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 першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи програмно визначаємого ЦОД на базі технологій Fujitsu”

Виконав здобувач вищої освіти
IV курсу, групи КІ-22-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Просінюк М.С.
«____» _____ 2026 р.

Керівник проекту
доктор філософії (PhD)
_____ Дрєєва Г.М.
«____» _____ 2026 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерна інженерія"*_

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

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Просінюку Максиму Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи програмно визначаємого ЦОД на базі технологій Fujitsu*

2. Керівник роботи *Дрєєва Ганна Миколаївна, доктор філософії (PhD)*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 133-02 від 27.02.2026 року

3. Строк подання студентом роботи до захисту *23.05.2026 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи програмно визначаємого ЦОД на базі технологій Fujitsu*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

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

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

7. Дата видачі завдання « 27 » лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2026 р.	
3.	Розробка моделі компонента	20.03.2026 р.	
4.	Розробка структур даних	25.03.2026 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2026 р.	
6.	Програмування алгоритмів	10.04.2026 р.	
7.	Оформлення ПЗ	17.04.2026 р.	
8.	Попередній захист роботи	23.05.2026 р.	

Дата видачі завдання
« 27 » лютого 2026 р.

Підпис керівника
_____ Дрєєва Г.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 27 » лютого 2026 р.

Підпис здобувача
_____ Просінюк М.С.
(прізвище та ініціали)

АНОТАЦІЯ

Просінюк М.С. Програмне забезпечення системи програмно визначаємого ЦОД на базі технологій Fujitsu. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи програмно визначаємого ЦОД на базі технологій Fujitsu.

Метою розробки є програмне забезпечення системи програмно визначаємого ЦОД на базі технологій Fujitsu.

Результат роботи – програмна реалізація системи програмно визначаємого ЦОД на базі технологій Fujitsu.

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'ютерна інженерія, програмно визначаємий ЦОД, Fujitsu

ABSTRACT

Prosiniuk M.S. Software for a software-defined data center system based on Fujitsu technologies. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for a software-defined data center system based on Fujitsu technologies.

The purpose of the development is software for a software-defined data center system based on Fujitsu technologies.

The result of the work is a software implementation of a software-defined data center system based on Fujitsu technologies.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software are provided.

The program can be used on a PC with Windows 10/11 OS.

The program was developed in the Python environment.

Keywords: computer engineering, software-defined data center, Fujitsu

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 2
ВСТУП..... 3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 4
1.1 Призначення системи..... 4
1.2 Область застосування..... 5
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 7
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти..... 7
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування..... 26
2.3 Розгорнута постановка завдання 31
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 32
3.1 Опис функціонування системи 32
3.2 Розробка структурної схеми..... 35
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
4.2 Захист розробленого програмного забезпечення..... 66
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 70
6 ОСНОВНІ ВИСНОВКИ..... 77
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 79

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ДПТМ	–	динамічний перерозподіл трафіку мережі
ЕВЕ	–	ефективність використання електроенергії
ЕЦП	–	електронний цифровий підпис
ДБЖ	–	джерело безперервного живлення
ІВК	–	інфраструктура відкритих ключів
ІТ	–	інформаційні технології
ЛОМ	–	локальна обчислювальна мережа
ПЗ	–	програмне забезпечення
СЗД	–	системи зберігання даних
СКЗІ	–	система комплексного захисту інформації
УК	–	управляючі компоненти
УЦ	–	удостовірюючий центр
ЦОД	–	центр обробки даних
DCIM	–	Datacenter Infrastructure Management – керування інфраструктурою ЦОД
DMaaS	–	Datacenter Management as a Service – керування дата-центром як послуга
IGMP	–	Internet Group Management Protocol
NLB	–	Network Load Balancing, балансування мережного навантаження
PKI	–	Public Key Infrastructure
VPN	–	віртуальна приватна мережа

ВСТУП

Актуальність теми. Оскільки цифрова трансформація охоплює всі галузі, центри обробки даних стали основою глобального прогресу. У 2026 році сектор зіткнеться з безпрецедентним імпульсом, зумовленим зростаючим попитом на штучний інтелект, хмарні та периферійні обчислення, а також невпинним прагненням до швидкості, ефективності та сталого розвитку. Ситуація швидко змінюється: на ринок виходять нові гравці, визнані лідери розширюють свою діяльність, і кожен регіон поспішає нарощувати потужності. Однак, з можливостями приходять і виклики. Обмеження влади, дефіцит талантів, тиск на ланцюги поставок та регуляторні перешкоди випробовують стійкість галузі. Водночас, проривні технології та сміливі стратегії переосмислюють можливості.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи програмно визначаємого ЦОД на базі технологій Fujitsu.

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

– Огляд існуючих систем програмно визначаємого ЦОД на базі технологій Fujitsu.

– Дослідження системи програмно визначаємого ЦОД на базі технологій Fujitsu.

– Програмна реалізація системи програмно визначаємого ЦОД на базі технологій Fujitsu.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі програмно визначаємого ЦОД на базі технологій Fujitsu.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи програмно визначаємого ЦОД на базі технологій Fujitsu, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Програмно-визначаємі мережі й віртуалізація мережних функцій – досить зрілі технології в плані реалізації функціональності L3 у центрах обробки даних. Однак жодна з них не пройшла повноцінну перевірку на відповідність загальним критеріям безпеки IT (Common Criteria). Останні являють собою механізм, за допомогою якого регулювальні органи можуть переконатися в безпеці систем. Реалізація їхніх вимог у програмному середовищі надзвичайно важке й коштовне.

Програмно-визначаємі компоненти важливі, але вони далеко не весь арсенал рішень. Так, програмно-визначаємі сховища далекі від зрілості. Я б не радив розміщати на них критичні навантаження. Разом з тим їх цілком можна довірити гіперконвергентній інфраструктурі. Стек Fujitsu VCF або стек Nutanix – це зовсім іншу справу, оскільки вони реалізують наскрізний контроль помилок. Гіперконвергентне рішення цілком життєздатне, у всякому разі для деяких навантажень, але цього, однак, не можна сказати про окремих програмно-визначаємих сховища, оскільки високорівневі мережні компоненти ще не готові. Взяти, наприклад, такі рішення на базі відкритого вихідного коду, як пропоновані Red Hat – у масштабних середовищах вони не працюють.

Хоча мені неодноразово намагалися пояснити, але я так і не зрозумів, що таке програмно-визначаємий сервер. Це якась фікція. Таким чином, у програмно-визначаємих рішень є своя ніша, але вони ще недостатньо зрілі. Гіперконвергентні рішення більше зрілі – той самий програмний стек реалізує функціональність серверів, мережі й зберігання. Однак і вони підходять не для всіх завдань – наприклад, не здатні ефективно здійснювати віддалене тиражування даних.

Таким чином, мова може йти про окремі елементи, але не про цілісне рішення.

1.2 Область застосування

Програмно-визначаємі центри обробки даних характеризуються не своїм наповненням, а способом керування ними. І це багато в чому вже реальність. Тут у гру вступає гібридний підхід. Якщо центр обробки даних здатний надавати застосунки, які можуть виконуватися як у хмарі, так і на площадці замовника, такий ЦОД по визначенню є програмно-визначаємим. І за його функціонування відповідають програми керування. Без сумніву, програмно-визначаємий ЦОД – важливий елемент цифрової інфраструктури майбутнього.

В остаточному підсумку апаратне забезпечення – це механізм виконання коду. Воно, звичайно, важливо, але час, коли застосунки залежали від платформ, на яких вони виконувалися, у минулому. Модель вертикального масштабування вмирає, тоді як модель горизонтального масштабування процвітає. Деякі застосунки пишуться таким чином, що для них необхідна багатопроцесорна обробка, але для цього є HPC.

Невелике застереження. Самі серверні архітектури міняються. Чи чули ви про дезагрегації? Давайте почнемо спочатку. Що таке сервер? Це гра в скільки знадобиться процесорів, пам'яті, портів для виконання типового завдання. При цьому обмеження визначаються не завданням, що треба буде розв'язати, а технологічною можливістю реалізувати необхідний функціональний набір.

У дійсності ж користувачі просто хотіли б мати досить апаратних ресурсів для виконання поточного завдання. Дезагрегацію можна розглядати як апаратний еквівалент програмної віртуалізації. Який би гіпервізор не застосовувався, він надає механізм для спільного використання наявних ресурсів сервера для рішення різних завдань. Але ефективність програмної віртуалізації обмежена, оскільки гіпервізор працює поверх доступної апаратної архітектури.

До появи гіпервізорів рівень використання ресурсів сервера був дуже низьким і не перевищував 12%. Сервер використовувався під конкретне завдання – у якості Web-сервера, сервера бази даних і т.п. Якщо він справлявся зі своїм завданням, те й чудово – нікого не турбувало, якщо він простоював.

У Великобританії є Департамент по податках і зборам Її Королівської Величності (Her Majesty's Revenue and Customs, HMRC). Ця урядова організація ввела процедуру самостійної оцінки. Кожний громадянин повинен заповнити форму, де потрібно вказати свої доходи, активи й т.д. – на основі цих даних розраховується податок. За законом дані повинні бути надані 10 січня (я не пам'ятаю точно, але це не суть важливо). Як неважко догадатися, декларації заповнюються в останній момент – 97% платників податків роблять це в останній тиждень.

Для прийому декларацій була створена величезна система, вона розрахована на пікове навантаження й розміщається в декількох ЦОД. Уряд не може собі дозволити, щоб система виявилася непрацездатної в піковий період, оскільки від цього залежить його дохід. Іншу частину року вона не робить нічого, крім як обігріває ЦОД. Це найбільш показовий приклад неефективності використання устаткування, що тільки можна привести.

Зміни неминучі. Незабаром у серверах з'явиться енергонезалежна оперативна пам'ять – цього року Intel повинна випустити пам'ять Apache Pass (Optane DIMM). Пам'ять буде надзвичайно швидкою й розташовуватися близько до процесора. Поряд із кремнієвою оптикою й подальшим розвитком програмно-визначаємих підходів, її поява буде сприяти дезагрегації – звичних «коробок» більше не буде. Процесори будуть перебувати в одному місці, пам'ять – в іншому, ввод-вивід – у третьому. Всі ці компоненти будуть розподілені так, як зручно замовникові, і зв'язані високошвидкісними з'єднаннями аналогічно тому, що в часи мейнфреймів називалося когерентною моделлю. Під конкретне завдання буде виділятися необхідний обсяг ресурсів апаратним гіпервізором.

Таким чином, апаратне забезпечення також зміниться. Але питання не в ньому. Головне – де перебуває мій застосунок, як воно надається клієнтам, яке час відгуку, як виконуються вимоги захисту даних і т.д.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи програмно визначаємого ЦОД на базі технологій Fujitsu, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

EcoStruxure IT – хмарне програмне забезпечення для керування інфраструктурою ЦОДів

Система дистанційного контролю – Asset Advisor

Відкрийте для себе хмарну систему цілодобового дистанційного контролю. Засновані на актуальних дані рекомендації від нашої команди сервісного обслуговування допоможуть вам краще контролювати свою систему.

EcoStruxure IT Expert

Підтримуйте безперебійну роботу системи й управляйте сигналами тривоги за допомогою хмарного моніторингу. Користуйтеся можливостями безпечного контролю системи з будь-якої точки світу.

Комерційно нейтральний моніторинг на базі хмарних рішень

Профілактичні рекомендації на основі даних про продуктивність і сигнали тривоги – для безпечного контролю, де б ви не перебували.

Інформація на основі даних

Відповідність галузевим стандартам для поліпшення розподілу ресурсів, планування, резервування й продуктивності.

Alsok пропонує робота Reborg-Z для моніторингу дата-центра

Інженери японської компанії Alsok запропонували проектувальникам, будівельниками й операторам ЦОД із усього світу новий інструмент для підвищення безпеки серверних ферм. Мова про робота Reborg-Z.

Машина здатна здійснювати патрулювання машзалів, моніторинг устаткування й інші процедури в центрі обробки даних. Охоронний робот

оснащений датчиками для виявлення небезпечних газів і має функцію пожежогасіння.

Пристрій уже був придбаний операторами ряду японських ЦОД. Зокрема, робот уже використовується в недавно відкрившемося дата-центрі компанії Internet Initiative Japan потужністю 50 Мвт у місті Широи, префектура Тиба (Японія).

Schneider Electric обновляє платформу EcoStruxture IT Advisor для аналітичного моніторингу ЦОД

Компанія Schneider Electric випустила нову версію своєї програмної платформи EcoStruxure, що спрощує керування інфраструктурою центрів обробки даних, а також планування використання ресурсів і моделювання дата-центрів.

Рішення з підзаголовком EcoStruxure IT Advisor, було представлено на заході DCD New York 2019. Розроблювачі змогли додатково спростити розгортання платформи, додавши можливість установки її як у хмарі, так і на локальному сервері. Клієнтам було запропоновано мінімізувати витрати на продукт завдяки моделі підписки.

Також був розширений функціонал продукту в сфері кібербезпеки й в інших областях. Зокрема, з'явилася можливість моделювання впливу різних інцидентів на вже встановлені в ЦОД IT-пристрої й допоміжну інфраструктуру.

APC by Schneider Electric пропонує нові системи спостереження Netbotz для дата-центра

Компанія APC by Schneider Electric представила нові продукти для ринку систем безпеки ЦОД, які можуть похвастати підтримкою Po і відеоспостереження із застосуванням 4-мегапіксельної камери. Мова про пристрої Netbotz 750 і NetBotz Camera Pod 165c.

Підтримка бездротових датчиків, який володіють продукти, усуває необхідність у прокладці стандартних мережних кабелів при розгортанні цих пристроїв. Системи були розроблені для захисту IT-середовищ від фізичних і

екологічних погроз, таких як перегрів і підвищена вологість, витоки води, дим і пожежа, а також несанкціонований доступ.

Оператори ЦОД тепер можуть віддалено через єдину IP-адресу контролювати такі функції як відеоспостереження, зчитування даних з підключених датчиків і керування смарт-картами для контролю доступу. Аварійні сигнали й дані датчиків з декількох пристроїв з лінійки NetBotz можуть централізовано оброблятися в хмарної DCIM-платформі EcoStruxure IT.

Raritan демонструє сенсорну технологію для моніторингу центрів обробки даних

Компанія Raritan представила продукти наступного покоління для моніторингу центрів обробки даних. Рішення з лінійки SmartSensor покликані спростити інженерами одержання інформації середовищу усередині ЦОД у режимі реального часу. Пристрою допомагають швидко визначати «гарячі точки», оптимально прохолоджувати IT-устаткування й запобігати дорогі простої ЦОД.

Інтелектуальні датчики дозволяють контролювати температуру, вологість, напрямок повітряних потоків і тиск повітря, а також виявляти витоку води, вібрації й несанкціоновані відкриття дверцят серверної стійки.

Крім того, пристрої здатні попереджати операторів ЦОД про потенційні ризики. Датчики з лінійки SmartSensor можна легко підключити до існуючої інфраструктури живлення усередині стійки, що допомагає оптимізувати витрати й прискорити розгортання.

Постачальник систем аналітичного моніторингу для дата-центрів Romonet куплений CBRE

Компанія Romonet, що спеціалізується на рішеннях для моніторингу й оптимізації інфраструктури центрів обробки даних, була придбана фірмою CBRE, що займається наданням послуг у сфері комерційної нерухомості й інвестицій. Ціна угоди не розголошується.

Компанія Romonet розробляє хмарне програмне забезпечення для моделювання й аналітики, що дозволяє замовникам знизити енергоспоживання свого центра обробки даних і підвищити надійність ЦОД. Цей продукт може виявитися досить корисним для фірми CBRE, що управляє більш ніж 8 сотнями ЦОД по усьому світі.

