Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції IT-систем із хмарними службами”

Виконав здобувач вищої освіти
II курсу, групи КН-25М
ОПП «Комп’ютерні науки»
спеціальності F3 «Комп’ютерні науки»
_____ Сидоренко А.С.
«____» _____ 2026 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Коваленко А.С.
«____» _____ 2026 р.

Нормоконтроль проекту
_____ Коваленко А.С.
«____» _____ 2026 р.

Рецензент _____
_____

Підписано КЕП
Центральноукраїнський національний технічний університет
Сидоренко Артем Сергійович
Сертифікат: 514B5C86A1E5DA1104000000DD2A9F00B6F24B05
Дійсний з 23.01.2026 19:22:17 по 23.01.2027 19:22:16

Центральноукраїнський національний технічний університет
Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Рівень вищої освіти *магістр*
Галузь знань _F_____*"*Інформаційні технології*"*_____
Спеціальність F3 *"*Комп'ютерні науки*"*
Освітньо-професійна (освітньо-наукова) програма _*"*Комп'ютерні науки*"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 1 » вересня 2026 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Сидоренку Артему Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції IT-систем із хмарними службами*

2. Керівник роботи *Коваленко Анна Степанівна, кандидат технічних наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 42-13 від 17.08.2026 року

3. Строк подання студентом роботи до захисту *2.12.2026 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції IT-систем із хмарними службами*

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

1. Призначення та область використання.	*5. Впровадження системи в промислову експлуатацію*
2. Перегляд аналогічних існуючих систем.	*6. Наукова новизна.*
3. Опис і обґрунтування проектних рішень.	*7. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*8. Висновки.*

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

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Марченко К.М., к.т.н., доцент	06.10.2026	16.11.2026

8. Дата видачі завдання « 1 » вересня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2026 р.	
3.	Розробка моделі компонента	20.10.2026 р.	
4.	Розробка структур даних	25.10.2026 р.	
5.	Розробка алгоритмів системи	30.10.2026 р.	
6.	Програмування алгоритмів	10.11.2026 р.	
7.	Розрахунки з охорони праці та техніки безпеки	15.11.2026 р.	
8.	Оформлення ПЗ	17.11.2026 р.	
9.	Попередній захист роботи	2.12.2026 р.	

Дата видачі завдання
« 1 » вересня 2026 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 1 » вересня 2026 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Сидоренко А.С. Дослідження та програмна реалізація системи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції ІТ-систем із хмарними службами. ФЗ Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції ІТ-систем із хмарними службами.

Метою розробки є дослідження та програмна реалізація системи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції ІТ-систем із хмарними службами.

Об'єктом дослідження є процес адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції ІТ-систем із хмарними службами.

Предметом дослідження є методи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції ІТ-систем із хмарними службами.

Методи дослідження базуються на методах хмарних обчислень, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції ІТ-систем із хмарними службами.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерні науки, програмне забезпечення, інтелектуальна інтеграція, ІТ-система, хмарні служби

ABSTRACT

Sydorenko A.S. Research and Software Implementation of an Adaptive Middleware System for the Intelligent Integration of IT Systems with Cloud Services. F3 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

This final qualification thesis, submitted for the second (master's) level of higher education, presents the development of software designed for an adaptive middleware system intended for the intelligent integration of IT systems with cloud services.

The objective of this development is the research and software implementation of an adaptive middleware system for the intelligent integration of IT systems with cloud services.

The object of the research is the process of adaptive middleware for the intelligent integration of IT systems with cloud services.

The subject of the research consists of methods for adaptive middleware used for the intelligent integration of IT systems with cloud services.

The research methodology is based on cloud computing techniques, mathematical statistics, and software development practices.

The outcome of the work is the software implementation of an adaptive middleware system for the intelligent integration of IT systems with cloud services.

During the development of the software model, an analysis of existing hardware and software tools was conducted. All components of the developed software are described in detail.

A user-friendly interface has been developed, and instructions for operating the software are provided.

The software is designed for use on personal computers running the Windows 10/11 operating system. The program was developed in the Python environment.

Keywords: computer science, software, intelligent integration, IT system, cloud services

ВІДОМІСТЬ ПРОЕКТУ

		Текстові документи			
1	А4	Пояснювальна записка	83	–	
2	А4	Робоча програма	54	–	Додаток А
		Графічні документи			
3	А4	Наукова новизна	1	–	
4	А4	Структурна схема	1	–	
5	А4	Функціональна схема роботи системи	1	–	
6	А4	Діаграма взаємодії процесів	1	–	
7	А4	Блок-схема основної програми	1	–	
8	А4	Блок-схема роботи підпрограми	1	–	

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 9
ВСТУП..... 10
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 12
1.1 Призначення системи..... 12
1.2 Область застосування 12
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 15
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 15
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування 27
2.3 Розгорнута постановка завдання 29
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 30
3.1 Опис функціонування системи 30
3.2 Розробка структурної схеми..... 33
3.3 Розробка функціональної схеми 40
3.4 Розробка діаграми процесів 43
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ 45
4.1 Розробка блок-схем та опис алгоритмів функціонування системи 45
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 58
6 НАУКОВА НОВИЗНА 63

7 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ 64
7.1 Вступ 64
7.2 Аналіз умов праці на робочому місці фахівця 66
7.3 Розробка заходів з умов поліпшення охорони праці 69
7.4 Розрахункова частина 70
7.5 Висновки до розділу 72
8 ОСНОВНІ ВИСНОВКИ 73
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 75

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ІТ	–	інформаційні технології
ПЗ	–	програмне забезпечення
ЦОД	–	центр обробки даних
QoS	–	механізми контролю й керування якістю

ВСТУП

Актуальність теми. Фрагментація даних є універсальною проблемою для сучасних підприємств. Оскільки SaaS-додатки, локальні системи, хмарні сховища даних та Інтернет речей генерують величезні обсяги даних, компанії стикаються з ізольованими наборами даних та непов'язаними аналітичними даними. Інтеграція хмарних даних вирішує цю проблему, об'єднуючи різнорідні джерела даних у спрощені, доступні та керовані канали передачі даних.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції IT-систем із хмарними службами.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції IT-систем із хмарними службами.

– Дослідження системи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції IT-систем із хмарними службами.

– Програмна реалізація системи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції IT-систем із хмарними службами.

Об'єктом дослідження є процес адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції IT-систем із хмарними службами.

Предметом дослідження є методи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції IT-систем із хмарними службами.

Методи дослідження базуються на методах хмарних обчислень, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції IT-систем із хмарними службами.

– Розроблено вітчизняний продукт адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції ІТ-систем із хмарними службами, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції ІТ-систем із хмарними службами.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVIII науково-технічній конференції здобувачів вищої освіти «Наука-виробництву», основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №17.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції ІТ-систем із хмарними службами, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Інтеграція хмарних даних – це практика об’єднання даних з кількох систем, як локальних, так і хмарних, у централізовані хмарні середовища, такі як озера даних, сховища даних або хмарні додатки.

Це дозволяє організаціям:

– Консолідація фрагментованих даних для створення єдиного джерела достовірної інформації.

– Підтримка аналітики в режимі реального часу та ініціатив у сфері штучного інтелекту/машинного навчання.

– Підвищте операційну ефективність, автоматизувавши потоки даних.

Інструменти інтеграції хмарних даних керують переміщенням, трансформацією, управлінням та безпекою даних у гібридних та мультихмарних архітектурах.

1.2 Область застосування

Чому інтеграція хмарних даних є пріоритетом бізнесу:

1. Швидке впровадження хмарних технологій та SaaS. Бізнеси покладаються на десятки SaaS-платформ, таких як Salesforce, HubSpot та NetSuite, одночасно запускаючи хмарні сервіси на AWS, Azure та GCP. Інтеграція узгоджує ці ресурси для отримання цілісного розуміння.

2. Вимоги бізнесу в режимі реального часу. Інтеграція потокових даних підтримує виявлення шахрайства, персоналізацію та операційну аналітику шляхом синхронізації даних у режимі реального часу.

3. Ефективність витрат та еластичність. Хмарні сервіси пропонують безсерверні варіанти, автоматичне масштабування та моделі ціноутворення з оплатою за використання, оптимізуючи як капітальні, так і операційні витрати.

4. Готовність до штучного інтелекту/машинного навчання та розширеної аналітики. Озера даних та сховища даних передають структуровані та неструктуровані дані в конвеєри машинного навчання, розкриваючи можливості прогнозування та автоматизації.

5. Управління та дотримання вимог. Хмарні рішення для інтеграції даних включають функції управління даними, необхідні для дотримання вимог GDPR, HIPAA та CCPA.

Архітектури інтеграції хмарних даних

Пакетна інтеграція:

– Опис: Заплановані передачі великих обсягів даних.

– Приклад використання: ETL для сховищ даних, історична звітність.

– Інструменти: AWS Glue, Фабрика даних Azure, Integrate.io

Потокове передавання в реальному часі:

– Опис: Безперервна синхронізація даних за допомогою потоків подій.

– Приклад використання: Моніторинг інформаційних панелей, виявлення шахрайства.

– Інструменти: Apache Kafka, AWS Kinesis, Integrate.io

Збір даних змін (CDC):

– Опис: Записує та відтворює поступові зміни з вихідних систем.

– Приклад використання: оновлення сховища даних з низькою затримкою.

– Інструменти: Fivetran, Debezium, Integrate.io

Віртуалізація даних:

– Опис: Дозволяє виконувати запити між системами без переміщення даних.

– Варіант використання: Легка інтеграція, галузі з високим рівнем управління.

– Інструменти: Denodo, віртуалізація даних Red Hat.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції IT-систем із хмарними службами, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

ONEiO

ONEiO – це керований iPaaS: керована послуга інтеграції, яка забезпечує результати iPaaS, підключені системи, контрольовані потоки та час безвідмовної роботи за угодою про рівень обслуговування (SLA), без необхідності участі клієнта в управлінні платформою. ONEiO проектує, запускає, контролює та підтримує інтеграції від початку до кінця. Самообслуговування iPaaS – це платформа, яку ви купуєте та експлуатуєте. Керований iPaaS забезпечує ті самі результати інтеграції, а постачальник керує ними. Можливості ті самі. Різниця лише в тому, хто нею керує.

Штаб-квартира: Гельсінкі, Фінляндія.

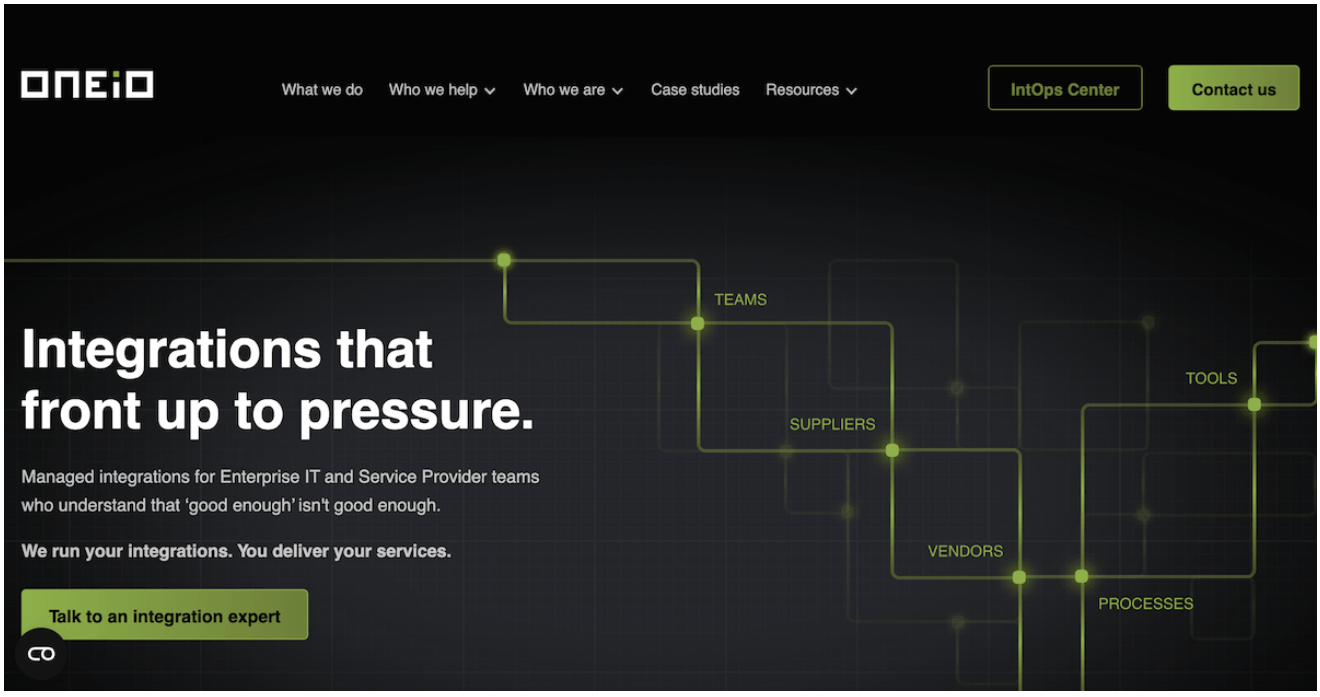

Рисунок 2.1 – Інтерфейс користувача ONEiO

ONEiO підключає кожну систему один раз до спільного рівня інтеграції, замість того, щоб створювати з'єднання "точка-точка", тому інструменти можна замінювати, а екосистеми можуть зростати без перебоїв у інтеграції. Доставка стандартизована та оцінюється як підписка, або на основі обсягу, або фіксована для кожної інтеграції, що забезпечує передбачуваність витрат. ONEiO має глибокий досвід у сфері ITSM, постачальників керованих послуг та сценаріїв B2B eBonding, де той самий запис про інцидент, запит або зміну має залишатися узгодженим в організаціях, які не мають спільної служби підтримки. Більшість постачальників інтеграції створюють його та повертають назад. ONEiO бере на себе операційну відповідальність назавжди, а клієнти вирішують, який рівень контролю вони залишають. Оберіть ONEiO, якщо ви хочете отримати результати інтеграції без найму команди інтеграції. Оберіть iPaaS самообслуговування, якщо у вас є зріла власна інженерія інтеграції та ви хочете максимального контролю над платформою. ONEiO проектує, запускає та підтримує інтеграції цілодобово за угодою про рівень обслуговування (SLA).

Переваги:

– Повністю керована та за підтримки SLA. ONEiO виконує повний життєвий цикл інтеграції, від проектування та розгортання до цілодобового моніторингу та обслуговування, тому надійна інтеграція не потребує власної команди інтеграторів.

– Створено для складних, мінливих середовищ. Кожна система підключається до ONEiO один раз замість прямої збірки, з глибоким досвідом роботи в ITSM, MSP та сценаріях B2B eBonding.

– Передбачувана вартість та швидка доставка. Фіксована ціна за підписку, залежно від обсягу або за інтеграцію, зі стандартизованою доставкою протягом тижнів, а не місяців.

Недоліки:

– Не платформа самообслуговування. Якщо у вас є зріла команда інтеграторів і ви хочете мати практичний контроль над платформою, iPaaS самообслуговування надасть вам більше можливостей.

– Немає повного пакету керування API. ONEiO не призначений для продуктізації API, такої як шлюзи, портали розробників або монетизація API.

– Тільки хмара. Локальні системи можна інтегрувати, але сам ONEiO не можна розгорнути локально, що є обмеженням для середовищ з обмеженим доступом або суворим зберіганням даних.

SnapLogic

SnapLogic – це low-code-платформа самообслуговування для побудови конвеєрів даних та додатків через візуальний інтерфейс. З 2026 року вона позиціонує себе як агентна інтеграційна компанія.

