рисунок 2.6 – Діаграма активностей для сценарію "Визначення моделі"

Опис діаграми активностей здійснимо таким чином:

1. Введення параметрів – користувач заповнює форму в графічному інтерфейсі для введення значень витрат, рівня контролю та безпеки.
2. Фазифікація – програма використовує функції приналежності для перетворення введених параметрів у нечіткі значення.
3. Аналіз правил.

4. Логічне виведення – система збирає результати всіх застосованих правил для кожної моделі, визначаючи узагальнені ступені відповідності.

5. Дефазифікація – система визначає чітке значення для кожної моделі.

6. Відображення результату.

Опишемо за допомогою діаграми послідовностей визначення провайдера. Діаграма послідовностей для визначення провайдера хмарного сервісу ілюструє взаємодію між користувачем, інтерфейсом програми та алгоритмом TOPSIS під час вибору найкращої альтернативи на основі заданих критеріїв (рис.2.7).

Опишемо основні взаємодії в даному сценарії

1. Введення даних – користувач вводить критерії та оцінки через GUI або завантажує їх із JSON-файлу. GUI перевіряє формат і передає дані алгоритму TOPSIS.

2. Розрахунок – TOPSIS отримує ваги та матрицю оцінок, нормалізує дані, обчислює рейтинги та визначає найкращу альтернативу.

3. Відображення результатів – GUI отримує результати обчислень і відображає їх у вигляді списку провайдерів із загальними балами.

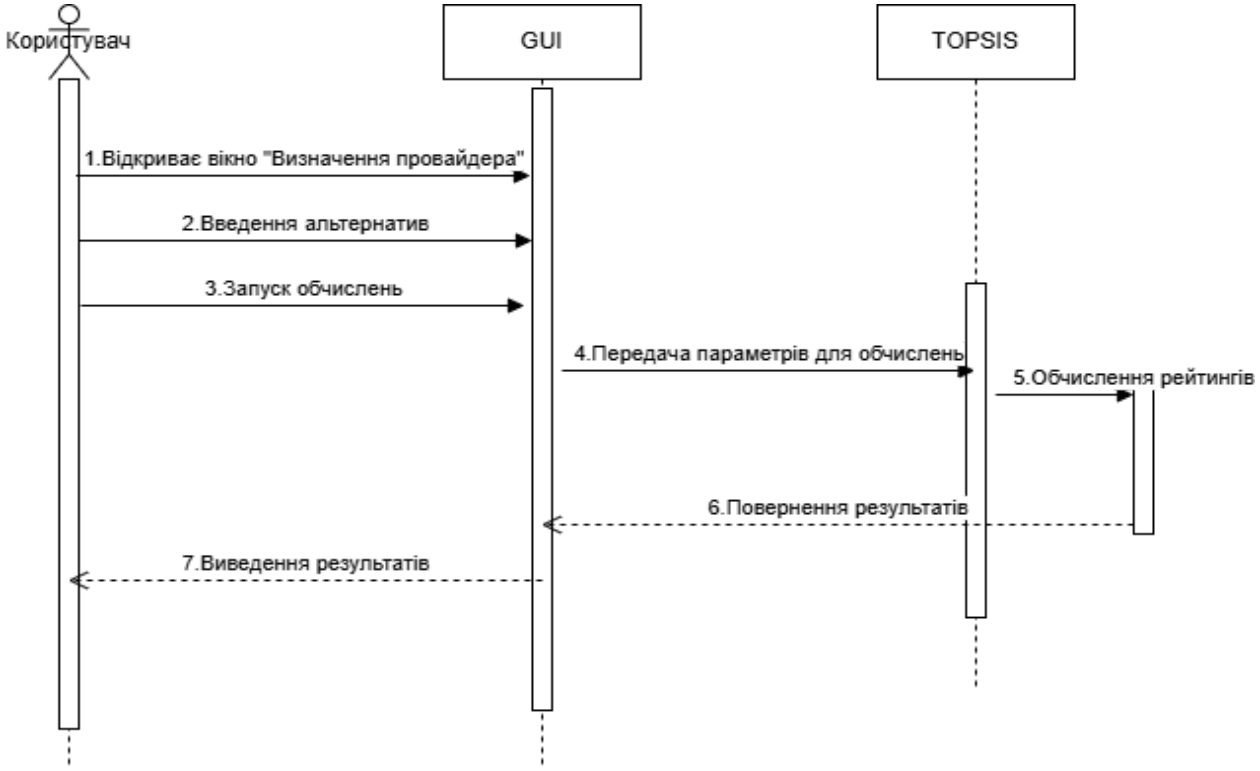

Рисунок 2.7 – Діаграма послідовностей для сценарію "Розрахунок рейтингу провайдерів"

Отже, послідовність дій в даному сценарію:

1. Користувач відкриває вікно GUI.
2. GUI запитує критерії й оцінки.
3. Користувач вводить дані вручну або завантажує JSON-файл.
4. GUI передає отримані дані алгоритму TOPSIS.
5. TOPSIS виконує обчислення та повертає рейтинги альтернатив.
6. GUI відображає результати вибору.

Таким чином, проведене проектування дало змогу описати поведінку системи.

Перейдемо до опису архітектури системи. Згідно вимог система повинна мати зручний інтерфейс та бути достатньо гнучкою. Виходячи з цих вимог, передбачається, що система буде мати багаторівневу архітектуру, що складається з таких компонентів (рис. 2.8):

1. Інтерфейс користувача (UI). Інтерактивний графічний інтерфейс для введення даних та відображення результатів.

2. Бізнес-логіка. Обчислювальний модуль. Реалізація відповідних алгоритмів.

3. Сховище даних. JSON-файли для зберігання інформації про альтернативи. Файли CSV для експорту результатів розрахунків.

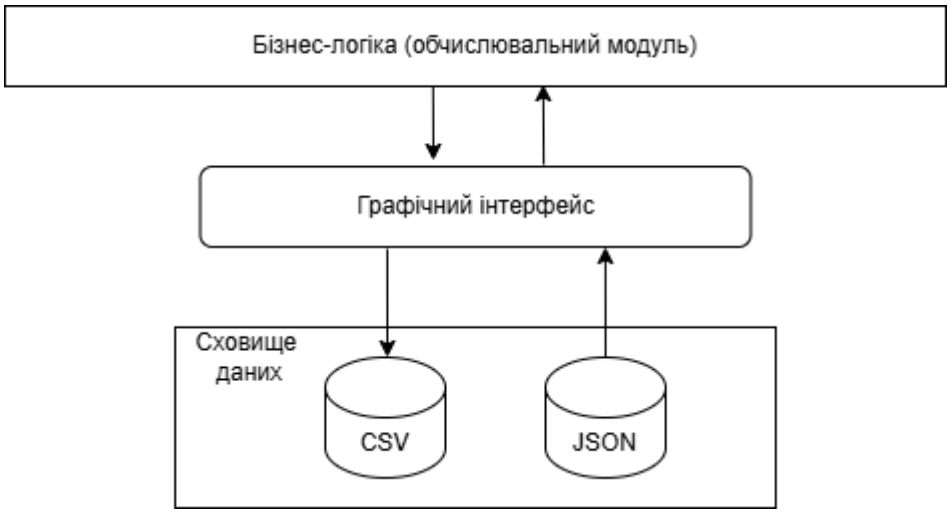

Рисунок 2.8 – Архітектура системи

Використання такої архітектури забезпечить меншу зв’язність між блоками програми, що буде сприяти кращому її розумінню та кращій супроводженості.

2.4 Висновки до розділу 2

Для оцінювання хмарних сервісів можна запропонувати різні характеристики, але в більшості ці характеристики є достатньо складними і не завжди легко вимірюваними або оціненими. Тому було запропоновано використати агреговані критерії, які достатньо легко оцінити, що дає можливість користувачу системи достатньо легко отримати відповідну рекомендацію.

Поєднання нечіткої логіки на першому етапі та алгоритму TOPSIS на другому забезпечує інтеграцію якісного аналізу із кількісним, що дозволяє:

1. Приймати рішення навіть за умов нечіткої або неповної інформації.
2. Гарантувати обґрунтованість і прозорість вибору на кожному етапі.
3. Знизити ризики помилкових рішень завдяки комплексному підходу до аналізу.

Використання об’єктно-орієнтованої моделі системи дало змогу описати її поведінку та визначити архітектуру. Результати проектування забезпечать вищу ефективність процесу розробки.

3 РОЗРОБКА СППР ДЛЯ ВИБОРУ ХМАРНИХ СЕРВІСІВ

3.1 Інформаційна технологія визначення хмарного сервісу

Інформаційна технологія (ІТ) – це сукупність методів, процесів, програмних і апаратних засобів, що використовуються для створення, збереження, обробки, передачі та використання інформації в різних формах. Інформаційні технології охоплюють широкий спектр діяльності, пов’язаної з управлінням і обробкою даних, і є основою для автоматизації процесів в багатьох сферах, таких як бізнес, наука, освіта, медицина, урядова діяльність та ін. [21]

Основними компонентами інформаційної технології виступають:

1. Програмне забезпечення (ПЗ) – це комплекси програм, що виконують певні завдання або функції, наприклад, операційні системи, додатки для обробки текстів, електронні таблиці, бази даних тощо.
2. Апаратне забезпечення (АЗ) – це фізичні пристрої та обладнання, які використовуються для зберігання, обробки та передачі даних. До цього класу належать комп’ютери, сервери, мережеве обладнання, периферійні пристрої (монітори, клавіатури, принтери тощо).
3. Мережі та комунікації – забезпечують передачу даних між комп’ютерами, пристроями та користувачами. Це можуть бути як локальні мережі (LAN), так і глобальні мережі, такі як Інтернет.
4. Бази даних – системи для зберігання та організації великих обсягів інформації, що дозволяють здійснювати ефективний доступ до даних, їх обробку і управління ними.
5. Алгоритми та методи обробки інформації – це математичні моделі і логічні процедури, за допомогою яких здійснюється аналіз і обробка даних, прийняття рішень, розв’язання задач тощо.

Опишемо технологію визначення провайдера сервісу.

Збирання інформації відбувається за допомогою залучених експертів, які допомагають користувачу визначити ваги критерії та альтернативи. Також користувач для цього може використовувати підготовлені дані, що зберігаються в форматі JSON.

Рішення задачі в системі здійснюється в 2 етапи.

Етап 1: Вибір моделі хмарного сервісу за допомогою нечіткої логіки

1.1. Визначення лінгвістичних змінних та їх термів.