Sunbird Software виводить на ринок нову версію DCIM-рішення dcTrack

Розроблювач інноваційних програмних інструментів для операторів центрів обробки даних Sunbird Software оголосив про доступність оновленої платформи для керування інфраструктурою серверної ферми (DCIM). Мова про рішення dcTrack, що обновилося до версії з порядковим номером 6.3.

Новий реліз може похвастати крім іншого наявністю вдосконаленої панелі бізнес-аналітики, що забезпечує цілісне інтегроване подання про інфраструктуру центра обробки даних.

Був поліпшений механізм інтеграції dcTrack 6.3 із хмарою й CMDB-платформами, щоб спростити об'єднання декількох систем для більше ефективного керування ресурсами дата-центра й планування періодичних відновлень, щоб підтримувати базу даних dcTrack в актуальному стані.

У новій версії також з'явилася можливість спільного використання моніторингових панелей, що налаштовуються. Система дозволяє створити необмежену кількість персоналізованих інформаційних панелей, посиланнями на які користувачі можуть ділитися з іншими членами своєї команди й керівництвом.

Fujitsu автоматизує весь життєвий цикл розробки програмного забезпечення за допомогою нової платформи розробки програмного забезпечення на основі штучного інтелекту

Платформа буде використана в модифікаціях усіх 67 програмних пакетів, що надаються клієнтам медичної та урядової галузі до кінця 2026 фінансового року.

Компанія Fujitsu Limited оголосила про розробку та запуск своєї платформи розробки програмного забезпечення на основі штучного інтелекту (AI-Driven Software Development Platform), нової ініціативи, спрямованої на перенесення розробки програмного забезпечення в епоху штучного інтелекту та сприяння сталому зростанню її клієнтів і суспільства. Ця платформа автоматизує весь процес розробки програмного забезпечення, від визначення та проектування вимог до впровадження та інтеграційного тестування. Використовуючи модель великих мов Takane (LLM) [1] та агентну технологію штучного інтелекту для масштабної розробки програмного забезпечення, розроблену Fujitsu Research, платформа розробки програмного забезпечення на основі штучного інтелекту дозволяє агентам штучного інтелекту розуміти складні, що розвиваються, великомасштабні системи, що належать підприємствам та державним організаціям. Платформа має на меті використовувати кілька агентів штучного інтелекту, які спільно виконують кожен етап розробки програмного забезпечення, досягаючи повної автоматизації всього процесу без втручання людини.

Fujitsu прагне використовувати цю платформу розробки програмного забезпечення на основі штучного інтелекту для внесення змін до всіх 67 типів медичних та урядових бізнес-програмних продуктів, що надаються Fujitsu Japan Limited, до кінця 2026 фінансового року. Зміни необхідні через законодавчі та регуляторні зміни. З січня 2026 року платформа використовується в Японії для модифікацій програмного забезпечення, що стали необхідними через перегляд медичних зборів 2026 року [2]. У документі про поєднання (PoC), який оновлював програмне забезпечення відповідно до перегляду медичних тарифів 2024 року, платформа продемонструвала значне скорочення часу розробки для одного з приблизно 300 запитів на зміни. Використовуючи традиційні методи розробки програмного забезпечення [3], модифікації зайняли б три людино-місяці. Завдяки цій технології цей час було значно скорочено до чотирьох годин, що дозволило досягти 100-кратного збільшення продуктивності.

Використовуючи цю платформу розробки на основі штучного інтелекту, Fujitsu значно покращить швидкість модифікацій програмного забезпечення, необхідних для законодавчих змін та змін у системі. Це значно зменшить навантаження, яке раніше було потрібно для перевірки системи під час модифікацій, тим самим звільняючи час для планування та розробки заходів і послуг, що призводять до покращення обслуговування пацієнтів, мешканців та клієнтів, а також дозволяючи клієнтам і партнерам постійно адаптуватися до постійно мінливих операцій та суспільних потреб. Крім того, Fujitsu передбачає, що ця технологія також допоможе клієнтам і партнерам реагувати на зростаючий попит на IT та пом'якшувати дефіцит IT-фахівців.

У розробці на основі штучного інтелекту Fujitsu позиціонує AI-Ready Engineering – процес підготовки ресурсів та знань для забезпечення правильного розуміння ШІ існуючих систем та досягнення високонадійної автоматизації – як ключовий. Завдяки спільній роботі AI-Ready Engineering та платформи розробки програмного забезпечення на основі ШІ, Fujitsu прискорить розробку програмного забезпечення на основі ШІ. Fujitsu сприятиме трансформації стилів роботи інженерів, зміцнюючи свій штат Forward Deployed Engineer (FDE) та змінюючи парадигму розробки програмного забезпечення від традиційного підходу, заснованого на людино-місяцях, до підходу, заснованого на цінності для клієнта.

У майбутньому Fujitsu планує розширити застосування платформи розробки програмного забезпечення на основі штучного інтелекту (AI-Driven Software Development Platform) на широкий спектр секторів, включаючи фінанси, виробництво, роздрібну торгівлю та державні послуги, до кінця 2026 фінансового року. Fujitsu також почне пропонувати цю послугу клієнтам та компаніям-партнерам, щоб вони могли швидко та гнучко розробляти системи, що адаптуються до змін у їхньому бізнес-середовищі. Завдяки цим зусиллям Fujitsu прагне перетворити процес розробки програмного забезпечення на модель, керовану штучним інтелектом, як галузевий стандарт.

Підтвердження від організацій та компаній

IDC прогнозує, що з 2026 року прискорення використання бізнесу на основі ШІ/агентів та модернізація існуючих систем будуть ключовими рушійними силами трансформації на японському IT-ринку. Заява Fujitsu спрямована на переосмислення складних застарілих системних активів у стан, коли ШІ може точно їх розуміти та обробляти, а також на автоматизацію всього процесу каскадної розробки. Очікується, що ця ініціатива забезпечить практичний шлях для багатьох вітчизняних підприємств, які стикаються з постійною проблемою підтримки та експлуатації застарілих активів, а також сприятиме переходу в розробці програмного забезпечення від трудомісткої моделі.

Я глибоко вражений концепцією автоматизації всього процесу розробки програмного забезпечення від початкового до кінцевого етапу за допомогою штучного інтелекту, і навіть довіри процесу верифікації штучному інтелекту. Це перевертає традиційне припущення, що людські перевірки зрештою незамінні, і я бачу великий потенціал, особливо в орієнтації на бізнес-пакети, які щороку зазнають складних системних змін. Наша компанія також багато років займається модифікацією бізнес-пакетів, і як якісно завершити системні редагування за короткий період завжди було серйозною проблемою. Ми вважаємо, що знання та досвід, накопичені під час цього процесу, можуть суттєво сприяти реалізації та розвитку цієї концепції. Наша компанія продовжуватиме робити свій внесок у розширення бізнесу Fujitsu та Fujitsu Japan завдяки постійній співпраці, не обмежуючись цим проектом.

Ця ініціатива з автоматизації штучного інтелекту, яку просуває Fujitsu, спрямована не лише на підвищення ефективності розробки; ми усвідомлюємо, що передача та розвиток обширних бізнес-знань та філософії дизайну, що культивувалися компаніями протягом багатьох років, є значним викликом для наступного покоління. Зокрема, концепція надання комплексної підтримки, від визначення вимог до проектування, впровадження та забезпечення якості, що

викликається змінами в законах і правилах, відкриває нові можливості в сферах, які традиційно спиралися на людський досвід і неявні знання. Ми бачимо велике значення у функціонуванні штучного інтелекту як основи, яка підтримує людське судження та креативність, а не замінює їх». У виробничій галузі такі проблеми, як зміни дизайну, дотримання нормативних вимог та розуміння масштабів впливу, з кожним роком стають дедалі складнішими. Підхід Fujitsu до просування як стандартизації знань, так і використання штучного інтелекту в цих сферах пропонує цінні знання для підвищення продуктивності та конкурентоспроможності всієї галузі. Kawasaki Heavy Industries щиро сподівається, що ця ініціатива стане вирішальним кроком у просуванні трансформації японського виробництва та широкого кола інших галузей промисловості, і ми всіляко підтримуємо її подальший розвиток.

Системи стають дедалі складнішими протягом багатьох років експлуатації та часто потребують значних зусиль з обслуговування. Хоча впровадження генеративного штучного інтелекту підвищило ефективність аудиту, його точність залишається недостатньою для надійного практичного застосування. У цій ситуації ми покладаємо великі сподівання на «багаторівневий контроль якості», який автоматично виправляє неоднозначності та упущення. Ми впевнені, що цей механізм, де сам штучний інтелект перевіряє якість та автономно повторює процеси, значно підвищить надійність розробки системи. Ми з нетерпінням чекаємо на його подальший розвиток.

Завдання модернізації існуючих систем вже давно є значним для інженерів. Платформа розробки програмного забезпечення на основі штучного інтелекту від Fujitsu має потенціал докорінно трансформувати працю, яка раніше була пов’язана з розумінням складних законів і нормативних актів, аналізом величезних історичних ресурсів і опануванням неявних знань у цій галузі. Особливої уваги заслуговує здатність штучного інтелекту автономно вивчати «людський інтелект», тим самим значно підвищуючи точність визначення вимог. Крім того, він може виконувати все: від аналізу структури програми та

стандартизації до фази широкого тестування з неймовірною швидкістю та повнотою. Це дозволяє створювати високоякісні продукти за короткий період. Оскільки роль штучного інтелекту розширюється та звільняє людей від рутинних завдань, інженери можуть зосередитися на більш творчій діяльності. Я маю великі очікування щодо зміни парадигми в оновленні систем, яку принесе це рішення.

Ця ініціатива щодо досягнення комплексної автоматизації в одному місці, від визначення вимог до валідації системи, є новаторською інновацією для галузі. Технологія дозволяє штучному інтелекту точно розуміти величезні ресурси, включаючи давно встановлені програми та проектну документацію, і ми раді потенціалу як для якості виробничого рівня, так і для виняткового підвищення продуктивності. Ми впевнені, що ця платформа стане новим стандартом розробки та прискорить цифрову трансформацію наших клієнтів. Ми залишаємося відданими співпраці з Fujitsu для вирішення соціальних проблем за допомогою штучного інтелекту.

Я вважаю, що система розробки на основі штучного інтелекту Fujitsu Limited має потенціал стати «новою парадигмою розробки систем». Цього неможливо досягти, просто вводячи існуючий код або інформацію про проектування в ШІ, і хоча існують різні перешкоди, такі як перетворення документації в Markdown та створення тестових середовищ, подолання цих перешкод може призвести до вирішення традиційних проблем розробки систем (таких як QCD). Що особливо привернуло мою увагу, так це не лише покращення в процесі розробки, а й те, що відбувається після генеративного ШІ, тобто включення детальних специфікацій та коду (логіки). Я бачу тут величезний потенціал як рішення найбільшої проблеми: візуалізація та передача неявних знань досвідчених інженерів-програмістів, яких традиційно бракує в документації. Ми очікуємо, що генеративний ШІ діятиме як радник для менш досвідчених інженерів-програмістів, легко відповідаючи на запитання в будь-який час, тим самим значно просуваючи передачу ноу-хау наступному

поколінню. З іншого боку, цей механізм також має потенціал різко змінити традиційну бізнес-модель системного інтелекту, і ми уважно стежимо за майбутнім розвитком подій.

Ця ініціатива щодо повної автоматизації розробки та обслуговування додатків є надзвичайно цінною трансформацією для нашої компанії. Вона формалізує та встановлює відтворюваний процес для завдань, які довгий час спиралися на неявні знання та досвід окремих співробітників, і ми маємо на неї великі очікування. Зокрема, вона має потенціал значно покращити якість обслуговування застарілих систем та втрачені можливості через затримку реагування на зміни. Крім того, здатність штучного інтелекту виконувати аналіз першопричин та виявляти необхідну додаткову інформацію, що розвивається, є важливим кроком до передових та ефективних системних операцій, що може змінити саму природу розробки систем. Ми поділяємо прагнення Fujitsu підвищити продуктивність у всій галузі та встановити нові стандарти розробки. Ми з нетерпінням чекаємо на її подальше активне просування як ініціативи, яка сприятиме розвитку всієї галузі.

Центральна лікарня префектури Сімане: «Платформа розробки програмного забезпечення на основі штучного інтелекту, представлена Fujitsu, пропонує практичний та надійний підхід до вирішення давніх проблем, з якими стикаються медичні установи: зростаюча складність розрахунку медичних гонорарів та зростаюче навантаження на обробку претензій. Механізм, за допомогою якого штучний інтелект аналізує юридичні документи та витягує відповідні області, чітко виділяючи моменти, відкриті для інтерпретації, щоб доповнити людське судження, особливо вражає. Ця конструкція демонструє глибоке розуміння операцій на місці та заслуговує на високу похвалу. Крім того, специфічна для Японії LLM та врахування безпеки є невід'ємними елементами використання штучного інтелекту в медичній сфері. Окрім претензій щодо медичних гонорарів, ця технологія має потенціал для інтеграції з суміжними галузями, такими як управління ліжками та розуміння вимог до продуктивності,

що зробить вагомий внесок у загальну операційну ефективність лікарні в майбутньому. Це перспективна ініціатива, яка заслуговує на позитивне розгляд для впровадження, щоб полегшити навантаження на медичних працівників.

Ми очікуємо, що платформа розробки програмного забезпечення на основі штучного інтелекту від Fujitsu стане ініціативою з потенціалом докорінно змінити сам процес розробки програмного забезпечення. Якщо вдасться реалізувати світ, де все, від визначення вимог до проектування, кодування та тестування, може виконуватися автоматично безперебійно та в одному місці, можна буде досягти як значного покращення швидкості, так і якості розробки». З 2024 року ми співпрацюємо з Fujitsu в деяких сферах цієї галузі. Завдяки цим ініціативам ми впевнені, що весь процес розробки буде повністю автоматизований найближчим часом. Реалізація цієї трансформації значно розширить можливості для послуг, які ми можемо надавати нашим клієнтам. Ми щодня обмірковуємо ідеї для нових послуг для наших клієнтів. Це дозволить нам швидко опрацьовувати ці ідеї та швидко надавати їх нашим клієнтам. Ми сподіваємося, що цей новий світ створення цінності з'явиться якомога швидше.

Ми твердо віримо, що оголошення Fujitsu знаменує собою значний крок вперед в еволюції розробки систем у Японії. Воно тісно узгоджується з баченням IBM Japan і являє собою важливу ініціативу, яка допоможе сформувати майбутнє галузі в цілому. Ми з нетерпінням чекаємо на спільну підтримку цього імпульсу та внесок у створення більш надійної та динамічної екосистеми.

Ми щиро вітаємо оголошення Fujitsu Limited про платформу розробки програмного забезпечення на основі штучного інтелекту (AI-Driven Software Development Platform) як новаторську ініціативу, яка відкриває новий розділ у розробці систем для ери штучного інтелекту. Завдяки об’єднанню кількох агентів штучного інтелекту для автоматизації життєвого циклу розробки від початку до кінця – від визначення вимог до постійного вдосконалення – та інтеграції забезпечення якості, керованого людиною, ця платформа втілює нову інженерну модель, в якій люди та штучний інтелект справді працюють разом. Ми вважаємо

цю ініціативу надзвичайно важливою, оскільки вона безпосередньо вирішує критичні проблеми, з якими стикається японська індустрія розробки систем, включаючи гостру нестачу талантів та зростаючу складність і витонченість сучасних систем. Ми твердо очікуємо, що ці сміливі зусилля сприятимуть розвитку японського бізнесу розробки систем і переростуть у модель трансформації з глобальним значенням. У майбутньому ми продовжуватимемо тісно співпрацювати з Fujitsu, поєднуючи сильні сторони обох компаній, щоб рішуче підтримувати наших клієнтів на їхньому шляху до становлення передовими фірмами.