Штаб-квартира: Сан-Матео, Каліфорнія

Інтелектуальна платформа інтеграції SnapLogic використовує конструктор конвеєрів з функцією перетягування та великою бібліотекою попередньо створених конекторів під назвою Snaps, що охоплюють інтеграцію даних у стилі ETL, інтеграцію додатків у реальному часі та створення API з мінімальним кодуванням. У липні 2026 року SnapLogic перезапустила SnapGPT як агентського помічника, що охоплює весь життєвий цикл інтеграції, від побудови та тестування конвеєрів до їх документування та усунення несправностей, разом зі шлюзом штучного інтелекту та ідентифікатором довіреного агента для керування тим, до чого дозволено звертатися агентам. Можливості B2B та EDI залишаються скромними. Ваша команда проектує, керує та обслуговує конвеєри.

Переваги:

– Візуальний конструктор конвеєрів, який швидко вивчається для роботи з поширеними шаблонами.

– Прозоре ціноутворення та присутність на основних хмарних торгових майданчиках.

– SnapGPT тепер охоплює повний життєвий цикл інтеграції, а не лише генерує конвеєр.

Недоліки:

– Обмежені можливості B2B та EDI.

– Грубі дозволи доступу для великих команд.

– Налагодження та розширене використання вимагають справжнього навчання.

Celigo

Celigo – це iPaaS самообслуговування, орієнтований на інтеграцію SaaS та автоматизацію процесів бек-офісу, орієнтований переважно на компанії середнього та вищого середнього бізнесу.

Платформа Celigo integrator.io поєднує велику бібліотеку шаблонів бізнес-процесів з автоматизованою обробкою помилок і є найсильнішою у фінансах, електронній комерції та бек-офісі, орієнтованому на NetSuite. У березні 2026 року Celigo запустила Ora, платформу для роботи з природною мовою, разом з Agent Builder, яка дозволяє бізнес-командам створювати та керувати агентами штучного інтелекту поверх існуючих інтеграцій. Інтеграція B2B та досвід розробників є заявленими пріоритетами дорожньої карти. Celigo підходить командам, яким потрібні бізнес-користувачі, які створюють автоматизацію під наглядом IT-відділу. Ваша команда проектує, запускає та підтримує інтеграції.

Переваги:

– Категоризація та вирішення помилок під час виконання за допомогою штучного інтелекту.

– Велика бібліотека шаблонів бізнес-процесів, найсильніша в NetSuite та електронній комерції.

– Ora та Agent Builder дозволяють людям, які не є розробниками, створювати керовані агенти штучного інтелекту.

Недоліки:

– Менше контролю підприємства у великих масштабах.

– Операційні дані важко експортувати до зовнішніх інструментів моніторингу.

– Складність інтерфейсу швидко зростає, виходячи за межі використання на основі шаблонів.

TIBCO

TIBCO має давній досвід у сфері корпоративної інтеграції та обміну повідомленнями, а також відома тим, що забезпечує роботу критично важливих високопродуктивних систем. Вона є частиною Cloud Software Group з 2022 року.

Штаб-квартира: Санта-Клара, Каліфорнія.

Платформа TIBCO об'єднує інтеграцію, обмін повідомленнями, управління API та продукти обробки даних для організацій, які працюють з великими обсягами та чутливими до затримки робочими навантаженнями, часто у сфері фінансових послуг, телекомунікацій та логістики. TIBCO є частиною Cloud Software Group з моменту об'єднання з Citrix у 2022 році. Її робота в галузі штучного інтелекту у 2026 році зосереджена на додаванні можливостей моделі та агента до існуючих потоків BusinessWorks, а не на запуску окремої платформи агентів, що підходить клієнтам, які захищають велику встановлену мережу. TIBCO впроваджує стратегії інтеграції, орієнтовані на дані, в підприємства, які вже використовують TIBCO. Ваша команда проектує, запускає та підтримує інтеграції.

Переваги:

– Перевірено при дуже великих обсягах повідомлень та низькій затримці.

– Широкий спектр стилів інтеграції в одному портфоліо.

– Ціноутворення на основі обсягу, яке змінюється залежно від використання.

Недоліки:

– Непослідовний досвід у портфоліо, що накопичувалося протягом багатьох років.

– Деякі можливості доступні лише на власному хостингу.

– Повільніша частота випусків, ніж у постачальників хмарних рішень.

IBM App Connect

IBM App Connect – це платформа корпоративної інтеграції для підключення програм, автоматизації робочих процесів та синхронізації даних у гібридних хмарних середовищах.

Штаб-квартира: Армонк, Нью-Йорк.

App Connect поєднує картографування та трансформацію за допомогою штучного інтелекту з гібридним розгортанням, що дозволяє підключати застарілі системи та хмарні додатки з однієї платформи. Він включає керування API та архітектуру, керовану подіями, для синхронізації в режимі реального часу, і входить до портфоліо автоматизації IBM поряд з webMethods, яке IBM придбала у Software AG у 2024 році. Він є найбільш привабливим для організацій, які вже використовують проміжне програмне забезпечення IBM. Ваша команда проектує, запускає та підтримує інтеграції.

Переваги:

– Потужні гібридні та локальні варіанти розгортання.

– Вбудоване керування API та засоби контролю безпеки.

– Підходить для організацій, які вже використовують проміжне програмне забезпечення та інструменти автоматизації IBM.

Недоліки:

– Крута крива навчання для команд без досвіду роботи з IBM.

– Перекриття портфоліо з webMethods ускладнює читання дорожньої карти.

– Деякі клієнти повідомляють, що ліцензування складне для моделювання.

Oracle Integration Cloud

Oracle Integration Cloud – це iPaaS для підключення додатків Oracle та сторонніх розробників із попередньо вбудованими адаптерами та автоматизацією процесів.

Штаб-квартира: Остін, Техас.

Oracle Integration Cloud поєднує візуальний конструктор інтеграцій, попередньо створені адаптери для Oracle Fusion та сторонніх додатків, а також автоматизацію процесів в одному сервісі. Засоби управління та безпеки створені для регламентованого використання в підприємствах, а Oracle розширює платформу картографуванням на основі штучного інтелекту та можливостями

агентів, пов'язаними з її додатками Fusion. Це природний вибір для організацій, стандартизованих на Oracle, і його складніше виправдати в інших місцях. Ваша команда проектує, запускає та підтримує інтеграції.

Переваги:

– Потужна інтеграція з Oracle

– Безпека корпоративного рівня

– Візуальний дизайнер, який зменшує ручне кодування стандартних шаблонів.

Недоліки:

– Менша гнучкість із інструментами, що не належать Oracle

– Може стати дорогим у великих масштабах

– Менша екосистема незалежних партнерів, ніж у лідерів ринку.

SAP Integration Suite

SAP Integration Suite – це компонент iPaaS платформи SAP Business Technology Platform, створений для інтеграції всередині екосистеми SAP та навколо неї.

Штаб-квартира: Вальдорф, Німеччина.

Пакет SAP Integration Suite охоплює інтеграцію додатків, даних, API та B2B, пропонуючи кілька тисяч готових інтеграцій для додатків SAP та менший набір для сторонніх систем. SAP узгоджує пакет зі своїми бізнес-агентами штучного інтелекту, використовуючи Business Technology Platform як керований рівень даних та процесів, через який діють ці агенти. Цей пакет чудово підходить для організацій, які значно інвестували в SAP, і його важче виправдати в інших місцях. Ваша команда проектує, запускає та підтримує інтеграції.

Переваги:

– Тисячі шаблонів, переважно для додатків SAP.

– Глобальна операційна та допоміжна присутність через SAP.

– Прозоре ціноутворення.

Недоліки:

– Сприймається переважно як інструмент, орієнтований на SAP.

– Бізнес-модель значною мірою схиляється до додатків SAP.

– Ціноутворення не адаптоване до всіх сегментів ринку.

Служби інтеграції Microsoft Azure

Служби інтеграції Microsoft Azure – це набір служб Azure для підключення програм, автоматизації робочих процесів та керування даними в хмарних та локальних середовищах.

Штаб-квартира: Сіетл, штат Вашингтон.

Служби інтеграції Azure – це набір послуг, а не окремий продукт: Logic Apps для автоматизації робочих процесів, API Management для публікації та керування API, Service Bus для обміну повідомленнями та Event Grid для маршрутизації подій. Покупці самостійно збирають та керують цією комбінацією, що забезпечує гнучкість та переносить роботу з архітектурою інтеграції на їхню власну команду. Microsoft розширює набір можливостями агента, пов’язаними з Copilot та Azure AI Foundry. Це вибір за замовчуванням для організацій, які вже стандартизовані в Azure. Ваша команда проектує, запускає та підтримує інтеграції.

Переваги:

– Глибока інтеграція з продуктами Microsoft

– Гнучка гібридна та мультихмарна підтримка

– Масштабований API та інструменти автоматизації робочих процесів

Недоліки:

– Найкраще підходить для компаній, які вже використовують сервіси Azure

– Деякі клієнти повідомляють, що ціноутворення на споживання важко прогнозувати у великих масштабах.

– Моніторинг та обробку помилок у всіх сервісах потрібно налаштовувати самостійно.

AWS Application Integration Services

AWS Application Integration Services – це група сервісів AWS для підключення додатків, маршрутизації подій та автоматизації робочих процесів в екосистемі AWS.

Штаб-квартира: Сіетл, штат Вашингтон.

AWS Application Integration Services – це група сервісів, а не єдина платформа: EventBridge для маршрутизації подій, Step Functions для оркестрації, SQS та SNS для обміну повідомленнями, а також AppFlow для переміщення даних до та з SaaS-застосунків. Це поєднання є потужним для архітектури, керованої подіями, всередині AWS, і воно передбачає інженерні можливості, оскільки ваша команда збирає, захищає та керує цими елементами. AWS додає можливості взаємодії з агентами через Bedrock та AgentCore. Це підходить для організацій, орієнтованих на AWS, з власними хмарними інженерами. Ваша команда проектує, запускає та підтримує інтеграції.

Переваги:

– Глибока інтеграція з сервісами AWS.

– Високомасштабована та подієво-орієнтована архітектура.

– Надійні інструменти для обміну повідомленнями та автоматизації.

Недоліки:

– Найкраще підходить для середовищ, орієнтованих на AWS.

– Немає попередньо створених конекторів для бізнес-застосунків, порівнянних з iPaaS.

– Витрати швидко зростають завдяки високій пропускній здатності подій.

MuleSoft Anypoint Platform

MuleSoft Anypoint Platform – це самообслуговувана корпоративна iPaaS, побудована на основі підходу до інтеграції, що ґрунтується на API. MuleSoft є компанією Salesforce, і платформа орієнтована на великі IT-організації та системні інтегратори, особливо тих, хто вже працює в екосистемі Salesforce.

Штаб-квартира: Сан-Франциско, Каліфорнія.

Платформа Anypoint поєднує розробку API, управління API та інтеграцію в одній платформі самообслуговування, з організаційним принципом підключення на основі API. Salesforce розширила її до агентного ШІ за допомогою MuleSoft Agent Fabric, яка реєструє, оркеструє та керує агентами ШІ від будь-якого постачальника, а з того часу додала автоматизоване виявлення агентів та площину керування для багатоагентного управління. MuleSoft підходить організаціям з власною інженерією інтеграції, які бажають стратегії на основі API та готові найняти персонал для роботи на платформі. Ваша команда проектує, запускає та підтримує інтеграції.

Переваги:

– Розробка, управління та життєвий цикл API на єдиній платформі.

– Велика бібліотека шаблонів та конекторів з автоматизацією та інструментами RPA.

– Agent Fabric додає реєстр, спостереження та контроль політик для агентів штучного інтелекту від кількох постачальників.

Недоліки:

– Одна з найвищих витрат на вхід на ринок iPaaS, зі складним ціноутворенням.

– Найсильніше вписується в екосистему Salesforce.

– Користувачі повідомляють про крутий графік навчання, а для його роботи потрібні спеціальні навички роботи з MuleSoft.

Ключові критерії вибору для хмарних інтеграцій

Оцінюючи платформи хмарної інтеграції, слід визначити пріоритети:

– Доступність конекторів: Чи легко платформа підключається до ваших поточних та запланованих систем? Перевірте наявність попередньо створених конекторів для інструментів ITSM (ServiceNow, Jira, BMC Remedy, Zendesk).

– Стиль інтеграції: Визначте, чи потрібна вам пакетна інтеграція даних (ETL), інтеграція подій у реальному часі, інтеграція API чи B2B/EDI. Зіставте

сильні сторони платформи з вимогами.

– Простота використання проти гнучкості: враховуйте, хто створює та підтримує інтеграції. Команди, що не є розробниками, отримують вигоду від інтерфейсів з низьким кодом або керованих підходів. Досвідчені команди розробників можуть надавати перевагу точному контролю.

– Масштабованість та продуктивність: Оцініть обсяги даних та вимоги до пропускної здатності. Платформи, орієнтовані на підприємства, ефективно обробляють великі обсяги.

– Модель ціноутворення: Порівняйте ціни за конектор, за транзакцію або за передплату. Прогнозуйте витрати на 3-5 років у міру зростання потреб в інтеграції. Передбачуваність проти оплати за використання є ключовим компромісом.

– Безпека та відповідність вимогам: Забезпечте надійний захист (шифрування, контроль доступу) та відповідні сертифікації (відповідність ISO27001, HIPAA, GDPR). Інтеграція часто обробляє конфіденційні дані.

– Рівень підтримки: Оцініть доступну підтримку. Керовані послуги включають експертів, які забезпечують роботу інтеграцій. Платформи самообслуговування вимагають потужних внутрішніх можливостей або партнерських мереж.

Перехід від інтеграцій на основі проектів до Integration Ops

Більшість традиційних підходів до хмарної інтеграції розглядають підключення як разові проекти. Команди збирають вимоги, створюють індивідуальні інтеграції, розгортають їх та рухаються далі. Коли API змінюються або нові інструменти потребують інтеграції, цикл повторюється.

Integration Ops трансформує цю модель, розглядаючи інтеграції як поточні операційні послуги. Замість розробки на основі проектів, Integration Ops встановлює проактивний рівень інтеграції в організаціях.

Як Integration Ops спрощує інтеграцію з хмарою

Платформи Integration Ops, такі як ONEiO, забезпечують швидше інтеграцію завдяки конфігурації, а не користувацькому коду. Що ще важливіше,

вони контролюють та підтримують інтеграції цілодобово за допомогою автоматизованих систем або постачальників послуг. Коли версії API змінюються або нові інструменти потребують інтеграції, це стає запитом на обслуговування, а не багатомісячним проектом.

Для управління хмарною інтеграцією Integration Ops усуває «вузьке місце інтеграції», яке затримує ініціативи цифрової трансформації. Корпоративним компаніям часто потрібно інтегрувати сервісні інструменти як внутрішньо, так і із зовнішніми постачальниками чи партнерами, що вимагає надійних можливостей роботи в режимі реального часу без навантаження на обслуговування.

Ключові переваги Integration Ops для хмарних інтеграцій включають:

– Швидше досягнення цінності: інтеграції розгортаються за лічені дні, а не місяці, що пришвидшує впровадження нових інструментів та адаптацію постачальників.

– Зменшення операційного навантаження: постачальники послуг або автоматизовані системи постійно обробляють оновлення, зіставлення та вирішення проблем у рамках підписок.

– Вища надійність: проактивний моніторинг та автоматизоване вирішення проблем зменшують час простою під час інтеграції, який може вплинути на угоди про рівень обслуговування (SLA).

– Більша адаптивність: Потреби в інтеграції розвиваються разом зі змінами в бізнесі. Integration Ops спрощує додавання або зміну підключень.

– Звільнені IT-потужності: команди зосереджуються на стратегічній роботі, а не на гасінні зламаних інтеграцій чи перебудові конекторів для нових інструментів.

Підсумок щодо найкращих платформ інтеграції хмари

Платформи хмарної інтеграції стали стратегічними можливостями для сучасних підприємств. Незалежно від того, чи ви обираєте комплексні рішення iPaaS, такі як SnapLogic або Celigo, пакети корпоративної інтеграції від Microsoft та SAP, чи керовані послуги Integration Ops, такі як ONEiO, правильний підхід

забезпечує підвищення ефективності, покращення гнучкості та покращення якості послуг.