Кожен критерій оцінюється через лінгвістичні змінні

1.2. Формування бази правил

Експертні знання відображаються у вигляді правил типу: "якщо…, то…"

1.3. Побудова функцій належності

1.4. Нечіткий висновок

На основі правил та значень критеріїв виконується нечіткий висновок за алгоритмом Мамдані:

Фазифікація – вводяться нечіткі значення критеріїв.

Агрегація правил – визначається ступінь виконання кожного правила.

Композиція – об’єднуються вихідні нечіткі множини.

Дефазифікація – отримується чітке значення (наприклад, IaaS, PaaS, SaaS).

1.5. Результат

Результатом цього етапу є обрана модель сервісу (наприклад, PaaS).

Етап 2: Вибір конкретного хмарного провайдера за методом TOPSIS

Нормалізація критеріїв.

Зважена нормалізація.

Визначення ідеального та антиідеального рішення.

Розрахунок відстаней та коефіцієнтів близькості.

Ранжування провайдерів.

Далі інформація щодо провайдера виводиться користувачу та може бути збережена у файлі.

Структурно-логічна схема операцій в інформаційній технології визначення хмарного сервісу наведена на рис. 3.1.

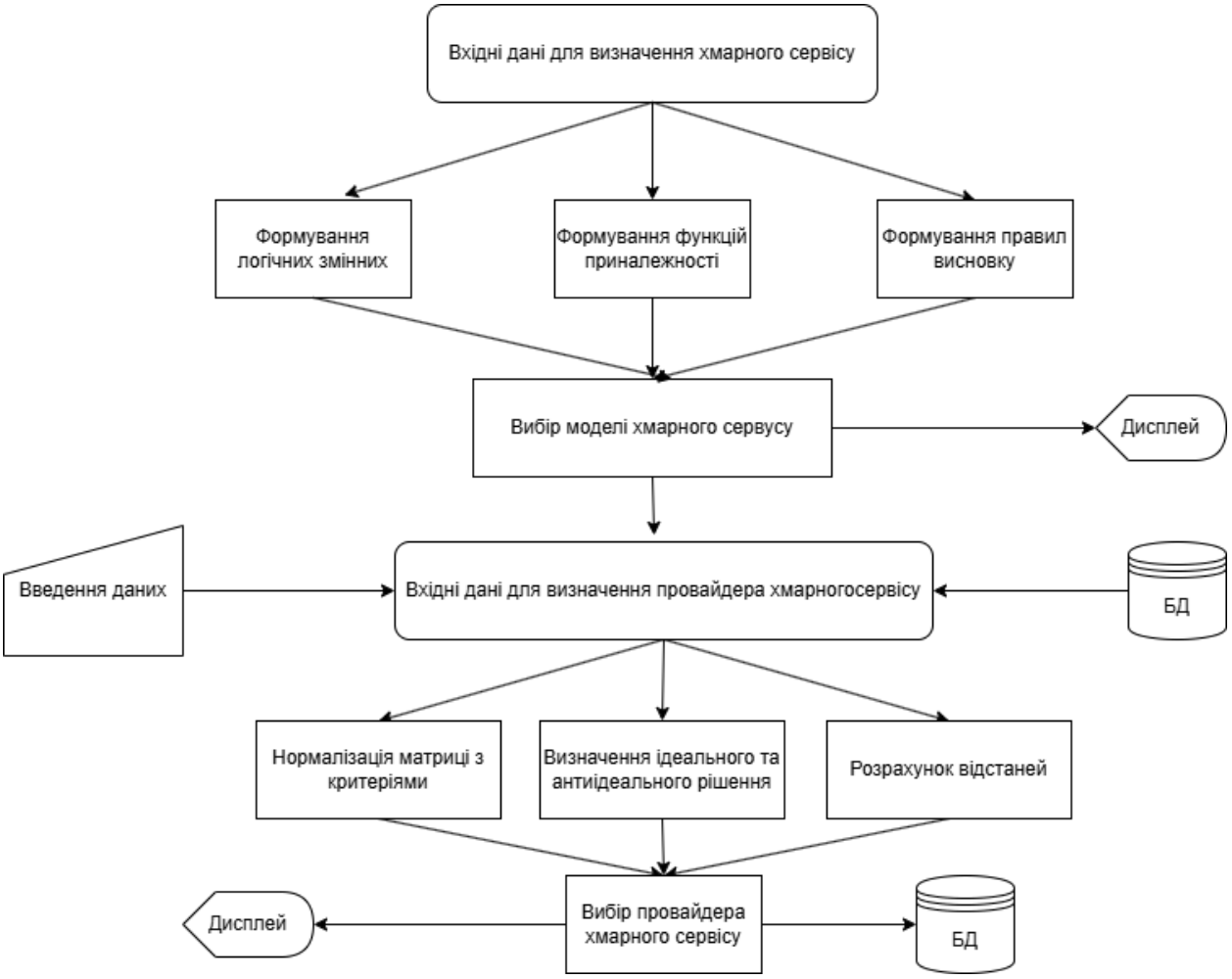

Рисунок 3.1. – Структурно-логічна схема операцій в інформаційній технології визначення хмарного сервісу

Для реалізації даної інформаційної технології передбачено розробка десктопного додатку та рівноцінного веб-застосунку. Додаток буде здійснювати розрахунок та визначення моделі хмарного сервісу та провайдера.

Десктопний додаток розробляється під Windows, не використовує спеціальних програм, а отже, немає специфічних вимог. З іншого боку, система спрямована на збереження результатів вибору провайдера, також програма може користуватися підготовленими даними, що зберігаються на

зовнішніх носіях. Обсяг програми з усіма бібліотеками складає 280Мб. Також програмна розробка не висуває високих вимог до використання графіки. Отже, виходячи з цього, можна визначити такі вимоги до технічного забезпечення розробленої інформаційно-управляючої системи (табл.3.1).

Додатковим варіантом є веб-застосунок, який може покрити потреби користувачів, що не мають змоги використовувати десктопний додаток.

Таблиця 3.1

Обмеження на застосування ІУС на базі десктопного додатку

Процесор	1 Ггц
Пам’ять (ОЗУ)	2 ГБ
Жорсткий диск	3 ГБ
Дисплей	1024x576
Windows OS	7/8/10

Таблиця 3.2

Обмеження на застосування ІУС на базі веб-застосунку

Процесор	1 Ггц
Пам’ять (ОЗУ)	2 ГБ
Доступ до мережі	Обов’язково
Дисплей	Довільний
OS	Без обмеження

Розроблена інформаційна технологія є достатньо гнучкою, оскільки може враховувати нечіткі дані та суб’єктивні оцінки. Вона краще моделює краще моделює складні рішення в умовах невизначеності. А експертам легше формулювати правила в лінгвістичній формі, ніж давати точні числові оцінки. Використання двохетапного вибору дає змогу більш точно відбирати сервіс, який задовольняє вимогам користувача.

3.2 Обґрунтування засобів розробки

Python обрано основним засобом розробки завдяки його простоті, зрозумілому синтаксису та широкому набору функціональних можливостей. Це сучасна мова програмування, яка забезпечує швидкість і зручність розробки, що особливо важливо для створення систем підтримки прийняття рішень (СППР). Її багатий набір бібліотек дозволяє інтегрувати математичні обчислення, аналіз даних і створення графічного інтерфейсу в одному середовищі. Наприклад, бібліотеки NumPy і SciPy забезпечують ефективні математичні операції, Tkinter дозволяє створювати інтуїтивно зрозумілий графічний інтерфейс, а Flask надає можливості з побудови вебдодатків. Кросплатформеність Python забезпечує можливість запуску програм на різних операційних системах, а велика спільнота користувачів гарантує доступ до ресурсів і швидке вирішення технічних питань.

Для роботи з нечіткими множинами та нечіткими контролерами використовувалась бібліотека skfuzzy. Дана бібліотека забезпечує створення функцій приналежності згідно заданими параметрам та нечіткий висновок на основі заданих правил. Її використання дозволяє реалізовувати достатньо складні структури , побудовані на нечіткій логіці.

Для роботи з даними використовується формат JSON, який забезпечує простоту структури, універсальність та гнучкість у зберіганні інформації. JSON підтримує вкладені структури, що дозволяє зберігати складні дані, такі як характеристики альтернатив і їхні оцінки. Він є стандартизованим форматом, що широко використовується у сучасних інформаційних системах. Python має вбудовану бібліотеку json, яка забезпечує легку взаємодію з JSON-файлами у програмі. Це дозволяє зчитувати дані з файлів, конвертувати їх у Python-структури та зберігати результати у вигляді JSON, зменшуючи складність реалізації та спрощуючи обмін інформацією між компонентами.

Засоби Python, такі як Tkinter для інтерфейсу, NumPy для математичних обчислень та json для роботи з файлами, забезпечують інтеграцію різних функціональних частин системи в єдину програму. Таке рішення дозволяє швидко розширювати функціональність, додаючи нові критерії, модулі

аналізу або підтримку інших форматів даних, наприклад, CSV або XML. Завдяки використанню цих засобів, розробка стає не лише швидшою, але й більш надійною, оскільки більшість складних алгоритмів і операцій реалізовані у готових бібліотеках.

Переваги обраних засобів розробки

1. Простота інтеграції компонентів. Python і його бібліотеки дозволяють швидко поєднувати різні функціональні частини системи, такі як інтерфейс, обробка даних і зберігання результатів.

2. Масштабованість. У разі необхідності можна легко додати нові критерії, модулі аналізу або змінити формати збереження даних (наприклад, додати підтримку CSV, XML).

3. Швидкість розробки. Використання готових бібліотек оптимізує зусилля для реалізації складних алгоритмів і забезпечує їхню надійність.

Таким чином, використання Python, JSON та відповідних бібліотек забезпечує створення ефективної, гнучкої та масштабованої системи підтримки прийняття рішень. Це дозволяє швидко розробити технологію, яка відповідає сучасним вимогам, є зручною для користувачів і адаптивною до змін у задачах чи умовах роботи.

3.3 Розробка СППР для вибору хмарних сервісів

При розробці СППР використовувалась подієво-орієнтована парадигма програмування. Подієво-орієнтоване програмування (ПОП) – це парадигма програмування, в якій виконання програми визначається подіями, такими як дії користувача (натискання кнопки, введення тексту), повідомлення від інших програм або змінні стану. ПОП широко застосовується у розробці графічних інтерфейсів користувача (GUI), реального часу систем і інтерактивних програм.

У ПОП програма складається з таких основних компонентів:

Події – це дії або умови, які ініціюють виконання певного коду. Наприклад, події можуть бути пов’язані з натисканням миші, введенням тексту, закриттям вікна або отриманням даних з мережі.