Ми вважаємо, що оголошена цього разу комплексна автоматизована розробка систем на основі штучного інтелекту являє собою не лише значне підвищення продуктивності, але й сміливий виклик для фундаментальної трансформації способу надання корпоративних IT-інфраструктури. Дозволяючи ШІ точно розуміти часто оновлювані правила та складні бізнес-знання, включаючи неявні ноу-хау, цей підхід автономно та безперешкодно виконує процеси від визначення вимог до модифікації системи. Він має потенціал для ефективного вирішення основних проблем, що виникають через застарілі системи, з якими стикаються багато японських підприємств. Ми з нетерпінням чекаємо на спільне вдосконалення цієї технології через практичний досвід та розвиток спільної творчості, використовуючи досвід Panasonic та Fujitsu на місцях, очікуючи, що вона стане новим стандартом розробки систем і буде широко впроваджена не лише в Panasonic, але й у всьому суспільстві.

Ми очікуємо, що ця ініціатива призведе до нової трансформації в розробці систем. Ця трансформація буде зумовлена застосуванням передових можливостей обробки японської мови, таких як розуміння юридичних документів, до різноманітних завдань, надійним виконанням кожного процесу за допомогою функцій аудиту якості та розширенням цих можливостей для розробки з нуля. Крім того, ми вважаємо, що AI Ready Engineering, формалізуючи експертні ноу-хау та знання предметної області в явні знання та перетворюючи їх на активи, що

використовуються ШІ, значною мірою сприятиме спадкоємності досвіду від дедалі обмеженішого кола кваліфікованих фахівців. Ми щиро сподіваємося, що спільна творчість між ШІ, що успадковує знання, та персоналом на місці створить нову цінність і стане наріжним каменем для інновацій у галузі розробки систем, та й взагалі у всіх галузях.

Аналіз ринку програмно-визначених центрів обробки даних з 2026 по 2035 рік

Глобальний ринок програмно-визначених центрів обробки даних оцінюється в 82,77 млрд доларів США у 2025 році та, як очікується, досягне приблизно 602,27 млрд доларів США до 2035 року, зростаючи зі середньорічним темпом зростання 21,95% з 2026 по 2035 рік.

Огляд ринку

За деякими суттєвими винятками, програмно-визначені центри обробки даних (SDDC) відрізняються від типових центрів обробки даних тим, що вони використовують віртуалізацію, об'єднання ресурсів, абстракцію та автоматизацію. Типовий центр обробки даних – це місце, де інфраструктура, мережі та організаційні дані централізовані та доступні. Він служить центральним центром для IT-операцій та серверів фізичної інфраструктури, пристроїв зберігання даних, мережевих компонентів та пристроїв безпеки. Він забезпечує програмний підхід до роботи звичайного центру обробки даних завдяки використанню віртуалізованого середовища. Подібно до давно усталених методів віртуалізації серверів, вони підвищують гнучкість бізнесу. Хмарні хост-сервери SDDC дозволяють створювати та налаштовувати центри обробки даних відповідно до вимог без необхідності фізичного розміщення або налаштування мережевого обладнання.

Фактори зростання ринку програмно-визначених центрів обробки даних

Очікується, що впровадження програмно-визначених центрів обробки даних (SDDC) буде зумовлене зниженням вартості обладнання. Компаніям, що

використовують програмно-визначені центри обробки даних (SDDC), не потрібно інвестувати в обладнання, вивчати навички, специфічні для певних постачальників, для його експлуатації та обслуговування або займатися модернізацією фізичних машин. Це покращує управління ресурсами всередині бізнесу, підвищуючи загальну ефективність.

Крім того, зростає популярність програмно-визначених центрів обробки даних (ПВЦД), щоб гарантувати кращий контроль та управління як програмним, так і апаратним забезпеченням. З іншого боку, підприємства можуть перебільшувати труднощі, пов'язані з використанням застарілих платформ та наглядом за сучасною IT-інфраструктурою. Банківські, фінансові та страхові компанії та корпорації BFSI у США швидко встановлюють програмно-визначені центри обробки даних у відповідь на зростаючий попит своїх споживачів та клієнтів на безпечні та швидкі транзакції. Для підприємств BFSI це пропонує ефективне відновлення даних, покращені рішення для безпеки та мобільності. Знижуючи затримку мережі, це спрощує високочастотну торгівлю та безпечні фінансові транзакції. Крім того, це допомагає підприємствам дотримуватися обтяжливих урядових правил, що вимагають більш гнучких та адаптивних центрів обробки даних. Щоб запобігти потенційним сліпим зонам безпеки, SDDC використовує як фізичні, так і віртуальні брандмауери. Потреба в рішеннях SDDC зумовлена зростанням організацій BFSI та нових фінтех-стартапів у США.

Одним з головних факторів, що стимулюють ринок програмно-визначених центрів обробки даних, є швидкозмінний IT-ландшафт. Організації використовують програмно-визначені центри обробки даних, щоб зменшити складність IT-інфраструктури та підвищити гнучкість IT. Забезпечуючи виділення ресурсів на основі політик та об'єднані ресурси інфраструктури, це сприятиме стандартизації інструментів управління на всіх рівнях інфраструктури та забезпечить швидше реагування на нові запити на IT-ресурси. Крім того, це дозволяє IT-менеджерам у компаніях-кінцевих користувачах заощаджувати операційні витрати, прокладати курс модернізації кінцевого використання та

підтримувати контроль над виділенням ресурсів. Це полегшує програмам самостійне виділення IT-ресурсів, а коли розгорнуто правильне програмне забезпечення, іноді це навіть дозволяє повністю усунути роботу адміністратора. Ці елементи збільшують розмір ринку.

Перспективи ринку:

– Огляд зростання галузі: Ринок програмно-визначених центрів обробки даних прискорюється, оскільки організації приділяють більше уваги гнучкості, економії коштів та автоматизації своєї IT-інфраструктури. Постійне зростання хмарних середовищ та використання гібридних IT-моделей стимулюють значне зростання розгортання програмно-визначених центрів обробки даних у всьому світі на рівні підприємства.

– Тенденції сталого розвитку: Рішення SDDC дозволяють підприємствам керувати робочими навантаженнями, використовуючи енергоефективні методи, оптимізувати використання серверів та зменшувати залежність від фізичного обладнання. Підприємства часто користуються можливістю використовувати програмні методи розподілу ресурсів для зменшення викидів вуглецю та досягнення своїх цілей сталого розвитку, пов'язаних з ESG, для центрів обробки даних.

– Глобальна експансія: У всіх регіонах організації продовжують впроваджувати рішення SDDC для побудови стандартизованої IT-інфраструктури. Постійне поширення рішень хмарних обчислень через кордони, а також постійне розширення центрів обробки даних у всьому світі, продовжує сприяти зростанню рівня проникнення SDDC у всьому світі.

– Екосистема стартапів: Багато стартапів розробляють інноваційні продукти, що забезпечують оркестрацію, безпеку контейнерів та управління робочими навантаженнями за допомогою автоматизації за допомогою штучного інтелекту. Багато з цих стартапів створюють легкі інструменти SDDC, які легко підключаються до гіпермасштабних платформ та гібридних рішень, і ці стартапи фінансуються венчурними капіталістами.

Охоплення ринку

Залежно від компонента, сегмент керованих послуг є домінуючим гравцем і, як очікується, матиме найбільший вплив на ринок. Підвищений акцент, який компанії приділяють досягненню оптимізованого управління робочим навантаженням зі збільшенням гнучкості, швидкості та безпеки без необхідності мати справу з труднощами навчання, інтеграції та розгортання, підтримки та обслуговування, є основною причиною зростання ринку керованих послуг. Попит на послуги, що виникає внаслідок використання програмно-визначених послуг, є основною причиною їх розширення в бізнесі.

Через зростання використання апаратного та програмного забезпечення очікується, що рішення SDDC займатимуть найбільшу частку ринку протягом прогнозованого періоду. Для реалізації рішень можна використовувати різні апаратні платформи, такі як конвергентна інфраструктура, гіперконвергентна інфраструктура, стандартне обладнання та налаштовувана інфраструктура. Крім того, це дозволяє програмам запускатися лише на логічно визначених ресурсах, відокремлених від базового обладнання.

Протягом прогнозованого періоду сегмент програмно-визначених обчислень, як очікується, буде найбільшим на ринку програмно-визначених центрів обробки даних. Завдяки впровадженню хмарних застосунків, SDC наразі має більшу частку ринку. Хмарні застосунки використовують принципи хмарних обчислень для зниження ризиків розгортання та підвищення гнучкості, продуктивності та якості послуг. Локальні та хмарні платформи підтримуються хмарними застосунками. Завдяки своїм відмінним особливостям, таким як спрощені та автоматизовані операції центру обробки даних, вони здобули популярність серед підприємств. Вони надаються кінцевим користувачам на умовах оренди для зберігання їхньої критично важливої інформації, до якої можна отримати доступ віддалено.

Очікується, що ринок програмно-визначених мереж SDN розвиватиметься найшвидшими темпами протягом прогнозованого періоду. Ринок розширюється

через зростання попиту на SDN для покращення мобільності підприємств, віртуалізації серверів та консолідації центрів обробки даних. Використання BYOD (принеси власні пристрої) зростає, і в результаті підприємства все більше покладаються на SDN-рішення для управління доступом до даних співробітників та оптимізації мобільної робочої сили.

Аналітика галузі

Очікується, що протягом прогнозованого періоду галузь IT та телекомунікацій матиме найбільшу частку ринку та найвищий середньорічний темп зростання (CAGR) завдяки своїй важливості на ринку. Через розширення великих даних та зростаючий попит на програмне забезпечення для зберігання даних та послуги, BFSI та уряд також, за прогнозами, займатимуть значну частину ринку.

Організації BFSI впроваджують різноманітні хмарні технології через значні проблеми, пов'язані зі зберіганням, відновленням та кібербезпекою даних, з якими вони повинні справлятися, а також обробкою величезних обсягів даних, які необхідно обробляти, зберігати та реплікувати неодноразово. Постійна вимога до масштабованості та безпеки даних у гібридних хмарних середовищах зумовлює попит на SDDC для автоматизації та управління IT-сервісами в гетерогенних хмарах.

Регіональні аналітичні дані

Розмір ринку програмно-визначених центрів обробки даних у США оцінюється в 25,49 мільярда доларів США у 2025 році та, за прогнозами, становитиме близько 188,53 мільярда доларів США до 2035 року, зростаючи зі середньорічним темпом зростання 22,15% з 2026 по 2035 рік.

Найбільша частка ринку та домінуюче становище в галузі програмно-визначених центрів обробки даних належить Північній Америці. Цей регіон включає США та Канаду. Присутні тут великі фірми, такі як VMware, Inc., Cisco Systems, Inc., IBM Corporation та інші, що сприяло розширенню. Крім того, очікується, що розширення глобального охоплення великих корпорацій шляхом

запуску продуктів та альянсів стимулюватиме розширення ринку в Північній Америці. На міжнародному ринку програмно-визначених центрів обробки даних США займають найбільшу частку. Прогнози свідчать про те, що Північна Америка продовжуватиме домінувати на ринку програмно-визначених центрів обробки даних протягом деякого часу.

Європа вважається другим за величиною регіоном на ринку програмно-визначених центрів обробки даних. Потреба в центрах обробки даних зумовлена сприятливими економічними умовами регіону, такими як податкові пільги, достатні запаси відновлюваної енергії та належні правила конфіденційності та захисту даних. Завдяки сприятливому законодавству щодо захисту даних та конфіденційності, європейські компанії швидко впроваджують програмно-визначені центри обробки даних. Крім того, скандинавські країни пропонують податкові пільги учасникам ринку центрів обробки даних, що збільшує попит на системи програмно-визначених центрів обробки даних. Щоб зміцнити свій глобальний охоплення, гравці ринку розширюють свій бізнес програмно-визначених центрів обробки даних у Європі.

Очікується, що Азіатсько-Тихоокеанський регіон зростатиме найшвидшими темпами протягом прогнозованого періоду. Регіон зазнав значного розширення в результаті зростання індустріалізації та розвитку ринків, що розвиваються. Основна IT-інфраструктура регіону Азіатсько-Тихоокеанського регіону розвивається швидше, що створює самовідтворюваний цикл, який заохочує додаткові інвестиції та розширення. Як наслідок, зростатиме потреба у збільшенні потужностей, ресурсів та відповідальності в управлінні IT. У цьому регіоні є кваліфіковані програмісти, які мають щонайменше таку ж кваліфікацію, як і в інших частинах світу. Інфраструктура, яку пропонують Китай та Індія, також є корисною. Сучасна телекомунікаційна інфраструктура забезпечує швидке спілкування з країнами всього світу. Широке використання англійської мови є ще однією значною перевагою. Англійська мова є єдиною мовою, що використовується у вищій освіті.

Впровадження програмно-визначених центрів обробки даних (SDDC) у Латинській Америці зростає відповідно до модернізації застарілих центрів обробки даних підприємств. Урядові ініціативи щодо цифрової трансформації, підвищена доступність хмарних сервісів та потреба в масштабованих IT-послугах стимулюють зростання в Латинській Америці, прокладаючи шлях для впровадження моделей SDDC підприємствами.

MEA: Центри обробки даних, що працюють на «розумних» технологіях

Агентство з питань економічного розвитку та інфраструктури (MEA) значно інвестує у створення рішень для «розумного» міста та впровадження гіпермасштабних центрів обробки даних. Технологія програмно-визначених центрів обробки даних (SDDC) забезпечує гнучкий метод розгортання інфраструктури для підтримки національних стратегій цифрової трансформації та підвищення операційної ефективності підприємств, що працюють у BFSI та державному секторі.

Аналіз ланцюжка створення вартості ринку програмно-визначених центрів обробки даних:

– Рівень розробки основних технологій та віртуалізації: ця частина ланцюжка включає гіпервізори, контролери SDN та програмне забезпечення для віртуалізації сховищ, що дозволяє користувачеві абстрагувати свої апаратні ресурси. Постійні дослідження та розробки, захист інтелектуальної власності та тестування сумісності є життєво важливими компонентами цього етапу. Ключові постачальники: VMware, Microsoft, Nutanix, Red Hat Systems.

– Послуги інтеграції та оркестрації: Інтегратори та постачальники об'єднують обчислювальні, сховищні, мережеві та безпекові ресурси в єдину інтегровану платформу SDDC для підприємств. Ця частина ланцюга зосереджена на автоматизації робочих навантажень, сумісності з кількома хмарами та управлінні життєвим циклом корпоративних систем. Ключові постачальники: IBM, HPE, Cisco Systems та Dell Technologies.

– Керовані послуги та оптимізація: ця частина ланцюга надає послуги з міграції в хмару, моніторингу систем на відповідність вимогам, оптимізації продуктивності та регулярного доходу через довгострокові контракти та керовані послуги на основі підписки. Ключові постачальники: Accenture, TCS, Infosys, Capgemini

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – це об’єктно-орієнтована мова програмування високого рівня загального призначення з відкритим кодом. Це визначення може бути важким для новачків, тому розглянемо кожну характеристику окремо, щоб зрозуміти, що вона означає:

– Відкритий вихідний код: це безкоштовно та доступно для подальших покращень, таких як додавання корисних функцій або виправлення помилок.

– Об’єктно-орієнтована: заснована не на функціях, але в об’єктах з певними атрибутами й методами.

– Високий рівень: зручний для людини, а не для комп’ютера.

– Загальне призначення: можна використовувати для створення будь-яких програм.

Ця мова використовується в будь-якому програмному забезпеченні, про яке ви тільки можете подумати. Ви можете використовувати його для створення веб-сайтів, штучного інтелекту, серверів, програмного забезпечення для бізнесу та багато іншого. Також застосовується в науці про дані, аналізі даних, машинному навчанні, інженерії даних, веб-розробці, розробці програмного забезпечення та інших галузях.