Успішна інтеграція у 2026 році вимагає відповідності можливостей платформи потребам організації, рівням кваліфікації та операційним моделям. Оскільки складність інтеграції зростає з поширенням SaaS, підходи, що спрощують підключення, одночасно забезпечуючи надійність, стають дедалі ціннішими.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для розробки архітектурних модулів адаптивного проміжного програмного забезпечення (Middleware), забезпечення інтелектуальної взаємодії гетерогенних IT-систем та інтеграції з хмарними службами обрано мову програмування Python. Це високорівнева мультиплатформна мова загального призначення, яка завдяки своїй гнучкості та розвиненій екосистемі активно застосовується провідними світовими компаніями (Google, IBM, NASA, Dropbox та ін.) для побудови хмарних сервісів і масштабних інтеграційних платформ.

Python є інтерпретованою об'єктно-орієнтованою мовою з динамічною типізацією, що дозволяє передавати структури даних без попереднього жорсткого оголошення типів. Інтерпретований характер виконання забезпечує швидке виявлення синтаксичних і логічних помилок на етапі розробки, а об'єктна модель із підтримкою повної інтроспекції дає можливість динамічно аналізувати внутрішню структуру компонентів під час виконання програм. До ключових переваг мови належать:

– Логічний та лаконічний синтаксис: забезпечує легкість читання коду, що є критично важливим при розробці та підтримці складних логічних шлюзів Middleware.

– Масштабованість і гнучкість: дозволяє адаптувати високорівневу інтеграційну логіку та розширювати функціональність у міру зміни вимог

хмарних платформ.

– Висока швидкість розробки: значно скорочує цикл створення прототипів і готових програмних рішень порівняно з більшістю низькорівневих мов.

– Кросплатформність: код виконується на будь-яких операційних системах за допомогою універсального інтерпретатора.

До умовних недоліків мови відносять підвищене споживання обчислювальних ресурсів і меншу швидкість виконання чистих обчислень порівняно з компільованими мовами (наприклад, C++). Проте для розробки проміжного програмного забезпечення та хмарних інтеграційних шлюзів швидкість адаптації, зручність роботи з мережевими протоколами та гнучкість управління потоками даних мають пріоритетне значення.

Для реалізації завдань інтелектуальної інтеграції, обробки запитів та маршрутизації даних у проміжному програмному забезпеченні використовується низка бібліотек та інструментів Python:

– Requests та мережеві пакети: забезпечують ефективну взаємодію з хмарними API, RESTful-сервісами та мікросервісними архітектурами.

– Pandas та NumPy: застосовуються для швидкої обробки, трансформації та аналізу структурованих потоків даних, що надходять від різнорідних IT-систем.

– Фреймворки (Django, Pyramid, Web2py): надають зручне, стандартизоване середовище для розгортання масштабованих вебзастосунків, серверних компонентів Middleware та управління асинхронними запитами без необхідності ручної реалізації базових рутинних механізмів.

Для організації процесу розробки, відлагодження й тестування програмних модулів використовуються сучасні інтегровані середовища (зокрема PyCharm, Visual Studio Code), які забезпечують високу ефективність супроводу інтеграційних проєктів.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції IT-систем із хмарними службами.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3.1 Опис функціонування системи

Інтеграція хмарних даних: найкращі практики

1. Спочатку визначте бізнес-результати.

Успішні проекти інтеграції даних орієнтовані на певну ціль. Перш ніж вибирати інструменти або розробляти пайплайни, визначте чіткі бізнес-цілі:

– Ви вмикаєте звітність у режимі реального часу?

– Вам потрібно об'єднати профілі клієнтів у різних системах?

– Чи є дотримання нормативних вимог (GDPR, HIPAA) вашим основним рушієм?

Ця чіткість гарантує, що вибір технологій, пакетна чи потокова передача, ETL чи ELT, набори інструментів узгоджені з вимірюваними результатами. Наприклад, конвеєри виявлення шахрайства в режимі реального часу пріоритезуватимуть потокові сервіси з низькою затримкою, такі як Kafka, тоді як квартальна фінансова звітність може спиратися на пакетну ETL.

2. Віддайте перевагу ELT над ETL.

З появою сучасних хмарних сховищ даних, таких як Snowflake, BigQuery та Amazon Redshift, традиційний підхід «Вилучення-Перетворення-Завантаження» (ETL) еволюціонував до підходу «Вилучення-Перетворення-Перетворення» (ELT). Ось чому:

– Масштабованість, властива хмарним ресурсам: хмарні сховища масштабуються еластично, що робить їх ідеальними для трансформації даних на складі.

– Швидша доступність даних: необроблені дані стають доступними одразу після завантаження, що дозволяє гнучке дослідження даних.

– Зниження накладних витрат на інфраструктуру: Зменшує потребу в виділених ETL-серверах та мінімізує складність конвеєра.

Впровадження ELT покращує актуальність даних, прискорює терміни доставки та використовує хмарні оптимізації продуктивності.

3. Забезпечте якість даних у джерелі.

Низька якість даних збільшує витрати на подальшому етапі. Інтегруйте профілювання, валідацію та очищення даних на ранніх етапах конвеєра:

– Валідація схеми запобігає пошкодженню конвеєрів через погані зміни вгорі.

– Дедуплікація зменшує витрати на зберігання та уникає спотвореної аналітики.

– Стандартизація та збагачення, такі як узгоджені формати дат або нормалізація адрес, підвищують зручність використання в подальшому процесі.

Автоматизовані системи контролю якості виявляють проблеми на ранній стадії, зменшуючи кількість ручної переробки та зміцнюючи довіру до аналітики та звітності.

4. Автоматизація оркестрації даних.

Ручні тригери конвеєра даних призводять до неефективності, помилок та низької масштабованості. Автоматизація забезпечує надійність та зручність обслуговування:

– Використовуйте інструменти оркестрації, такі як Apache Airflow, AWS Step Functions або Azure Data Factory, для планування, моніторингу та керування робочими процесами.

– Впроваджуйте тригери, керовані подіями, для реагування в режимі реального часу, наприклад, ініціювання надходження даних, коли файли потрапляють у хмарне сховище.

– Застосовуйте логіку повторних спроб та сповіщення про невдачі для стійких, самовідновлювальних конвеєрів.

Це зменшує операційні накладні витрати та забезпечує плавну адаптацію конвеєрів до змінних обсягів даних та бізнес-вимог.

5. Створюйте з урахуванням масштабованості.

Обсяги даних та потреби в інтеграції неминуче зростають. Створюйте

гнучкість з самого початку:

– Безсерверні сервіси (наприклад, AWS Lambda, Google Cloud Functions) дозволяють автоматично масштабувати конвеєри без попередньо налаштованих серверів.

– Потокові сервіси, такі як Kafka, масштабуються горизонтально для обробки високопродуктивних конвеєрів реального часу.

– Хмарні групи автоматичного масштабування для обчислювальних ресурсів запобігають вузьким місцям під час пікових навантажень.

Це забезпечує майбутнє вашої архітектури, оптимізуючи витрати під час низького використання та забезпечуючи потужність під час пікового попиту.

6. Підтримка управління та відповідності вимогам.

Нормативні зобов'язання не підлягають обговоренню в підприємствах, що працюють на основі даних. Найкращі практики включають:

– Відстеження походження даних за допомогою каталогів метаданих (наприклад, AWS Glue Data Catalog, Azure Purview) для повного відстеження.

– Детальний контроль доступу з дозволами на основі ролей (наприклад, AWS IAM, GCP IAM).

– Шифрування як даних у стані спокою, так і даних під час передачі, за допомогою служб керування ключами (KMS) корпоративного рівня.

– Маскування даних та токенізація для захисту конфіденційних полів, таких як особиста інформація.

Це мінімізує ризик витоків великих даних, забезпечує готовність до аудиту та підтримує дотримання вимог кількох юрисдикцій під час передачі даних.

Інтеграція хмарних даних перетворилася на фундаментальну можливість для сучасних організацій. Об'єднуючи фрагментовані джерела даних, компанії можуть:

– Пришвидшуйте прийняття рішень завдяки аналітиці в режимі реального часу.

– Забезпечення ініціатив зі штучного інтелекту/машинного навчання за

допомогою консолідованих наборів даних.

– Підвищте гнучкість за допомогою самообслуговування та автоматизації.

– Забезпечення відповідності нормативним вимогам за допомогою вбудованих функцій управління.

У 2026 році команди обробки даних з перспективним мисленням використовуватимуть хмарну інтеграцію даних не лише для переміщення даних, але й як стратегічний важіль для інновацій, масштабованості та конкурентної переваги.

3.2 Розробка структурної схеми

Перехід до підключених цифрових екосистем

Програмні системи більше не функціонують як ізольовані інструменти. Сучасні організації залежать від кількох цифрових платформ для управління своїми операціями, клієнтами, фінансами, аналітикою та комунікаціями. Протягом останнього десятиліття перехід до хмарного програмного забезпечення змінив те, як працюють підприємства. Замість того, щоб підтримувати великі локальні системи, компанії тепер покладаються на спеціалізовані SaaS-платформи, які задовольняють різні операційні потреби.

Майбутнє інтеграційної екосистеми 2026 року

Підключена екосистема інтеграції, де системи взаємодіють у режимі реального часу, забезпечуючи безперебійний потік даних, автоматизацію та розумніші бізнес-операції.

Чому інтеграція зараз є пріоритетом бізнесу

Цей зсув створив новий виклик. Хоча кожна система ефективно виконує свою роль, справжня цінність проявляється, коли ці системи працюють разом. Дані повинні переміщуватися між платформами, щоб узгоджувати операції. Сервісним командам потрібна інформація про клієнтів з платформ CRM. Фінансовим системам потрібні дані про виконання робіт з платформ

обслуговування. Системи управління запасами повинні оновлювати інформацію про використання запасних частин у режимі реального часу. Тому інтеграція стала одним з найважливіших компонентів сучасної цифрової інфраструктури.

Оскільки організації продовжують впроваджувати хмарні технології, роль інтеграції лише зростатиме. До 2027 року стратегії інтеграції вийдуть за рамки простих системних з'єднань і перейдуть до більш інтелектуальних та автоматизованих цифрових екосистем. Бізнес більше не розглядатиме окремі програмні платформи як окремі інструменти. Натомість вони створюватимуть пов'язані середовища, де системи безперервно взаємодіють та підтримують операційні робочі процеси в усій організації.

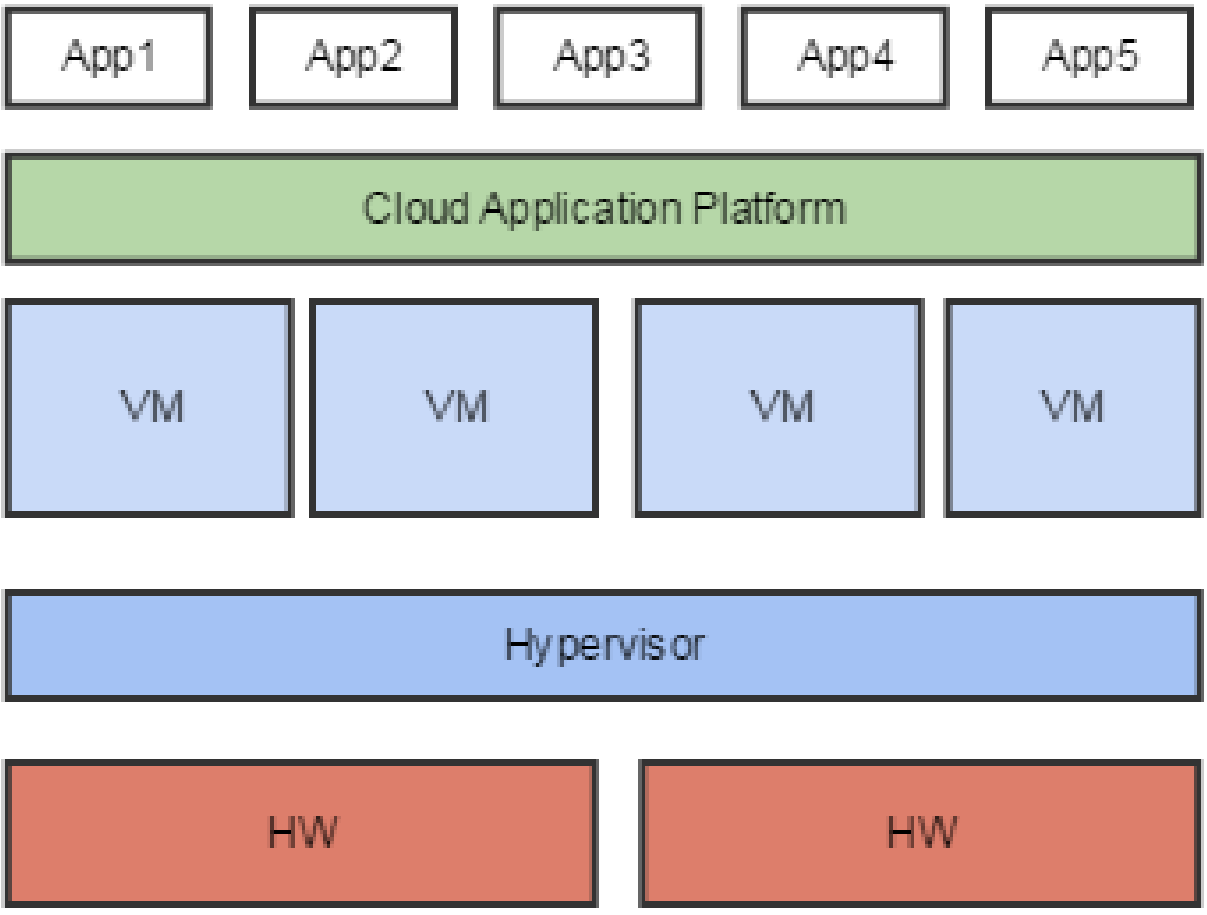

Рисунок 3.1 – Структурна схема системи

Майбутнє інтеграції формується кількома важливими тенденціями. Ці тенденції зосереджені на підвищенні гнучкості, посиленні безпеки, забезпеченні обміну даними в режимі реального часу та спрощенні способів підключення організацій до своїх цифрових систем. Для сервісних компаній, які керують складними операціями з розподіленими командами, ці розробки матимуть прямий вплив на те, як планується, виконується та контролюється робота:

1. Зростання API-орієнтованої архітектури. Однією з найбільш помітних тенденцій в інтеграції є зростаюче впровадження архітектури API first. У попередніх програмних середовищах інтеграції часто створювалися як вторинні функції після розробки основного застосунку. Такий підхід ускладнював підтримку інтеграцій та обмежував способи взаємодії систем одна з одною. В архітектурі API first можливості інтеграції з самого початку проектуються як основна частина системи. Коли платформа побудована на основі API, її легше підключати до інших систем. До кожної функції платформи можна отримати доступ через чітко визначені кінцеві точки API, що дозволяє зовнішнім програмам безпечно взаємодіяти із системою. Такий підхід забезпечує гнучкість для організацій, які покладаються на кілька програмних платформ. Замість того, щоб створювати власні інтеграції щоразу, коли впроваджується нова система, компанії можуть покладатися на стандартизовані API, які спрощують процес інтеграції. Платформи, що підтримують архітектуру на основі API, дозволяють організаціям створювати пов’язані екосистеми, де дані вільно обмінюються між системами. Наприклад, платформа польового обслуговування, така як Wello, може підключатися до бухгалтерського програмного забезпечення, систем управління клієнтами, аналітичних інструментів та платформ управління запасами через структуровані API. Такий тип інтеграції гарантує, що операційні дані залишаються синхронізованими між відділами, зменшуючи ручну роботу та підвищуючи точність.