Обробники подій – це функції або методи, які виконуються у відповідь на певні події. Наприклад, натискання кнопки може викликати метод, який змінює текст у текстовому полі або виконує певний розрахунок.

Цикл подій – це основний механізм програми, який постійно перевіряє, чи виникла якась подія, і запускає відповідний обробник. Цей цикл працює до завершення програми.

Ключові особливості ПОП визначено таким чином.

Асиметричність керування – на відміну від класичного структурного програмування, де потік виконання визначений послідовно, у ПОП виконання залежить від подій, які можуть відбуватися у будь-який момент.

Керування подіями – основна логіка програми зосереджена в обробниках подій, що дозволяє розділити код на частини, які відповідають за окремі взаємодії.

Гнучкість – легко додавати нові обробники або змінювати поведінку програми, додаючи чи модифікуючи події.

При розробці програми було виділено декілька модулів.

Модуль start_step1() реалізовував створення графічного вікна для організації вибору моделі хмарного сервісу. В модулі було передбачено створення відповідних полів введення для завдання критеріїв.

Модуль start_step2() розроблявся для реалізації графічного вікна для організації вибору хмарного провайдера. Модуль містить відповідні для введення, кнопки та таблицю для введення даних та відображення результатів.

Для вибору моделі хмарного сервісу був реалізований модуль setup_fuzzy_model(). В модулі задаються нечіткі лінгвістичні змінні, створюються правила, визначені згідно наведених у розділі 2.2. Здійснюється

логічний висновок. Реалізація виведення отриманого висновку здійснюється в модулі start_step1().

Приклад створення нечітких множин та лінгвістичних змінних наведений на рис.3.2.

```
        # Визначення лінгвістичних змінних
        cost = ctrl.Antecedent(np.arange(0, 11, 1), 'cost')
        control = ctrl.Antecedent(np.arange(0, 11, 1), 'control')
        security = ctrl.Antecedent(np.arange(0, 11, 1), 'security')

        model = ctrl.Consequent(np.arange(0, 101, 1), 'model')

        # Нечіткі множини для витрат
        cost['low'] = fuzz.trapmf(cost.universe, [0, 0, 2, 4])
        cost['medium'] = fuzz.trimf(cost.universe, [2, 5, 8])
        cost['high'] = fuzz.trapmf(cost.universe, [6, 8, 10, 10])

        # Нечіткі множини для контролю
        control['low'] = fuzz.trapmf(control.universe, [0, 0, 2, 4])
        control['medium'] = fuzz.trimf(control.universe, [3, 5, 7])
        control['high'] = fuzz.trapmf(control.universe, [6, 8, 10,
10])

        # Нечіткі множини для безпеки
        security['low'] = fuzz.trapmf(security.universe, [0, 0, 2, 4])
        security['medium'] = fuzz.trimf(security.universe, [3, 5, 7])
        security['high'] = fuzz.trapmf(security.universe, [6, 8, 10,
10])
```

Рисунок 3.2 – Лінгвістичні змінні та нечіткі множини

Набір правил нечіткої логіки для оцінки критеріїв і прийняття рішення про модель наведений на рис. 3.3.

```
  # Генерація правил
  for c in cost_levels:
      for ctrl_level in control_levels:
          for sec in security_levels:
              if c == 'low' and ctrl_level == 'high':
                  rules.append(ctrl.Rule(cost[c] &
control[ctrl_level] & security[sec], model['IaaS']))
              elif c == 'medium' and ctrl_level == 'medium':
                  rules.append(ctrl.Rule(cost[c] &
control[ctrl_level] & security[sec], model['PaaS']))
              elif c == 'high' and ctrl_level == 'low':
                  rules.append(ctrl.Rule(cost[c] &
control[ctrl_level] & security[sec], model['SaaS']))
              else:
                  rules.append(ctrl.Rule(cost[c] &
control[ctrl_level] & security[sec], model['PaaS']))

  # Створення системи управління
  model_ctrl = ctrl.ControlSystem(rules)
  model_simulation = ctrl.ControlSystemSimulation(model_ctrl)
```

Рисунок 3.3 – Створення набору правил

Для роботи з вагами критеріїв та альтернативами метод TOPSIS був реалізований як окремий модуль.

Для роботи з даними було розроблено два модуля.

1 Модуль завантаження даних із зовнішніх джерел (JSON) (рис.3.4).

```
    def load_from_file():
        """Завантажує альтернативи з JSON-файлу."""
        try:
            file_path = filedialog.askopenfilename(filetypes=[("JSON Files",
"*.json")])
            if not file_path:
                return
            with open(file_path, "r") as file:
                data = json.load(file)
                if not isinstance(data, list):
                    raise ValueError("Формат файлу некоректний: очікується список
альтернатив.")

                global alternatives
                alternatives = []
                for item in data:
                    if "name" not in item or "scores" not in item:
                        raise ValueError("Кожна альтернатива повинна містити 'name' і
'scores'.")
                    # Перетворення оцінок на числовий формат
                    scores = list(map(float, item["scores"]))
                    alternatives.append({"name": item["name"], "scores": scores})

                update_table()
                messagebox.showinfo("Успіх", f"Дані завантажено з файлу
'{file_path}'.")
        except Exception as e:
            messagebox.showerror("Помилка", f"Помилка завантаження даних: {e}")
```

Рисунок 3.4 – Модуль обробки зовнішніх даних

2. Модуль збереження результатів у файл (рис.3.5).

```
    def save_results():
        """Зберігає результати в файл."""
        try:
            with open("topsis_results.txt", "w") as file:
                file.write("Результати TOPSIS:\n")
                file.write("Назва провайдера, Продуктивність, Вартість,
Функціональність, Наявність підтримки, Загальний бал\n")
                for alt in alternatives:
                    file.write(f"{alt['name']}, {', '.join(map(str,
alt['scores']))}, {alt.get('score', 0):.4f}\n")
            messagebox.showinfo("Успіх", "Результати збережено у файл
'topsis_results.txt'.")
        except Exception as e:
            messagebox.showerror("Помилка", f"Помилка збереження файлу:
{e}")
```

Рисунок 3.5 – Модуль збереження результатів обчислення у файл

Продемонструємо роботу з СППР.

При запуску програми з’являється вікно для виконання першого етапу (рис.3.6).

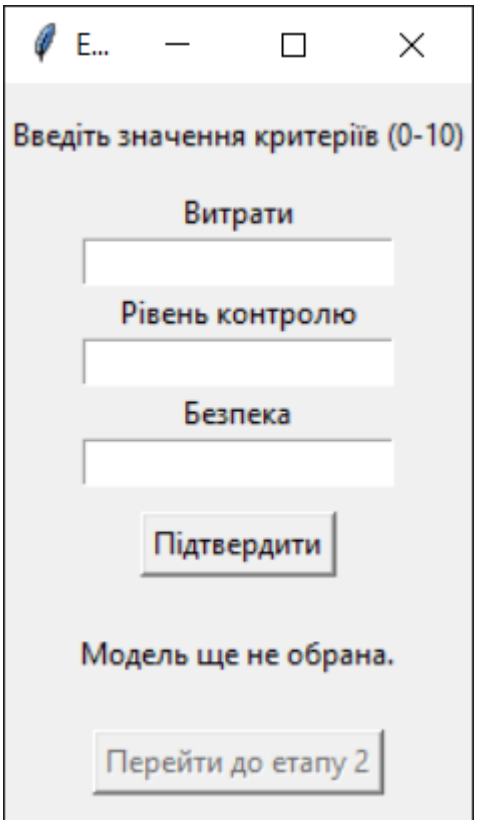

Рисунок 3.6 – Вікно програми для вибору моделі

Користувач у вікні вводить свої переваги відносно витрат, рівня контролю та вартості сервісу. Переваги оцінюється в порядковій шкалі від 0 до 10, де 0 – така характеристика абсолютно неважлива, 10 – така характеристика надзвичайно важлива. Використання порядкової шкали замість абсолютної значно спрощує роботу з програмою, оскільки не потребує спеціальних знань в області хмарних технологій, а зосереджуються на вимогах бізнесу.

Наприклад, користувач віддає переваги недорогому сервісу з підвищеною безпекою та середнім рівнем контролю. Після введення даних та натискання кнопки Підтвердити користувач отримує рекомендацію щодо моделі хмарного сервісу та може переходити до вибору конкретного

провайдера (рис.3.7). В цьому випадку, згідно визначених правил, користувачу пропонується обрати модель PaaS.

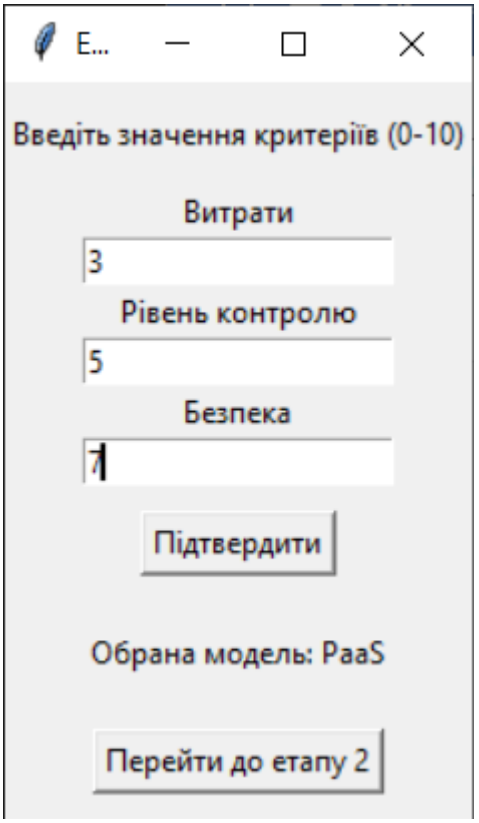

Рисунок 3.7 – Рекомендації щодо обраної моделі

При натисканні кнопки Перейти до етапу 2 на екрані з’являється друге вікно, де користувач може обирати конкретного провайдера (рис.3.8).