Переваги та недоліки Python

Переваги:

– Її легко читати, вчити та писати. Це мова програмування високого рівня з англійським синтаксисом. Це полегшує читання та розуміння коду. Її дійсно легко зрозуміти і вивчити, тому багато людей рекомендують Python новачкам. Вам потрібно менше рядків коду для виконання того ж завдання в порівнянні з іншими основними мовами, такими як C/C++ та Java.

– Підвищує продуктивність. Це дуже продуктивна мова. Завдяки її простоті розробники можуть зосередитися на розв'язанні проблеми. Їм не потрібно витрачати багато часу на розуміння синтаксису або поведінку мови програмування. Ви пишете менше коду та виконуєте більше завдань.

– Інтерпретована мова. Python мова, що інтерпретується, а це означає, що вона безпосередньо виконує код по рядку. Якщо сталася помилка, вона зупиняє подальше виконання та повідомляє про її виникнення. Вона показує лише одну помилку, навіть якщо у програмі їх кілька. Це спрощує налагодження.

– Динамічно типізована. Python не визначає тип змінної, доки ми не запустимо код. Вона автоматично надає тип даних, коли відбувається процес виконання. Фахівець може не турбуватися про оголошення змінних та типи даних.

– Безкоштовна та з відкритим вихідним кодом. Ця мова постачається під схваленою OSI ліцензією з відкритим вихідним кодом. Це робить його безкоштовним для використання та розповсюдження. Ви можете завантажити вихідний код, змінити його та навіть розповсюджувати свою версію. Це корисно для організацій, які хочуть використати свою версію для розробки.

– Підтримка великих бібліотек. Стандартна бібліотека Python є величезною, ви можете знайти майже всі функції, необхідні для вашого завдання. Таким чином ви не залежите від зовнішніх бібліотек.

– Портативність .У багатьох мовах, таких як C/C++, потрібно змінити свій код, щоб запустити програму на різних платформах. З Python все інакше. Ви тільки пишете один раз і запускаєте її будь-де.

Недоліки:

– Низька швидкість. Вище ми обговорювали, що це інтерпретована мова з динамічною типізацією. Порядкове виконання коду часто призводить до повільного виконання. Динамічна природа Python також є причиною її низької швидкості, оскільки їй доводиться виконувати додаткову роботу при виконанні коду. Тому вона не підходить для цілей, де швидкість важливий аспект проєкту.

– Неефективна для пам’яті. Ця мова програмування використовує великий обсяг пам’яті, це може бути недоліком при створенні програм, коли віддають перевагу оптимізації пам’яті.

– Слабка у мобільних обчисленнях. Python зазвичай використовується у серверному програмуванні. Ми не бачимо – її на стороні клієнта або в мобільних програмах з таких причин: вона не заощаджує пам’ять і має повільну обчислювальну потужність у порівнянні з іншими мовами.

– Доступ до бази даних. Програмувати на цій мові легко, але коли ми взаємодіємо з базою даних, її не вистачає. Рівень доступу до бази даних у Python примітивний та недостатньо розвинений у порівнянні з іншими популярними технологіями.

– Помилки виконання. Це мова з динамічною типізацією, тому тип даних змінної може змінюватись у будь-який час. Змінна, що містить ціле число, у майбутньому може містити рядок, що може призвести до помилок виконання.

Застосування Python:

– Для аналізу даних. Дані стали цінним активом у будь-якій сучасній галузі, і більшість компаній зацікавлені у збиранні, обробці та аналізі релевантних даних, щоб витягти з них цінну інформацію для бізнесу. І тут Python виходить за межі будь-якої конкуренції. Python особливо цінна тим, що крім великої стандартної бібліотеки надає величезний набір додаткових модулів, розроблених спеціально для аналітичних цілей. Найвідоміші бібліотеки Python для аналізу даних – це pandas і NumPy . Ці інструменти дозволяють робити з

вашими даними майже все, наприклад, очищати і аналізувати їх, вивчати статистику або візуалізувати приховані тенденції у ваших даних.

– Для візуалізації даних. Візуалізація даних – це окрема частина аналізу даних, яка допомагає нам подавати інформацію, необроблену чи очищену, у більш змістовній формі. Тут Python знову входить у гру, пропонуючи широкий спектр інструментів візуалізації даних. Найпопулярніші з них – matplotlib і заснований на ній seaborn. Використовуючи їх, ми можемо створювати буквально всі види візуалізації: від найпростіших до складніших.

– Для машинного навчання. Машинне навчання (ML) є основою більшості завдань науки даних. Він є областю штучного інтелекту, пов'язаною з використанням алгоритмів, що дозволяють машинам вивчати закономірності та тенденції на основі історичних даних, щоб робити прогнози на основі невідомих даних. – Використовуючи методи ML, ми можемо створювати моделі, які можуть точно передбачити швидкість відтоку клієнтів компанії, оцінити ризик виникнення у людини певного захворювання, визначити оптимальне розташування автомобілів таксі й т.д. За допомогою Python ми можемо побудувати модель ML, використовуючи лише три рядки коду.

– Для розробки програмного забезпечення. Крім свого багатостороннього застосування в галузях науки про дані, Python використовується на кожному етапі розробки програмного забезпечення, включаючи контроль складання, автоматичну безперервну компіляцію, прототипування, відстеження помилок, тестування та обслуговування програмного забезпечення. За допомогою цієї мови можемо створювати аудіо- або відеопрограми на основі методів штучного інтелекту, машинного навчання, API (інтерфейсів прикладного програмування), GUI (графічних інтерфейсів) або будь-якого іншого типу програмного забезпечення.

– Для веброзробки. У той час як для створення візуальної частини вебсайту ми переважно будемо використовувати такі мови, як HTML, CSS та JavaScript, для його невидимої частини ми часто вибираємо Python. Серед

масштабних вебсайтів та програм, створених за допомогою цієї мови, варто згадати Google, Facebook, Instagram, YouTube, Dropbox та Reddit.

– Для автоматизації задач/скриптингу. Це відмінний інструмент для написання програм для автоматизації різних завдань, що повторюються. Цей процес називається скриптингом. Зокрема, можна робити скрипти для роботи з файлами та папками. Наприклад, можна створювати, перейменовувати, перетворювати, розділяти, об'єднувати або видаляти файли, перевіряти їх на наявність помилок. Ви також можете використовувати автоматизацію Python для пошуку та завантаження інформації з Інтернету, заповнення та надсилання онлайн-форм та надсилання регулярних повідомлень або електронних листів.

Яким фахівцям потрібно володіти Python:

– Фахівець з даних.

– Аналітик даних.

– Інженер даних.

– Інженер з машинного навчання.

– Журналіст даних.

– Архітектор даних.

– Повний стек веб-розробника.

– Backend-розробник.

– DevOps-інженер.

– Інженер-програміст.

Можемо зробити висновок, що Python ще довго буде популярною мовою, хоч і має низку недоліків. Цю мову використовують для створення вебсайтів, штучного інтелекту, серверів, програмного забезпечення для бізнесу, аналізу даних, машинного навчання, інженерії даних та для багатьох інших областей. Це перспективна і затребувана навичка, яка необхідна у всіх галузях.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи програмно визначаємого ЦОД на базі технологій Fujitsu.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) 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 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Програмно-визначений центр обробки даних (SDDC) застосовує цілісний підхід до IT-інфраструктури організацій, абстрагуючи кожен елемент центру обробки даних в єдиний хмарний ресурс.

Міграція в хмару забезпечила сучасним підприємствам безпрецедентну гнучкість та масштабованість, але оскільки компанії все більше використовують ці переваги, вони ризикують призвести до ненавмисної фрагментації та складності центру обробки даних.

Щоб вирішити цю проблему, SDDC інтегрує віртуалізовану інфраструктуру, оптимізуючи виділення та управління ресурсами. У разі успішного впровадження це єдина архітектура, яка враховує та координує всі важливі компоненти центру обробки даних.

Традиційний центр обробки даних проти віртуалізації

Традиційно вважалося, що центри обробки даних складаються з трьох основних елементів IT-інфраструктури: обчислювальних ресурсів, сховищ даних та мереж. Колись усі три ці елементи були фізичними ресурсами, часто розташованими географічно, але для деяких організацій це досі так.

Однак переважна більшість підприємств перенесла принаймні частину своєї інфраструктури в хмару, віртуалізуючи той чи інший ресурс заради масштабованості, гнучкості та економічної ефективності. Сьогодні віртуалізовані ресурси можуть мати кілька різних форм:

– **Обчислення:** Будь-який бізнес, який покладається на хмарні обчислювальні ресурси, такі як AWS або Azure, зробив крок до віртуалізації. Це велика група, і неважко зрозуміти, чому. Фізичні сервери повільно розгортаються та потребують ресурсів для обслуговування, і це може перетворити

масштабування на повільну гру вгадайок. Віртуалізовані обчислювальні ресурси можна розгортати (або згортати) за потреби.

– **Зберігання даних:** Більшість із нас сьогодні використовує хмарне сховище даних у особистому житті, і ми бачимо багато його переваг, таких як легкість резервного копіювання та підвищена доступність. Але корпоративні центри обробки даних мають свої власні проблеми, такі як дотримання вимог безпеки та нормативних актів у всій своїй інфраструктурі. Віртуалізація сховища даних може оптимізувати та стандартизувати цей процес.

– **Мережева взаємодія:** Тут ми починаємо бачити синергію, яка виникає внаслідок інтеграції хмарних ресурсів. Як тільки організація використовує віртуальні машини, вона може спростити мережеве об'єднання між ними та іншими машинами за допомогою віртуалізованого мережевого рівня – таким чином, виділення мережевих ресурсів може відбуватися так само швидко, як і виділення самих машин, і все це без турботи про власне обладнання.

Зі зростанням впровадження хмарних технологій стало зрозуміло, що весь центр обробки даних підприємства може бути перенесений у хмару, що надасть величезні переваги бізнесу. Але з цими можливостями приходять і виклики, і SDDC пропонує модель для їх вирішення.

Переваги використання програмно-визначеного центру обробки даних (SDCC)

Загалом, переваги використання програмно-визначеного центру обробки даних схожі на переваги віртуалізації певного ресурсу, поєднані та оптимізовані завдяки продуманій координації ресурсів.

– **Вартість:** На кожному рівні вашого центру обробки даних обслуговування спрощується та стандартизується, що знижує витрати та звільняє команди розробників та операцій для виконання важливіших завдань. Для менших організацій SDDC робить IT-інфраструктуру, яка раніше була б недоступною, доступною.

– **Енергія:** Віртуалізований центр обробки даних використовує лише ту енергію, яка йому потрібна в будь-який момент часу, що різко контрастує з фізичним центром обробки даних з негнучкими базовими показниками та неоптимальним використанням обладнання. Зменшуючи фізичну площу центру обробки даних, підприємства одночасно зменшують фінансові та екологічні витрати.

– **Простір:** Оскільки організації в різних секторах працюють над тим, щоб зрозуміти, які фізичні ресурси дійсно необхідні, центр обробки даних стає ключовим ресурсом, який вже підходить для розподіленого та віртуалізованого підходу до роботи. Усі складнощі управління фізичною власністю усуваються, абстрагуючи центр обробки даних до функціональності, яка дійсно важлива.

– **Безпека:** Незалежно від того, чи належить ваше підприємство до високорегульованого сектору, чи ні, безпека є важливою, і SDDC може застосовувати єдині стандарти до кожного елемента вашої інфраструктури, не вимагаючи від вашої операційної команди постійного оновлення. Хоча підтримка безпечної хмарної інфраструктури вимагає ретельного підходу до прийняття рішень та вибору постачальників послуг, вона зрештою може посилити безпеку та спростити дотримання вимог.

Разом ці переваги є переконливими, але вони не позбавлені труднощів. Тож як ці елементи поєднуються між собою, і як організації можуть ефективно створити SDDC?

Проблеми використання програмно-визначеного центру обробки даних (SDCC)

SDDC – це не чарівна паличка; це архітектурна модель, яку потрібно ретельно впроваджувати, щоб досягти бажаних переваг для вашої організації.

Оскільки віртуалізовані ресурси з'являлися (і часто впроваджувалися) по частинах, від різних постачальників послуг, вони часто не розроблені для простої інтеграції та сумісності.

Багато великих підприємств, які розглядають можливість використання

SDDC, використовують комбінацію публічних, приватних та гібридних хмар – від серверів, орієнтованих на клієнта, до середовищ розробки для вільного впровадження інновацій та платформ для застарілих додатків. Це створює складність, але також і можливості; це ресурси, які ви хочете мати змогу плавно координувати з єдиного рівня управління.

Ефективний SDDC – такий, що відповідає обіцянкам концепції – усуне складнощі, мінімізуючи обмеження, дозволяючи вибору вашої команди впливати на форму ваших IT-операцій. На практиці це має означати, що у вас є можливості віддаленого керування через API та прості у використанні програмні інтерфейси.

3.2 Розробка структурної схеми

Архітектура SDDC

Програмно-визначений центр обробки даних об'єднує віртуалізовані обчислення, сховище та мережу під єдиним рівнем управління. По суті, ця архітектура розроблена для того, щоб зруйнувати ізольованість, яка традиційно розділяє ці три стовпи IT-інфраструктури. Натомість кожен з них розглядається як пул ресурсів, що управляються та надаються через централізований програмний інтерфейс або API.

Типова архітектура SDDC включає:

– Віртуальний інфраструктурний рівень: абстрагує та об'єднує ресурси, такі як сервери, диски та мережеві комутатори, щоб їх можна було динамічно розподіляти залежно від потреб.

– Рівень управління: забезпечує єдиний інтерфейс для забезпечення, моніторингу та автоматизації завдань на всіх віртуальних ресурсах, позбавляючи команд необхідності мати справу з окремими власними апаратними засобами керування.

– Механізм безпеки та політик: застосовує узгоджені правила безпеки та відповідності на кожному рівні центру обробки даних, незалежно від того, чи

знаходяться ресурси локально, у публічній хмарі чи в гібридному середовищі.

– Автоматизація та оркестрація: Забезпечує програмне керування інфраструктурою, тому такі процеси, як запуск нових робочих навантажень або скорочення невикористовуваних ресурсів, відбуваються на льоту, керуючись попередньо визначеними політиками або аналітикою в режимі реального часу.

Фактично, архітектура SDDC дозволяє організаціям розглядати кожен рівень центру обробки даних як окреме програмне забезпечення. Такий підхід не лише оптимізує використання ресурсів, але й забезпечує більш оперативну та гнучку роботу, дозволяючи вам коригувати свою інфраструктуру майже в режимі реального часу відповідно до розвитку потреб бізнесу.

Майбутнє програмно-визначених центрів обробки даних

Зі зростанням темпів цифрової трансформації майбутнє SDDC все більше переплітається з контейнеризацією, безсерверними обчисленнями та розгортанням на периферії мережі. Підприємства продовжують досліджувати способи швидшої та ефективнішої доставки додатків, і SDDC розвиваються, щоб задовольнити ці потреби. Ви, ймовірно, побачите глибшу інтеграцію Kubernetes, фреймворків автоматизації, оптимізації інфраструктури на основі штучного інтелекту та більш детального контролю політик, який поширюється аж на периферію мережі.

Крім того, нові концепції, такі як архітектури безпеки Zero Trust, стануть стандартними в багатьох SDDC, що призведе до ще більшої зосередженості на стратегіях глибокого захисту для віртуальних середовищ. Загальна тенденція очевидна: оскільки компанії впроваджують новіші технології та розподіляють свої центри обробки даних по кількох хмарах та локальних ресурсах, SDDC стає об'єднуючою моделлю, яка може постійно адаптуватися, забезпечуючи безперебійну роботу та стабільну безпеку.