2. Обмін даними в режимі реального часу: від пакетної обробки до миттєвої. Ще однією важливою тенденцією, яка формує інтеграцію у 2026 році, є зростаючий попит на обмін даними в режимі реального часу. У традиційних

моделях інтеграції дані часто передавались запланованими партіями. Системи експортували інформацію через фіксовані проміжки часу та періодично імпортували оновлення. Хоча цей підхід працював для базової звітності, він не підходив для операційних середовищ, де рішення залежать від негайної інформації. Інтеграція в режимі реального часу дозволяє системам миттєво обмінюватися даними в міру виникнення подій. Коли технік завершує сервісне завдання, платформа польового обслуговування негайно оновлює запис клієнта, запускає створення рахунку-фактури та реєструє історію обслуговування обладнання. Диспетчери можуть бачити хід виконання завдання, не чекаючи ручних оновлень. Фінансові команди можуть обробляти рахунки швидше, оскільки необхідна інформація стає доступною миттєво. Інтеграція в режимі реального часу покращує операційну видимість в організації. Менеджери можуть контролювати ефективність послуг, відстежувати продуктивність технічних спеціалістів та виявляти операційні вузькі місця в міру їх виникнення. Такий рівень оперативності особливо цінний для сервісних компаній, які керують великими обсягами щоденних робочих замовлень або працюють у кількох регіонах.

3. Автоматизація в робочих процесах інтеграції. Автоматизація також відіграватиме більшу роль у стратегіях інтеграції в найближчі роки. Оскільки організації підключають більше систем, ручне керування цими підключеннями стає дедалі складнішим. Автоматизація дозволяє робочим процесам інтеграції працювати незалежно після їх налаштування. Передача даних, тригери процесів та оновлення системи відбуваються автоматично без необхідності постійного нагляду. Наприклад, сервісна організація може налаштувати автоматизований робочий процес, де нові облікові записи клієнтів, створені на платформі CRM, автоматично відображаються в системі польового обслуговування. Реєстрація обладнання може ініціювати графіки профілактичного обслуговування. Виконання робочого замовлення може генерувати рахунок-фактуру у фінансовій системі. Ці автоматизовані процеси зменшують адміністративне навантаження,

водночас забезпечуючи узгодженість операційної інформації на всіх платформах.

4. Архітектура, керована подіями: системи, що реагують. Ще однією тенденцією, що з'являється в інтеграційному ландшафті, є зростання архітектур, керованих подіями. У системі, керованій подіями, програмні платформи реагують на події, а не покладаються виключно на прямі запити даних. Коли відбувається певна подія, така як створення нового робочого замовлення або завершення завдання обслуговування, система автоматично повідомляє інші підключені платформи. Ці платформи потім вживають відповідних дій на основі події. Інтеграція на основі подій покращує швидкість реагування системи та зменшує потребу в постійному опитуванні між системами. Замість того, щоб постійно запитувати про наявність нових даних, системи просто реагують, коли відбуваються події. Такий підхід покращує продуктивність та зменшує непотрібний мережевий трафік. Він також дозволяє організаціям розробляти робочі процеси, які динамічно реагують на операційні зміни.

5. Зміцнення безпеки інтеграції у 2026 році. Безпека залишатиметься центральною проблемою в міру розширення інтеграційних мереж. Кожна нова інтеграція створює ще один шлях, через який дані переміщуються між системами. Організації повинні забезпечити захист цих шляхів від несанкціонованого доступу. У 2026 році безпека інтеграції буде зосереджена переважно на надійній автентифікації, зашифрованому зв'язку та детальному моніторингу взаємодії систем. Методи автентифікації на основі токенів продовжуватимуть замінювати статичні облікові дані як кращий метод для забезпечення безпеки інтеграцій. Ці токени забезпечують обмежений у часі доступ, що зменшує ризик неправомірного використання облікових даних. Протоколи шифрування захищатимуть дані під час передачі, гарантуючи, що конфіденційна інформація залишиться нечитабельною у разі перехоплення під час передачі. Детальні системи реєстрації та моніторингу дозволять організаціям відстежувати кожну взаємодію між підключеними платформами.

6. Зростаюча важливість управління даними. Управління даними також стане важливим елементом стратегій інтеграції. Оскільки системи обмінюються більшою кількістю інформації, організації повинні контролювати , як ці дані використовуються, зберігаються та поширюються. Чіткі політики управління допомагають визначити, які системи можуть отримувати доступ до певних типів даних і за яких умов. Це гарантує, що інтеграції підтримують операційну ефективність, зберігаючи при цьому дотримання правил захисту даних.

7. Інтеграційні платформи як централізовані центри. Ще однією тенденцією, що розвивається, є використання інтеграційних платформ, які спрощують процес підключення організацій до своїх систем. Ці платформи діють як центральні вузли, де можна налаштовувати, контролювати та керувати інтеграціями. Замість того, щоб створювати окремі інтеграції для кожної пари систем, організації можуть керувати з'єднаннями через єдине середовище інтеграції. Такий підхід зменшує складність і дозволяє командам підтримувати прозорість своєї цифрової інфраструктури. Для сервісних організацій інтеграційні платформи допомагають оптимізувати операції, забезпечуючи безперебійний переміщення інформації між системами. Дані клієнтів, записи про обладнання, звіти про обслуговування та фінансові транзакції можуть синхронізуватися автоматично. Коли робочі процеси інтеграції правильно розроблені, команди витрачають менше часу на керування даними та більше часу зосереджуються на наданні послуг.

8. Інтеграція Інтернету речей та прогнозного обслуговування. Розширення підключених пристроїв також вплине на майбутнє інтеграції. Багато галузей промисловості починають використовувати системи моніторингу обладнання, які збирають дані з машин, що працюють у польових умовах. Датчики можуть відстежувати такі показники продуктивності, як температура, тиск, вібрація та робочі години. Ці системи моніторингу генерують великі обсяги даних, які необхідно інтегрувати з сервісними платформами, щоб отримати змістовну аналітику. Завдяки інтеграції із застосунками для польового

обслуговування, системи моніторингу обладнання можуть автоматично запускати завдання технічного обслуговування. Якщо машина демонструє ознаки потенційної несправності, платформа моніторингу може надіслати сповіщення, яке створить сервісний запит у системі польового обслуговування. Диспетчери можуть призначити техніків для перевірки обладнання до того, як станеться поломка. Цей тип прогнозного технічного обслуговування підвищує надійність обладнання, одночасно зменшуючи незапланований час простою.

9. Як забезпечується безперебійна інтеграція. Такі платформи здатні підтримувати ці тенденції інтеграції, забезпечуючи структуровану комунікацію між сервісними операціями та зовнішніми системами. Завдяки безпечним API та механізмам контрольованого обміну даними, платформа дозволяє організаціям пов’язувати свої робочі процеси з фінансовими системами, клієнтськими платформами, інструментами аналітики та рішеннями для моніторингу обладнання. Ця можливість гарантує, що сервісні операції залишаються узгодженими з ширшим цифровим середовищем бізнесу.

10. Створення підключених систем, готових до майбутнього. Майбутнє інтеграції зрештою буде зосереджено на створенні пов’язаних цифрових екосистем, а не ізольованих додатків. Організації проектуватимуть свої технологічні середовища таким чином, щоб системи безперервно співпрацювали. Інформація автоматично переміщуватиметься між платформами, дозволяючи командам працювати з аналітикою в режимі реального часу та скоординованими робочими процесами. До 2027 року компанії, які оберуть інтеграцію як основну стратегію, отримають значні операційні переваги. Їхні системи будуть ефективно взаємодіяти, їхні дані залишатимуться узгодженими між відділами, а їхні команди зможуть швидко реагувати на зміну операційних умов. Компанії, які продовжують працювати з розрізненими системами, зіткнуться зі зростаючими викликами, оскільки цифрова інфраструктура стає все складнішою.

11. Інтеграція як стратегічна бізнес-перевага. Інтеграція вже не є лише технічним питанням для IT-відділів. Вона стала стратегічним компонентом

бізнес-операцій. Оскільки програмні платформи продовжують розвиватися, здатність безпечно та ефективно підключати системи визначатиме, наскільки ефективно організації зможуть масштабувати свої операції, надавати послуги та підтримувати довіру клієнтів. Компанії, які готуються до майбутнього інтеграції сьогодні, створять цифрові середовища, здатні підтримувати довгострокове зростання. Інвестуючи в безпечні API, обмін даними в режимі реального часу, автоматизовані робочі процеси та гнучкі архітектури інтеграції, організації можуть створювати технологічні екосистеми, що підтримують як операційну ефективність, так і постійні інновації.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи наведена на рисунку 3.2. Процес розв'язання задачі забезпечення адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції гетерогенних IT-систем із хмарними службами охоплює чотири послідовні етапи:

1. Визначення загальної множини маршрутів передачі інформації.

2. Знаходження оптимальної підмножини маршрутів для передачі цифрових даних у телекомунікаційній мережі.

3. Обчислення коефіцієнтів розподілу інформаційного потоку та динамічного управління навантаженням комп'ютерної мережі.

4. Створення та регулярне оновлення таблиць маршрутизації.

5. Запобігання зацикленню та ієрархічна маршрутизація. Під час пошуку найкоротших шляхів передачі даних у мережі традиційні алгоритми часто стикаються з проблемою «зациклення», що призводить до зростання затримок та втрати пакетів. Для усунення цього ефекту запропоновано використовувати формальні обмеження (умову постійної відсутності циклів):

$$d(k, j) < d(i, j),$$

де $d(k, j)$ – мінімальний час передачі пакетів від проміжного вузла «k» до адресата «j», а $d(i, j)$ – відстань від джерела «i» до адресата через вузол «k».

Процес пошуку маршрутів без циклів реалізується у два етапи: розрахунок попередньої топології на основі ієрархічної структури мережі та розповсюдження попередньої топології за допомогою спрощеного механізму синхронізації між суміжними вузлами, що суттєво зменшує час збіжності алгоритмів. За відсутності флуктуацій трафіку розроблений підхід демонструє час затримки, зіставний із наявними аналогами, проте в рази перевершує їх в умовах різких коливань вхідного трафіку.

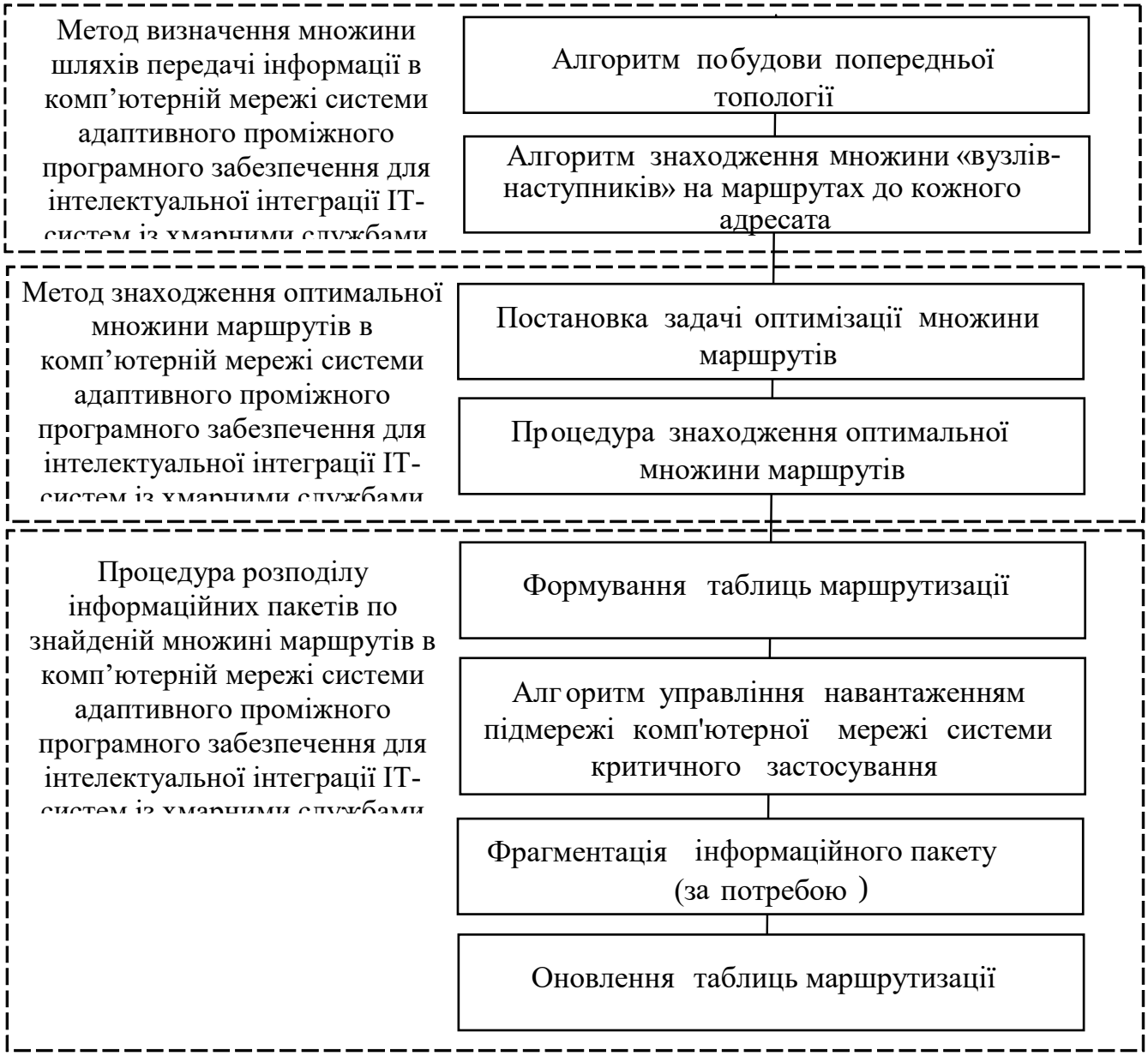

Рисунок 3.2 – Функціональна схема системи

6. Оптимізація та розподіл навантаження. Збільшення кількості використовуваних маршрутів (M) призводить до зростання ймовірності спотворення інформації та ускладнення таблиць маршрутизації. Тому виникає необхідність вибору оптимальної сукупності маршрутів, яка забезпечує максимально можливу достовірність доставки даних. Для визначення початкового завантаження підмережі розраховуються коефіцієнти розподілу потоку інформації між вузлами-наступниками, де враховується час передачі пакетів від вузла-наступника до адресата. Такий адаптивний розподіл дозволяє оптимізувати пропускну здатність, мінімізувати ймовірність спотворення даних та збалансувати навантаження під час флуктуацій трафіку.

7. Оцінка ефективності та результати моделювання. Для практичного застосування розроблено структуру процесу реального часу, яка включає адаптивне кодування, зниження надмірності та управління швидкістю передачі. Застосування алгоритмів компенсації втрачених пакетів дозволяє використовувати протоколи транспортного рівня (RTP та UDP).

Оцінка ефективності показала значні переваги розробленого методу порівняно з традиційними:

– ймовірність доведення інформації до одержувача у допустимий час вища до 3 разів;

– середній час доставки пакетів менший до 15 разів.

Достовірність результатів підтверджена імітаційним моделюванням у повнозв'язній мережі з бездротовими каналами зв'язку. За критерієм згоди Персона доведено нормальний закон розподілу прийнятих пакетів, а отримані довірчі інтервали підтверджують високу точність математичної моделі.

Після розгляду всіх блоків функціональної схеми перейдемо до аналізу діаграми взаємодії процесів, які відбуваються у розробленій системі.

3.4 Розробка діаграми процесів

Діаграма процесів (Data Flow Diagram, DFD) розробленої системи наведена на рисунку 3.3. Її детальний розгляд дозволяє простежити логіку інформаційного обміну, маршрутизації та виконання функціональних завдань у межах адаптивного проміжного програмного забезпечення (Middleware) для інтеграції IT-систем із хмарними службами.

У межах структурного проєктування діаграма взаємодії процесів використовується для візуалізації процедур обробки та передачі даних. Згідно з інженерною методологією, розробка розпочинається з побудови контекстної діаграми (діаграми нульового рівня), яка визначає загальні межі системи та її інтерфейси із зовнішнім середовищем. Надалі ця модель підлягає ітераційній декомпозиції шляхом деталізації базових процесів, сховищ та потоків з метою повної візуалізації архітектури розроблюваного комплексу.



