МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Центральноукраїнський національний технічний університет

Кафедра кібербезпеки та програмного забезпечення

На правах рукопису

Середа Максим Леонідович

Програмне забезпечення системи мережевих систем моніторингу на базі DevOps-архітектури

Спеціальність: 123 «Комп'ютерна інженерія»

Освітній ступінь: бакалавр

Науковий керівник:

Доренський Олександр Павлович _____

(підпис) (дата)

кандидат технічних наук

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри

_____ О.А. Смірнов

(підпис) ПІБ

«____»_____2021 р.

Кропивницький – 2021

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Спеціальність *123 Комп'ютерна інженерія*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____О.А.Смірнов
« 11 » січня 2021 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Середі Максиму Леонідовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи мережевих систем моніторингу на базі DevOps-архітектури*

керівник роботи *Доренський Олександр Павлович, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 204-02 від 28.12.2020 року

2. Строк подання студентом роботи до захисту *22.05.2021 р.*

3. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмне забезпечення системи мережевих систем моніторингу на базі DevOps-архітектури*

4. 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

5. 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*

6. Дата видачі завдання « 11 » січня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів кваліфікаційної бакалаврської роботи	Примітка
1.	Аналіз існуючих систем	10.03.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2021 р.	
3.	Розробка моделі компонента	20.03.2021 р.	
4.	Розробка структур даних	25.03.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2021 р.	
6.	Програмування алгоритмів	10.04.2021 р.	
7.	Оформлення ПЗ	17.04.2021 р.	
8.	Попередній захист роботи	14.05.2021 р.	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Середа М.Л. Програмне забезпечення системи мережевих систем моніторингу на базі DevOps-архітектури. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній кваліфікаційній бакалаврській розроблено програмне забезпечення, яке призначено для системи мережевих систем моніторингу на базі DevOps-архітектури.

Метою розробки є програмне забезпечення системи мережевих систем моніторингу на базі DevOps-архітектури.

Результат роботи – програмна реалізація системи мережевих систем моніторингу на базі DevOps-архітектури.

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 архітектури IBM PC з ОС Windows XP/Vista/7/8/10.

Програму розроблено в середовищі RAD Studio Delphi 10.4.

Ключові слова: комп'ютерна інженерія, DevOps

ABSTRACT

Sereda M.L. Network monitoring system software based on DevOps-architecture. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this bachelor's qualification the software which is intended for system of network monitoring systems on the basis of DevOps-architecture is developed.

The purpose of the development is the software of the network monitoring system based on the DevOps-architecture.

The result is a software implementation of a network monitoring system based on the DevOps architecture.

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.

Developed user-friendly interface. Instructions for working with software are given.

The program can be used on an IBM PC with Windows XP / Vista / 7/8/10.

The program is developed in the environment of RAD Studio Delphi 10.4.

Keywords: computer engineering, DevOps

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ПЗ	–	Програмне забезпечення
API	–	Application Programming Interface, інтерфейс програмування застосунків
DevOps	–	Злиття розробки й експлуатації програмного продукту
IT	–	Інформаційні технології
PaaS	–	Platform as a Service, платформа як послуга
QA	–	забезпечення якості
ППІ	–	прикладний програмний інтерфейс

ВСТУП

Актуальність теми. У наш час комбінація культурної трансформації й автоматизації привела до переосмислення ідей того, як розроблювачі, співробітники IT-підрозділів і фахівці з інформаційної безпеки можуть разом працювати. Орієнтуючись на активну взаємодію й інтеграцію інженерів-програмістів, тестувальників і системних адміністраторів, методологія DevOps базується на ідеї про тісну взаємозалежність розробки й експлуатації програмних продуктів і сервісів, з метою їх більш швидкого створення й відновлення. Але для того, щоб такий підхід став успішним, життєво необхідне здійснення всебічного моніторингу в режимі реального часу.

Методологія DevOps у буквальному (акронім DevOps походить від комбінації двох англійських слів – development і operations) і образному змістах являє собою злиття двох раніше роз'єднаних процесів: розробки й експлуатації програмного продукту. Протягом багатьох років ці дві групи були роз'єднані між собою границями, обумовленими як специфічними ідеологічними відмінностями, так і наборами оперуємих знань, що особливо було помітно у великих IT-організаціях корпоративного масштабу.

Цей поділ був дуже простим: фокус уваги розроблювачів не поширювався далі їхнього програмного коду, а системні адміністратори хоч і працювали з кодом, наданим їм першою групою, але їх завданням було брати цей код і переконатися в його працездатності для конкретних умов. Повний розрив між цими двома групами звичайно є основною причиною тривалих циклів забезпечення якості (QA) і мінімальною кількістю виробничих розгортань через побоювання простою або страху просто що-небудь «зіпсувати».

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи мережевих систем моніторингу на базі DevOps-архітектури.

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

– Огляд існуючих систем мережевих систем моніторингу на базі DevOps-архітектури.

– Дослідження системи мережевих систем моніторингу на базі DevOps-архітектури.

– Програмна реалізація системи мережевих систем моніторингу на базі DevOps-архітектури.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі мережевих систем моніторингу на базі DevOps-архітектури.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи мережевих систем моніторингу на базі DevOps-архітектури, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

З відносно статичною кодовою базою системним адміністраторам не потрібні були найбільш чутливі застосунки для моніторингу. Ті ж, що використовувалися, найчастіше оперували тільки з основними статистичними даними у виробничому середовищі. При використанні такої каскадної методології міг пройти рік або навіть більше між основними відновленнями програмного забезпечення. У багатьох організаціях і донині продовжують дотримуватися подібного підходу.

Проте, протягом останніх десяти років відбулося значне зрушення в підході, який кардинально змінив увесь IT-ландшафт. Розроблювачі, що утомилися від постійного очікування зворотного зв'язку від виробництва для вдосконалення свого коду, почали писати програмне забезпечення, яке автоматизує операційні завдання. У той же час фахівці IT-підрозділів почали делегувати свої глибокі знання й багатий досвід у програмне забезпечення, написане розроблювачами.

Сьогодні колись дуже чіткі границі між розробкою й експлуатацією почали зникати. Це приводить до прискорення всього життєвого циклу програмного забезпечення, більш коротким сесіям QA і багатьом іншим змінам. У нас навіть з'явилися нові процеси, такі як керування безперервною інтеграцією й безперервним розгортанням. Багато великих програмних застосунків, розроблені за допомогою методики DevOps, розгортаються по декілька раз у день, а не кілька раз у рік, як було раніше.

І це не якась новомодна фішка, яка скоро пройде. Ця тенденція в автоматизації робочих процесів прогресувала поступово протягом останніх 10 років, поки програмне забезпечення й процеси не досягли своєї зрілості. І вплив,

який DevOps уже виявляє й продовжить виявляти в майбутньому на інформаційні технології підприємства, необхідно ретельно проаналізувати з погляду всіх залучених учасників цього процесу.

На даний момент здійснення моніторингу є однієї із самих недооцінених областей при прийнятті методології DevOps. Часта зміна програмного коду міняє підхід до роботи служби моніторингу, роблячи щоденною нормою здійснення таких завдань, як контроль передачі потоків у режимі реального часу, відтворення історії подій, просунута візуалізація і т.д. Здійснення постійного і якісного моніторингу стає критично важливим компонентом усіх застосунків. Якщо ви не розумієте, чому моніторинг так важливий для DevOps, ви, швидше за все, не готові до тем змінам, які принесе із собою DevOps, а значить ваша компанія залишиться в аутсайдерах на конкурентному ринку, що стрімко розвивається.

1.2 Область застосування

Перш ніж ми поглибимося в міркування про те, як і чому DevOps драматично змінить вимоги до сучасних застосунків для моніторингу, давайте розберемося, як ця методологія вплинула на розробку застосунків. Зрештою, якщо ви не знайомі з інструментами розробки DevOps, вам складно буде зрозуміти, чому підхід до моніторингу повністю зміниться.