Як різні галузі використовують SDCC

Переваги SDDC – гнучкість, масштабованість та централізоване управління – є привабливими для широкого кола секторів, кожен з яких

використовує ці переваги по-різному:

– **Фінанси:** Швидко адаптуйтеся до змін у нормативних актах та вимогах ринку, створюючи безпечні та відповідні середовища для конфіденційних фінансових програм.

– **Охорона здоров'я:** Оптимізуйте управління даними пацієнтів, забезпечте суворе управління даними та підтримуйте робочі навантаження телемедицини та досліджень на спільній, але безпечній віртуальній інфраструктурі.

– **Роздрібна торгівля:** Миттєво реагуйте на сплески онлайн-трафіку під час сезонних акцій або блискавичних розпродажів, збільшуючи або зменшуючи ресурси залежно від вимог, забезпечуючи безперебійний клієнтський досвід.

– **Технологічні компанії та компанії, що займаються розробкою програмного забезпечення як послугою (SaaS):** пришвидшення процесів безперервної інтеграції/безперервного розгортання (CI/CD), що забезпечує швидкі цикли розробки та тестування, що створюють нові середовища за лічені хвилини.

– **Виробництво та логістика:** Підтримка операцій «точно в строк» та аналізу даних шляхом розширення віртуального центру обробки даних на віддалені місця або філії, що забезпечує стабільну продуктивність на всіх об’єктах.

Незалежно від галузі, об'єднуючою рисою є гнучкість. Централізуючи ресурси у віртуалізованому пулі, організації можуть адаптуватися до раптових змін у вимогах до робочого навантаження, підтримувати доступність життєво важливих систем та зменшувати накладні витрати, пов'язані з виділенням та обслуговуванням фізичного обладнання.

Міграція до віртуалізованої архітектури центру обробки даних

Для тих, хто має великі фізичні центри обробки даних, думка про перенесення всіх цих ресурсів у хмару може бути лякаючою. З чого почати? І як переконатися, що ви правильно впроваджуєте свій SDDC, повністю

використовуючи всі можливості моделі?

Хмарне рішення для центру обробки даних, таке як Mirantis OpenStack для Kubernetes, надає набір інструментів для підтримки на кожному етапі процесу. В основі цього рішення лежить безпечна платформа керування контейнерами, яка дозволяє розгортати та масштабувати гібридні мультихмари по всій вашій організації. Використовуйте єдине «скляне» середовище для координації між OpenStack, vSphere та локальною інфраструктурою bare metal, а також публічними хмарами AWS, Azure та GCP, що оптимізує операції та звільняє вашу організацію від залежності від постачальника.

Сервісне обслуговування аутсорсингу центрів обробки даних та керованих послуг гібридної інфраструктури Fujitsu по всьому світу

Аутсорсинг центрів обробки даних та керовані послуги гібридної інфраструктури Fujitsu – це послуга, розроблена для управління та оптимізації середовищ IT-інфраструктури підприємства шляхом поєднання традиційного аутсорсингу центрів обробки даних з гібридним хмарним управлінням. Послуга пропонує централізований нагляд за фізичними та віртуальними серверами, сховищами та мережевими ресурсами, а також підтримує інтеграцію з публічними та приватними хмарними платформами. Вона забезпечує безперервний моніторинг, управління інцидентами та обслуговування систем для підвищення операційної ефективності та зниження ризиків у складних IT-ландшафтах. Послуга вирішує бізнес-завдання, пов'язані з масштабованістю, продуктивністю та відповідністю вимогам, за допомогою автоматизованих процесів та передових практик управління інфраструктурою, що дозволяє організаціям ефективно адаптуватися до технологічних вимог, що постійно змінюються. Аутсорсинг центрів обробки даних та керовані послуги гібридної інфраструктури Fujitsu – це послуга, яка використовує індивідуальну модель ціноутворення, що базується на таких факторах, як обсяг, масштаб та складність керованої інфраструктури, при цьому плата зазвичай структурується як регулярні платежі, що визначаються залежно від використання, рівня обслуговування або

конкретних вимог клієнта.

Рисунок 3.1 – Структурна схема системи

Планування подальших дій: що далі для центрів обробки даних

Оскільки індустрія центрів обробки даних прискорюється у 2026 році, лідери стикаються з ландшафтом, сповненим як складності, так і можливостей. Тенденції, що формують сьогоднішній ринок, – це не просто виклики, які потрібно подолати, а каталізатори переосмислення. Ось як організації, що мислять прогресивно, можуть підготуватися до майбутнього:

– Прискорення завдяки гнучкості: Використовуйте модульне проектування, цифрове будівництво та стратегічні партнерства в ланцюгах поставок, щоб швидко реалізовувати проекти, залишаючись гнучкими до змін на ринку та нових технологій.

– Інвестуйте в таланти майбутнього: створюйте надійні портфелі талантів, співпрацюючи з навчальними закладами та галузевими партнерами, забезпечуючи стабільний приплив кваліфікованих фахівців та майбутніх лідерів.

– Лідер у сфері сталого розвитку: виходьте за рамки дотримання вимог, інтегруючи гібридні енергетичні рішення та передові технології охолодження, встановлюючи нові стандарти енергоефективності та водоефективності.

– Зміцнення стійкості ланцюга поставок: моніторинг глобальних ризиків та інвестування в циркулярність, переробку та альтернативні джерела постачання для забезпечення критично важливих матеріалів для довгострокового зростання.

– Підготуйтеся до наступної хвилі інновацій: використовуйте управління проектами на основі штучного інтелекту та почніть закладати основу для інтеграції квантових обчислень, щоб ваші операції були готові до наступного стрибка в продуктивності.

Заглядаючи в майбутнє, переможцями стануть ті, хто передбачає зміни, інвестує в переосмислення та діє сміливо.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.2. Створена функціональна модель мережі системи програмно визначаємого ЦОД на базі технологій Fujitsu дозволяє вирішити наступні завдання:

– точно визначити вхідні, вихідні й керуючі впливи на підсистему динамічного перерозподіл трафіку мережі системи програмно визначаємого ЦОД на базі технологій Fujitsu, що надалі дозволяє провести аналітичне й імітаційне моделювання інформаційного потоку;

– сформувати наочне візуальне подання взаємодії основних процесів, що відбуваються у мережі системи програмно визначаємого ЦОД на базі технологій Fujitsu;

– зробити чітке визначення ролі системи поліпшення функціонування в загальній системі системи програмно визначаємого ЦОД на базі технологій Fujitsu;

– сценарій динамічного перерозподілу трафіку мережі системи програмно визначаємого ЦОД на базі технологій Fujitsu, розроблений у ході виконання магістерської роботи, є основою алгоритму імітаційного моделювання.

З рисунку видно, що розроблена система складається з наступних частин:

– Блок розрахунку вектору значень інтегрального критерію оптимізації.

– Блок ініціалізації граничних значень параметрів моделювання.

– Блок формування матриці динамічний перерозподіл трафіку мережі системи програмно визначаємого ЦОД на базі технологій Fujitsu.

– Блок формування матриці транзитних вузлів.

– Блок формування сигналів для елементів мережі системи програмно визначаємого ЦОД на базі технологій Fujitsu.

– Згенеровані звіти по поточному завантаженню елементів мережі системи програмно визначаємого ЦОД на базі технологій Fujitsu.

Інтернет-технології прийняли широке поширення, і використовуються як основа побудови корпоративних і критично важливих додатків, таких як WEB-вузли, вузли потокового мультимедіа-віщання й сервери віртуальних приватних мереж. Служба динамічного перерозподілу трафіку мережі системи програмно визначаємого ЦОД на базі технологій Fujitsu (ДПТМ) є оптимальним і ефективним рішенням, що забезпечує масштабованість і високу відказостійкість таких додатків як в Інтернеті, так і в інтрамережах. Служба ДПТМ дозволяє системним адміністраторам створювати кластери, що включають до 32 вузлів, між якими будуть розподілятися вступні від клієнтів запити. При цьому з погляду клієнтів кластер нічим не відрізняється від звичайного сервера; серверні додатки також не вимагають адаптації для роботі в кластері.

Служба ДПТМ постачена всіма необхідним адміністраторам способами керування, у тому числі можливістю (після уведення пароля) віддалено управляти кластером з будь-якого комп'ютера в мережі системи програмно визначаємого ЦОД на базі технологій Fujitsu. Крім того, адміністратори мають можливість набудовувати кластер під спеціальні завдання, управляючи потоком даних на рівні портів. Вузли додаються й виключаються із кластера без припинення обслуговування. Крім того, програмне забезпечення на вузлах кластера можна обновляти без припинення обробки клієнтських запитів.

Рисунок 3.2 – Функціональна схема системи

Служба ДПТМ використовує для розподілу навантаження між вузлами повністю розподілений алгоритм. На відміну від рішень на основі диспетчеризації така архітектура забезпечує високу продуктивність і низькі накладні витрати ресурсів на розподіл потоку запитів від клієнтів. Крім того, для цієї архітектури характерна висока відказостійкість (N-1) при числі вузлів N. Всі ці характеристики досягаються без необхідності використовувати спеціальні апаратні або програмні рішення.

Тести продуктивності демонструють, що використання програмної служби ДПТМ дає низькі накладні витрати на обробку потоку даних і чудові можливості масштабування продуктивності, обмежені тільки пропускною

здатністю підмережі системи програмно визначаємого ЦОД на базі технологій Fujitsu.

Служба ДПТМ демонструє пропускну здатність більше 200 Мбіт/с у реальних рішеннях по обслуговуванню електронної торгівлі з більш ніж 800 млн. запитів протягом дня.

Служба ДПТМ періодично розсилає ритмічні повідомлення, призначені для інформування кожного члена кластера про наявність інших вузлів. Збій будь-якого вузла виявляється протягом п'яти секунд, а відновлення обслуговування клієнтів виконується протягом десяти секунд.

Як при відключенні працюючого вузла, так і при додаванні нового вузла в кластер навантаження автоматично й прозоро перерозподіляється між членами кластера.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі системи програмно визначаємого ЦОД на базі технологій Fujitsu.

3.4 Розробка діаграми процесів

Розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3. Основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей. Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників. Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграми потоків даних містять чотири типи елементів:

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

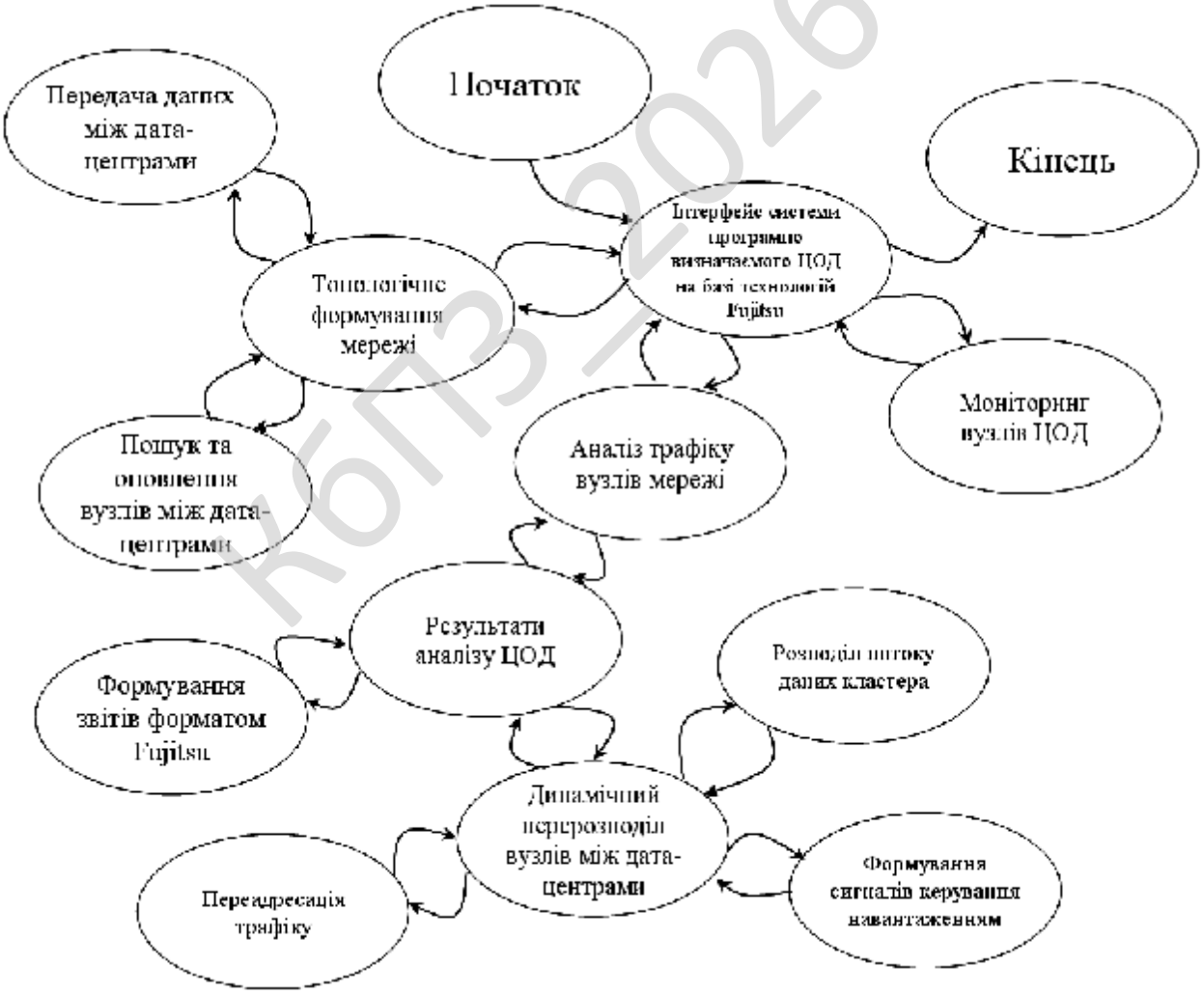

Рисунок 3.3 – Діаграма взаємодії процесів

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо реалізацію бакалаврської дипломної роботи. Були проведені розрахунки і підібрані набори тестових даних для перевірки правильності реалізації проектних рішень.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підсистеми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було створено блок-схеми роботи системи. Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо.

Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції.

Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

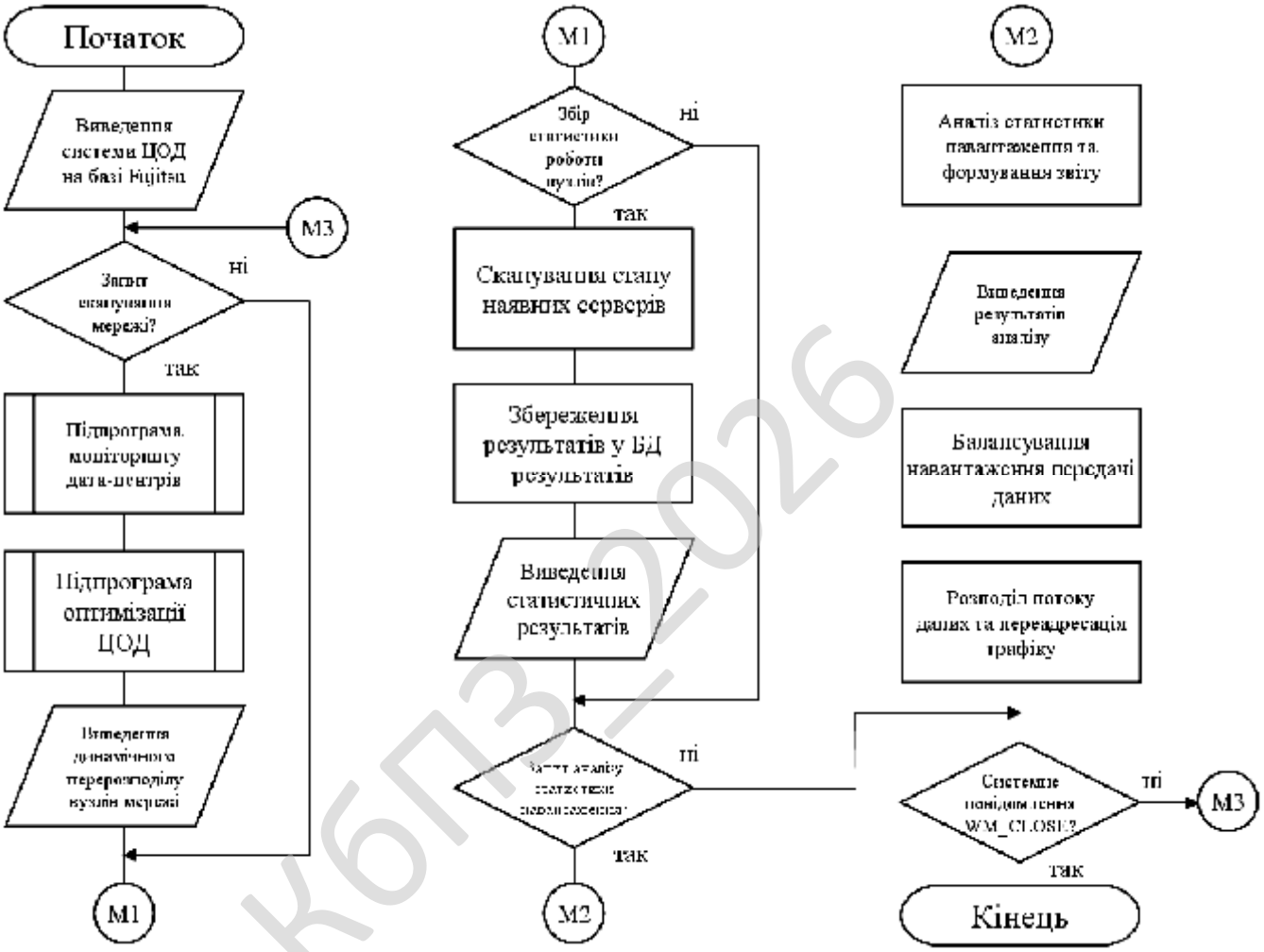

Рисунок 4.1 – Блок-схема основної програми

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач. Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

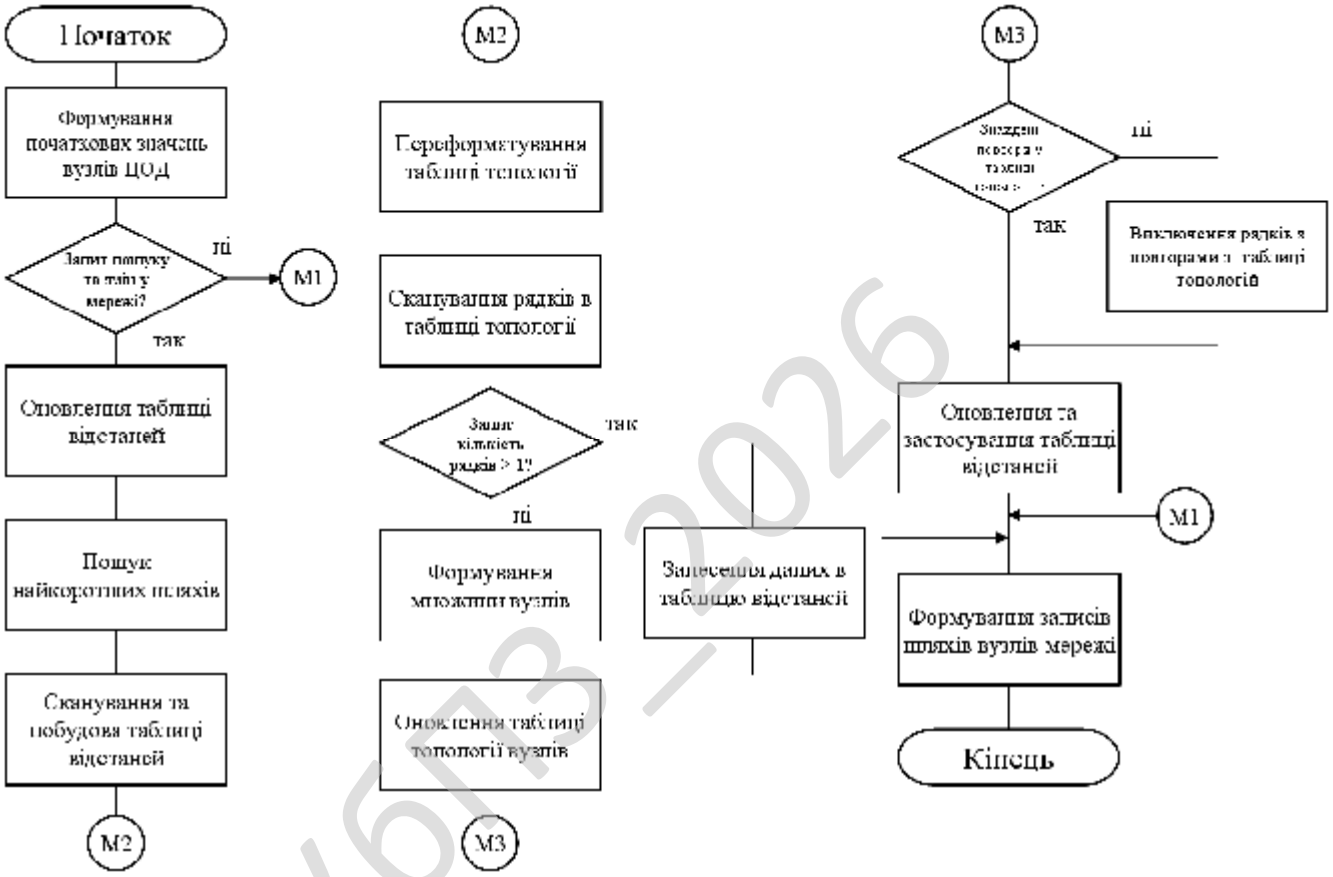

Рисунок 4.2 – Блок-схема роботи підпрограми

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції. Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу.

Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується.

Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи. Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи). З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми.

Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу); Діаграма класів.

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись

один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Розглянемо визначення API. Це прикладни́й програ́мний інтерфе́йс (Application Programming Interface, API) – набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення.

Спрощено – це набір чітко визначених методів для взаємодії різних компонентів. API надає розробнику засоби для швидкої розробки програмного забезпечення. API може бути для веб-базованих систем, операційних систем, баз даних, апаратного забезпечення, програмних бібліотек.

Одним з найпоширеніших призначень API є надання набору широко використовуваних функцій, наприклад для малювання вікна чи іконок на екрані. API є абстрактним поняттям – програмне забезпечення, що пропонує деякий API, часто називають реалізацією (implementation) даного API. У багатьох випадках API є частиною набору розробки програмного забезпечення, водночас, набір розробки може включати як API, так і інші інструменти/апаратне забезпечення, отже ці два терміни не є взаємозамінювані.

Високорівневі API часто програють у гнучкості. Виконання деяких функцій нижчого рівня стає набагато складнішим, або навіть неможливим.

В об'єктно-орієнтованих мовах, прикладний програмний інтерфейс зазвичай включає в себе опис набору визначень класу, з набором форм поведінки, пов'язаних з цими класами. Це абстрактне поняття пов'язане з реальними функціями, які надані або надаватимуться, класами, які реалізуються в методах класу.

Прикладний програмний інтерфейс в даному випадку можна розглядати як сукупність всіх методів, які публічно доступні в класах (зазвичай званий інтерфейс класу). Це означає, що прикладний програмний інтерфейс вказує методи, за допомогою яких взаємодіє з об'єктами, отриманими з визначень класів і обробляє їх.

У більш загальному плані можна визначити Прикладний Програмний Інтерфейс як сукупність усіх видів об'єктів, які можна вивести з визначення класу, і пов'язаних з ними можливих варіантів поведінки.

Наприклад: клас, що представляє Stack, може просто виставити публічно два методи Push() (для додавання нового елемента в стек) і *Pop()* (для вилучення останнього пункту, ідеально розташований на вершині стека).

У цьому випадку Прикладний Програмний Інтерфейс може бути інтерпретованим як два методи *pop()* і *push()*, або, більш широко, використовується варіант, коли можна використовувати елемент типу Stack, який реалізує поведінку стека, надаючи йому можливість для додавання / видалення елементів з вершини. Друга інтерпретація видається більш доречною в дусі об'єктно-орієнтованого підходу.

Якість документації, пов'язаної з Прикладним Програмним Інтерфейсом, є часто ключовим фактором, що визначає його успішність з точки зору простоти використання.

ППІ, як правило, пов'язаний із бібліотеками програмного забезпечення: ППІ описує і вказує очікувану поведінку в той час, як бібліотека є фактичною

реалізацією даного набору правил. Один ППІ може мати декілька реалізацій (або жодної, будучи абстрактним) у вигляді різних бібліотек, які мають такий же інтерфейс.

Прикладний програмний інтерфейс також може бути пов'язаним з платформами програмування: платформа може бути заснована на кількох бібліотеках реалізує декілька інтерфейсів ППІ, але на відміну від звичайного використання ППІ, доступ до поведінки вбудований в платформу опосередкований шляхом розширення його змісту новими класами і вставлений в саму платформу. Крім того, загальний потік управління програми може бути під контролем абонента.

Прикладний програмний інтерфейс може бути також реалізацією протоколу.

Коли ППІ реалізує протокол, він може бути заснованим на проксі-методах віддалених викликів, що засновані на протоколі зв'язку. Роль ППІ може заключатися саме в тому, щоб приховати деталі транспортного протоколу. Наприклад: RMI є ППІ, який реалізує протокол або JRMP IIOP як RMI-IIOP.

Протоколи, як правило, розподіляються між різними технологіями і зазвичай дозволяють різним технологіям обмінюватися інформацією, діючи як абстракція між двома світами. ППІ, як правило, є специфічним для конкретної технології: звідси, інтерфейси даної мови не можуть бути використані на інших мовах, якщо виклики функції не будуть перетворені з конкретної адаптації бібліотеки.

Деякі мови, серед яких такі, що працюють на віртуальних машинах (наприклад: мови, сумісні з NET CLI середовища CLR і JVM сумісних мов у віртуальній машині Java) можуть ділитися програмними інтерфейсами.

У цьому випадку віртуальна машина дозволяє мові взаємодії завдяки спільному знаменнику віртуальної машини, що абстрагується від конкретної мови, використовувати проміжний байт-код і його мову.

При використанні прикладного програмного інтерфейсу в контексті веб-розробки, як правило, ППІ визначається набором повідомлень запиту HTTP, також визначається структура повідомлень-відповідей, зазвичай у розширенні мови розмітки XML або в форматі об'єктного запису JavaScript (JSON). У той час як прикладний програмний інтерфейс у Web історично був практично синонімом для веб-служби, останнім часом тенденція змінилась (так званий Web 2.0) на відхід від Simple Object Access Protocol (SOAP) на основі веб-сервісів і сервіс-орієнтованої архітектури (SOA) на більш прямі передачі репрезентативного стану (REST) стилів веб-ресурсів та ресурсів-орієнтованої архітектури (ROA).

Частина цієї тенденції пов'язана з рухом Семантичного веб-ресурсу до Опису Платформ (RDF), Концепції розвитку веб-технологій інженерних онтологій. Прикладні програмні інтерфейси у Web, що дозволяють комбінувати декількома прикладними програмними інтерфейсами в нові додатки називають гібридними.

Розглянемо формат що використовується – **JSON** (JavaScript Object Notation, укр. запис об'єктів JavaScript, вимовляється джейсон) – це текстовий формат обміну даними між комп'ютерами.

JSON базується на тексті, може бути прочитаним людиною. Формат дозволяє описувати об'єкти та інші структури даних. Цей формат головним чином використовується для передачі структурованої інформації через мережу (завдяки процесу, що називають серіалізацією). Розробив і популяризував формат Дуглас Крокфорд.

JSON знайшов своє головне призначення у написанні веб-програм, а саме при використанні технології AJAX. JSON, що використовується в AJAX, виступає як заміна XML (використовується в AJAX) під час асинхронної передачі структурованої інформації між клієнтом та сервером. При цьому перевагою JSON перед XML є те, що він дозволяє складні структури в атрибутах, займає менше місця і прямо інтерпретується за допомогою JavaScript в об'єкти.

JSON з'явився через необхідність обміну даними з сервером у реальному часі без використання плагінів для браузерів, flash-додатків або Java-аплетів, які використовувались скрізь на початку 2000-х років. Дуглас Крокфорд був тим, хто активно просував новий на той час формат. Він з колегами хотів створити технологію, яка використовувала б можливості звичайного браузера давала б веб-розробникам можливість створювати веб-додатки із постійним двостороннім зв'язком із веб-сервером.

JSON вперше був використаний в проекті в Communities.com для Cartoon Network, він дозволяв обмінюватись повідомленнями і одночасно маніпулювати DHTML-елементами.

Веб-сайт JSON.org було запущено 2002 року. У грудні 2005-го року Yahoo! почав переводити деякі зі своїх веб-сервісів на роботу з JSON. Google взялася до роботи з технологією у своєму веб-протоколі GData у грудні 2006-го року.

Використання

За рахунок своєї лаконічності в порівнянні з XML, формат JSON може бути більш придатним для серіалізації складних структур.

Якщо говорити про веб-застосунки, в такому ключі він доречний в задачах обміну даними як між браузером і сервером (AJAX), так і між самими серверами (програмні HTTP-інтерфейси).

Формат JSON так само добре підходить для зберігання складних динамічних структур в реляційних базах даних або файловому кеші.

Приклад використання JSON (JavaScript)

```
const ajaxData = '{"name": "wiki",
"fname": "pedia", "rates": [1, 4, 5, 6]}'
const ajaxObj = JSON.parse(ajaxData)
console.log(`${ajaxObj.name}_${ajaxObj.rates[2]}`) /* Виведе "wiki_5"*/
```

Синтаксис

JSON будується на двох структурах:

1. Набір пар назва/значення. У різних мовах це реалізовано як об'єкт, запис, структура, словник, хеш-таблиця, список з ключем або асоціативним масивом.

2. Впорядкований список значень. У багатьох мовах це реалізовано як масив, вектор, список, або послідовність.

Це універсальні структури даних. Теоретично всі сучасні мови програмування підтримують їх у тій чи іншій формі. Оскільки JSON використовується для обміну даними між різними мовами програмування, то є сенс будувати його на цих структурах.

У JSON використовуються такі їхні форми:

1. Об'єкт – це послідовність пар назва/значення. Об'єкт починається з символу { і закінчується символом } . Кожне значення слідує за : і пари назва/значення відділяються комами.

2. Масив – це послідовність значень. Масив починається символом [і закінчується символом]. Значення відділяються комами.

3. Значення може бути рядком в подвійних лапках, або числом, або логічними true чи false, або null, або об'єктом, або масивом. Ці структури можуть бути вкладені одна в одну.

4. Рядок – це послідовність з нуля або більше символів юнікода, обмежена подвійними лапками, з використанням escape-послідовностей, що починаються зі зворотної косої риски \. Символи представляються простим рядком. Тип Рядок (String) дуже схожий на String в мовах C і Java. Число теж дуже схоже на C- або Java-число, за винятком того, що вісімкові та шістнадцяткові формати не використовуються. Пропуски можуть бути вставлені між будь-якими двома лексемами.

Наступний приклад показує JSON представлення об'єкта, що описує людину. У об'єкті є рядкові поля імені і прізвища, об'єкт, що описує адресу, і масив, що містить список телефонів.

```
{
   "firstName": "Іван",
   "address": {
       "city": "Київ",
       "postalCode": 21000
   }}
```

Використання JSON в AJAX

Наступний фрагмент коду JavaScript показує, як клієнт може використати XMLHttpRequest для запиту об'єктів в форматі JSON з серверу. (Серверна частина коду пропущена, вона просто повертає на запит URL рядок у JSON форматі).