Рисунок 3.3 – Діаграма взаємодії процесів

Відповідно до методології структурного аналізу, побудовані діаграми потоків даних містять чотири базові типи елементів:

– Процеси: функціональні перетворення і трансформації даних у межах описуваної інтеграційної системи.

– Сховища даних (репозиторії): логічні сховища (бази даних, кеш-пам'ять, таблиці маршрутизації), де зберігається інформація між сеансами обробки.

– Зовнішні сутності: джерела або приймачі інформації, що знаходяться поза межами розроблюваної системи (зовнішні IT-системи, хмарні служби, користувачі).

– Потоки даних: канали пересилання інформації між зазначеними елементами системи.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Функціонування системи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції IT-систем із хмарними службами ґрунтується на використанні багаторівневої архітектури оброблення запитів, яка забезпечує взаємодію програмних компонентів із різними інтерфейсами, форматами даних і механізмами автентифікації.

Основним призначенням алгоритмічного забезпечення є автоматизований вибір способу інтеграції, перетворення інформаційних повідомлень, керування міжсервісними запитами та забезпечення узгодженості операцій у гетерогенному обчислювальному середовищі.

Для виконання цих завдань проміжне програмне забезпечення повинно аналізувати характеристики вхідного запиту, визначати доступні хмарні служби та обирати допустимий маршрут оброблення. Загальну логіку функціонування програмного комплексу представлено на рисунку 4.1.

Інтеграція корпоративних IT-систем із хмарною інфраструктурою ускладнюється відмінностями між програмними інтерфейсами постачальників послуг.

Окремі служби використовують REST API, інші підтримують gRPC або асинхронний обмін через черги повідомлень. Відрізняються також структури запитів, формати результатів, правила обмеження інтенсивності звернень та способи повідомлення про помилки.

Без проміжного програмного рівня кожна корпоративна система повинна реалізовувати власні механізми взаємодії з конкретним хмарним сервісом.

Це створює залежність прикладного програмного забезпечення від зовнішніх інтерфейсів і ускладнює зміну постачальника послуг.

Запропоноване проєктне рішення передбачає використання уніфікованого інтеграційного інтерфейсу, через який внутрішні застосунки передають функціональні запити до відповідних хмарних служб. Middleware приймає повідомлення, перевіряє його коректність, встановлює вимоги до оброблення та визначає сумісні програмні адаптери.

Після вибору маршруту система перетворює дані до формату цільового сервісу, виконує необхідні мережеві операції та нормалізує отриманий результат. Завдяки такій організації прикладні компоненти взаємодіють із логічними функціями інтеграції, а не з окремими низькорівневими інтерфейсами постачальників.

Адаптивність програмного комплексу забезпечується механізмом контекстно залежного вибору способу оброблення. Під час прийняття рішення враховуються тип операції, вимоги до конфіденційності, допустима затримка, доступність хмарної служби та її поточні експлуатаційні характеристики. Якщо основний сервіс недоступний або не відповідає встановленим обмеженням, middleware визначає можливість використання альтернативного маршруту. При цьому заміна постачальника допускається лише за наявності підтвердженої функціональної та семантичної сумісності.

Послідовність роботи підпрограми адаптивної маршрутизації інтеграційних запитів наведено на рисунку 4.2. Важливою особливістю системи є відокремлення алгоритму маршрутизації від механізмів фактичної взаємодії із зовнішніми службами. Модуль маршрутизації визначає допустимий маршрут на основі формалізованих характеристик запиту й доступних адаптерів. Адаптер виконує перетворення формату, формування службових заголовків, автентифікацію та передавання повідомлення. Модуль контролю виконання відстежує результат, реєструє затримку та визначає можливість повторної спроби. Такий розподіл дозволяє змінювати логіку вибору маршруту незалежно від програмної реалізації конкретних хмарних інтеграцій.

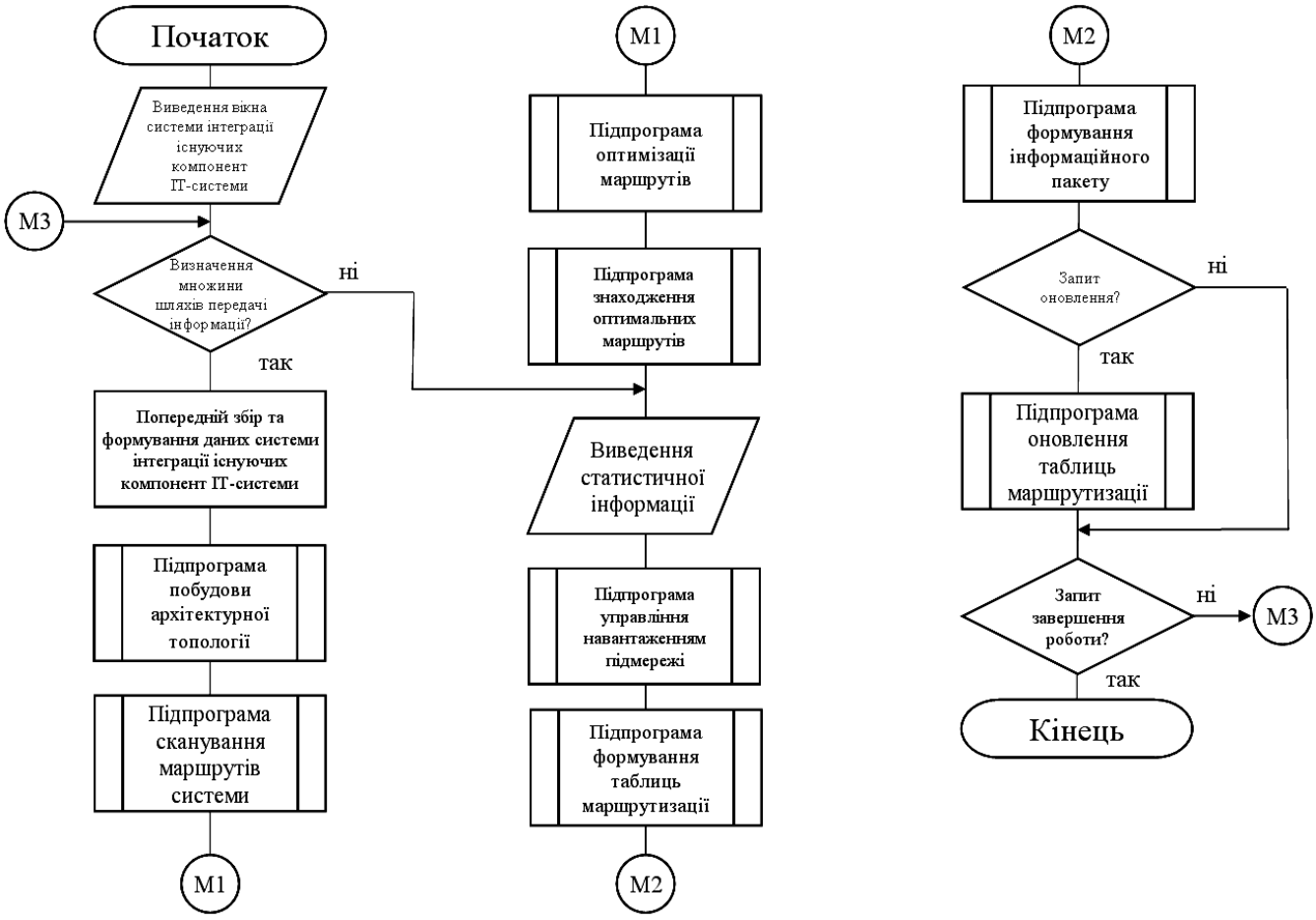

Рисунок 4.1 – Блок-схема основної програми

Для програмної реалізації розглядається використання Python із серверним середовищем FastAPI та бібліотекою асинхронної HTTP-взаємодії HTTPX. Зберігання конфігурації, метаданих маршрутів і результатів виконання може забезпечуватися PostgreSQL.

Для асинхронних завдань доцільно передбачити брокер повідомлень, наприклад RabbitMQ. Контроль експлуатаційних характеристик може здійснюватися засобами OpenTelemetry.

Вказані компоненти характеризують можливий технологічний варіант реалізації, а не підтверджений факт їхнього використання в готовому програмному продукті.

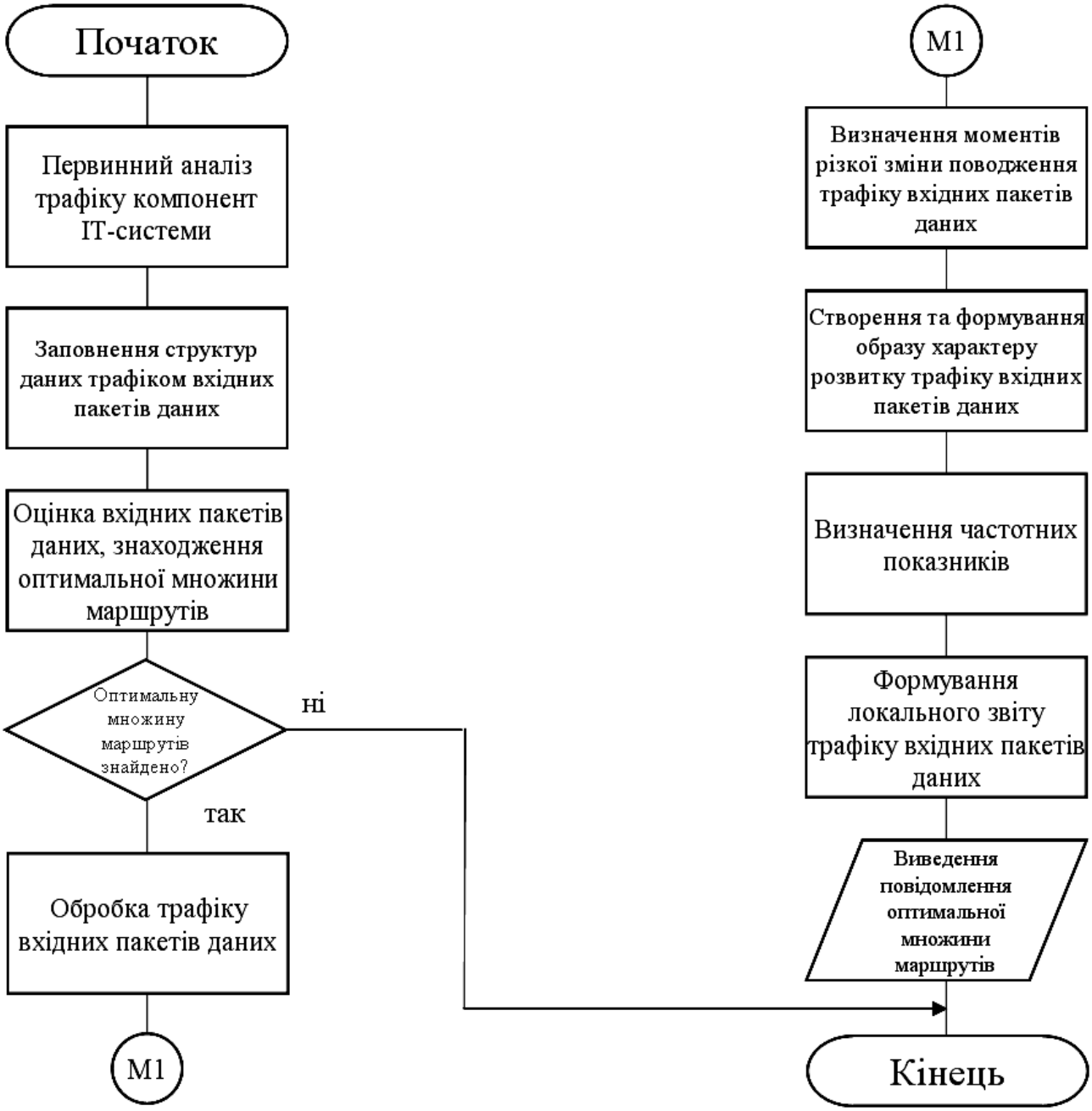

Рисунок 4.2 – Блок-схема роботи підпрограми

Побудова інтеграційного середовища на основі незалежних сервісних адаптерів

На архітектурному рівні проміжне програмне забезпечення доцільно представити як сукупність взаємодіючих функціональних компонентів. Вхідний шлюз відповідає за приймання запитів від корпоративних інформаційних систем. Модуль перевірки контрактів встановлює правильність структури повідомлень.

Каталог інтеграційних можливостей містить відомості про підтримувані хмарні операції. Компонент адаптивного вибору визначає маршрут виконання. Адаптери забезпечують безпосередній обмін із зовнішніми службами. Результати оброблення надходять до модуля нормалізації та повертаються ініціатору.

Для підключення нового хмарного сервісу необхідно визначити набір операцій, які він підтримує. До них можуть належати збереження файлів, отримання об'єктів, надсилання повідомлень, запуск обчислювальних завдань і виконання запитів до керованих баз даних. Кожна операція повинна мати формалізований опис входу, виходу та можливих помилок. Додатково необхідно визначити вимоги до автентифікації, максимальний розмір повідомлення й допустиму частоту звернень.

Уніфікований інтерфейс повинен описувати функціональну операцію незалежно від конкретного постачальника. Наприклад, операція object.store означає необхідність збереження інформаційного об'єкта. Її реалізація може відрізнятися залежно від хмарного сховища, але логічний результат повинен містити ідентифікатор об'єкта, статус завершення та доступні метадані. Для іншої операції, зокрема message.publish, визначається власний контракт, що містить інформацію про повідомлення та логічний канал його передавання.

Під час побудови каталогу необхідно відокремити функціональну сумісність від технологічної. Два хмарні сервіси можуть підтримувати HTTP API, але виконувати різні за змістом операції. Навіть однакові назви функцій не гарантують тотожної семантики. Наприклад, підтвердження приймання повідомлення брокером не завжди означає його доставлення кінцевому споживачу. Тому каталог повинен містити не лише тип транспортного протоколу, а й характеристики гарантій виконання та способу підтвердження результату.

Для кожного адаптера доцільно визначати стани AVAILABLE, DEGRADED, UNAVAILABLE та DISABLED. Стан AVAILABLE означає можливість приймання нових запитів відповідно до конфігурації. DEGRADED характеризує часткове порушення експлуатаційних показників. UNAVAILABLE використовується за підтвердженої недоступності сервісу. DISABLED відповідає

адміністративному вимкненню адаптера. Окремо може зберігатися стан механізму захисту від каскадних відмов, який визначає можливість виконання нових звернень.

Інформація про можливості адаптера має бути пов'язана з версією зовнішнього програмного інтерфейсу. Хмарні постачальники можуть змінювати формати відповідей, додавати обов'язкові поля або припиняти підтримку окремих операцій. Для запобігання порушенням необхідно зберігати версію контракту та перевіряти її сумісність із поточною реалізацією. Оновлення адаптера доцільно виконувати незалежно від основного компонента маршрутизації.

Для складних інтеграційних процесів одного синхронного HTTP-запиту може бути недостатньо. Наприклад, передавання великого інформаційного об'єкта до хмарного сховища може потребувати декількох послідовних операцій. Виконання обчислювального завдання може повертати ідентифікатор процесу, результат якого стає доступним пізніше. Тому програмний контракт повинен розрізняти завершену операцію, прийняте до виконання завдання та помилку оброблення.

Взаємодію між компонентами middleware доцільно організувати через внутрішні програмні інтерфейси. Це дозволяє окремо перевіряти правила маршрутизації, коректність перетворення даних та роботу конкретного адаптера. У випадку розподіленого розгортання компоненти можуть обмінюватися повідомленнями через захищені API або брокер. Якщо всі функції розміщені в одному процесі, доцільно зберегти логічний поділ на модулі без обов'язкового створення додаткових мережевих сервісів.

Для синхронних команд доцільно використовувати окремий контракт, оскільки подія та команда мають різну семантику. Подія описує факт, який уже відбувся. Команда містить вимогу виконати певну дію. Змішування цих понять ускладнює оброблення повторних повідомлень та контроль завершення операцій. Тому middleware повинен зберігати явну ознаку типу взаємодії.