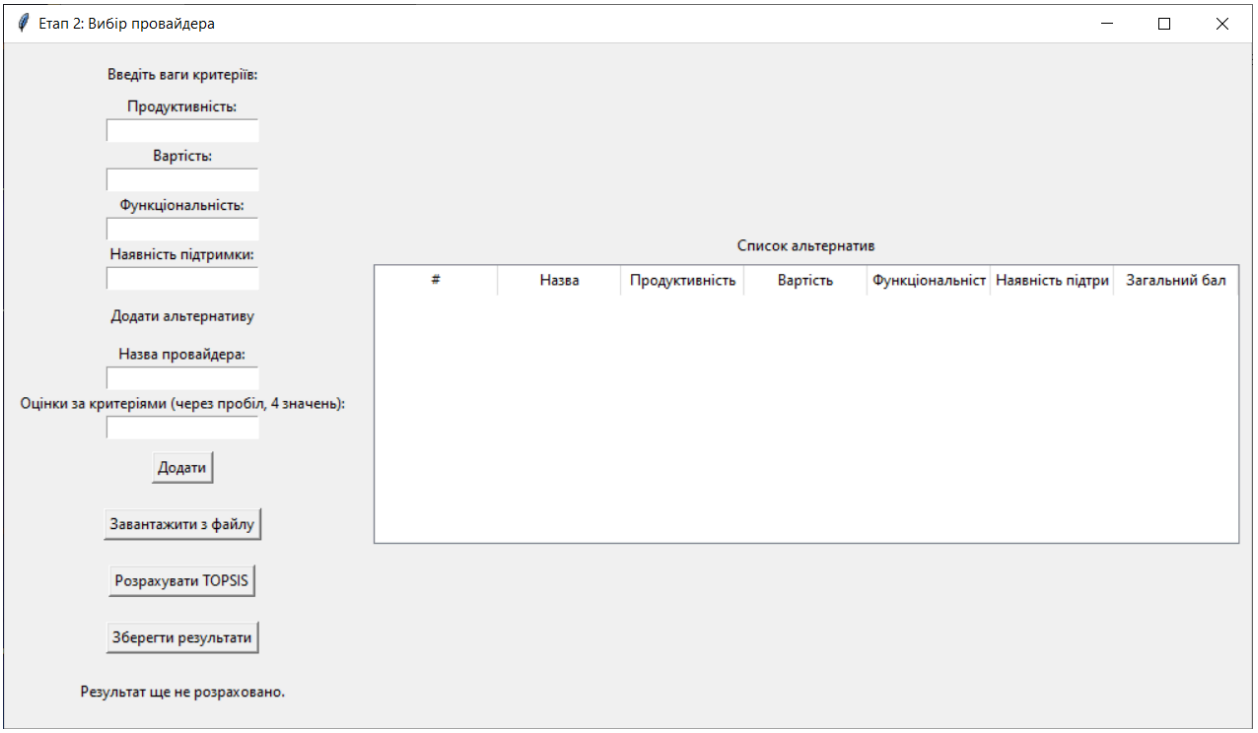

Рисунок 3.8 – Вікно для вибору конкретного провайдера

Приклади сервісів PaaS опишемо так.

1) Heroku
   a) Проста у використанні платформа для розробки, розгортання та масштабування додатків.
   b) Підтримує мови – Python, Ruby, Node.js, Java, Go тощо.
   c) Ідеально для стартапів та швидкого прототипування.
2) Google App Engine (GAE)
   a) Платформа від Google для автоматичного масштабування додатків.
   b) Підтримка Python, Java, PHP, Node.js та інших мов.
   c) Інтеграція з іншими сервісами Google Cloud.
3) Microsoft Azure App Service
   a) Інструмент для створення веб-додатків, API, мобільних застосунків.
   b) Підтримує .NET, Java, Ruby, Python, PHP, Node.js.
   c) Інтеграція з іншими сервісами Azure.
4) AWS Elastic Beanstalk
   a) Проста у використанні платформа для розгортання та управління веб-додатками.

b) Підтримка мов: Java, .NET, PHP, Node.js, Python, Ruby, Go, Docker.
c) Частина Amazon Web Services.

5) Red Hat OpenShift
   a) Kubernetes-базована платформа для розробки та розгортання додатків.
   b) Підтримує контейнери, мікросервіси та різноманітні мови програмування.
6) IBM Cloud Foundry
   a) Потужний інструмент для розробки корпоративних додатків.
   b) Орієнтована на безпеку та інтеграцію з іншими IBM-сервісами.
7) Salesforce Platform (Force.com)
   a) Платформа для розробки корпоративних додатків.
   b) Інтегрується з CRM Salesforce.
8) Oracle Cloud Platform
   a) Пропонує розробку додатків, аналітику, управління базами даних.
   b) Інтеграція з Oracle Database.
9) Alibaba Cloud PaaS
   a) Рішення для ринку Азії.
   b) Платформа для роботи з великими даними, AI та корпоративними додатками.
10) DigitalOcean App Platform
   a) Платформа для створення, розгортання та масштабування хмарних додатків.
   b) Легкий у використанні інтерфейс, підтримка Docker та Kubernetes

Оцінки сервісів наведені в таблиці 3.2. Використання бальної шкали дає можливість користувачу зосередитись на агрегованих оцінках обраних характеристик. В той же час перед користувачем не стоїть задача попарного порівняння, яка не є тривіальною вже при наявності 4 альтернатив.

Таблиця 3.2

Оцінювання сервісів PaaS

Сервіс	Продуктивність	Функціональність	Вартість	Підтримка
Heroku	8	8	6	9
Google App Engine	9	9	7	9
Microsoft Azure App Service	8	9	6	9
AWS Elastic Beanstalk	9	8	7	8
Red Hat OpenShift	8	9	6	9
IBM Cloud Foundry	8	8	5	8
Salesforce Platform	7	9	5	9
Oracle Cloud Platform	7	8	5	8
Alibaba Cloud PaaS	8	8	6	7
DigitalOcean App Platform	7	7	8	7

1. Alibaba Cloud PaaS – сильна підтримка в Азії, але для інших ринків – обмежений вплив.
2. AWS Elastic Beanstalk – стабільна продуктивність і широкий набір функцій, хоча використання AWS може бути складним для новачків.
3. DigitalOcean App Platform – проста у використанні і доступна за ціною, але поступається у функціональності гігантам.
4. Google App Engine – висока продуктивність і глибока інтеграція з іншими сервісами Google.
5. Heroku – ідеальний для стартапів, але зростання вартості при масштабуванні може бути стримуючим фактором.
6. IBM Cloud Foundry – хороший вибір для корпоративних користувачів із великою кількістю інтеграцій.
7. Microsoft Azure App Service – потужний інструмент, але вартість може швидко зрости залежно від конфігурації.
8. Oracle Cloud Platform – більше підходить для великих організацій, які вже використовують Oracle Database.
9. Red Hat OpenShift – орієнтований на підприємства, особливо для тих, хто використовує контейнери та Kubernetes.
10. Salesforce Platform – потужний для розробки бізнес-додатків, але вартість висока.

Для проведення аналізу користувач може завантажити інформацію з файлі в форматі JSON, де вже попередньо були виставлені (рис.3.9).

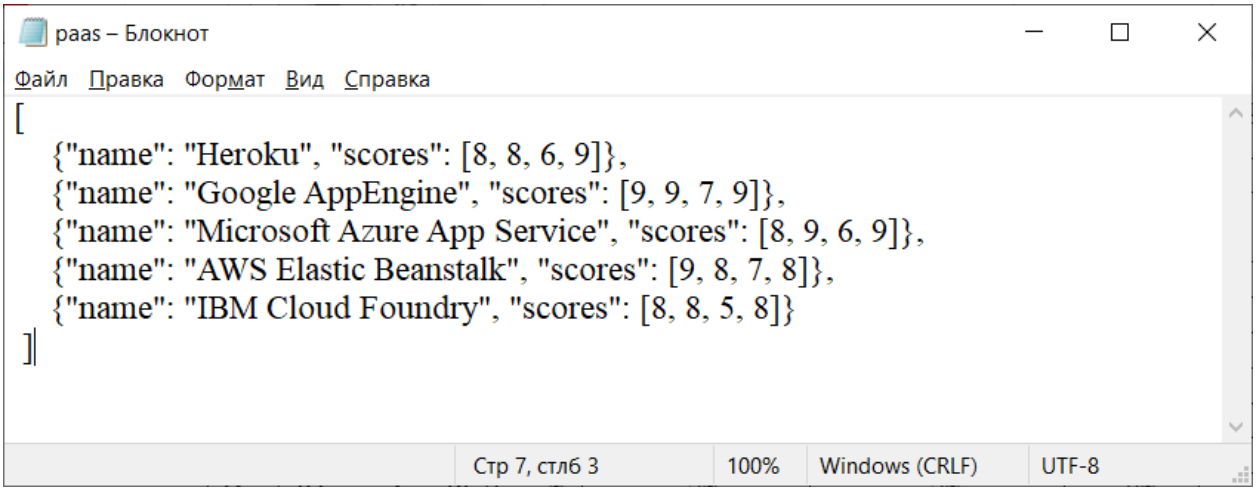

Рисунок 3.9 – Вхідні дані в форматі JSON

Для завантаження інформації з файлу користувачу потрібно натиснути відповідну кнопку і у вікні, що з’явилось вибрати потрібний файл (рис.3.10).

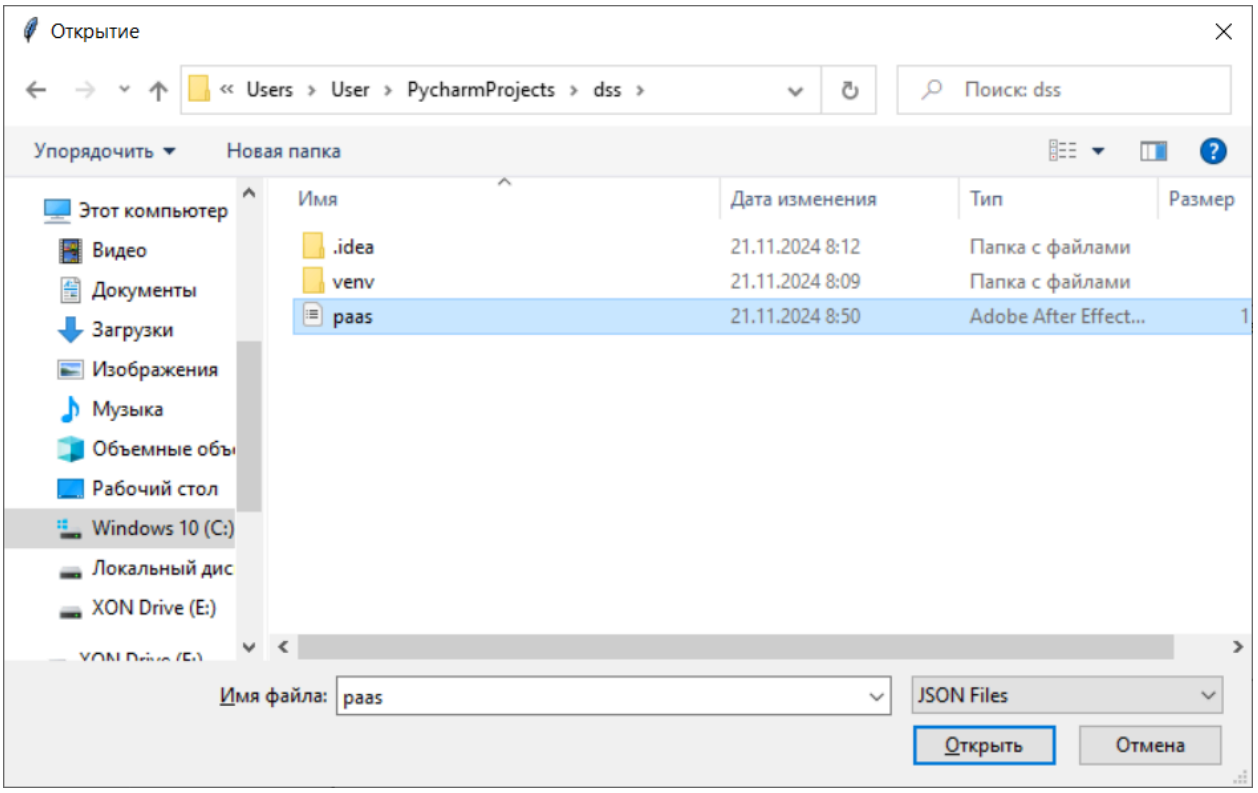

Рисунок 3.10 – Вибір файлу з даними для оцінки

Після цього оцінки завантажуються в систему (рис.3.11).