Рух DevOps зародилося в 2009 році, але його джерела йдуть ще глибше. Так, фахівці як приклад інструментарію-прабатька називають популярне кросплатформений клієнт-серверний застосунок Puppet, перша версія якого з'явилася ще в 2005 році. Люку Канису (Luke Kanies), нині засновникові компанії Puppet Labs і розроблювачеві на Ruby у той час, набридло вручну налаштовувати Linux і вносити зміни у файли конфігурації. Він мріяв про спосіб, який би дозволив надавати й налаштовувати Unix-подібні системи більш програмним і

відтвореним способом. Таким чином, він написав скрипт на Ruby, який робив це замість нього, і назвав його Puppet.

Пізніше на ринку з'явився аналогічний інструментарій, включаючи Chef, Ansible, Saltstack і багато інші. Крім того, навколо цих інструментів утворювалися співтовариства; розроблювачі й системні адміністратори акумулювали свої знання у вигляді «рецептів», що дозволяють вам легко й швидко настроїти програмне забезпечення незалежно від використовуваного основного дистрибутива Unix.

За допомогою подібного інструментарію розроблювачі можуть створювати самодостатні програмні сценарії того, як запустити застосунок. Вони можуть включати всі залежності й запускатися на різних дистрибутивах Unix за допомогою простого запуску скрипту. Те, що раніше займало кілька тижнів налаштування в ручному режимі й виконувалося тільки висококваліфікованими фахівцями, у цей час робиться в лічені години за допомогою звичайного скрипту.

Хоча розроблювачі й одержали можливість розвертати свій код швидше й простіше, чим раніше, вони зіштовхнулися з іншою проблемою. Тому що програмісти стали менше залежати від своїх колег з операційного відділу, на розроблювачів лягла більша відповідальність за працездатність їх власних запущених застосунків.

Ця проблема спричинила поява на ринку ще одного інструментарію, який згодом одержав широке поширення: модель надання хмарних обчислень «платформа як послуга» (Platform as a Service, PaaS). Хоча спочатку даний інструментарій просувався такими компаніями, як Salesforce і Google, перші реалізації PaaS вимагали від розроблювачів писати спеціалізований код, який замикав застосунки на їхній платформі. PaaS не був широко популяризований, поки компанія Heroku (в 2010 році поглинена Salesforce) не представила однойменну хмарну PaaS-платформу, що підтримує ряд мов програмування. З її допомогою розроблювачі могли запускати свій код лише з невеликою кількістю істотних модифікацій, якщо такі взагалі були необхідні.

Системи PaaS виявилися на вершині принципів автоматизації DevOps. Більше того, у наші дні більшість представлених на ринку PaaS-платформ використовують DevOps-інструменти для налаштування й запуску. Різниця в тому, що на PaaS-платформі застосунку, які запущені на ній, повністю керовані. Ви можете запускати, зупиняти, а також здійснювати масштабування й моніторинг застосунків у рамках PaaS-платформи через інтерфейс програмування застосунків (Application Programming Interface, API). В DevOps ви можете створити набір інструментів для керування вашими застосунками, а з P aaS ви одержуєте вже попередньо налаштований інструментарій.

І, нарешті, зараз жодне з обговорень DevOps не обходитися без згадування Docker і Linux контейнерів. Мабуть, найбільшим недоліком використання моделі надання хмарних обчислень PaaS є те, що вона дуже строго визначає архітектуру застосунку. І якщо ви прагнете одержати більше контролю над середовищем виконання, саме Linux контейнери вам її нададуть, не вплинувши при цьому на швидкість і гнучкість виконання коду. Створення середовища з нуля за допомогою інструментарію Chef або Puppet може зайняти кілька годин, причому ви не зможете бути до кінця певен у тому, що в підсумку ви працюєте з точною її копією. Використання ж Linux контейнерів дозволить розроблювачам у лічені секунди відтворити середовище Linux, при цьому ви можете бути впевненими, що це її точна копія.

Інженери одержали величезний виграш до своєї продуктивності за рахунок багаторічного розвитку й поліпшення інструментарію DevOps. Дні розроблювачів, що не обертають увагу на проблеми операційного функціонування й масштабування своїх застосунків, підходять уводити, увести до ладу кінцю, тому що мистецтво запуску застосунків неухильно перетворюється в повністю автоматизований комп'ютерний код.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи мережевих систем моніторингу на базі DevOps-архітектури, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми кваліфікаційної бакалаврської роботи

Чому важливий моніторинг? Команди розроблювачів кодять швидше, продукт потрібно тестувати оперативніше, робити релізи скоріше й не просідати в рівні якості. Але часті зміни коду тягнуть лавинообразн ий ріст помилок. Для одержання повної картини необхідно мати проактивний моніторинг ПЗ .

Інструменти для моніторингу

Гарна моніторингова платформа дозволить відстежити продуктивність усієї системи й застосунків локально, у хмарі або в контейнерних середовищах. Ефективний набір інструментів підвищить продуктивність і допоможе скоротити або навіть усунути час простою. Ви зможете планувати відновлення й нові проекти, краще розподіляти час і інші ресурси, а найважливіше – виявляти й вирішувати проблеми до того, як їх побачить користувач.

Інструменти:

- Sensu – гнучкий і масштабований застосунок для перевірки працездатності телеметрії й служб. Використовується для моніторингу серверів, контейнерів, застосунків, функцій і підключених пристроїв.
- Prometheus – інструмент із вбудованою базою даних, що використовує pull-метод для збору інформації.
- Nagios – старий надійний інструмент для моніторингу комп'ютерних систем і мереж з відкритим вихідним кодом моніторинг, що задав моду на, у цілого покоління інженерів.

Керування конфігураціями

Засобу керування конфігами дозволяють автоматизувати підготовку й розгортання систем, примусово застосовувати необхідні конфігурації й усувати баги. Моделюючи інфраструктуру коду, ви можете застосовувати контроль версій, автоматизоване тестування й безперервну доставку в застосунки. Автоматизація рутинної, підданої помилкам роботи приводить до підвищення швидкості, передбачуваності й масштабованості. Також забезпечуються стандартизовані конфігурацій в test, dev і prod середовищах.

Інструменти:

– Ansible – інструмент на Python. Застосовується без агента й використовує імперативний підхід.

– Chef – інструмент на Ruby, імперативний підхід керування конфігураціями.

– Puppet – декларативний підхід, використовує специфічний «доменний» мову й архітектуру агент/майстер.

Інструменти оповіщення

Система оповіщення повинна бути досить чутливої, щоб реагувати на перебої, але не настільки чутливої, щоб спамити вас повідомленнями про дрібні падіння, які не помітять користувачі. Інструменти оповіщення допомагають закласти основу для політик безпеки, а також настроїти: кого повідомляти, як мониторити і як установлювати пріоритети відновлення.

Інструменти:

– Pagerduty – ситуаційна платформа керування з доповненнями для аналітики, аналізу подій і автоматичного реагування на інциденти.

– Servicenow – використовує автоматизовані робочі процеси для ITSM (IT Service Management), обслуговування клієнтів і бізнес-процесів.

– Slack – можна повісити оповіщення на ту ж платформу, яку ви використовуєте для групових чатів і спільної роботи.

Метрики

Як тільки ви автоматизуєте керування конфігурацією, настроїте оповіщення й моніторинг, у вашому розпорядженні будуть дані, які можна аналізувати. Встає питання: « як надійно зберігати й аналізувати інформацію?». Потрібна система зберігання, яка дозволить поєднувати й вивчати показники системи, поведінка користувачів, рівень обслуговування й ризики.

Інформація, одержувана з метрик, допомагає ухвалювати рішення на всіх рівнях бізнес-процесу, поліпшуючи вашу здатність відповідати SLA (Service Level Agreement), задовольняти очікування клієнтів і обґрунтовувати потребу в інвестиціях.