Програмну реалізацію перевірки вхідного контракту можна виконати через структури даних Python. Наведений приклад демонструє створення уніфікованого

запиту, перевірку базових полів і нормалізацію часової позначки. Він не виконує безпосереднього звернення до хмарної служби.

```
from dataclasses import dataclass
from datetime import datetime, timezone
from math import isfinite
from uuid import UUID

@dataclass(frozen=True)
class IntegrationRequest:
    request_id: str
    tenant_id: str
    operation: str
    deadline: datetime
    sensitivity: str
    payload: dict
    idempotency_key: str | None

SUPPORTED_OPERATIONS = {
    "object.store",
    "object.read",
    "message.publish",
    "task.submit"
}

SENSITIVITY_LEVELS = {
    "PUBLIC", "INTERNAL", "CONFIDENTIAL"
}

def normalize_request(data: dict) -> IntegrationRequest:
    # Перевірка структури вхідних даних
    if not isinstance(data, dict):
        raise TypeError("Очікується об'єкт запиту")

    request_id = str(UUID(str(data["request_id"])))

    tenant_id = data.get("tenant_id")
    if not isinstance(tenant_id, str) or not tenant_id:
        raise ValueError("Не визначено організаційний сегмент")

    operation = data.get("operation")
    if operation not in SUPPORTED_OPERATIONS:
        raise ValueError("Непідтримувана операція")

    sensitivity = data.get("sensitivity")
    if sensitivity not in SENSITIVITY_LEVELS:
        raise ValueError("Некоректний клас конфіденційності")
    payload = data.get("payload")
    if not isinstance(payload, dict):
        raise ValueError("Некоректне тіло повідомлення")

    # Часовий параметр обов'язково містить зону
    raw_deadline = data.get("deadline")
    if not isinstance(raw_deadline, str):
        raise ValueError("Не визначено граничний час")
    deadline = datetime.fromisoformat(
        raw_deadline.replace("Z", "+00:00")
    )

    if deadline.tzinfo is None:
        raise ValueError("Не визначено часову зону")
```

```
deadline = deadline.astimezone(timezone.utc)

if deadline <= datetime.now(timezone.utc):
    raise ValueError("Термін виконання вже минув")

# Ключ ідемпотентності використовується
# для підтримуваних операцій зі зміною стану
key = data.get("idempotency_key")
if key is not None:
    if not isinstance(key, str) or not key.strip():
        raise ValueError("Некоректний ключ ідемпотентності")
return IntegrationRequest(
    request_id=request_id,
    tenant_id=tenant_id,
    operation=operation,
    deadline=deadline,
    sensitivity=sensitivity,
    payload=payload,
    idempotency_key=key
)
```

Наведена функція забезпечує початкову структурну перевірку. Для практичного застосування необхідно доповнити її контролем максимального розміру повідомлення, перевіркою прав організаційного сегмента та окремими схемами корисного навантаження для кожної операції. Значення tenant_id повинно звірятися з автентифікованим контекстом користувача. Його не можна безумовно приймати як достовірне лише на підставі вхідного JSON.

Під час перетворення даних необхідно забезпечити збереження їхнього значення. Наприклад, відмінність між відсутнім полем і полем зі значенням null може бути важливою для зовнішнього API. Аналогічно перетворення грошових величин до типу з плаваючою комою може створювати похибки. Для таких значень доцільно використовувати десяткове представлення з визначеною точністю. Усі перетворення одиниць вимірювання повинні бути явними та перевірюваними.

Окремим завданням є нормалізація помилок. Хмарні сервіси можуть повертати різні коди та структури діагностичних повідомлень. Внутрішній контракт доцільно доповнити уніфікованими категоріями помилок, які охоплюють некоректні дані, відсутність повноважень, перевищення квоти, тимчасову недоступність, перевищення часу очікування та невизначений результат виконання. При цьому вихідний код провайдера необхідно зберігати окремо для діагностики.

Математична модель адаптивного вибору хмарного маршруту

Після нормалізації інтеграційного запиту система повинна визначити найбільш придатний спосіб його виконання. Це завдання розглядається як багатокритеріальний вибір серед множини доступних сервісних адаптерів. Для кожного кандидата оцінюються відповідність функціональному контракту, затримка виконання, надійність, орієнтовна вартість і допустимість передавання відповідних даних. Рішення приймається лише серед маршрутів, які задовольняють обов'язкові обмеження.

Нехай $R = \{r_1, r_2, \dots, r_m\}$ є множиною зареєстрованих маршрутів інтеграції. Для кожного маршруту визначаються функціональні можливості F_j, регіон розміщення G_j, прогнозована затримка L_j, вартість операції C_j та частка помилок E_j. Запит q містить необхідну операцію, клас конфіденційності, допустимі регіони, часові обмеження та параметри корисного навантаження.

На першому етапі формується множина допустимих маршрутів

$$R_{\text{valid}}(q) = \{r_j \in R \mid F(q) \subseteq F_j,\ G_j \in G(q), P(q, r_j) = 1\},$$

де $F(q)$ визначає необхідні функції, $G(q)$ є множиною дозволених регіонів, а $P(q, r_j)$ характеризує виконання політик безпеки та сумісності. Якщо множина виявляється порожньою, система повинна повернути результат про неможливість допустимого маршруту. Не можна автоматично послаблювати вимоги до конфіденційності або територіального розміщення даних заради виконання запиту.

Прогнозована затримка може визначатися за історичними результатами виконання запитів. Для початкового варіанта доцільно використовувати експоненціальне ковзне середнє

$$\widehat{L_t} = \alpha L_t + (1 - \alpha)\widehat{L_{(t-1)}},$$

де L_t є останнім виміряним часом оброблення, а $\alpha \in (0,1]$ визначає швидкість адаптації до нових результатів. Мале значення α забезпечує повільніше реагування на короткочасні коливання. Більше значення дозволяє швидше враховувати зміни стану хмарної служби. Для контролю ризику порушення

часових обмежень доцільно додатково використовувати високі процентилі затримки.

Важливо враховувати, що середнє значення не характеризує всі режими виконання. Сервіс може забезпечувати невелику типову затримку, але періодично демонструвати значні відхилення. Для критичних запитів доцільно застосовувати оцінку 95-го процентиля. Якщо кількість вимірювань недостатня, прогноз повинен супроводжуватися ознакою низької достовірності. У такому випадку можна використовувати консервативну початкову оцінку або обмежену пробну експлуатацію сервісу.

Вартість операції повинна визначатися відповідно до тарифної моделі конкретного хмарного постачальника.

Для деяких служб вона залежить від кількості запитів, для інших від тривалості обчислень, обсягу збережених даних або міжрегіонального передавання. Тому загальна функція оцінювання повинна використовувати вартість конкретного типу операції, а не абстрактну ціну сервісу. Якщо вартість невідома, система повинна застосовувати визначену політику оброблення невизначеності.

Для програмної реалізації вибору маршруту можна використати структури Python, які зберігають профілі сумісних адаптерів і поточні експлуатаційні оцінки.

Наступний фрагмент демонструє відбір кандидатів за обов'язковими вимогами та їхнє подальше ранжування.

Значення ваг і порогів наведено як параметри моделі, які потребують експериментального налаштування.

```
from dataclasses import dataclass
from math import isfinite
@dataclass(frozen=True)
class CloudRoute:
    route_id: str
    operations: frozenset[str]
    region: str
    allowed_sensitivity: frozenset[str]
    p95_latency_ms: float
    estimated_cost: float
    error_rate: float
    uncertainty: float
    available: bool

@dataclass(frozen=True)
```

```
class RoutingContext:
    operation: str
    sensitivity: str
    allowed_regions: frozenset[str]
    max_latency_ms: float
    max_cost: float

def select_cloud_route(
    routes: list[CloudRoute],
    context: RoutingContext
) -> CloudRoute | None:
    # Не допускаються некоректні обмеження запиту
    if context.max_latency_ms <= 0 or context.max_cost <= 0:
        raise ValueError("Некоректні часові або цінові межі")
    candidates = []
    for route in routes:
        # Спочатку перевіряються жорсткі вимоги
        if not route.available:
            continue
        if context.operation not in route.operations:
            continue
        if context.region not in context.allowed_regions:
            continue
        if context.sensitivity not in route.allowed_sensitivity:
            continue
        metrics = (
            route.p95_latency_ms,
            route.estimated_cost,
            route.error_rate,
            route.uncertainty
        )
        if not all(isfinite(value) for value in metrics):
            continue
        if not (
            route.p95_latency_ms >= 0
            and route.estimated_cost >= 0
            and 0 <= route.error_rate <= 1
            and 0 <= route.uncertainty <= 1
        ):
            continue
        if route.p95_latency_ms > context.max_latency_ms:
            continue

        if route.estimated_cost > context.max_cost:
            continue
       # Нормалізована багатокритеріальна оцінка
        score = (
            0.40 * route.p95_latency_ms / context.max_latency_ms
            + 0.25 * route.estimated_cost / context.max_cost
            + 0.25 * route.error_rate
            + 0.10 * route.uncertainty
        )
        candidates.append((score, route.route_id, route))
    if not candidates:
        return None
    # Детермінований вибір із розв'язанням рівності
    return min(candidates, key=lambda item: item[:2])[2]
```

Наведена функція демонструє відбір із урахуванням функціональної сумісності, регіону, конфіденційності, затримки та вартості. Вона не виконує безпосередніх хмарних операцій. Для повноцінної реалізації додатково необхідно

враховувати обмеження розміру даних, поточні квоти, семантику підтвердження виконання та вимоги до узгодженості результатів. Параметр available повинен визначатися на основі актуальних перевірок стану адаптера, а не лише статичної конфігурації.

Обчислювальна складність наведеного алгоритму становить $O(m)$, де m є кількістю зареєстрованих маршрутів. Кожен кандидат перевіряється один раз. Така складність є прийнятною для невеликої кількості хмарних адаптерів. За великого каталогу доцільно використовувати попередню індексацію за типом операції, регіоном та політикою доступу. Це дозволяє зменшити кількість кандидатів, які надходять до етапу ранжування.

Для програмної реалізації механізму можна використати окремий клас, який контролює кількість невдалих викликів і час очікування перед пробною операцією. Наступний фрагмент демонструє базову логіку переходів між станами. Для багатопотокового середовища або декількох процесів її необхідно доповнити синхронізацією спільного стану.

```
from dataclasses import dataclass
from time import monotonic

@dataclass
class CircuitBreaker:
    failure_limit: int = 5
    recovery_timeout: float = 30.0
    state: str = "CLOSED"
    failures: int = 0
    opened_at: float | None = None
    probe_in_progress: bool = False
    def allow_request(self) -> bool:
        # У звичайному стані виклики дозволені
        if self.state == "CLOSED":
            return True
        now = monotonic()
        # Перевірка завершення періоду блокування
        if self.state == "OPEN":
            if self.opened_at is None:
                return False
            if now - self.opened_at < self.recovery_timeout:
                return False
            self.state = "HALF_OPEN"
        # У пробному режимі дозволяється один виклик
        if self.state == "HALF_OPEN":
            if self.probe_in_progress:
                return False
            self.probe_in_progress = True
            return True
        return False
    def record_success(self):
        # Успішна перевірка відновлює робочий стан
```

```
        self.state = "CLOSED"
        self.failures = 0
        self.opened_at = None
        self.probe_in_progress = False
    def record_failure(self):
        # Помилка пробного виклику знову блокує сервіс
        self.probe_in_progress = False
        if self.state == "HALF_OPEN":
            self.open()
            return
        self.failures += 1
        if self.failures >= self.failure_limit:
            self.open()
    def open(self):
        self.state = "OPEN"
        self.opened_at = monotonic()
        self.probe_in_progress = False
```

Наведений механізм демонструє принцип обмеження звернень до несправної служби. Для робочої конфігурації необхідно враховувати не лише послідовні помилки, а й частку невдалих запитів у часовому вікні. Окремо слід визначити, які саме помилки впливають на стан Circuit Breaker. Наприклад, некоректний запит користувача не є доказом несправності провайдера та не повинен безумовно збільшувати лічильник відмов.

Важливо не змішувати тимчасову недоступність сервісу з перевищенням установленої квоти. У разі отримання відповіді HTTP 429 або іншого підтвердження обмеження інтенсивності звернень система повинна враховувати правила провайдера. Заголовок Retry-After, якщо він наявний, може визначати рекомендований час до наступної спроби. Повторне передавання без урахування таких обмежень здатне погіршити ситуацію.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Впровадження системи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції IT-систем із хмарними службами передбачає розгортання серверного середовища, налаштування інтеграційних адаптерів і забезпечення взаємодії корпоративних застосунків із зовнішніми програмними інтерфейсами. Програмний комплекс призначений для приймання інтеграційних запитів, перевірки їхньої структури, автоматизованого вибору хмарного маршруту та перетворення даних між гетерогенними інформаційними системами.

Додатково передбачається контроль доступності зовнішніх сервісів, оброблення помилок і збереження результатів виконання операцій. Головне вікно програми показано на рисунку 5.1.

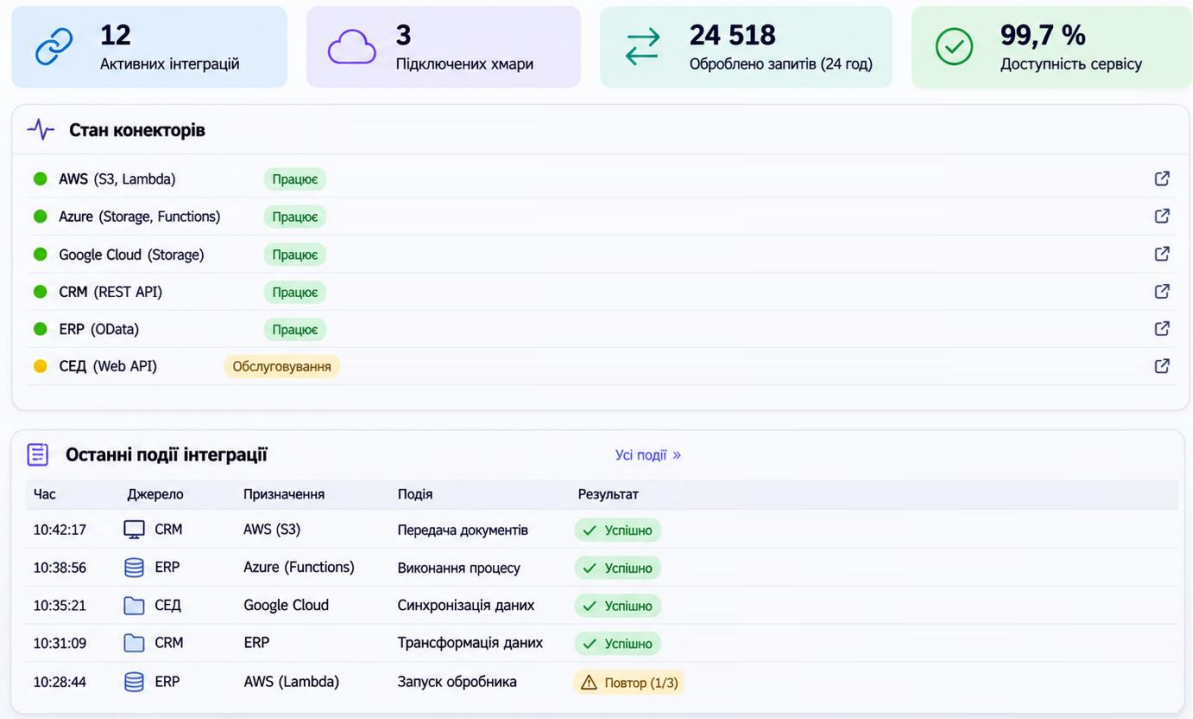

Рисунок 5.1 – Головне вікно ПЗ

Для розгортання комплексу доцільно використовувати серверну операційну систему Linux із підтримкою Python. Серверне середовище повинно забезпечувати одночасне обслуговування інтеграційних запитів, виконання асинхронних мережевих операцій і взаємодію з базою даних. Апаратні вимоги визначаються кількістю корпоративних систем, інтенсивністю звернень і складністю перетворення повідомлень. Особливу увагу необхідно приділити обсягу оперативної пам'яті та стабільності мережевого з'єднання. Для масштабованої конфігурації окремі компоненти можуть розміщуватися на незалежних віртуальних машинах або в контейнеризованому середовищі.