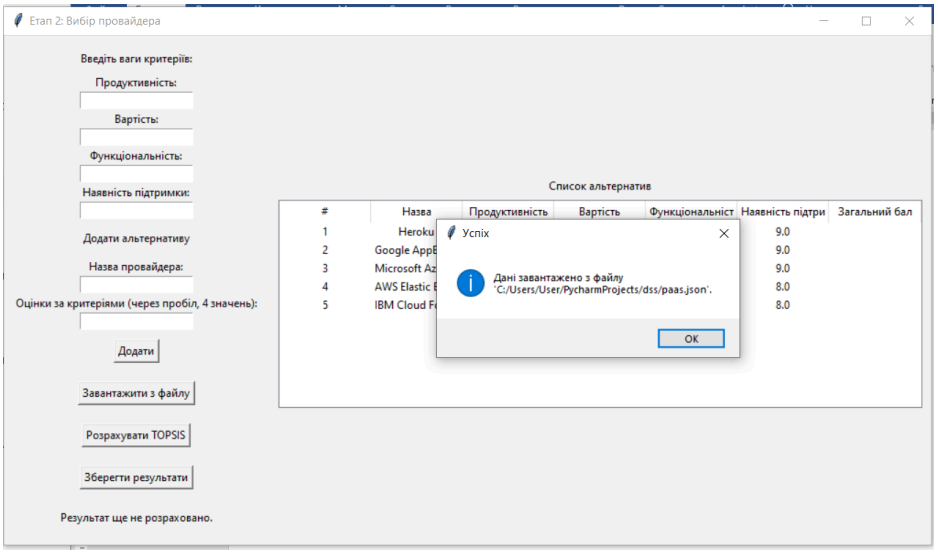

Рисунок 3.11 – Завантаження даних з файлу

Користувач може додати до завантажених даних ще альтернативи. Для цього у відповідних полях потрібно ввести назву альтернативи та бали для характеристик (рис.3.12).

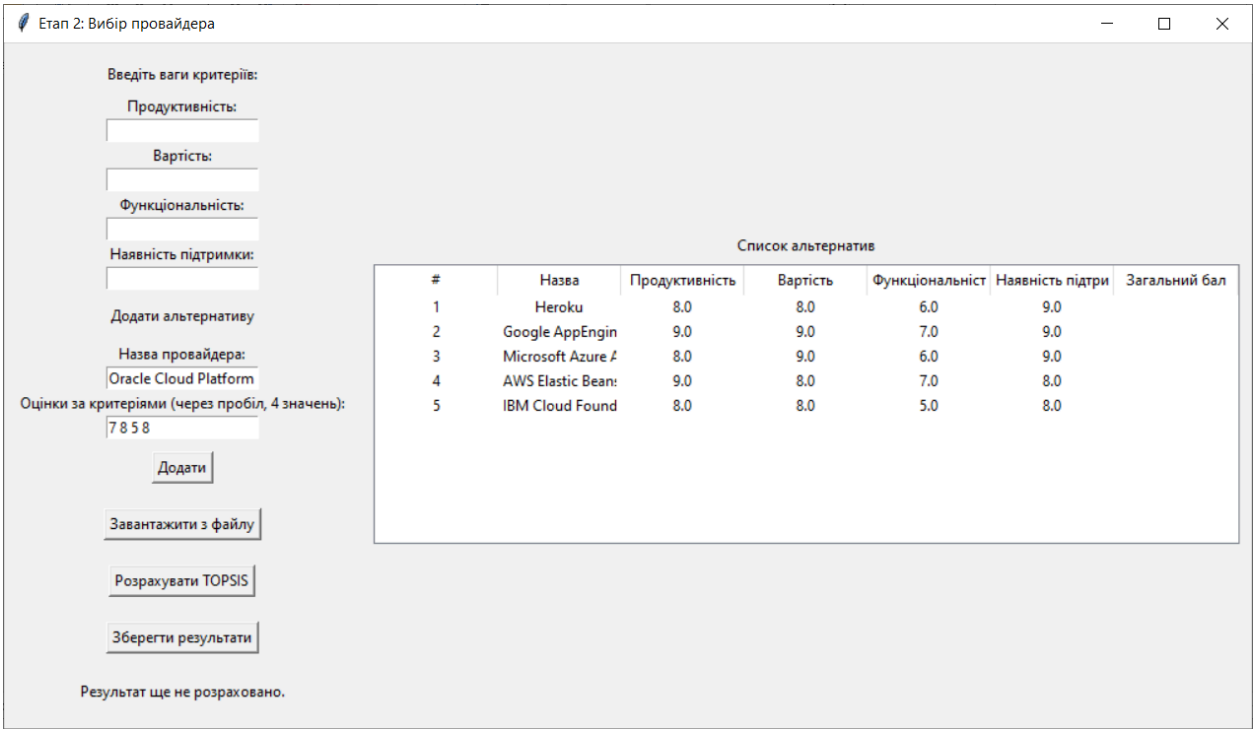

Рисунок 3.12 – Введення даних про альтернативу

Натиснення кнопки Додати додає дані до таблиці альтернатив (рис.3.13)

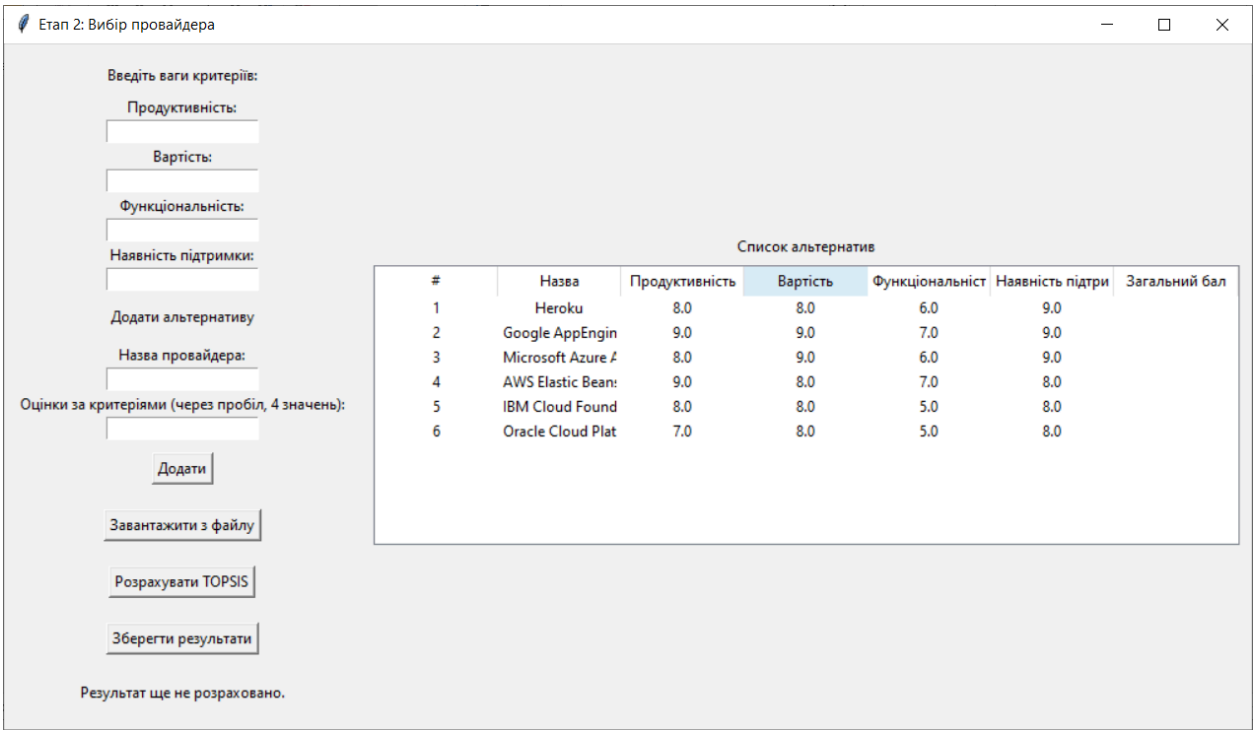

Рисунок 3.13 – Додана альтернатива

Така організація ввдення даних дає можливість користувачу, як скористатися попередніми наробками, так і доповнити їх якимось новими сервісами.

Після цього користувач задає ваги для кожної характеристики, визначаючи, що саме для нього є найважливішим на даний момент (рис.3.14). Загальна сума ваг повинна дорівнювати 1.