Інструменти:

– Influxdb – база даних для роботи з тимчасовими рядами, що підходить для довгострокового зберігання даних.

– Splunk – використовує модель бази даних, як у пошукових системах.

Візуалізація

Інструменти візуалізації можуть розглядатися як основу для набору інструментів, використовуваних при моніторингу. Ви можете поєднувати дані, сортувати й візуалізувати їх у різних панельках. Тонке налаштування дозволить команді створювати й спільно використовувати власні наробітки в області моніторингу.

Інструменти:

– Grafana – може використовуватися поверх різних сховищ даних, включаючи Graphite, Influxdb і Elasticsearch.

Наступні кроки: оцінка поточних інструментів

Незалежно від того, де ви перебуваєте у своєму DevOps-подорожі, розумно переглянути поточну ситуацію й визначити місця, у яких можна щось змінити. Добірка інструментів DevOps і метод їх використання визначає ваші звички, культуру командної роботи, якість вашого продукту й цінність, яку ви приносите своїм користувачам.

Рисунок 2.1 – Інтерфейс користувача Grafana

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000

виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64 -bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.
- Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на кваліфікаційну бакалаврську роботу, реалізації підлягає програмне забезпечення, яке призначено для системи мережевих систем моніторингу на базі DevOps-архітектури.

В процесі розробки кваліфікаційної бакалаврської роботи 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

Хоча розроблювачі й стояли в джерел революції DevOps, системні адміністратори й фахівці з технологічних операцій стали ключовою ланкою його популяризації. Зрештою, цей інструментарій допомагає підвищувати ефективність і їх роботи, а не тільки програмістів. У підсумку використання інструментів DevOps трансформувало зону відповідальності сучасної, гнучкої команди фахівців з технологічних операцій і внесло кардинальні зміни у виконувану ними роботу.

До появи методології DevOps системні адміністратори й інженери по технологічних операціях підтримували окремі застосунки у великому масштабі. Це вимагало виконання таких завдань, як налаштування баз даних і веб-серверів, налаштування балансування навантаження, керування безпекою, керування системами кешування й проведення багатьох інших робіт.

Інструментарій DevOps забезпечив високий ступінь стандартизації, пропонуючи ефективні способи розгортання, налаштування й запуску багатьох серверів за допомогою всього лише декількох автоматизованих інструментів, а не покладаючись на втручання операторів. Усе частіше роль команди по технологічних операціях переорієнтується на розгортання й обслуговування автоматизованих застосунків-сервісів, надаваних на вимогу, приміром, через модель PaaS або кластер контейнерів Linux. Розроблювачі розгортають і масштабують індивідуальні застосунки в мережі пристроїв, залишаючи команді системних адміністраторів запуск і масштабування цих мереж.

DevOps платформи й інструменти створили буфер, який дозволив командам розроблювачів і фахівцям з технологічних операцій працювати незалежно друг від друга, а не залежати друг від друга. До появи цього

інструментарію існував якийсь послідовний конвеєр роботи над програмним забезпеченням. Так, перш ніж програмне забезпечення виходило остаточно розгорнути, окремі застосунки для їхнього використання потребували додаткових індивідуальних налаштувань і доробках уже після здійснення закупівлі, а сервера необхідно було зконфігурувати.

З появою інструментів DevOps розроблювачі одержали певні квоти, у рамках яких змогли розвертати застосунки на вимогу й у режимі реального часу. Системним адміністраторам більше не потрібно було турбується про розгортання окремих застосунків. Так, команди по технологічних операціях як і раніше здійснювали закупівлю устаткування, а також налаштовували й управляли серверами, але робили вони це на набагато більших масштабах, чому рівень окремого програмного забезпечення. Їхній обов'язок став полягати в керуванні автоматизованими DevOps-Мережами, до яких розроблювачі одержали більш вільний доступ.

Цей технологічний буфер забрав строгу послідовність із життєвого циклу розробки й впровадження програмного забезпечення, дозволивши розроблювачам і системним адміністраторам працювати більш тісно один з одним. На перший погляд може здатися, що прибирання тісного взаємозв'язку між двома командами для їхньої більшої інтеграції суперечить здоровому глузду. Дозволивши розроблювачам працювати в рамках чітко певної системи (наприклад, побудова коду усередині віртуальної мережі пристроїв), у той час як команда фахівців з технологічних операцій працює поза контейнером Linux, гарантує, що ваші програмісти одержали й займаються різними завданнями.

Але золотою серединою між розроблювачами й системними адміністраторами стала спільна екосистема (самі інструменти DevOps), де в наші дні стирається існуюче раніше чітке розмежування обов'язків. Команди фахівців з технологічних операцій одержують більш глибокі знання про кращих практиків запуску й підтримки складних програмних систем. А розроблювачі, у свою чергу,

одержують більше можливостей для навчання комп'ютерів, впроваджуючи більш глибокі знання про реалізацію технологічних процесів без втручання людини.

Тому не дивно, що роботодавці все частіше шукають інженерів з розумінням технологічних операцій, а системних адміністраторів з досвідом у програмуванні. У цілому ж, однак, найбільший вплив, який DevOps виявляє на команди фахівців з технологічних операцій, полягає в тому, що на системних адміністраторів усе більше лягати відповідальність за запуск мережі пристроїв, який забезпечить автоматичне розгортання програмних опцій на вимогу, реалізованих розроблювачами.

Як DevOps змінює підхід до моніторингу?

Як ви, швидше за все, уже встигнули переконатися, методологія DevOps багато в чому є сучасним еволюційним процесом, породженим зі старого способу ведення справ. Ця методологія сприяє появі все більшої кількості автоматизованих процесів на всіх рівнях життєвого циклу програмного забезпечення, тому час виходу на ринок нових застосунків значно зменшується. Тем ні менш, усі ці зміни приводять до серйозних наслідків. Через тектонічне зрушення у вимогах до розробки, тестування й розгортанню програмного забезпечення міняються потреби у функціональних можливостях сучасних хмарних систем моніторингу.

DevOps прискорює весь життєвий цикл програмного забезпечення від розробки до процесів забезпечення якості. Відносно статичне раніше прикладне програмне забезпечення сьогодні може обновлятися до декількох раз у день. Це, у свою чергу, ставить перед розроблювачами й системними адміністраторами безліч завдань, частина з яких старі, а частина – нові.

Розроблювачам довелося адаптуватися до нових умов у тому числі за рахунок написання більш детальних автоматизованих тестів для свого коду, у такий спосіб процеси QA стають усе більш автоматизованими, наскільки це взагалі можливо. Забезпечення якості було поставлено у тверду залежність від безперервної інтеграції, яка автоматично запускає всі програмні елементи й

інтеграційні тести щораз, коли передається новий код. Тому в цей час системи моніторингу стають усе більш підготовленими до роботи з кожною частиною всього набору необхідних інструментальних засобів DevOps.

До прийняття методології DevOps нові відновлення програмного забезпечення ретельно тестувалися й ухвалювалися висококваліфікованими технічними фахівцями. Безперервне розгортання разюче поміняло прийнятий раніше підхід. Зараз процес будується на максимальній автоматизації всього інструментарію DevOps, і код уводиться в роботу щораз, як тільки він проходить усі автоматизовані тести.

Якщо такий підхід представляється вам диким, і ви припускаєте, що його застосовують в основному до найменш малих і незначних застосунків, то вам варто довідатися, що, приміром, компанія Facebook уже довгий час є прихильником такого роду гнучкого розгортання систем. Будь-який програміст знає, що, якщо його код якимось образом поламає Facebook, це буде відслідковано до нього через історію керування версіями, і він понесе за це відповідальність.