Треба відзначити, що тут використання XMLHttpRequest не є кросс-браузерним (за деталями звертайтеся на сторінку XMLHttpRequest), і код зазнаватиме незначних модифікацій в різних версіях оглядачів Internet Explorer, Opera, Safari або Mozilla. Використання запиту XMLHttpRequest обмежено правилом одного джерела (same origin policy): URL, що відповідає на запит, має посилатися на той же сайт, що обслуговує сторінку, що ініціювала запит.

Оглядачі можуть також використовувати тег <iframe> для асинхронного запиту JSON-даних в кросс-браузерному варіанті, або використати просте перенаправлення <form action="url_to_cgi_script" target="name_of_hidden_iframe">. Такий підхід був поширений до приходу популярного нині запиту XMLHttpRequest.

Динамічний тег <script> також можна використати для підвантаження JSON-даних. Ця техніка можлива, щоб обійти надто суворе правило одного джерела, але вона не є безпечною. Запит JSONRequest пропонується, як безпечніша альтернатива.

Питання безпеки

Хоча JSON призначений для передачі даних в серіалізованому вигляді, його синтаксис відповідає синтаксису JavaScript і це створює низку проблем безпеки. Часто для обробки даних, отриманих від зовнішнього джерела у форматі JSON, до них застосовується функція eval() без якої-небудь попередньої перевірки.

JavaScript eval()

Оскільки JSON представляється синтаксично правильним фрагментом коду JavaScript, природним способом розбору JSON-даних в JavaScript-програмі є використання вбудованої в JavaScript функції eval(), яка призначена для

обчислення JavaScript-виразів. При цьому підході відпадає необхідність у використанні додаткових парсерів.

Техніка використання eval() робить систему вразливою, якщо джерело JSON-даних, що використовуються, не відноситься до надійних. Такими даними може виступати шкідливий JavaScript-код для атак за допомогою ін'єкції коду. За допомогою цієї вразливості можливо здійснювати крадіжку даних, підробку автентифікації. Проте, вразливість можна усунути за рахунок використання додаткових засобів перевірки даних на коректність. Наприклад, до виконання eval() отримані від зовнішнього джерела дані можуть перевірятися за допомогою регулярних виразів. У RFC, що визначає JSON пропонується використовувати такий код для перевірки його відповідності формату JSON

```
const my_JSON_object = !(/[^,:{}\[\]0-9.\-+Eaeflnr-u \n\r\t]/.test(
text.replace(/"(\\.|[^"\\])*"/g, ''))) &&
eval('(' + testedData + ')');
```

Як безпечніша альтернатива eval() була запропонована нова функція parseJSON(), здатна обробляти тільки JSON-дані.

Вбудований JSON

Останні версії веб-браузерів мають вбудовану підтримку JSON і здатні його обробляти без виклику функції eval(), що призводить до описаної проблеми. Обробка JSON у такому разі зазвичай здійснюється швидше.

Принаймні дві популярні бібліотеки JavaScript використовують вбудований JSON у разі його доступності: jQuery; Dojo.

Підробка крос-доменного запиту

Непродумане використання JSON робить сайти вразливими до підробки міжсайтових запитів (CSRF або XSRF). Оскільки тег <script> допускає використання джерела, що не належить до того ж домену, що і використовуваний ресурс, це дозволяє виконувати код даних, представлених у форматі JSON, в контексті довільної сторінки, що робить можливою компрометацію паролів або іншої конфіденційної інформації користувачів, що пройшли авторизацію на іншому сайті.

Це є проблемою тільки у разі вмісту в JSON-даних конфіденційної інформації, яка може бути компрометована третьою стороною і якщо сервер розраховує на політику одного джерела, блокуючи доступ до даних при виявленні зовнішнього запиту. Це не є проблемою, якщо сервер визначає допустимість запиту, надаючи дані тільки у разі його коректності. HTTP cookie не можна використовувати для визначення цього. Виключне використання HTTP cookie використовується підробкою міжсайтових запитів.

Розширення, JSONP

JSONP або «JSON з підкладкою» є розширенням JSON, коли назва функції зворотного виклику вказується як вхідний аргумент. Спочатку ідея була запропонована в блозі MacPython в 2005 році, і в наш час використовується багатьма Web 2.0 застосунками, такими, як Dojo Toolkit Applications, Google Toolkit Applications і zanox Web Services.

Подальші розширення цього протоколу були запропоновані з урахуванням введення додаткових аргументів, як, наприклад, у разі JSONPP за підтримки S3DB вебсервісів.

Оскільки JSONP використовує скрипт-теги, виклики по суті відкриті світові. З цієї причини JSONP може бути недоречними для зберігання конфіденційних даних.

Включення скриптових тегів від віддалених сайтів дозволяє їм передати будь-який контент на сайті. Якщо віддалений сайт має вразливості, які дозволяють виконати ін'єкції JavaScript, то початковий сайт також може бути ними зачеплений.

Розширення, BSON

BSON – це бінарна форма представлення простих структур даних і асоціативних масивів (які називають об'єктами або документами). Назва «BSON» заснована на визначенні JSON і неофіційно означає «Binary JSON» (бінарний JSON).

JSON Reference

Стандарт JSON не описує посилання на інші об'єкти або частини, але існує чернетка стандарту IETF для посилань на об'єкти на основі JSON. Посилання дозволяють здійснювати трансклюзію – вставляти одні документи в інші.

JSON Reference – це JSON-об'єкт з ключем "$ref" (всі інші ключі ігноруються) і значенням стрічкового типу що містить URI, наприклад:

```
{ "$ref": "http://example.com/example.json#/foo/bar" }
```

Якщо URI містить ідентифікатор фрагмента (в прикладі вище "/foo/bar"), він інтерпретується як JSON Pointer.

Модуль dojox.json.ref в Dojo toolkit, забезпечує підтримку декількох форм JSON Reference.

Систе́ма керування ба́зами да́них (СКБД, Database Management System, DBMS) – набір взаємопов'язаних даних (база даних) і програм для доступу до цих даних. Надає можливості створення, збереження, оновлення та пошуку інформації в базах даних з контролем доступу до даних.

Першим поколінням СКБД прийнято вважати ієрархічні й мережеві системи. Ці системи отримали широке поширення в 1970-х роках, а першою комерційною системою цього типу була система IMS компанії IBM.

У 1980-х роках ці системи були витіснені системами другого покоління – повсюдно використовуваними і донині реляційними СКБД. У цих системах використовувалися непроцедурні мови управління даними (SQL) і передбачався значний ступінь незалежності даних. Реляційні системи внесли значні удосконалення в управління даними: графічний користувацький інтерфейс (GUI), клієнт-серверні застосунки, розподілені бази даних, паралельний пошук даних та інтелектуальний аналіз даних.

Але вже до кінця 1980-х років існуюча тоді реляційна модель перестала задовольняти розробників в силу низки обмежень. Відповіддю на зростаючу складність програм баз даних стали два нових напрямки розвитку СКБД: об'єктно-орієнтовані СКБД і об'єктно-реляційні СКБД.

У 1991 був утворений консорціум ODMG, основною метою якого стало вироблення промислового стандарту об'єктно-орієнтованих баз даних. Між 1993 та 2001 роками ODMG опублікувала п'ять ревізій своїх специфікацій. Остання версія стандарту має індекс 3.0, після чого група розпустилася. До кінця 1990-х існувало близько десяти компаній, що виробляли комерційні продукти, що позиціонуються на ринку як ООСКБД. Найбільш відомими системами даного класу стали Objectivity, Versant виробництва однойменних компаній, а також СКБД Jasmine, випущена компанією CA. Незважаючи на переваги, що дозволяють ефективніше вирішувати певний ряд завдань, об'єктно-орієнтовані системи так і не змогли завоювати значущу частку ринку СКБД, залишившись «нішевим» продуктом.

Постачальниками традиційних реляційних СКБД також була проведена значна робота з об'єднання об'єктно-орієнтованих і реляційних систем. Розробники постаралися розширити мову SQL, щоб включити в неї концепції об'єктно-орієнтованого підходу, зберігаючи переваги реляційної моделі (об'єктні розширення мови SQL були зафіксовані в стандарті SQL:1999).

Основний принцип – це еволюційний розвиток можливостей СКБД без поломки попередніх підходів та зі збереженням наступності з системами попереднього покоління.

Поняття СКБД третього покоління, якими, власне кажучи, і є об'єктно-реляційні СКБД, з'явилося після опублікування групою відомих фахівців в області баз даних «Маніфесту систем баз даних третього покоління». Основні принципи СКБД третього покоління, позначені в маніфесті:

Крім традиційних послуг з управління даними, СКБД третього покоління повинні забезпечити підтримку розвиненіших структур об'єктів і правил. Розвинутіша структура об'єктів характеризує засоби, необхідні для зберігання і маніпулювання нетрадиційними елементами даних (тексти, просторові дані, мультимедіа).

СКБД третього покоління повинні включити в себе СКБД другого покоління. Системи другого покоління внесли вирішальний вклад у двох областях – непроцедурний доступ за допомогою мови запитів SQL і незалежність даних. Ці досягнення обов'язково повинні враховуватися в системах третього покоління.

СКБД третього покоління повинні бути відкриті для інших підсистем. Це включає оснащення різноманітними інструментами підтримки прийняття рішень, доступом з багатьох мов програмування, інтерфейсами до існуючих популярних систем і бізнес-застосунків, можливістю запуску програм з бази даних на іншій машині і розподілені СКБД. Весь набір інструментів і СКБД має ефективно функціонувати на різноманітних апаратних платформах з різними операційними системами.

Крім того, СКБД, що розраховує на широку сферу застосування, повинна бути оснащена мовою четвертого покоління (4GL).

У середині 1990 років було лише кілька дослідних прототипів СКБД, які поєднали найкращі риси реляційних і об'єктно-орієнтованих СКБД. Першим комерційним продуктом, якому були властиві об'єктно-реляційні риси, став Universal Server компанії Informix(згодом була поглинена IBM). В даний час більшість цих ідей вже втілено в реальних комерційних рішеннях, в тому числі і в продуктах основних постачальників СКБД (Oracle Database і IBM DB2).

Розвиток індустрії систем керування базами даних базується на значних фундаментальних наукових дослідженнях. Найчастіше, між самими дослідженнями та їхньою конкретною реалізацією в прикладних рішеннях минають роки, а іноді й десятиліття. Роботу в області управління даними проводять як університетські дослідницькі групи (MIT, Berkeley), так і центри розробок основних постачальників СКБД (Oracle, IBM, Microsoft). Інвестування в управління даними – це довгострокове, і разом з тим, вигідне вкладення коштів. В даний час дослідники мають у своєму розпорядженні засоби, що дозволяють

ефективно реалізувати найскладніші запити, що маніпулюють терабайтами й петабайтами різних даних.

Основними тенденціями, які дали привід для проведення різних масштабних досліджень в області баз даних стали:

Експонентний ріст даних. Обсяг даних, у тому числі синтетичних, що генеруються автоматизованими системами, значно зріс. Збільшилося і число прикладних областей, в яких вимагається обробка великих обсягів даних. До таких областей тепер відносяться не тільки традиційні корпоративні програми, пошук у веб, але також і наукові дослідження, обробка природних мов, аналіз соціальних мереж тощо.

Значне ускладнення структур використовуваних даних. Прості види даних у вигляді чисел і символьних рядків стали доповняться численною мультимедійною інформацією, просторовими, процедурними даними та великою кількістю інших складних форматів.

Широке поширення дешевих високопродуктивних апаратних засобів. Щорічно ми спостерігаємо зростання обчислювальних можливостей мікропроцесорів, збільшення ємності і зниження вартості доступних і зручних в експлуатації пристроїв дискової оперативної пам'яті.

Активний розвиток засобів комунікації та «всесвітньої павутини» World Wide Web. WWW стає єдиним інформаційним середовищем, що пронизує весь світ і об'єднує величезне число користувачів та електронних пристроїв.

Поява нових важливих областей застосування СКБД. У першу чергу, це пов'язано з інтелектуальним аналізом даних, сховищами даних, а останнім часом – з паралельними обчисленнями і хмарними технологіями.

Основні характеристики СКБД

1. Контроль за надлишковістю даних.
2. Несуперечливість даних.
3. Підтримка цілісності бази даних (коректність та несуперечливість).
4. Цілісність описується за допомогою обмежень.

5. Незалежність прикладних програм від даних.

6. Спільне використання даних.

7. Підвищений рівень безпеки.

Можливості СКБД

1. Дозволяється створювати БД (здійснюється за допомогою мови визначення даних DDL (Data Definition Language)).

2. Дозволяється додавання, оновлення, видалення та читання інформації з БД (за допомогою мови маніпулювання даними DML, яку часто називають мовою запитів)

3. Можна надавати контрольований доступ до БД за допомогою: Системи забезпечення захисту, яка запобігає несанкціонованому доступу до БД; Системи керування паралельною роботою прикладних програм, яка контролює процеси спільного доступу до БД; Система відновлення – дозволяє відновлювати БД до попереднього несуперечливого стану, що був порушений в результаті збою апаратного або програмного забезпечення.

Основні компоненти середовища СКБД

1. Апаратне забезпечення.

2. Програмне забезпечення.

3. Дані.

4. Процедури – інструкції та правила, які повинні враховуватись при проектуванні та використанні БД.

5. Користувачі: адміністратори даних (керування даними, проектування БД, розробка алгоритмів, процедур) та БД (фізичне проектування, відповідальність за безпеку та цілісність даних); розробники БД; прикладні програмісти; кінцеві користувачі.

Архітектура СКБД

Існує трирівнева система організації СКБД ANSI-SPARC, при якій існує незалежний рівень для ізоляції програми від особливостей представлення даних на нижчому рівні.

Рівні:

1. Зовнішній – представлення БД з точки зору користувача.

2. Концептуальний – узагальнене представлення БД, описує які дані зберігаються в БД і зв'язки між ними. Підтримує зовнішні представлення, підтримується внутрішнім рівнем.

3. Внутрішній – фізичне представлення БД в комп'ютері.

Логічна незалежність – повна захищеність зовнішніх моделей від змін, що вносяться в концептуальну модель.

Фізична незалежність – захищеність концептуальної моделі від змін, які вносяться у внутрішню модель.

MySQL – вільна система керування реляційними базами даних. Розробка та підтримка сайта MySQL здійснює корпорація Oracle, яка отримала права на торговельну марку разом з поглиненої Sun Microsystems, яка раніше придбала шведську компанію MySQL AB.

Продукт поширюється як під GNU General Public License, так і під власною комерційною ліцензією. Крім цього, розробники створюють функціональність за замовленням ліцензійних користувачів. Саме завдяки такому замовленню майже в найраніших версіях з'явився механізм реплікації.

MySQL є рішенням для малих і середніх додатків. Входить до складу серверів WAMP, AppServ, LAMP і в портативні збірки серверів Денвер, XAMPP, VertrigoServ. Зазвичай MySQL використовується як сервер, до якого звертаються локальні або видалені клієнти, проте в дистрибутив входить бібліотека внутрішнього сервера, що дозволяє включати MySQL в автономні програми.

Гнучкість СКБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів.

Більш того, СКБД MySQL поставляється із спеціальним типом таблиць EXAMPLE, що демонструє принципи створення нових типів таблиць. Завдяки

відкритій архітектурі і GPL-ліцензуванню, в СКБД MySQL постійно з'являються нові типи таблиць.

26 лютого 2008 року Sun Microsystems придбала MySQL AB за 1 млрд доларів, 27 січня 2010 року Oracle придбала Sun Microsystems за 7,4 млрд доларів і включила MySQL в свою лінійку СКБД.