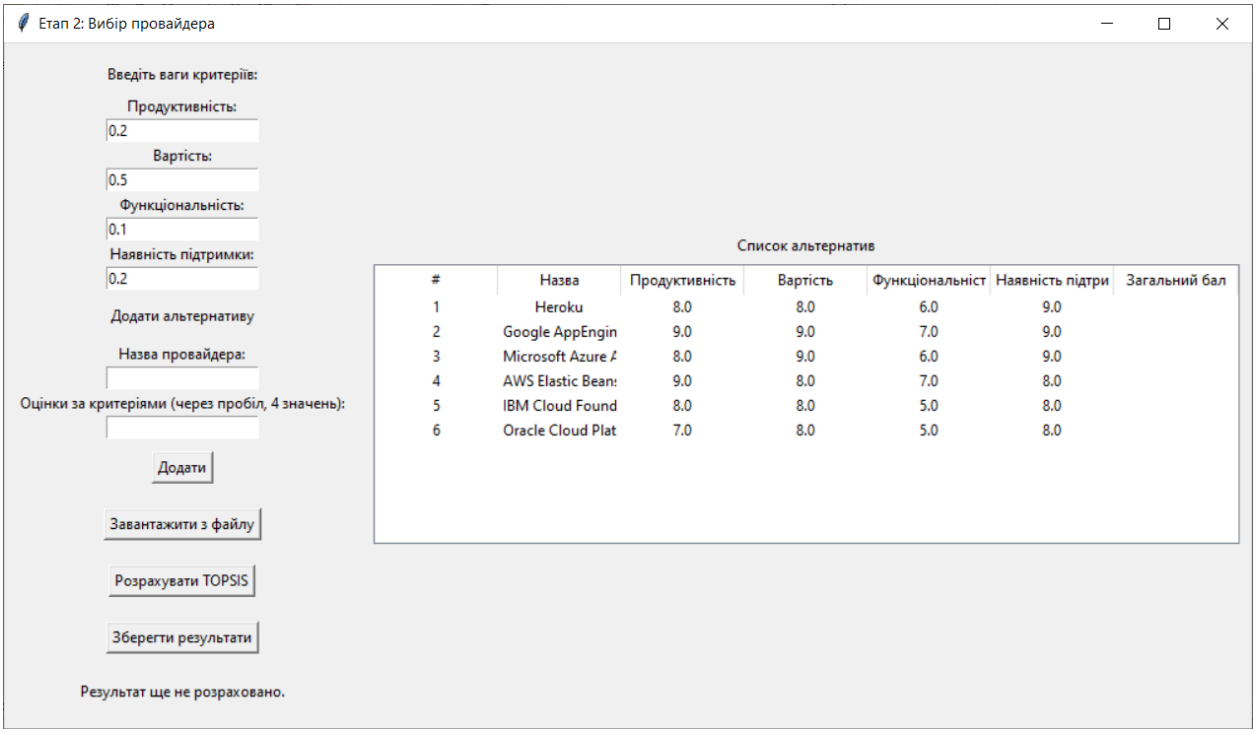

Рисунок 3.14 – Встановлення ваг для критеріїв

Після натискання кнопки Розрахувати TORSIS в таблиці з’являються бали, набрані кожною альтернативою (рис.3.15). Також альтернатива з максимальною кількістю балів потрапляє в рекомендації.

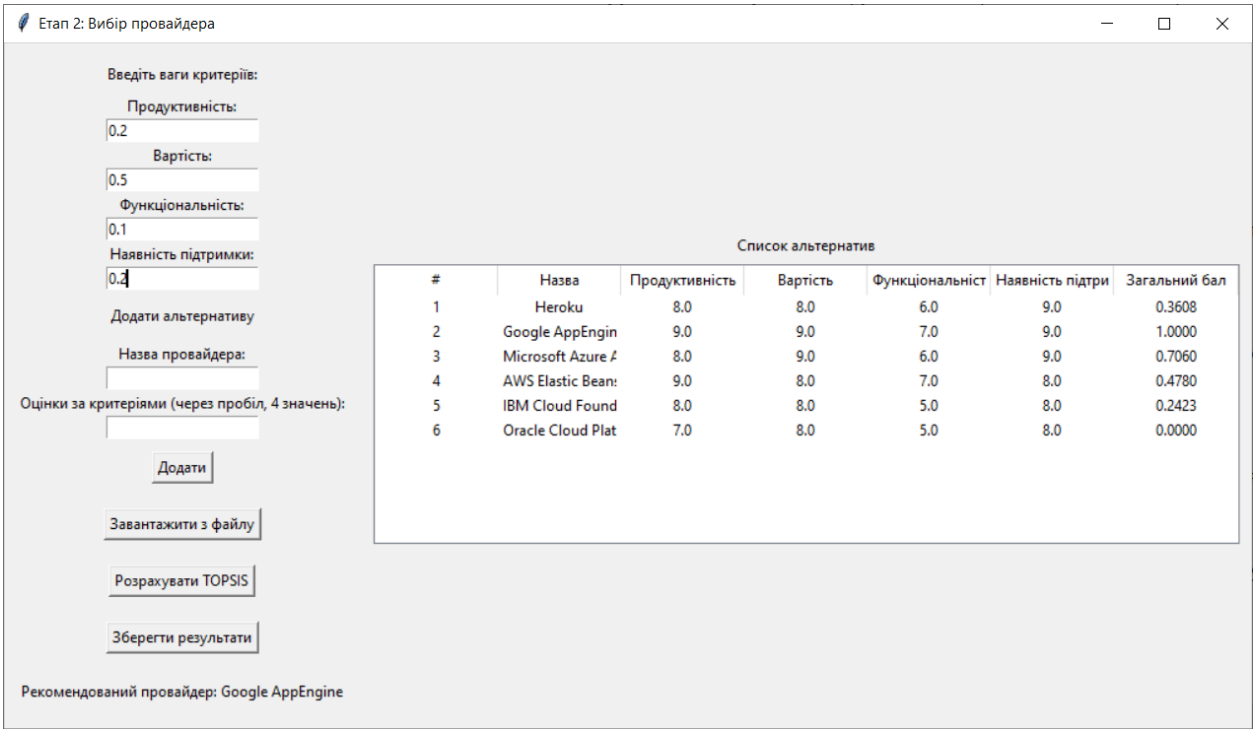

Рисунок 3.15 – Розрахунок балів для кожної альтернативи

Зміна ваг критеріїв може призвести до перерозподілу набраних балів (рис.3.16).

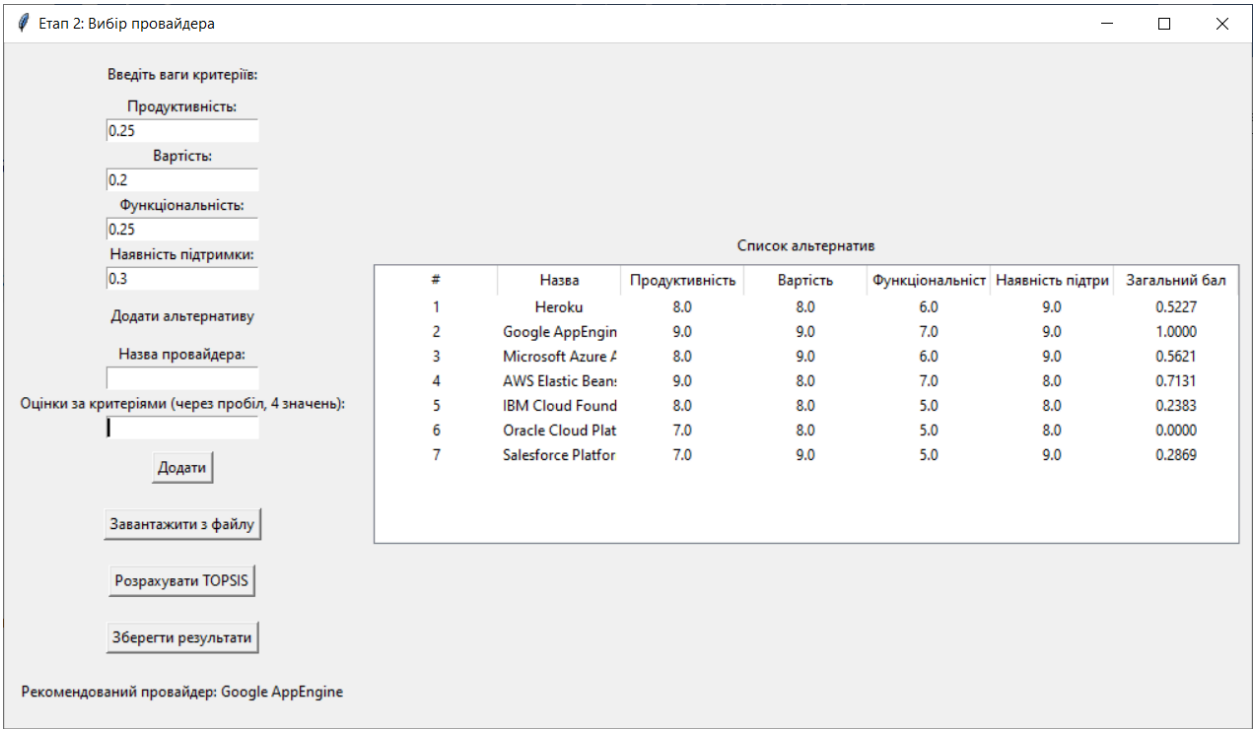

Рисунок 3.16 – Новий розрахунок загальних оцінок для провайдерів (для порівняння доданий ще один провайдер в ручному режимі)

Отримані результати користувач може записати у файл, натиснувши кнопку Зберегти результати. Вміст файлу наведений на рис.3.17.

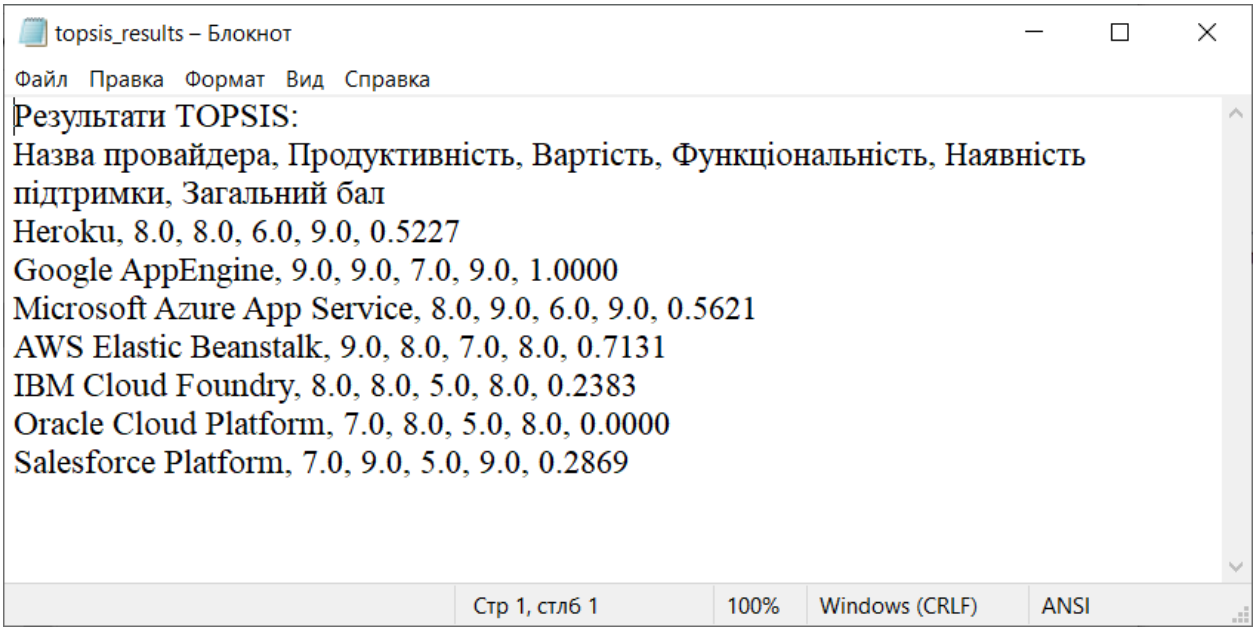

Рисунок 3.17 – Вміст файлу з результатами обчислень

Отже, розроблена програма дає можливість користувачу визначити оптимальний хмарний сервіс, який відповідає його вимогам. Така програма розрахована саме на клієнтів хмарних послуг.