Але багато організацій не можуть дозволити собі сліпо довіряти нікому чорному ящику, який автоматично розвертає код, робота якого з вашої сторони контролюється тільки на рівні надії на досвід інженерів-розроблювачів. Правильно реалізовані системи моніторингу хмарних сервісів можуть забезпечити вам настільки необхідне розуміння всіх процесів, допомагаючи вам перетворити автоматизований програмний Дикий Захід у щось те саме що Центру керування космічними польотами.

Сучасні системи моніторингу забезпечать вам більше розуміння в режимі реального часу функціонування кожної частини застосунку, чому коли-або раніше. Розроблювачі сучасного програмного забезпечення пишуть керований через інтерфейси API код, що привело до того, що ці ж інтерфейси програмування застосунків зараз стали доступні для систем моніторингу. Також

багато сервісів моніторингу можуть використовувати спеціальні кодові зачіпки в самій логіці програмного забезпечення.

Крім того, сервіси моніторингу розширили своє уваги від середовища виробничої експлуатації до всього набору процесів виходу застосунку. Вони зараз торкаються етапів компіляції, тестування модульних частин, інтеграційні тести, перевірку того, як код виконується при великому навантаженні й багато чого іншого. Приміром, у компанії Google сервіси моніторингу розгортання програмного коду навіть знають, як стежити за програмним забезпеченням для керування проектами, виглядаючи й відзначаючи окремі файли, які мають статистично більша кількість повідомлень про помилки під час розробки й тестування, чому інші, позначаючи їх як гарячі точки, щоб стежити за ними в майбутньому.

Належний моніторинг в DevOps є активним, а не просто реактивним. Він знаходить способи поліпшити якість ваших застосунків, перш ніж проблеми навіть дадуть про себе знати. Тому що такі системи моніторингу також стежать і за набором інструментів DevOps, це може допомогти поліпшити цей інструментарій за рахунок виділення областей, які, можливо, потребують більшої автоматизації.

Коли у вас є складний застосунок, який обновляється й розвертається по декілька раз на день і перетерплює швидкі цикли QA, вам би хотілося б ідентифікувати проблеми настільки швидко, наскільки це можливо. Здійснення складного моніторингу стає першою лінією захисту від простоїв. Таким чином, системи моніторингу повинні постійно розвиватися, щоб брати до уваги всі нові дані. Це приводить нас до дуже важливого питання. У чому різниця між сервісами моніторингу старої школи й сервісами моніторингу готовими до DevOps? Якщо ви помилитеся у виборі інструментарію для здійснення моніторингу, ви зіштовхнетеся із частими й тривалими простоями й упустите конкурентні переваги від застосування нових, набагато більш гнучких технологій.

Еволюція сучасних хмарних систем моніторингу

Очевидно, що багато чого змінилося в сучасному підході до побудови моделі життєвого циклу програмного забезпечення і його розгортанню, але багато постачальників застосунків для моніторингу усе ще не встигнули адаптуватися до нових умов і застрягли в минулому. На кожного вендора хмарної системи моніторингу, готового до DevOps, існують ще десяток, які не відповідають усім вимогам сучасного ринку. Тому дуже важливо знати, на що звертати увагу, коли ви оцінюєте представлені на ринку застосунки для моніторингу.

У сучасних DevOps-архитектурах комплексного програмного забезпечення існує безліч даних, які необхідно відслідковувати. Більше недостатньо відслідковувати тільки найпростіші статистичні дані, такі як RAM, CPU і дисковий I/O. Тепер ваш застосунок для моніторингу повинне працювати з API і передавати дані безпосередньо із самих застосунків. Для того, щоб зрозуміти всю цю інформацію, одним із критеріїв пошуку потрібної вам сучасної системи моніторингу повинні стати можливості аналізу в реальному часі потокових даних, відтворення історії змін і ще кращі засоби візуалізації.

Інструменти візуалізації, зокрема, мають важливе значення для цілісного розуміння стану всіх ваших застосунків. Можливість виявляти проблеми в гнучкій DevOps-середовищі найчастіше залежить від якості ваших інструментів візуалізації. Відстеження проблем способом перевірки окремих балка-файлів, коли у вас з'явилося настільки багато мінливих частин, більше не є ефективною стратегією роботи фахівців з технологічних операцій.

Наступним найбільш важливим способом оцінити хмарну систему моніторингу є кількість і якість підтримуваних модульних інтеграцій. Роботу зі скількома мовами програмування підтримує система моніторингу? Багато старих систем моніторингу працюють лише з декількома, з особливим фокусом на Java і .Net, незважаючи на те, що високорівневі мови програмування стають усе більш важливими для розробки корпоративних застосунків. Швидше за все, якщо не

прямо зараз, те найближчим часом ви захочете собі нову систему моніторингу, яку можна буде зав'язати на популярні сценарні мови, такі як Python, Ruby, PHP і Go.

Чи дозволяє програмне забезпечення для моніторингу підключатися безпосередньо до інструментів управління конфігурацією, таким як Puppet і Chef. Чи підтримує воно зв'язок з програмним забезпеченням, розміщених на PaaS-платформах, таким як Cloud Foundry і Openshift А як відносно Docker и Linux контейнеров?

По суті, один погляд на те, як програмне забезпечення для моніторингу підтримує роботу з Docker контейнерами відразу скаже вам дуже багато про підтримку інших сучасних популярних інструментів. Хоча програмна платформа Docker і є одним з новітніх інструментів DevOps, вона вже встигнула застолбити за собою звання одного з найбільше швидко зростаючих сервісів з погляду прийняття ринком.

В «важкому» сегменті програмного забезпечення рівня підприємства дотепер домінують традиційні інструменти й інфраструктури, такі як Java, .Net, Oracle, IIS, Websphere і Microsoft SQL Server. Але припускати, що хмарним системам моніторингу досить буде підтримувати роботу тільки із цими «старожилами» ринку, буде величезною помилкою. Нові програмні стеки все частіше «стають більш важкими» за допомогою Linux, PHP, Python, Ruby, Perl, Go, Nginx, Apache, Redis, Memcached, MySQL й PostgreSQL. Навіть великомасштабне програмне забезпечення для автоматизації виробничого процесу зараз усе частіше починає використовувати цей інструментарій. Як і найбільші світові корпорації: Facebook, як відомо, побудований на PHP, Google часто й багато використовує Python і Go.

У недалекому майбутньому корпоративне програмне забезпечення стане набагато більш різним, що ми можемо це спостерігати сьогодні. Провідні виробники хмарних систем моніторингу повинні з розумом підійти до цих змін, щоб підготуватися до нових вимог ринку. Щоб не канути в лету, їм потрібно

охопити всі прийдешнє багатомовне майбутнє й розробити інструменти, які можна масштабувати зі швидкістю гнучких DevOps-Розгортань.

Знамениті слова Марка Андриссена (Marc Andreessen) про те, що «програмне забезпечення пожирає мир», однаково слушні для процесів запуску й моніторингу застосунків, незалежно від того, використовуються останні для виклику таксі або бронювання місць у готелі. Десятиліттями завдання запуску програмного забезпечення було привілеями елітної касти висококваліфікованих геніїв – системних адміністраторів. Але зараз це стало мейнстримом.

Раніше більша частина використовуваних для цього знань не була задокументована й передавалася в усній формі, як народний фольклор, з покоління в покоління. Усі файли конфігурації мали свої власні ексцентричні формати. Іноді ці формати були настільки дивні, що одні конфігураційні файли компілювалися в інші додаткові конфігураційні файли. Знання ж, як саме й навіщо налаштовувати ці файли, залишалися загадкою для всіх, крім декількох вибраних.

DevOps став причиною ретельного документування всіх цих езотеричних операційних знань у відкритих стандартах, що дозволяють записати й відстежити всі процеси й метрики протягом часу. Ці знання можна запрограмувати за допомогою логіки, використовуваної для побудови сітки програмного забезпечення на платформах, таких як PaaS і кластери контейнерів Linux, і їх можна повторювати й сполучати з іншими.