Як технологічну основу серверної частини пропонується використання FastAPI. Цей фреймворк забезпечує створення уніфікованого програмного інтерфейсу для приймання запитів від корпоративних застосунків. Для асинхронної HTTP-взаємодії з хмарними службами може застосовуватися бібліотека HTTPX. Зберігання інформації про маршрути, адаптери, завдання та результати виконання доцільно організувати засобами PostgreSQL. За необхідності асинхронного оброблення передбачається використання RabbitMQ. Для збору діагностичної інформації може застосовуватися OpenTelemetry. Зазначені технології формують рекомендовану конфігурацію відповідно до алгоритмічної архітектури системи.

Підготовка програмного середовища починається зі встановлення сумісного інтерпретатора Python та створення окремого віртуального середовища. Після цього встановлюються залежності серверного API, бібліотеки роботи з HTTP, компоненти взаємодії з PostgreSQL та засоби оброблення конфігурації. Якщо використовується брокер повідомлень, необхідно додатково встановити відповідний клієнтський модуль. Після завершення підготовки перевіряється можливість завантаження програмних компонентів і встановлення підключень до необхідних служб. Конкретні версії бібліотек повинні відповідати конфігурації програмного проєкту.

Перед запуском middleware необхідно підготувати базу даних. У ній передбачається зберігання реєстру хмарних адаптерів, підтримуваних операцій,

параметрів маршрутизації та історії інтеграційних запитів. Окремо доцільно зберігати стани асинхронних завдань і результати адміністративних операцій. Після створення необхідних структур перевіряються повноваження службового облікового запису. Доступ до бази повинен надаватися лише авторизованим програмним компонентам. Для забезпечення можливості відновлення необхідно передбачити резервне копіювання конфігурації та критичних метаданих.

Наступним етапом є налаштування інтеграційних адаптерів. Для кожного хмарного сервісу визначаються адреса програмного інтерфейсу, перелік підтримуваних операцій, допустимі формати повідомлень та спосіб автентифікації. Додатково встановлюються часові обмеження, параметри повторних звернень і правила оброблення помилок. Для адаптерів, які підтримують однакові функціональні операції, необхідно перевірити семантичну сумісність результатів. Лише після цього відповідні маршрути можуть використовуватися як альтернативні способи виконання інтеграційного запиту.

Окремої уваги потребує підготовка механізмів автентифікації. Для підключення до хмарних служб можуть використовуватися службові облікові записи, токени доступу або інші підтримувані постачальником механізми. Конфіденційні параметри необхідно зберігати в захищеному сховищі секретів. Передавання даних повинно здійснюватися через захищені HTTPS-з'єднання з перевіркою сертифікатів. Для корпоративних застосунків також необхідно налаштувати перевірку повноважень. Вхідний ідентифікатор організаційного сегмента повинен зіставлятися з автентифікованим контекстом запиту.

Після підготовки адаптерів виконується налаштування канонічної моделі даних. Для кожної інтеграційної операції визначаються обов'язкові поля, допустимі типи значень і правила перетворення. Необхідно перевірити відповідність форматів часу, числових характеристик та ідентифікаторів. Для подійної інтеграції може застосовуватися представлення CloudEvents. Для синхронних команд доцільно використовувати окремі контракти. Перед початком експлуатації необхідно перевірити правильність перетворення контрольних повідомлень між внутрішнім форматом і структурами зовнішніх хмарних API.

Налаштування адаптивної маршрутизації передбачає створення каталогу доступних маршрутів. Для кожного з них визначаються підтримувані операції, регіон розміщення, допустимі класи конфіденційності та експлуатаційні обмеження. Додатково задаються вагові коефіцієнти критеріїв вибору, які характеризують затримку, вартість і надійність. Початкові значення повинні встановлюватися відповідно до вимог інтеграційних процесів. Якщо використовується навчена модель прогнозування затримки, необхідно окремо перевірити сумісність її вхідних характеристик із поточними даними.

Для забезпечення стійкості до відмов налаштовуються механізми контролю часу очікування та повторного виконання запитів. Окремо визначаються пороги активації Circuit Breaker і тривалість перевірки відновлення сервісу. Повторні спроби повинні застосовуватися лише до операцій, для яких вони є допустимими. Для запитів зі зміною стану необхідно враховувати підтримку ключів ідемпотентності. Якщо результат попередньої операції невідомий, система не повинна автоматично повторювати її без відповідної перевірки.

За використання асинхронних завдань необхідно підготувати RabbitMQ та визначити логічні черги для передавання повідомлень між компонентами. Після цього налаштовуються виконавчі процеси й механізм обліку станів завдань. Необхідно перевірити правильність підтвердження оброблення повідомлень і поведінку за тимчасової недоступності брокера. Для запобігання втраті завдань доцільно передбачити узгодження записів у базі даних із передаванням повідомлень. Повторне отримання одного завдання не повинно спричиняти неконтрольоване дублювання хмарних операцій.

Рекомендований порядок запуску передбачає початкову перевірку доступності бази даних і сховища секретів. Після цього активуються необхідні допоміжні служби, зокрема брокер повідомлень за наявності асинхронної обробки. Наступним запускається серверний компонент middleware, який завантажує конфігурацію інтеграційних адаптерів і перевіряє готовність основних залежностей. Після підтвердження працездатності запускаються фонові виконавці

завдань. Завершальним етапом активуються засоби моніторингу та користувацький інтерфейс. Перед прийманням робочих запитів необхідно підтвердити готовність усіх обов'язкових компонентів.

Первинну перевірку працездатності доцільно виконувати на контрольованих інтеграційних сценаріях. Спочатку корпоративний застосунок передає тестовий запит через уніфікований API. Після цього перевіряється правильність його нормалізації та визначення допустимих хмарних маршрутів. Обраний адаптер повинен сформувати повідомлення відповідно до контракту зовнішнього сервісу. Після отримання відповіді необхідно перевірити її перетворення до внутрішнього формату та збереження результату. Окремо перевіряються випадки непідтримуваних операцій, некоректних параметрів і відсутності дозволеного маршруту.

Ідентифікація програмного продукту передбачає відображення відомостей про розробника та належність програмного забезпечення до відповідної магістерської кваліфікаційної роботи. Авторську інформацію доцільно розміщувати в окремому інформаційному розділі користувацького інтерфейсу. Вона повинна бути відокремлена від конфігурації хмарних адаптерів, службових облікових даних і внутрішніх параметрів корпоративних систем. Відомості про авторство розробленого програмного забезпечення представлено на рисунку 5.2.

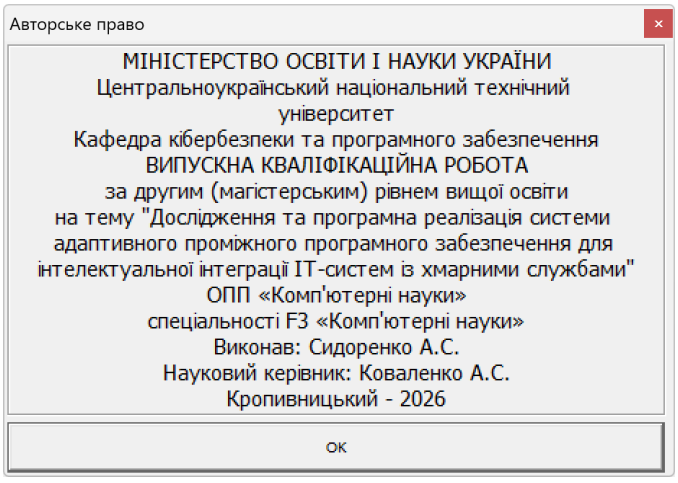

Рисунок 5.2 – Шаблона форма авторського права

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції ІТ-систем із хмарними службами.

Метою розробки є дослідження та програмна реалізація системи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції ІТ-систем із хмарними службами.

Об'єктом дослідження є процес адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції ІТ-систем із хмарними службами.

Предметом дослідження є методи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції ІТ-систем із хмарними службами.

Методи дослідження базуються на методах хмарних обчислень, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції ІТ-систем із хмарними службами.

– Розроблено вітчизняний продукт адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції ІТ-систем із хмарними службами, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

7.1 Вступ

Охорона праці в галузі ІТ-технологій дозволяє зберегти здоров'я, а іноді й життя працівників, тому є актуальною задачею.

Під час роботи з електронно-обчислювальними машинами (ЕОМ) на користувача у тій чи іншій мірі можуть впливати такі фізичні чинники:

– підвищені рівні змінного електромагнітного і електростатичного полів;

– підвищений рівень статичної електрики;

– підвищений рівень низько енергетичного (м'якого) рентгенівського іонізуючого випромінювання;

– підвищені рівні ультрафіолетового та інфрачервоного випромінювання;

– підвищений вміст позитивних аероіонів у повітрі робочої зони;

– знижений вміст негативних аероіонів;

– аномальний рівень освітленості робочої зони;

– підвищена яскравість фрагментів світлового зображення або світла, що потрапляє у поле зору користувача;

– підвищена нерівномірність розподілу яскравості у полі зору користувача;

– підвищена зовнішня освітленість екрану;

– підвищені пульсації світлового потоку джерел світла або світлового потоку, що випромінюється екраном;

– несприятливий для роботи спектр випромінювання джерел світла;

– підвищена часова нестабільність зображення; мерехтіння екрану;

– зміна яскравості свічення екрана;

– підвищена пряма блисткість, викликана попаданням у поле зору працюючого надмірно яскравого світла різних випромінюючих об'єктів;

– підвищена відбита блисткість, обумовлена наявністю дзеркальних відбиття (відблисків), у тому числі від екрану;

– підвищений рівень шуму;

– аномальні температура, вологість та рухливість повітря робочої зони;

– підвищене значення напруги в електричному ланцюзі, замикання якого може статися через тіло людини;

– пожежа.

До недоліків умов праці користувачів комп’ютерної техніки можна віднести також недостатню площу і обсяг виробничого приміщення, недотримання вимог, мікроклімату на робочих місцях.

Праця користувачів ПЕОМ відносять до психічних форм праці з високим ступенем навантаження.

Комп’ютер, як і будь-який електричний прилад, особливо при його неправильному підключенні, може бути джерелом ураження оператора електричним струмом. Саме тому всі працівники, які працюють з персональним комп’ютером, повинні мати першу(або другу) групу допуску з електробезпеки.

Ці шкідливі фактори можуть привести до професійних захворювань.

Відповідно до ст.14 Закону «Про охорони праці» [3] на роботодавця покладено обов'язок забезпечити:

– безпеку працівників при експлуатації устаткування;

– застосування коштів індивідуальної захисту працівників;

– відповідні вимоги охорони праці, умови праці в кожному робоче місце;

– дотримання режиму праці та відпочинку працівників;

– навчання безпечним методам і прийомам виконання;

– інструктажі з охорони праці;

– організацію контролю над станом умов праці в робочих місць;

– проведення атестації робочих місць в умовах праці.

Максимально зменшити кількість шкідливих впливів на людину при високій продуктивності праці, створити комфортні умови для роботи людей – одна з головних задач охорони праці [5].

7.2 Аналіз умов праці на робочому місці фахівця

Робота програміста пов'язана з постійною роботою на ЕОМ, яка відбувається у кімнаті розмірами 4,4 м×6,2 м×2,9 м. Одна з її більших стін має шість двостулкових вікон, розмірами 2 м×1,8 м, які виходять на північний захід. Вікна розташовані рівномірно по всій довжині стіни. Підлога в кімнаті покрита лінолеумом, всі стіни пофарбовані світло оранжевого кольору до висоти 2,8 м, а далі підвісна стеля. Уздовж стін розташовані комп'ютерні столи. На них розташовуються 2 персональні комп'ютери й інша оргтехніка (сканер принтери, телефони й ксерокс). Столи мають пластикове покриття. Габарити їхньої робочої поверхні 1255 мм×845 мм. Висота столів 760 мм. Висота стільців від рівня підлоги становить 430 мм.

Згідно НПАОП 0.00 – 1.28 – 10 «Правила охорони праці під час електронно-обчислювальних машин» площа повинна задовольняти умові – не менш 6 $м^2$ на одне робоче місце. Кратність повітрообміну в приміщенні вузла також регламентується ДСанПіН 3.3.2.007-98 [2], вона повинна становити 20 $м^3$/годину на одне місце. Виконання даних вимог забезпечить підтримку в приміщенні вузла оптимального значення вологості й складу повітря.

Відповідно ДБН В.2.5 – 28 – 2006 [1] роботу програміста можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення вузла можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при сполученому висвітленні), повинен становити 0,5%, освітленість при штучному висвітленні повинна становити 300 лк.

За результатами виміру освітленості відділом охорони праці величина освітленості від системи загального штучного висвітлення лежить у межах 200-250 лк, що не відповідає вимогам, які пред'являються до приміщення.

Відповідно ДСанПіН 3.3.2.007-98 [2] рівні звукового тиску в робочому приміщенні не повинні перевищувати в октавних смугах із середньо геометричними частотами наступних значень, наведених у таблиці 7.1.

Таблиця 7.1 – Допустимі спектри рівнів звукового тиску

Робоче місце	Рівень звукового тиску, дБ, в октавних смугах із середньогеометричними частотами, Гц								Рівень звуку і еквівалентний рівень звуку, дБА
	63	125	250	500	1000	2000	4000	8000	
Приміщення конструкторських бюро, програмістів обчислювальних машин, лабораторій для теоретичних робіт і опрацювання експериментальних даних, прийому хворих в медпунктах	71	61	54	49	45	42	40	38	50

У приміщенні перебувають наступні джерела шуму: електродвигуни внутрішнього вентилятора ЕОМ; працюючі принтери; працюючі дисководи. Шум, вироблений вентилятором можна класифікувати як постійний, всі інші джерела шуму, як імпульсні. Відповідно паспорта на приміщення рівень звуку, Дб(А), обмірюваний за шкалою (А) шумоміра досяг величини 28,3 Дб(А) при роботі всього встаткування вузла, включаючи й ксерокс. Це дозволяє зробити

висновок про відповідність рівня звуку в приміщенні вимогам нормативних актів.

Ергономічні вимоги до робочого місця працюючого з ВДТ ЕОМ і ПЕОМ нормуються НПАОП 0.00 – 1.28 – 10**.** Оптимальне положення тіла того, що працює забезпечується відповідною конструкцією робочого місця, а також регуляцією висоти робочої поверхні, сидіння, простори й підставки для ніг. Даного місця програміста не мають регульованих параметрів. Відмінності реальних параметрів робочого місця від параметрів відповідні вимоги нормативного акту дані в таблиці 7.2.

Таблиця 7.2 – Відмінності реальних параметрів робочого місця від параметрів відповідні вимоги нормативного акту

Ріст людини, см	Висота робочої поверхні мм,	Висота простору для ніг, мм	Висота робочого сидіння, мм
175	765(740)	655(600)	450(440)

У дужках зазначені реальні значення параметрів робочого місця; всі вони не відповідають параметрам, зазначеним у стандарті.

Параметри мікроклімату можуть мінятися в широких межах, тоді як необхідною умовою життєдіяльності людини є підтримка сталості температури тіла завдяки властивості терморегуляції, тобто здатності організму регулювати віддачу тепла в навколишнє середовище.

У приміщеннях, де встановлені комп'ютери, повинні дотримуватися певні параметри мікроклімату. У санітарних нормах ДСН 3.3.6.042 – 99 [4] встановлені величини параметрів мікроклімату, що створюють комфортні умови. Ці норми встановлюються в залежності від пори року, характеру трудового процесу і характеру виробничого приміщення (табл. 7.3).