Спільнотою розробників MySQL створені різні відгалуження коду, такі як Drizzle (англ.), OurDelta, Percona Server і MariaDB. Всі ці відгалуження вже існували на момент поглинання компанії Sun корпорацією Oracle.

MySQL має подвійне ліцензування. MySQL може розповсюджуватися відповідно до умов ліцензії GPL. Але за умовами GPL, якщо якась програма використовує бібліотеки MySQL, то вона теж повинна розповсюджуватися за ліцензією GPL. Проте це може розходитися з планами розробників, які не бажають відкривати сирцеві тексти своїх програм. Для таких випадків передбачена комерційна ліцензія компанії Oracle, яка також забезпечує якісну сервісну підтримку.

В разі використання та розповсюдження програмного забезпечення з іншими вільними ліцензіями, такими як BSD, Apache, MIT та інші, MySQL дозволяє використання бібліотек MySQL за ліцензією GPL.

MySQL виникла як спроба застосувати mSQL до власних розробок компанії: таблиць, для яких використовувалися ISAM – підпрограми низького рівня для індексного доступу до даних. У результаті був вироблений новий SQL-інтерфейс, але API-інтерфейс залишився в спадок від mSQL. Звідки походить назва «MySQL» – достеменно не відомо. Розробники дають два варіанти: або тому, що практично всі напрацювання компанії починалися з префікса My, або на честь дівчинки на ім'я My, дочки Майкла Монті Віденіуса, одного з розробників системи.

Логотип MySQL у вигляді дельфіна носить ім'я «Sakila». Він був обраний з великого списку запропонованих користувачами «імен дельфіна». Ім'я «Sakila» було відправлено Open Source-розробником Ambrose Twebaze.

За час розвитку під орудою Oracle дедалі більше відокремлює MySQL від спільноти і робить процес розробки все менш прозорим. Наприклад, повернута практика поставки власницьких розширених функцій в Enterprise-версії MySQL, спостерігається приховування інформації про вразливості, зі складу виключений тестовий набір, закритий доступ до більшої частини системи відстеження помилок та припинено публікація згрупованого логу змін, що дозволяє судити про прив'язку патчів до конкретних змін.

Можливості

MySQL – компактний багатопотоковий сервер баз даних. Характеризується високою швидкістю, стійкістю і простотою використання.

MySQL вважається гарним рішенням для малих і середніх застосувань. Сирцеві коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатопоточності, що підвищує продуктивність системи в цілому.

Можливості сервера MySQL:

1. Простота у встановленні та використанні.
2. Підтримується необмежена кількість користувачів, що одночасно працюють із БД.
3. Кількість рядків у таблицях може досягати 50 млн.
4. Висока швидкість виконання команд.
5. Наявність простої і ефективної системи безпеки.

4.2 Захист розробленого програмного забезпечення

Дані які використовуються у даній роботі захищаються алгоритмом ДСТУ 7624:2014 («Калина»). Одним із основних алгоритмів симетричного блокового шифрування, що використовуються в Україні, є ДСТУ 7624:2014 («Калина»), який визначає сучасний алгоритм симетричного блокового перетворення для

забезпечення конфіденційності й цілісності інформації при її обробці та встановлює режими його роботи.

В алгоритмі шифрування даних «Калина» використовуються криптографічні перетворення, які відповідають сучасним вимогам до рівня криптостійкості та швидкодії.

Даний стандарт розроблено з урахуванням існуючих та потенційних загроз, подальшого інтенсивного розвитку інформаційних технологій та необхідності активного використання протягом кількох наступних десятиліть.

Стандарт блокового симетричного шифрування ДСТУ 7624:2014 визначає десять різних режимів роботи, що широко поширені відповідно до міжнародних стандартів ISO/IEC 10116:2006.

Це спрямовано на забезпечення широкого застосування ДСТУ 7624:2014, у тому числі для захисту інформації, що передається комп'ютерними мережами, прозорого шифрування жорстких дисків і змінних носіїв, електронних документів, ключових даних.

Ефективність реалізації систем, засобів та протоколів криптографічного захисту інформації в інформаційно-телекомунікаційних системах різного призначення може бути забезпечена саме наявністю такої кількості режимів роботи алгоритму.

До блокового шифру «Калина» ставляться такі вимоги: високий рівень криптографічної стійкості з достатнім запасом у разі появи нових атак протягом тривалого часу; висока швидкодія програмної реалізації на сучасних та перспективних платформах; компактність програмної та програмно-апаратної реалізації; можливість ефективної інтеграції декількох алгоритмів в одному засобі криптографічного захисту; прозорість проектування, консервативний підхід до забезпечення стійкості; вища (або однакова) ефективність порівняно з найкращими світовими рішеннями. Криптографічні алгоритми, які визначаються стандартами ДСТУ 7624:2014 і ДСТУ 7564:2014, є гнучкими, підтримують розмір блоку і довжину ключа від 128 до 512 біт.

Стандарт симетричного блокового шифрування «Калина» є результатом багаторічної плідної співпраці Державної служби спеціального зв'язку та захисту інформації України та провідних українських вчених.

Даний алгоритм шифрування враховує досвід і результати проведення міжнародних та відкритих національних конкурсів криптографічних алгоритмів.

Алгоритм ДСТУ 7624:2014 забезпечує досить високий рівень криптостікості порівняно з міжнародним стандартом AES (ISO/IEC 18033-3:2010), оскільки дає можливість застосовувати блок даних і ключ шифрування розміром аж до 512 біт.

Крім того, він має аналогічну або навіть більш високу швидкодію на сучасних і перспективних програмних та програмно-апаратних платформах.

На даний момент продовжуються роботи зі стандартизації вітчизняних криптографічних алгоритмів та протоколів.

При цьому не обмежується застосування гармонізованих стандартів у сфері захисту конфіденційної інформації.

Також зусилля зосереджено на використанні кращих практик застосування стандартів шифрування даних для захисту інформації в інформаційно-комунікаційних системах [10, 11].

Оскільки в стандартах симетричного блокового шифрування «Калина» та AES використовуються аналогічні криптографічні перетворення, на наш погляд, буде доцільним порівняти ці два алгоритми.

Основними відмінностями «Калина» від «Rijndael» (AES) є: збільшена кількість циклів шифрування (запас стійкості); використання додавання за модулем 264 і за модулем 2 для введення ключової інформації (захист від алгебричних атак, лінійного та диференціального криптоаналізів, інтерполяційної атаки тощо); використання чотирьох блоків нелінійного перетворення (S-блоків) замість одного (додатковий захист від алгебричних атак, поліпшення властивостей розсіювання алгоритму – покращені статистичні властивості, відповідно, більш високий рівень стійкості до диференціального та лінійного

криптоаналізів тощо); використання випадково сформованих чотирьох блоків, відібраних критеріями стійкості до диференціального, лінійного криптоаналізів, ступені нелінійності булевих функцій (на відміну від S-блоку Rijndael/Camellia та інших шифрів, що використовують звернення в полі та, відповідно, квадратичні залежності між входом і виходом, – захист від алгебричних атак); принципово нова схема створення підключів (захист від усіх відомих атак на схеми створення підключів); досить висока продуктивність; можливість відновлення сеансового ключа за окремим підключем (додатковий захист від атак, що виконують відновлення підключів).

Усі поліпшення спрямовані на збільшення стійкості та запобігання потенційним вразливостям відносно Rijndael, виявленим в останні роки [12].

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено розроблене у бакалаврської дипломної роботі система програмно визначаємого ЦОД на базі технологій Fujitsu. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи: Функції представлені у графічному вигляді; Розділу виведення результату роботи системи; Розділу обрання групи; Верхнього меню; Функціональних кнопок ПЗ.

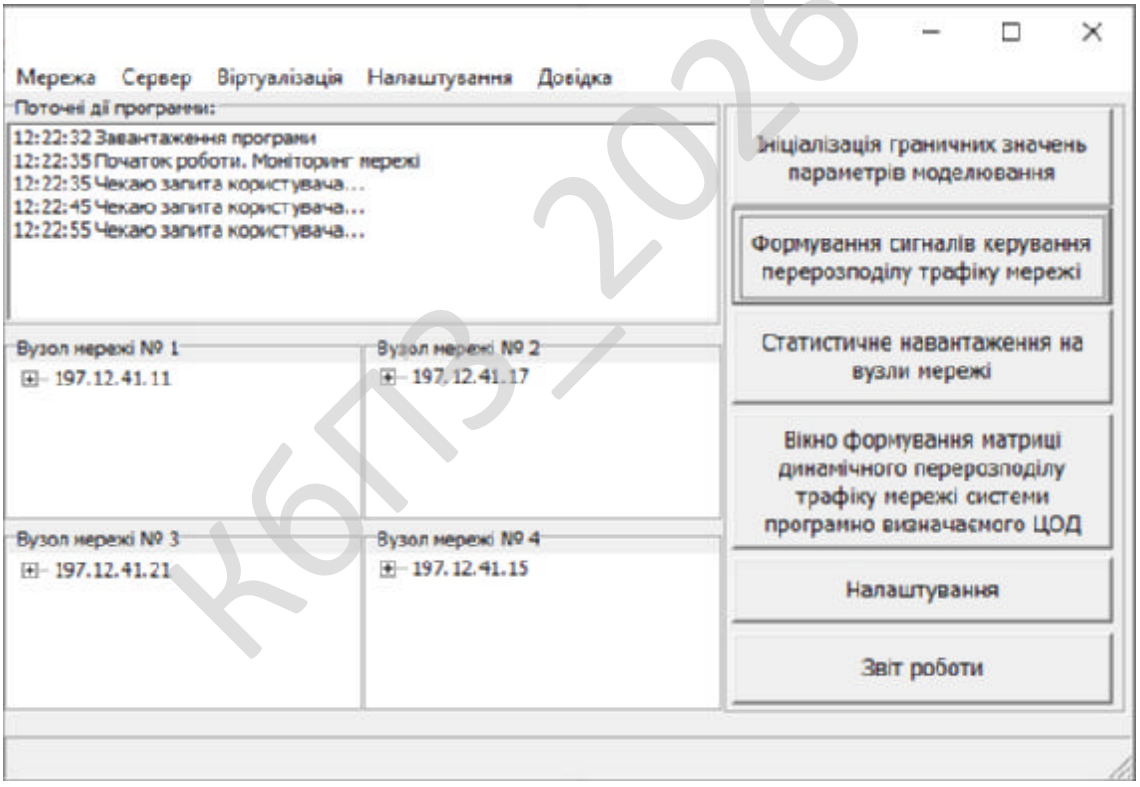

Рисунок 5.1 – Головне вікно ПЗ

Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму,

оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

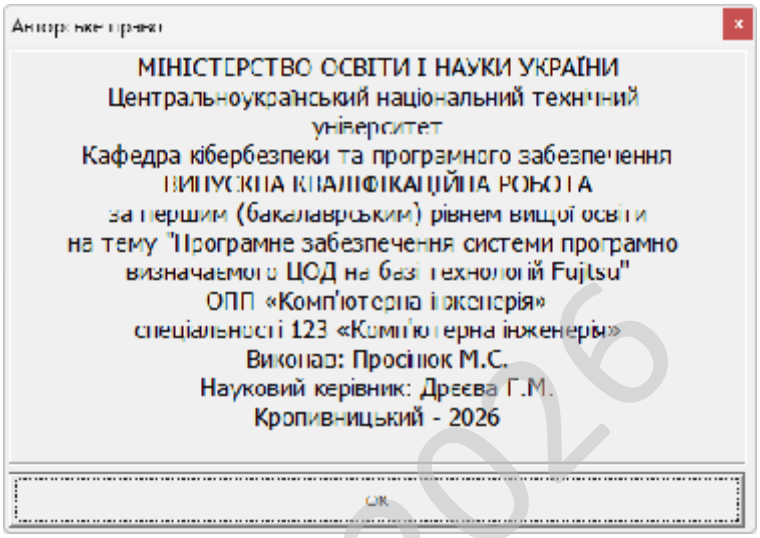

Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних

вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати ІТ рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене ІТ рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в ІТ рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути

невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Проводилось тестування чорної скриньки.

Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ITc).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його OT).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ITc;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору OT.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій.

– Помилок інтерфейсу.

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних.

– Помилок характеристик (необхідна ємність пам'яті і т.д.).

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – Shareware. Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження

комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання.

Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму.

В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи програмно визначаємого ЦОД на базі технологій Fujitsu.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем програмно визначаємого ЦОД на базі технологій Fujitsu.

– Досліджена система програмно визначаємого ЦОД на базі технологій Fujitsu.

– На основі отриманих результатів досліджень створена програмна реалізація системи програмно визначаємого ЦОД на базі технологій Fujitsu.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання програмно визначаємого ЦОД на базі технологій Fujitsu.

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 призначені для системи програмно визначаємого ЦОД на базі технологій Fujitsu. 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.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ДСТУ 7624:2014.

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. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 1». Cisco Press. 2020. – 848 p.

2. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 2 Premium Edition eBook and Practice Test». Cisco Press. 2020. – 624 p.

3. Scott Jernigan «CompTIA Network+ Certification All-in-One Exam Guide, Eighth Edition». 2022. – 976 p.

4. Doug Lowe «Networking For Dummies 12th Edition». 2020. – 480 p.

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

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

7. Вінтенко Б., Смірнов О., Миронець І., Смірнова Т., Смірнов С. «Імітаційна модель шляхів вхідних даних комп'ютерної інтелектуальної системи підтримки оператора енергоблоку АЕС». *Комбінаторні конфігурації та їхні застосування: Матеріали XXVII Міжнародного науково-практичного семінару, присвяченого 125-річчю Національного університету «Запорізька політехніка»* (Запоріжжя-Кропивницький-Київ, 4-6 червня 2025 р.). Запоріжжя: НУ «Запорізька політехніка», 2025. С.82-91.

8. 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.

9. 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.

10. 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.

11. 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.

12. 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.

13. 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.

14. 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.

15. 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.

16. 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.

17. 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.

18. 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.

19. 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.

20. 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.

21. 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.

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

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. Smirnov A.A., Kuzne tsov 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.

39. 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

40. 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.

41. 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.

42. 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.

43. 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.

44. 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.

45. 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.

46. 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).

47. 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

48. 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.

49. 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.

50. 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.

51. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

52. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

53. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Алгоритми формування безлічі маршрутів передачі метаданих у антивірусні хмарні системи. Збірник наукових праць "Системи обробки інформації". - Випуск 5 (142). - Х.: ХУПС - 2016. - С. 148-152.

54. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". - Випуск 3 (140). - Х.: ХУПС - 2016. - С. 36-39.

55. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. - 2016. - С. 121-127.

56. Смірнов О.А., Смірнов С.А., Дідик А.К. Метод безпечної маршрутизації метаданих у хмарні антивірусні системи. Системи озброєння та військова техніка. - Випуск 2 (46) - Х.: ХУПС - 2016. - С. 146-149.

57. Смірнов О.А., Кавун С.В., Доренський О.П., Вялкова В.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 151 с.

58. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

59. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп'ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

60. Смірнов О.А., Стасєв Ю.В. Бараннік В.В. Захист інформації в автоматизованих системах управління. Навчальний посібник – Харків: ХУПС, 2015. – 264 с.

61. Смірнов О.А., Мелешко Є.В., Семенов С.Г. Методи та засоби обробки сигналів і даних в інформаційних системах. Навчальний посібник. – Кіровоград: КНТУ 2012. – 250 с.

62. Смірнов О.А., Євсеєв С.П., Жукарев В.Ю., Король О.Г., Сорокін В.Є., Мелешко Є.В. Технології і стандарти комп'ютерних мереж. Навчальний посібник. – Кіровоград: КНТУ 2012. – 454 с