У той час як розроблювачі починали усе краще розбиратися в технологічних операціях, а системні адміністратори – у програмуванні, їх обов'язки, в остаточному підсумку, дуже тісно переплелися. Границі остаточно розмилися, і це відкрило дорогу для появи нового інструментарію для спільної роботи між групами, одна з основних цілей використання якого полягала в мінімізації часу взаємодії всіх залучених у процес груп.

Docker платформа, що лідирує на ринку, на основі контейнерів Linux для автоматизації розгортання й керування застосунками в середовищі віртуалізації

на рівні операційної системи, є ще одним прикладом цієї еволюції. Github, найбільший веб-сервіс для хостинга IT-проектів і їх спільної розробки, надав розроблювачам відкрите середовище для спільного використання коду один одного й більш легкого співробітництва, чому коли-або колись. Хмарний сервіс Docker Hub тільки зараз відкриває подібного виду інструменти для спільної роботи для системних адміністраторів, закладаючи початок для нового підходу в керуванні конфігураціями з відкритим вихідним кодом.

Інструменти для здійснення моніторингу повинні взяти до уваги всі ці зміни. Вони повинні розвиватися паралельно з технічними тенденціями й, у підсумку, зуміти задовольнити потреби як розроблювачів, так і системних адміністраторів, надавши необхідну видимість для прийняття нових технологій, таких як Docker, у виробниче середовище.

Через усе більш стислі життєві цикли програмного забезпечення, наявність правильної системи для здійснення хмарного моніторингу в режимі реального часу стає критично важливим наріжним каменем запуску й використання інструментів DevOps. Розуміння всіх мінливих частин і їх взаємодії між собою дасть вам необхідну основу, щоб розв'язати, який застосунок для моніторингу необхідно вашої організації.

Методологія DevOps продовжить розвиватися. Там завжди найдеться місце для нових інструментів, різних ст руктур і крутих трендів, але основа, яка буде їх усе разом зв'язувати, полягає у виборі й правильному використанні програмного забезпечення для автоматизації.

3.2 Розробка структурної схеми

Вирішити, який інструмент використовувати – завжди непросте завдання – вам необхідно порівняти всі доступні інструменти і їх пропозиції з різних точок зору. Нижче наведені декілька факторів, які відіграють важливу роль при виборі доступних варіантів або при створенні власного застосунку:

– Набір підтримуваних інструментів (плагіни або збирачі даних).

– Безкоштовне або платне.

– Підтримка налаштування збору даних.

– Багатий користувацький інтерфейс для представлення даних і налаштування полів і представлень.

– Підтримуваний діапазон показників.

– Легкість установки й налаштування.

– Політика ліцензування й поширення.

Підхід до застосунку для побудови вашого моніторингу DevOps

Якщо ви вирішили створити або розширити який-небудь існуючий інструмент, нижче наведені деякі міркування:

– Визначите інструменти, які ви повинні відслідковувати: перелічите інструменти DevOps, які ви повинні відслідковувати; список повинен охоплювати всі області DevOps (складання, репо, якість коду, розгортання, моніторинг і т.д.), щоб одержати повну картину.

– Визначите метрики: як тільки у вас буде список інструментів, визначите важливі метрики для кожного інструмента. Ваші метрики повинні забезпечувати KPI областей DevOps.

– Визначите API або інтерфейс командного рядка для одержань даних метрики: вам необхідно добре розбиратися в інструментах і їх реалізації, щоб визначити кращий спосіб одержань даних. Це можуть бути визначені API, REST API або будь-який інтерфейс командного рядка. Переконаєтеся, що ви одержуєте всі необхідні дані.

– Зіставте метрики з бізнес-цінністю або визначите DevOps KPI: ваші метрики повинні представляти KPI і бізнес-цінність усіх інструментів, задіяних у досягненні DevOps. Панель керування повинна охоплювати всі аспекти й представляти поточний статуси й пробіли.

– Визначите, як зберігати й обробляти ці метрики: вам потрібно визначити схему для зберігання даних у певній базі даних; ваша схема повинна

бути досить загальної, щоб обслуговувати будь-які дані з різних інструментів. Для обробки даних вам знадобиться проміжний компонент між базою даних і користувацьким інтерфейсом для обробки в презентабельному форматі.

– Створіть табло на основі показників. Використовуйте багатий графічний інтерфейс для відображення табло в різних форматах, таких як таблиці, графіки й діаграми.

– Підтримка високої доступності: інструмент повинен бути розроблений для підтримок високої доступності, щоб уникнути простоїв і втрати крапок даних.

Як інструменти моніторингу збирають дані

Більшість інструментів, які ми використовуємо для реалізації DevOps і CI / CD, являють собою веб-інструменти, такі як Jenkins, Sonar, Git, SVN, Udeploy, Gerrit, Jira, Confluence або інші. Ці інструменти надають REST API, з яких ми одержуємо дані в описаному форматі – або в json, або в xml. Деякі інструменти підтримують визначені API або інтерфейси командного рядка для підключення й одержання даних. Як тільки ви визначите механізм зв'язку з кінцевою системою кроком, що випливає, буде написання надійного коду, який зможе витягати необхідні дані метрик і обробляти всі випадки, щоб аналізувати дані й поміщати їх у реляційну або нереляційну базу даних.

Коли у в ас є дані в базі даних, вам знадобиться графічний інтерфейс для відображення їх у табличному й графічному форматах і компонент проміжного програмного забезпечення, який може витягати дані з бази даних і надавати їхньому графічному інтерфейсу в бажаному форматі.

Одержання даних із системи, яку ви прагнете відслідковувати, є реальною проблемою у всій цій вправі, тому що все інше залежить від цього. Нижче наведено кілька проблем, з якими ви можете зіштовхнутися при розробці системи моніторингу ланцюжка інструментів:

– Визначите API або будь-який інший механізм для одержань даних.

– Визначите метрики для моніторингу.

– Визначите, як ви прагнете, щоб дані були представлені в графічному інтерфейсі користувача – будь те діаграма тенденцій, кругова діаграма або просто таблична вистава.

– Визначите механізм автентифікації для системи моніторингу, оскільки кожна з них підтримує різні типи. Іноді потрібно визначений токен, а в інших випадках токен необхідно згенерувати під час виконання.

Проста архітектура моніторингу конвеєрів DevOps

Нижче представлена структурна схема у вигляді загальної архітектури, на якій працює більшість інструментів моніторингу. На рисунку 3.1 різні інструменти DevOps перебувають ліворуч; ці інструменти – кінцева система, звідки ми прагнемо доправити дані. Як приклад показано лише кілька інструментів.