3.4 Висновки до розділу 3

Розроблена інформаційна технологія для вибору хмарного сервісу є достатньо простою у використанні і не потребує спеціальних засобів для своєї реалізації.

Було запропоновано використовувати Python як мову розробки для даної системи. Це пояснюється наявністю бібліотек, які дозволили скоротити час розробки а також гнучко реалізувати запропоновані алгоритми рішення задачі.

Розроблена система відповідає вимогам. Вона надає користувачу можливість обирати хмарний сервіс як з запропонованих, так і з нових, які були додані самим користувачем.

Результати аналізу можна зберігати, що дозволяє проводити подальший детальний аналіз обраних альтернатив.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено систему підтримки прийняття рішень (СППР) для вибору моделі хмарного сервісу та конкретного хмарного провайдера.

1. В ході виконання даної роботи були проаналізовані моделі надання хмарних послуг. На даному етапі існує декілька різних способів використання хмарних послуг. Всі вони мають переваги та недоліки, тому вибір однієї з таких альтернатив може виявитися складною задачею.

2. Технології систем підтримки прийняття рішень мають досить довгу історію, але й зараз таке програмне забезпечення є актуальним. В сфері ІТ СППР дозволяє вирішувати складні задачі при управлінні різними інформаційними проєктами.

3. Результатом виконання роботи стала СППР для вибору хмарного провайдера. Розробка спиралася на інтеграцію методів нечіткої логіки для першого етапу (вибір моделі) та методу TOPSIS для другого етапу (ранжування провайдерів). Проєкт охоплює всі основні аспекти створення інформаційних систем, включаючи концептуальне проектування, вибір методів, технічну реалізацію та забезпечення зручності використання.

4. На першому етапі було використано алгоритм нечіткої логіки, який дозволяє враховувати невизначеність та суб’єктивність критеріїв оцінки. Для реалізації моделі були визначені лінгвістичні змінні, такі як "витрати", "рівень контролю" і "безпека", з відповідними функціями приналежності. Це забезпечило гнучкість і адаптивність у врахуванні різних вимог користувача до хмарної моделі (IaaS, PaaS або SaaS). Всі можливі комбінації значень критеріїв були враховані у системі правил, що гарантує повноту охоплення різних сценаріїв.

5. На другому етапі було застосовано метод TOPSIS, який базується на оцінці альтернатив за близькістю до ідеального рішення. Його перевагою є

здатність обробляти багатокритеріальні задачі, враховуючи ваги критеріїв. У цьому проєкті було передбачено чотири критерії для вибору провайдера: "продуктивність", "вартість", "функціональність" і "наявність підтримки". Це забезпечило об’єктивний підхід до визначення найкращого провайдера на основі реальних даних.

6. З технічної точки зору реалізація виконана мовою програмування Python, що забезпечує простоту інтеграції різних бібліотек, таких як scikit-fuzzy для роботи з нечіткою логікою, і гнучкість у розробці графічного інтерфейсу за допомогою Tkinter. Для зручності користувача система підтримує введення даних вручну та завантаження їх із JSON-файлів, а також дозволяє зберігати результати розрахунків у вигляді текстових файлів.

7. Запропоноване рішення демонструє ефективність інтеграції методів штучного інтелекту та багатокритеріального аналізу у вирішенні задач вибору. Воно може бути застосоване не тільки в контексті хмарних сервісів, але й для прийняття рішень у інших сферах, де є потреба враховувати множинні критерії і невизначеність.

Таким чином, створена СППР є ефективним інструментом для прийняття рішень у складних умовах, що дозволяє зменшити час і ресурси, необхідні для аналізу, та забезпечує високу якість вибору.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sikka R. et al. An Overview Of Cloud Computing. *IJIRCST*. 2021. P. 135-138. URL: https://doi.org/10.55524/ijircst.2021.9.6.31 (дата звернення 14.11.2024)

2. 4 Types of Cloud Computing and Their Advantages. URL: https://www.consensus.com/blog/types-of-cloud-computing/ (дата звернення 14.11.2024)

3. What Is Cloud Computing: Principal Models, Their Benefits, and Limitations. URL: https://adevait.com/blog/workplace/cloud-computing-models (дата звернення 14.11.2024)

4. Qu L., Wang Y., Orgun M. Cloud Service Selection Based on the Aggregation of User Feedback and Quantitative Performance Assessment. *IEEE Int. Conf. Serv. Comput*. 2013. P. 152–159.

5. Wilson, B.M.R., Khazaei, B., Hirsch, L.: Towards a cloud migration decision support system for small and medium enterprises in Tamil Nadu. *CINTI 2016 - 17th IEEE Int. Symp. Comput. Intell. Informatics Proc*. 2017. 341–346.

6. Eisa M., Younas M., Basu K., Zhu H. Trends and directions in cloud service selection. *Proc. IEEE Symp. Serv. Syst. Eng*. *SOSE*. 2016. P. 423–432.

7. Shen Z., Subbiah S., Gu X., Wilkes J. Cloudscale: elastic resource scaling for multitenant cloud systems, Proc. *2nd ACM Symposium on Cloud Computing*. 2011. № 5. P.1-14.

8. Madni S. H. H., Latif M. S. A., Coulibaly Y., Resource scheduling for infrastructure as a service (IaaS) in cloud computing: Challenges and opportunities, Network and Computer Applications. 2016. № 68. P. 173–200.

9. Ilieva G. Decision analysis for big data platform selection. *Engineering Sciences*, LVI. 2019. № 2. P. 5-18.

10. Alabool H., Kamil A., Arshad N., Alarabiat, D. Cloud service evaluation method-based Multi-Criteria Decision-Making: A systematic literature review. *Journal of Systems and Software*. 2018. № 139. P. 161-188.

11. Lee S., Seo K.-K. A Hybrid Multi-Criteria Decision-Making Model for a Cloud Service Selection Problem Using BSC, Fuzzy Delphi Method and Fuzzy AHP. *Wirel. Pers. Commun*. 2016. № 86. P. 57–75.

12. CISMIC: Service Measurement Index Framework Version 2.1. Carnegie Mellon University Silicon Valley, 2014. 125 p.

13. Garg S. K., Versteeg S., Buyya, R. SMICloud: A framework for comparing and ranking cloud services. *In Utility and Cloud Computing (UCC), Fourth IEEE International Conference on Cloud Computing*, 2011. P. 210-218.

14. Sun C. Research on investment decision-making model from the perspective of “Internet of Things + Big data”. *Future Generation Computer Systems*. 2020. № 107. P. 286-292.

15. Shahsavarani AM, Azad Marz Abadi E. The bases, principles, and methods of decision-making: A review of literature. *International Journal of Medical Reviews*. 2015. № 2(1). P. 214-225.

16. Sabaei D, Erkoyuncu J, Roy R. A review of multi-criteria decision making methods for enhanced maintenance delivery. *Procedia CIRP*. 2015. № 37. P. 30-35.

17. Saaty TL. What is the analytic hierarchy process? Springer; 1988 320 p.

18. Методи та системи штучного інтелекту: навч. посіб. / укл. Д.В. Лубко, С.В. Шаров. Мелітополь: ФОП Однорог Т.В., 2019. 264 с.

19. Михайленко В.С. Порівняльний аналіз методів аналізу ієрархій та нечіткої логіки в системі підтримки прийняття рішень. *Інформаційні інтелектуальні системи: матеріали XV Міжнарод. наук.-техн. конф.* Харків, ХПІ, 2011. С. 297-298.

20. Taherdoost Hamed, Madanchian, Mitra. Decision Making: Models, Processes, Techniques. *Cloud Computing and Data Science.* 2023. № 1-14. 10.37256/ccds.5120233284.

21. Rösch Manuel. Selecting the Right Cloud Service(s). 2016. 10.13140/RG.2.1.5071.4000.

22. Демиденко М. А. Системи підтримки прийняття рішень: навч. посіб. Дніпро : Національний гірничий університет, 2016. 104 с.

23. Карпенко М.Ю., Манакова Н.О., Гавриленко І.О. Технології створення програмних продуктів та інформаційних систем: навч. посібник. Харків : ХНУМГ ім. О. М. Бекетова, 2017. 93 с.