Таблиця 7.3 – Параметри мікроклімату для приміщень, де встановлені комп'ютери

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні	22 – 24°С
	Відносна вологість	40 – 60%
	Швидкість руху повітря	до 0,1 м/с
Теплий	Температура повітря в приміщенні	23 – 25°С
	Відносна вологість	40 ... 60%
	Швидкість руху повітря	0,1 ... 0,2 м / с

7.3 Розробка заходів з умов поліпшення охорони праці

Провівши аналіз умов праці в розглянутому вище приміщенні, було отримано наступні результати:

– значення мікроклімату в приміщенні не перевищує норму;

– розрахунки розміру робочого місця на одного працівника відповідають нормі;

– рівень шуму в приміщенні не становить вище норми.

З вище перерахованих результатів можна зробити висновок, що основний вплив на продуктивність ІТ-фахівців є його психологічний стан. Тому є доцільним зменшити рівень стресу на робочому місці.

Рекомендовані наступні заходи: За потреби особливої концентрації уваги під час виконання робіт суміжні робочі місця операторів необхідно відділяти одне від одного перегородками висотою 1,5 – 2 м. Конструкція робочого місця користувача персонального комп’ютера має забезпечити підтримання оптимальної робочої пози офісного працівника. Конструкція робочого столу має відповідати сучасним вимогам ергономіки і забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання (дисплея, клавіатури, принтера) і документів. Висота робочої поверхні робочого столу має регулюватися в межах 680-800 мм, а ширина і глибина – забезпечувати

можливість виконання операцій у зоні досяжності моторного поля (рекомендовані розміри: 600-1400мм, глибина – 800-1000мм). Робочий стіл повинен мати простір для ніг заввишки не менше ніж 600 мм, завширшки не менше ніж 500 мм, у глибину (на рівні колін) не менше ніж 450 мм, на рівні простягнутої ноги не менше ніж 650 мм. Робочий стілець має бути підйомно-поворотним, регульованим за висотою, з кутом і нахилу сидіння та спинки і за відстанню від спинки до переднього краю сидіння поверхня сидіння має бути плоскою, передній край – заокругленим [7].

7.4 Розрахункова частина

Для захисного штучного заземлення виберемо вертикальні електроди: металевий куток 80х80х8 мм, (згідно з ДСТУ 2251:2018 «Кутики сталеві гарячекатані рівнополичні. Сортамент») довжиною L=1,6 м та горизонтальний електрод – металева полоса з перетином 60х5 мм. Напруга – 220/380 В. Розрахункова схема розташування заземлюючих електродів – у ряд (рис. 7.1).

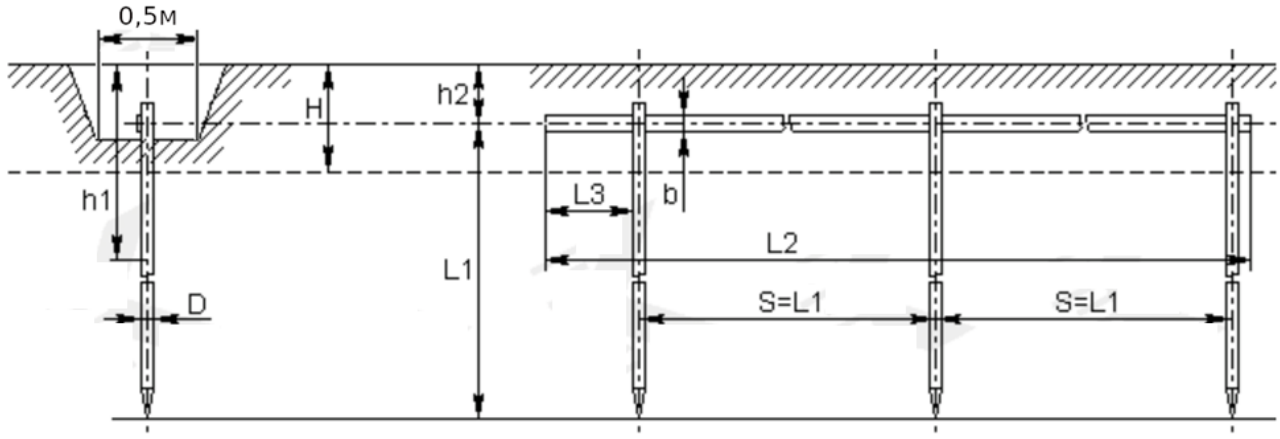

Рисунок 7.1 – Схема штучного заземлення.

Розрахунок проведемо за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару ґрунту – чорнозем, нижнього шару ґрунту – глина (питомий опір ρ_2 = 40 Ом·м). Умовна товщина верхнього шару ґрунту: H=0,5 м. Відстань між вертикальними заземлювачами (електродами) A=1,6 м. Глибина закладення горизонтального контура заземлення t=0,6 м. Опір заземлювача, який нормується: $R_{ЗН}$ = 4 Ом. Необхідно визначити необхідну кількість вертикальних заземлювачів та довжину полоси (горизонтального заземлювача).

Виконаємо розрахунок.

Відстань від центра вертикального заземлювача до поверхні землі:

$$T = t + L/2 = 0,6 + 1,6/2 = 1,4 \text{ м.}$$

Розрахунковий питомий опір ґрунту (з врахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі ґрунту):

$$\rho = \psi \cdot \rho = 1,36 \cdot 40 = 54,5 \text{ Ом·м.}$$

де ψ = 1,36 – табличне значення коефіцієнта сезонності для відповідної климатичной зони у багатошаровому грунті [6]; ρ_2 = 40 Ом·м. – табличне значення питомого опору нижнього шару ґрунту (глина) [7].

Еквівалентний діаметр вертикального електрода (кутка) [7]:

$$D_в = 0,95 \cdot К = 0,95 \cdot 80 = 76,85 \text{ мм} = 0,076 \text{ м.}$$

де К= 80 мм – розмір полиці металевого кутка.

Відношення A/L=1,6/1,6=1.

Опір розтіканню електричного струму одного електрода вертикального заземлювача з урахуванням заглиблення заземлювача [7]:

$$R_О = 0,366 \cdot (\rho/L) \cdot [\lg(2L/D_в) + (1/2) \cdot \lg((4T+L)/(4T-L))] =$$

$$= 0,366 \cdot (54,5/1,6) \cdot [(\lg(2 \cdot 1,6/0,076) + (1/2) \cdot \lg((4 \cdot 1,4+1,6)/(4 \cdot 1,4-1,6))] = 21,8 \text{ Ом.}$$

Визначаємо коефіцієнт екранування вертикальних електродів Кев=0,8 при орієнтовній кількості вертикальних електродів, яка дорівнює 5 [7].

Визначаємо необхідну кількість вертикальних електродів заземлювача (без врахування горизонтального заземлювача), при $R_{ЗН}$ = 4 Ом:

$$N = R_О / (Кев \cdot R_{ЗН}) = 21,8 / (0,8 \cdot 4) = 6,8 \approx 7 \text{ шт.}$$

Визначаємо довжину з’єднуючої полоси:

$$L_П = 1{,}05 \cdot A \cdot N = 1{,}05 \cdot 1{,}6 \cdot 7 = 12{,}2 \approx 12 \text{ м.}$$

Опір розтіканню електричного струму з’єднуючої полоси з урахуванням кліматичного коефіцієнту питомого опору ґрунту $K_П$ [7]:

$$R_П = 0{,}366 \cdot (\rho \cdot K_П / L_П) \cdot \lg(2 \cdot L_П^{\,2} / (B \cdot t)) =$$

$$= 0{,}366 \cdot (40 \cdot 5/12) \cdot \lg((2 \cdot 12^{\,2})/(0{,}06 \cdot 0{,}6)) = 20{,}16 \text{ Ом.}$$

де

$K_П$=5 – табличне значення кліматичного коефіцієнту питомого опору ґрунту для відповідної кліматичної зони для з’єднуючої полоси [7]:

B = 60 мм. = 0,06 м – ширина з’єднуючої полоси (задана).

Загальний опір розтіканню електричного струму заземлювача [7]:

$$R = (R_O \cdot R_П)/(R_O \cdot \eta_П + N \cdot R_П \cdot Кев) =$$

$$= (21{,}8 \cdot 20{,}16)/(21{,}8 \cdot 0{,}75 + 9 \cdot 20{,}16 \cdot 0{,}8) = 3{,}46 \text{ Ом.}$$

де $\eta_П = 0{,}75$ – табличне значення коефіцієнта екранування з’єднуючої полоси [7].

Умова $R \leq R_{ЗН}$ виконується (3,46 Ом ≤ 4 Ом).

Оскільки при 7 вертикальних електродах R суттєво менше $R_{ЗН}$, зменшимо кількість вертикальних електродів N до 6 і виконаємо перерахунок. У результаті остаточно отримали: R =3,96 Ом. при кількості вертикальних електродів N=6.

7.5 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров’я працівників, а також підвищує ефективність виробництва в цілому.

З цих міркувань було здійснено аналіз умов праці на робочому місці програміста, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи.

Виконано розрахунок захисного штучного заземлення, як одного з ключових факторів безпеки програміста.

8 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції ІТ-систем із хмарними службами.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції ІТ-систем із хмарними службами.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції ІТ-систем із хмарними службами.

– Досліджена система адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції ІТ-систем із хмарними службами.

– На основі отриманих результатів досліджень створена програмна реалізація системи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції ІТ-систем із хмарними службами.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції ІТ-систем із хмарними службами.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сидоренко А.С. Дослідження та програмна реалізація системи адаптивного проміжного програмного забезпечення для інтелектуальної інтеграції IT-систем із хмарними службами // Збірник праць молодих науковців ЦНТУ. – Вип. 17. – Кропивницький: ЦНТУ, 2027.

2. Ramon Nastase «Computer Networking: The Beginner's guide for Mastering Computer Networking, the Internet and the OSI Model». 2018. – 186 p.

3. Russ White & Ethan Banks «Computer Networking Problems and Solutions: An Innovative Approach to Building Resilient, Modern Networks». 2017. – 832 p.

4. Vintenko, B., Myronets, I., Smirnov, O., Smirnova, T. «Enhancing the Reliability of Input Data for Nuclear Power Plant Operator Support Systems». *International Conference on Sensors, Devices and Systems. SDaS 2025. Lecture Notes in Electrical Engineering*, 2026. vol 1617. pp 3-12.

5. Kuznetsov O., Frontoni E., Kuznetsova Y., Chevardin V., Smirnov O. «Architectural foundations for adaptive security in edge computing systems». *Cybersecurity Defensive Walls in Edge Computing*, 2025. pp. 21-61.

6. Kuznetsov, O., Frontoni, E., Kuznetsova, K., Arnesano, M., Smirnov, O. «A secure biometric authentication architecture for blockchain-driven cyber-physical systems». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 193–224.

7. Kuznetsov, O., Smirnov, O., Kuznetsova, T., Shaikhanova, A., Svatowsky, I. «Privacy-utility trade-offs in IoT networks: A comparative analysis of differential privacy mechanisms for sensor data aggregation». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 589–622.

8. Kuznetsov, O., Atzeni, G., Arnesano, M., Randieri, C., Smirnov, O. «Secure IoT-based smart wheelchair system: From implementation to security

enhancement strategy». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 225–257.

9. Богаченко, С., Смірнов, О., Буравченко, К., Смірнова, Т., Конопліцька-Слободенюк, О., Якименко, Н., Смірнов, С. «Обмеження ресурсів мобільних та edge-платформ при розробці програмного забезпечення для реалізації та оптимізації моделей машинного навчання». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»*, 2026, т. 2 № 34, С. 677-686.

10. Вінтенко Б.Ю., Смірнова Т.В., Миронець І.В., Смірнов О.А., Буравченко К.О. «Метод оцінки функціональної стійкості комп'ютерно-орієнтованих процедур системи підтримки оперативного персоналу АЕС». *Системи управління, навігації та зв'язку*, 2026, вип. 2(84), С. 62-69.

11. Смірнова Т.В., Усік П.С., Лисенко І.А., Буравченко К.О., Смірнов О.А. «Методологія підтримки технологічних процесів у критичній інфраструктурі з забезпеченням безпеки інформації на основі хмарних технологій». *Безпека інформації.* 2025. Том 31 № 1. С. 49-60.

12. Смірнова Т.В., Буравченко К.О., Добринчук О.А., Смірнов С.А., Якименко Н.М., Смірнов О.А. «Хмарна технологія моніторингу ключових індикаторів ефективності технологічних процесів у критичній інфраструктурі». *Центральноукраїнський науковий вісник. Технічні науки.* 2025. № 12(43), ч. II. С. 36-54.

13. Вінтенко, Б.Ю., Миронець, І.В., Смірнов, О.А., Коваленко, О.В., Усік, П.С., Буравченко, К.О., Лисенко, І.А. «Логіко-структурна модель комп'ютерно-орієнтованої процедури системи підтримки оперативного персоналу АЕС». *Кібербезпека: освіта, наука, техніка*. 2025. Том 2 № 30. С. 413-427.

14. Вінтенко Б.Ю., Смірнов О.А., Миронець І.В., Смірнова Т.В., Коваленко О.В., Мацуй А.М. «Модель шляхів отримання вхідних даних комп'ютерної інтелектуальної системи підтримки оперативного персоналу АЕС». *Центральноукраїнський науковий вісник. Технічні науки.* 2025. Вип.

11(42), ч. II. С.52-62.

15. Вінтенко, Б., Миронець, І., Смірнов, О., Коваленко, А., Конопліцька-Слободенюк, О., Смірнова, Т., Константинова, Л. «Дослідження застосування систем підтримки оперативного персоналу об'єкту критичної інфраструктури при керуванні енергоблоком АЕС з реактором типу ВВЕР-1000». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2024. № 2(26), С. 6-26.

16. Ткаченко, О., Ільєнко, А., Улічев, О., Мелешко, Є., Смірнов, О. «Правові засади поширення інформаційних впливів в соціальних мережах». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2024. № 2(26), С. 170–188.

17. Вінтенко Б.Ю., Миронець І.В., Смірнов О.А. Коваленко О.В., Смірнов С.А., Буравченко К.О., Якименко Н.М. «Дослідження інформаційного забезпечення та технологічних регламентів процесів керування критичною інфраструктурою енергоблоку АЕС з реактором типу ВВЕР-1000». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка».* 2024. № 1(25), С. 253–278.

18. Смірнова Т.В., Константинова Л.В., Конопліцька-Слободенюк О.К., Козлов Я.О., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження сучасного стану SIEM-систем». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка».* 2024. № 1(25), С. 6-18.

19. Al-Azzeh, J., Ayyoub, B., Mesleh, A., Smirnova, T., Gnatyuk, S., Drieiev, O., Smirnov, O., Dorenskyi, O. «Cloud-Based Information System for Evaluating Caverns in the Process of Blasting Metal Surfaces of Details». *International Review on Modelling and Simulations* 18 (1), 2025. pp. 32-42.

20. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.

21. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О.

«Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

22. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

23. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

24. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

25. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». *Lecture Notes on Data Engineering and Communications Technologies*, 2023, 178, pp. 208–223.

26. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

27. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399.

28. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

29. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А.,

Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

30. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

31. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

32. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». *Communications in Computer and Information Science*, 2021, vol 1486. Springer, Cham. pp 169-184.

33. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

34. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

35. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

36. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

37. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

38. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379.

39. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645.

40. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». *International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019*; Odessa; Ukraine; 9-13 September 2019. P.22-28.

41. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

42. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019. P.517-522.

43. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE*

International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

44. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

45. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

46. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

47. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

48. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

49. Smirnov A.A., Kuznetsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

50. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

51. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

52. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

53. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

54. О.А. Смірнов, П.С. Усік, «дослідження перспектив використання технологічних рішень в мережах 5g» у *Кібербезпека та інформаційні технології: монографія.* – Х.: ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

55. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

56. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

57. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

58. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та

інформаційні технології: монографія / за заг. ред. В. С. Пономаренка. – Х.: Вид. Рожко С.Г. 2019. С. 123-139

59. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

60. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

61. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.