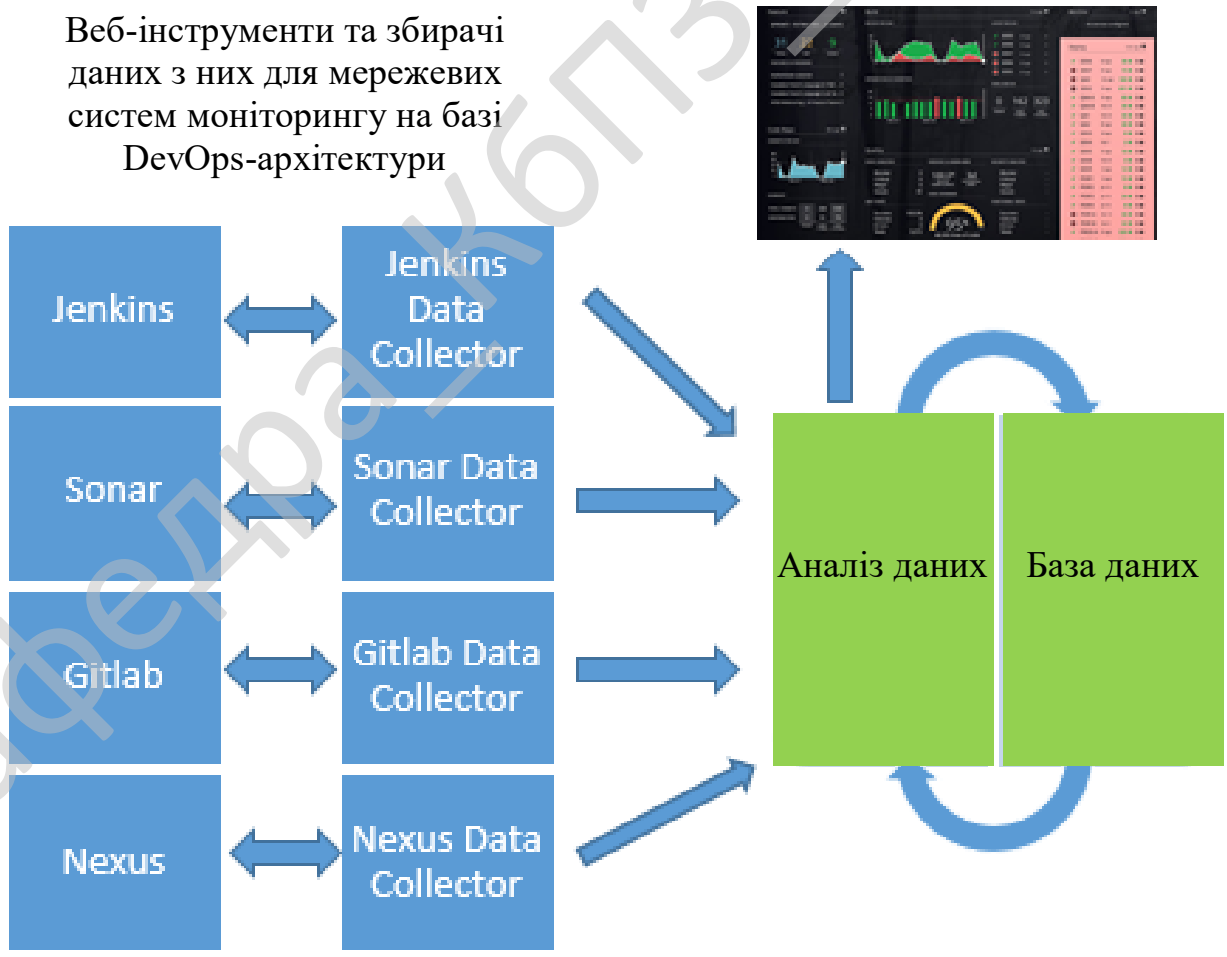

Рисунок 3.1 – Структурна схема системи

Збирач даних приносить дані з кожної кінцевої системи й обмінюється даними з кінцевою системою за допомогою REST API або будь -якого API, надаваного цим інструментом або CLI, залежно від випадку. Використовуючи певний механізм зв'язку, він доправить дані й передасть їхньому синтаксичному аналізатору, щоб одержати необхідні дані й помістити їх у базу даних у бажаному форматі.

При розробці збору даних вам необхідно добре розуміти інструмент, який ви збираєтеся відслідковувати, а також доступні API, включаючи REST, CLI, SMI або будь-який заздалегідь певний API, створений у цьому інструменті.

Рівень аналізу даних буде витягати бажане значення метрики з неопрацьованих даних, зібраних збирачем даних. Неопрацьовані дані можуть бути в будь-якому форматі, включаючи пари «ключ-значення», json, xml або csv. Методологія синтаксичного аналізу буде відрізнятися в кожному випадку залежно від типу даних. Ви можете використовувати будь-який доступний загальний API для аналізу.

Ви повинні знайти гарний API для аналізу даних, доступних в іншому форматі й різної складності.

База даних буде містити дані метрик кожного інтервалу для кожного інструмента й облікових даних користувача.

Графічний інтерфейс запросить дані в компонента синтаксичного аналізу. Він буде представляти його в табличному або графічному форматі, зручному для користувача.

Інші функції: коли в нас є дані, ми можемо мати графік тенденцій для прийняття деяких важливих застосунків, ми можемо генерувати сигнал тривоги при порушенні будь-яких певних граничних значень і інтегрувати його з електронною поштою або SMS, щоб вжити негайних заходів.

Інструмент моніторингу високої доступності

Якщо ми покладаємося на наш інструмент моніторингу для одержання звітів про всі інструменти, використовувані в DevOps, ми не можемо дозволити

собі простої. Ми можемо втратити крапки даних і тенденції, якщо інструмент моніторингу вийде з ладу, і керівництво може не одержати фактичний статус усього проекту без необхідності вручну збирати дані.

Щоб уникнути такої ситуації, в інструменті моніторингу слід реалізувати високу доступність (HA). У випадку налаштування високої доступності ми продовжимо збір даних, якщо вузол моніторингу вийде з ладу.

Щоб реалізувати HA, нам необхідно розглянути два аспекти: HA GUI і проміжного програмного забезпечення (компонент бізнес-логіки) і HA бази даних.

Висока доступність у графічному інтерфейсі й проміжному програмному забезпеченні може бути досягнута за допомогою кластера Apache або Nginix. У випадку контейнерних середовищ ми можемо використовувати будь-яку функцію платформи оркестровки за замовчуванням для досягнення високої доступності, таку як Kubernetes. Точно так само в базі даних ми можемо реалізувати функцію кластеризованної бази даних, надавану більшістю доступних баз даних.

Моніторинг інструментального ланцюжка DevOps дуже корисний. Він надає керівництву гарний знімок стану проекту й може визначити будь-які проблеми із блокуванням, які можуть вплинути на випуск, щоб їх можна було виправити. Моніторинг може допомогти виявити болючі крапки, на яких слід зосередитися при визначенні дорожньої карти. Організації, які додержуються практики DevOps і використовують інструменти DevOps, повинні мати інструмент для моніторингу конвеєра DevOps.

3.3 Розробка функціональної схеми

Перейдемо до розгляду функціональної схеми. Сформулюємо й вирішимо завдання моніторингу на базі DevOps-архітектури. Для цього розроблена концепція проведення аналізу, вирішені завдання формального опису структури й обчислення її характеристик: навантаження на канали зв'язку, вузли й

структуроутворююче устаткування мережевих систем моніторингу на базі DevOps-архітектури.