24. Модуль SkFuzzy. URL: https://scikit-fuzzy.readthedocs.io/en/latest/ (дата звернення 14.11.2024)

25. Модуль NumPy. URL: https://numpy.org/ (дата звернення 14.11.2024)

26. Модуль MatPlotLib. URL: https://matplotlib.org/ (дата звернення 14.11.2024)

ДОДАТКИ

Додаток А

Лістинг програми

```
import tkinter as tk
from tkinter import ttk, messagebox
import numpy as np
from skfuzzy import control as ctrl
import skfuzzy as fuzz

# ---- Нечітка логіка для вибору моделі ----
def setup_fuzzy_model():
    global model_ctrl, model_simulation

    # Визначення лінгвістичних змінних
    cost = ctrl.Antecedent(np.arange(0, 11, 1), 'cost')
    control = ctrl.Antecedent(np.arange(0, 11, 1), 'control')
    security = ctrl.Antecedent(np.arange(0, 11, 1), 'security')

    model = ctrl.Consequent(np.arange(0, 101, 1), 'model')

    # Нечіткі множини для витрат
    cost['low'] = fuzz.trapmf(cost.universe, [0, 0, 2, 4])
    cost['medium'] = fuzz.trimf(cost.universe, [2, 5, 8])
    cost['high'] = fuzz.trapmf(cost.universe, [6, 8, 10, 10])

    # Нечіткі множини для контролю
    control['low'] = fuzz.trapmf(control.universe, [0, 0, 2, 4])
    control['medium'] = fuzz.trimf(control.universe, [3, 5, 7])
    control['high'] = fuzz.trapmf(control.universe, [6, 8, 10, 10])

    # Нечіткі множини для безпеки
    security['low'] = fuzz.trapmf(security.universe, [0, 0, 2, 4])
    security['medium'] = fuzz.trimf(security.universe, [3, 5, 7])
    security['high'] = fuzz.trapmf(security.universe, [6, 8, 10, 10])

    # Нечіткі множини для моделі
    model['IaaS'] = fuzz.trimf(model.universe, [0, 20, 40])
    model['PaaS'] = fuzz.trimf(model.universe, [30, 50, 70])
    model['SaaS'] = fuzz.trimf(model.universe, [60, 80, 100])

    # Правила для всіх можливих комбінацій
    rules = []

    cost_levels = ['low', 'medium', 'high']
    control_levels = ['low', 'medium', 'high']
    security_levels = ['low', 'medium', 'high']

    # Генерація правил
    for c in cost_levels:
```

```
        for ctrl_level in control_levels:
            for sec in security_levels:
                if c == 'low' and ctrl_level == 'high':
                    rules.append(ctrl.Rule(cost[c] & control[ctrl_level] & security[sec], model['IaaS']))
                elif c == 'medium' and ctrl_level == 'medium':
                    rules.append(ctrl.Rule(cost[c] & control[ctrl_level] & security[sec], model['PaaS']))
                elif c == 'high' and ctrl_level == 'low':
                    rules.append(ctrl.Rule(cost[c] & control[ctrl_level] & security[sec], model['SaaS']))
                else:
                    rules.append(ctrl.Rule(cost[c] & control[ctrl_level] & security[sec], model['PaaS']))

    # Створення системи управління
    model_ctrl = ctrl.ControlSystem(rules)
    model_simulation = ctrl.ControlSystemSimulation(model_ctrl)

# ---- TOPSIS метод ----
def topsis(data, weights):
    norm_data = data / np.sqrt((data**2).sum(axis=0))
    weighted_data = norm_data * weights

    ideal_solution = np.max(weighted_data, axis=0)
    anti_ideal_solution = np.min(weighted_data, axis=0)

    distances_to_ideal = np.sqrt(((weighted_data - ideal_solution)**2).sum(axis=1))
    distances_to_anti_ideal = np.sqrt(((weighted_data - anti_ideal_solution)**2).sum(axis=1))

    closeness = distances_to_anti_ideal / (distances_to_ideal + distances_to_anti_ideal)
    return closeness

# ---- Графічний інтерфейс ----
def start_step1():
    def submit_criteria():
        global model_simulation, chosen_model
        try:
            cost = int(cost_entry.get())
            control = int(control_entry.get())
            security = int(security_entry.get())

            model_simulation.input['cost'] = cost
            model_simulation.input['control'] = control
            model_simulation.input['security'] = security
            model_simulation.compute()

            chosen_model = model_simulation.output['model']
            if chosen_model < 40:
                chosen_model_label.config(text="Обрана модель: IaaS")
            elif chosen_model < 70:
                chosen_model_label.config(text="Обрана модель: PaaS")
            else:
                chosen_model_label.config(text="Обрана модель: SaaS")
            next_button.config(state='normal')
```

```
            except Exception as e:
                messagebox.showerror("Помилка", f"Помилка введення даних: {e}")

        def go_to_step2():
            root.destroy()
            start_step2()

        root = tk.Tk()
        root.title("Етап 1: Вибір моделі")

        tk.Label(root, text="Введіть значення критеріїв (0-10)").pack(pady=10)
        tk.Label(root, text="Витрати").pack()
        cost_entry = tk.Entry(root)
        cost_entry.pack()

        tk.Label(root, text="Рівень контролю").pack()
        control_entry = tk.Entry(root)
        control_entry.pack()

        tk.Label(root, text="Безпека").pack()
        security_entry = tk.Entry(root)
        security_entry.pack()

        submit_button = tk.Button(root, text="Підтвердити", command=submit_criteria)
        submit_button.pack(pady=10)

        chosen_model_label = tk.Label(root, text="Модель ще не обрана.")
        chosen_model_label.pack(pady=10)

        next_button = tk.Button(root, text="Перейти до етапу 2", state='disabled', command=go_to_step2)
        next_button.pack(pady=10)

        root.mainloop()

    import json
    from tkinter import filedialog

    def start_step2():
        def update_table():
            """Оновлює таблицю зі списком введених альтернатив."""
            for row in tree.get_children():
                tree.delete(row)
            for idx, alt in enumerate(alternatives, start=1):
                tree.insert("", "end", values=(idx, alt["name"], *alt["scores"]))

        def add_alternative():
            """Додає альтернативу до списку."""
            try:
                name = alt_name_entry.get()
                scores = list(map(float, alt_scores_entry.get().split()))
                if len(scores) != len(criteria):
```

```
            raise ValueError("Кількість оцінок має відповідати кількості критеріїв.")
        alternatives.append({"name": name, "scores": scores})
        messagebox.showinfo("Успіх", f"Альтернатива '{name}' додана.")
        alt_name_entry.delete(0, tk.END)
        alt_scores_entry.delete(0, tk.END)
        update_table()
    except Exception as e:
        messagebox.showerror("Помилка", f"Помилка додавання альтернативи: {e}")

def load_from_file():
    """Завантажує альтернативи з JSON-файлу."""
    try:
        file_path = filedialog.askopenfilename(filetypes=[("JSON Files", "*.json")])
        if not file_path:
            return
        with open(file_path, "r") as file:
            data = json.load(file)
            if not isinstance(data, list):
                raise ValueError("Формат файлу некоректний: очікується список альтернатив.")

            global alternatives
            alternatives = []
            for item in data:
                if "name" not in item or "scores" not in item:
                    raise ValueError("Кожна альтернатива повинна містити 'name' і 'scores'.")
                # Перетворення оцінок на числовий формат
                scores = list(map(float, item["scores"]))
                alternatives.append({"name": item["name"], "scores": scores})

            update_table()
            messagebox.showinfo("Успіх", f"Дані завантажено з файлу '{file_path}'.")
    except Exception as e:
        messagebox.showerror("Помилка", f"Помилка завантаження даних: {e}")

def calculate_topsis():
    """Розраховує TOPSIS та виводить результат."""
    try:
        weights = [
            float(weight_perf_entry.get()),
            float(weight_cost_entry.get()),
            float(weight_func_entry.get()),
            float(weight_support_entry.get()),
        ]
        if len(weights) != len(criteria):
            raise ValueError("Кількість ваг має відповідати кількості критеріїв.")

        data = np.array([alt['scores'] for alt in alternatives])
        closeness = topsis(data, weights)
        for i, alt in enumerate(alternatives):
            alt["score"] = closeness[i]
```

```
            best_index = np.argmax(closeness)
            result_label.config(text=f"Рекомендований провайдер: {alternatives[best_index]['name']}")
            update_table_with_scores()
        except Exception as e:
            messagebox.showerror("Помилка", f"Помилка обчислення TOPSIS: {e}")

    def save_results():
        """Зберігає результати в файл."""
        try:
            with open("topsis_results.txt", "w") as file:
                file.write("Результати TOPSIS:\n")
                 file.write("Назва провайдера, Продуктивність, Вартість, Функціональність, Наявність підтримки, Загальний бал\n")
                for alt in alternatives:
                    file.write(f"{alt['name']}, {', '.join(map(str, alt['scores']))}, {alt.get('score', 0):.4f}\n")
            messagebox.showinfo("Успіх", "Результати збережено у файл 'topsis_results.txt'.")
        except Exception as e:
            messagebox.showerror("Помилка", f"Помилка збереження файлу: {e}")

    def update_table_with_scores():
        """Оновлює таблицю, включаючи загальні бали альтернатив."""
        for row in tree.get_children():
            tree.delete(row)
        for idx, alt in enumerate(alternatives, start=1):
            tree.insert("", "end", values=(idx, alt["name"], *alt["scores"], f"{alt.get('score', 0):.4f}"))

    root = tk.Tk()
    root.title("Етап 2: Вибір провайдера")

    # Ліва частина - введення даних
    left_frame = tk.Frame(root)
    left_frame.pack(side=tk.LEFT, padx=10, pady=10)

    tk.Label(left_frame, text="Введіть ваги критеріїв:").pack(pady=5)
    tk.Label(left_frame, text="Продуктивність:").pack()
    weight_perf_entry = tk.Entry(left_frame)
    weight_perf_entry.pack()

    tk.Label(left_frame, text="Вартість:").pack()
    weight_cost_entry = tk.Entry(left_frame)
    weight_cost_entry.pack()

    tk.Label(left_frame, text="Функціональність:").pack()
    weight_func_entry = tk.Entry(left_frame)
    weight_func_entry.pack()

    tk.Label(left_frame, text="Наявність підтримки:").pack()
    weight_support_entry = tk.Entry(left_frame)
    weight_support_entry.pack()

    tk.Label(left_frame, text="Додати альтернативу").pack(pady=10)
```

```
    tk.Label(left_frame, text="Назва провайдера:").pack()
    alt_name_entry = tk.Entry(left_frame)
    alt_name_entry.pack()

    tk.Label(left_frame, text=f"Оцінки за критеріями (через пробіл, {len(criteria)} значень):").pack()
    alt_scores_entry = tk.Entry(left_frame)
    alt_scores_entry.pack()

    tk.Button(left_frame, text="Додати", command=add_alternative).pack(pady=10)
    tk.Button(left_frame, text="Завантажити з файлу", command=load_from_file).pack(pady=10)
    tk.Button(left_frame, text="Розрахувати TOPSIS", command=calculate_topsis).pack(pady=10)
    tk.Button(left_frame, text="Зберегти результати", command=save_results).pack(pady=10)

    result_label = tk.Label(left_frame, text="Результат ще не розраховано.")
    result_label.pack(pady=10)

    # Права частина - таблиця альтернатив
    right_frame = tk.Frame(root)
    right_frame.pack(side=tk.RIGHT, padx=10, pady=10)

    tk.Label(right_frame, text="Список альтернатив").pack(pady=5)
    columns = ["#", "Назва"] + criteria + ["Загальний бал"]
    tree = ttk.Treeview(right_frame, columns=columns, show="headings")
    tree.pack()

    for col in columns:
        tree.heading(col, text=col)
        tree.column(col, width=100, anchor="center")
    root.mainloop()

# Глобальні змінні
criteria = ["Продуктивність", "Вартість", "Функціональність", "Наявність підтримки"]
alternatives = []
# Запуск програми
setup_fuzzy_model()
start_step1()
```