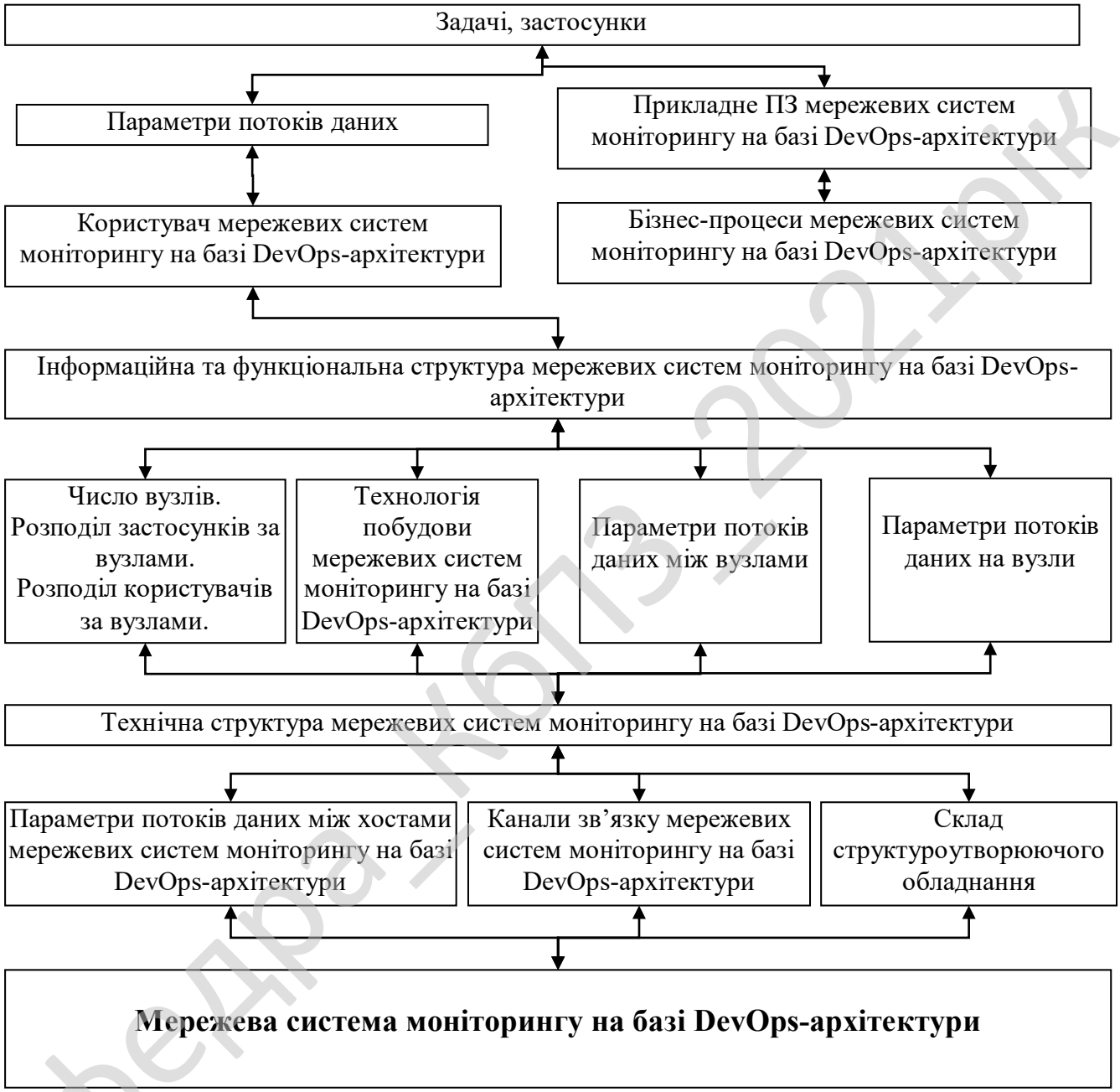

Рисунок 3.2 – Функціональна схема системи

Основною метою аналізу структури є визначення параметрів потоків даних, що проходять по каналах зв'язку мережевих систем моніторингу на базі DevOps-архітектури й вступні на вузли мережевих систем моніторингу на базі

DevOps-архітектури. Однак, завдання структури мережевих систем моніторингу на базі DevOps-архітектури тільки як сукупності вузлів і зв'язків між ними, не дозволяє досліджувати потоки даних, оскільки потоки даних формуються розв'язуваними на мережевих систем моніторингу на базі DevOps-архітектури завданнями, точніше додатками, які запускаються на вузлах мережевих систем моніторингу на базі DevOps-архітектури й обмінюються між собою даними. Отже, для аналізу мережевих систем моніторингу на базі DevOps-архітектури необхідно відомості про функціональну структуру доповнити відомостями про застосунки і їхнє розміщення в мережевих систем моніторингу на базі DevOps-архітектури.

Результатами аналізу повинні стати математичні залежності, що дозволяють обчислювати чисельні значення характеристик мережевих систем моніторингу на базі DevOps-архітектури з урахуванням особливостей конкретної структури мережевих систем моніторингу на базі DevOps-архітектури.

Запропоновано концептуальний підхід до аналізу функціональної структури мережевих систем моніторингу на базі DevOps-архітектури, заснований на дослідженні взаємодії застосунків (завдань) як незалежних джерел і приймачів даних. У цьому випадку спочатку визначаються параметри потоків даних між додатками при виконанні всього комплексу завдань (будується інформаційна модель), а потім, залежно від розміщення застосунків по вузлах мережевих систем моніторингу на базі DevOps-архітектури й використовуваного устаткування, визначаються параметри потоків даних між вузлами мережевих систем моніторингу на базі DevOps-архітектури (будується технічна модель). При цьому не тільки повністю враховуються всі взаємодії між додатками, але й з'являється можливість проведення аналізу складних ієрархічних мережних структур, шляхом декомпозиції на підмережевих систем моніторингу на базі DevOps-архітектури, що часто застосовується в технологіях VLAN і VPN. Даний підхід розвивається й застосовується в бакалаврській дипломній роботі.

Відзначимо, що отримані результати аналізу потоків даних, можна надалі використовувати для проведення більше глибоких досліджень із застосуванням відомих методів і моделей, наприклад, СеМО.

Для аналізу функціональної структури й розрахунку характеристик мережевих систем моніторингу на базі DevOps-архітектури визначені правила її формального опису, що однозначно задають властивості застосунків і структуру мережевих систем моніторингу на базі DevOps-архітектури, що дозволяють будувати моделі для проведення розрахунків.

Відповідно до концептуального підходу до проведення моніторингу на базі DevOps-архітектури виділені дві складові її структури інформаційна й технічна:

– Інформаційна структура визначає інформаційні потоки між інформаційними вузлами, на яких установлене програмне забезпечення й представляє сукупність вузлів і інформаційних ресурсів, розміщених на цих вузлах. Маючи у своєму розпорядженні дані про інформаційну структуру мережевих систем моніторингу на базі DevOps-архітектури можна приймати рішення про організацію каналів зв'язку між вузлами мережевих систем моніторингу на базі DevOps-архітектури, визначати необхідні параметри телекомунікаційної системи, формувати структуру мережевих систем моніторингу на базі DevOps-архітектури.

– Технічна структура – сукупність структуроутворюючого устаткування, технічних вузлів мережевих систем моніторингу на базі DevOps-архітектури й каналів зв'язку. Технічному вузлу можуть відповідати трохи інформаційних.

Таким чином, для повноцінного аналізу функціональної структури реальної мережевих систем моніторингу на базі DevOps-архітектури необхідно провести аналіз складових її інформаційної й технічної структур і зв'язати результати аналізу.

Зв'язування результатів аналізу інформаційної й технічної структур має на увазі відображення характеристик інформаційної структури в характеристики

технічної структури й визначення параметрів технічної структури на основі параметрів і характеристик інформаційної структури.

На рисунку 3.2 представлена функціональна схема системи, які відповідає запропонованому підходу.

Як міра ефективності ієрархічної структури запропоновані коефіцієнти поглинання інтенсивностей потоків даних на кожному рівні. Коефіцієнт поглинання на кожному рівні відбиває частку сумарної інтенсивності потоків, що локалізується усередині рівня.

Результати аналізу інформаційної структури дозволяють визначати параметри потоків даних між логічними об'єднаннями вузлів мережевих систем моніторингу на базі DevOps-архітектури. Інформаційна структура реалізується конкретними технічними засобами й втілюється у вигляді технічної структури. При цьому можливо невідповідність інформаційної й технічної структур, пов'язане з технічними характеристиками й можливостями апаратури. Наприклад, на одному технічному вузлі можуть бути встановлені кілька інформаційних вузлів. У зв'язку із цим розроблені методи й засоби аналізу технічної структури корпоративної мережевих систем моніторингу на базі DevOps-архітектури, створюваної на базі інформаційної структури.

Для аналізу роботи реальної мережевих систем моніторингу на базі DevOps-архітектури розроблений метод формального опису її технічної структури. Структура реальної мережевих систем моніторингу на базі DevOps-архітектури формується із застосуванням структуроутворюючого устаткування (комутатори, маршрутизатори), до якого підключаються вузли мережевих систем моніторингу на базі DevOps-архітектури, при цьому створюється багаторівнева (часто трьохрівнева) мережа. Такий підхід застосовується, як правило, при створенні корпоративної мережевих систем моніторингу на базі DevOps-архітектури на базі технології VLAN. При цьому групи першого рівня інформаційної структури становлять віртуальні локальні мережевих систем

моніторингу на базі DevOps-архітектури. Другий і третій рівні інформаційної структури призначені для з'єднання цих мереж між собою.

Число комутаторів, використовуваних для з'єднання вузлів технічної структури корпоративної мережевих систем моніторингу на базі DevOps-архітектури при створенні груп першого рівня (комутатори першого рівня), визначається особливостями реальної мережевих систем моніторингу на базі DevOps-архітектури, технічними можливостями комутаторів.

Якість рішення завдань залежить від того, яка частина пропускної здатності каналу (смуга) виділена для кожного завдання, що звичайно досягається шляхом застосування режиму гарантованої якості обслуговування (QoS). Тому при рішенні завдань розрахунку потоків із застосуванням систем з гарантованою якістю обслуговування проводиться розширення множини PST(ST) шляхом додавання множини коефіцієнтів поділу каналів – PKST. Це дозволяє визначити залежність характеристик мережевих систем моніторингу на базі DevOps-архітектури від смуги пропущення в каналі (комутаторі), що відводиться для завдання.

Також проведене узагальнення отриманих результатів для випадку мультисервісної мережевих систем моніторингу на базі DevOps-архітектури й для випадку, коли для кожної групи завдань застосовуються свої правила передачі даних.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Відповідно до методичних рекомендацій розроблення графічної частини кваліфікаційної бакалаврської роботи розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3.

Рисунок 3.3 – Діаграма взаємодії процесів

Розроблена діаграма взаємодії процесів використовується для представлення та візуалізації процесів обробки даних тобто структурного проектування бакалаврської роботи.

Основні складові елементи діаграми взаємодії процесів це потоки даних:

– Репозиторії, потік сховища даних.

– Потоки зовнішні по відношенню до системи сутності.

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

– Потоки даних гібридні між елементами трьох попередніх типів.

Відповідно до документації 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.

Для схематичного представлення системи що розробляється необхідно спочатку представити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи в цілому у подальшому. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Розроблена діаграма взаємодії процесів системи в подальшому уточнюється шляхом деталізації процесів та потоків даних з метою показати систему що розробляється. Таким чином у результаті після розгляду, вищеописаної системи, схеми структурної, функціональної, діаграми взаємодії процесів перейдемо до опису та розгляду блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо алгоритм роботи основної програми. Його блок-схема зображена на рисунку 4.1. З рисунку видно, що після запуску програми спочатку відбувається ініціалізація динамічних бібліотек системи далі проходе виведення вікна завантаження програми з зображенням завантаження модулів системи з подальшим виведенням основного вікна програми.

Потім здійснюється перевірка наявності файлу мережної взаємодії з послідуючим підключенням при знайдені та проходе пошук зі послідуючим створенням в системі об’єктів локальної чи глобальної мережі.

Далі проходе читання та формування точок моніторингу з виведенням на екран схеми мережі та отриманням з послідуючим аналізом мережних даних та записом до журналу роботи.

Після цих дій проходе запит на формування точок моніторингу з формуванням початкових даних та викликом підпрограми формування точок моніторингу та запитом збору статистики та викликом підпрограми збору статистики мережі.

Робота підпрограми у вигляді блок-схеми зображена на рисунку 4.2. Після проходження циклу роботи системи проходе запит на завершення ПЗ після чого ПЗ продовжує опрацьовувати запити чи завершує свою роботу.

Одна з під задач системи що розроблялась, а саме системи мережевих систем моніторингу на базі DevOps-архітектури є виведення списку комп'ютерів підключених до локальної мережі.

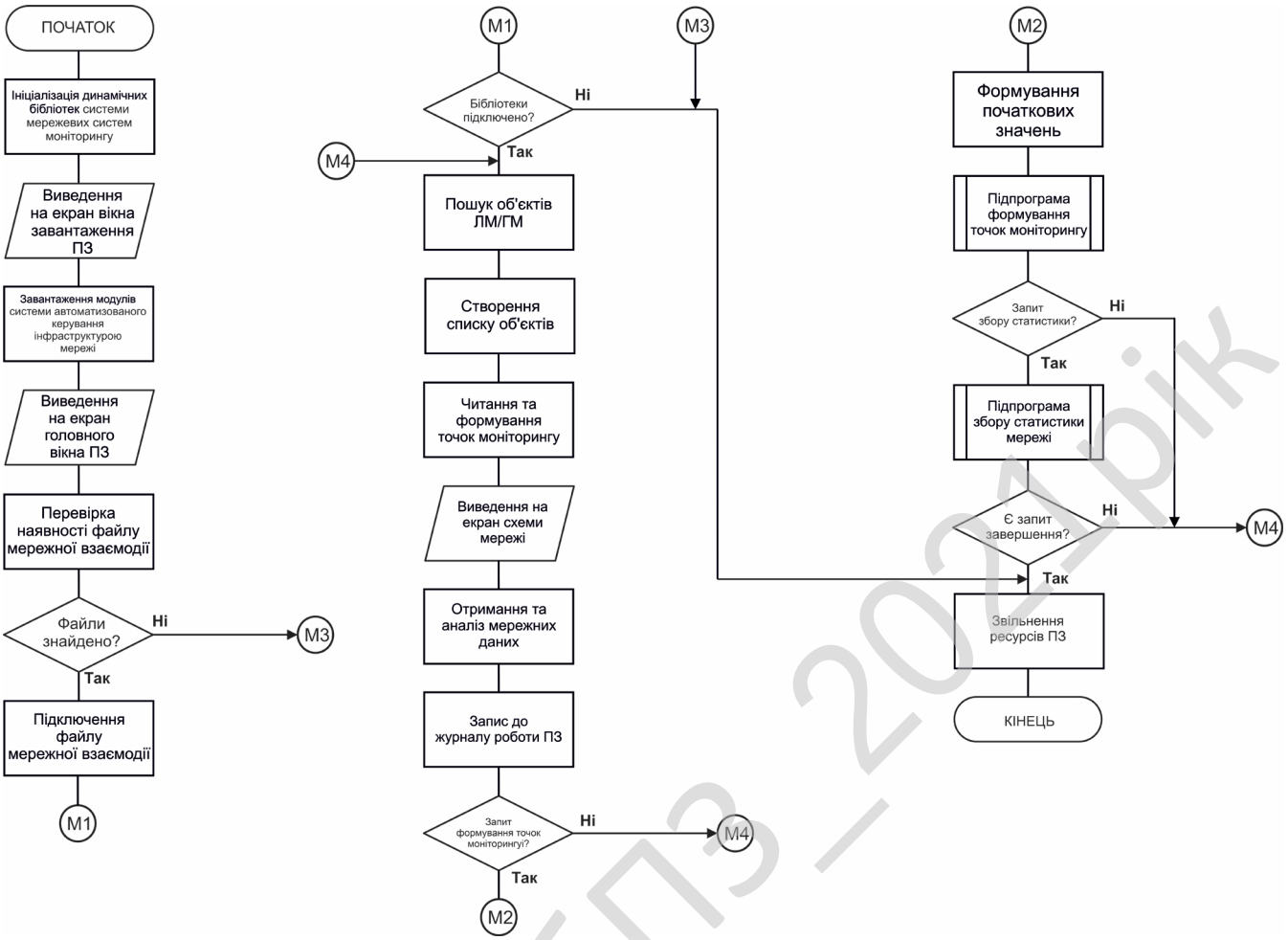

Рисунок 4.1 – Блок-схема основної програми

Розглянемо реалізацію цієї підзадачі у вигляді вихідного коду:

```
unit U3;
// Модуль U3

interface
// об’ява інтерфейсна
Uses // бібліотеки підключаємо
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ComCtrls, StdCtrls, ExtCtrls, ImgList, ShellApi;

Type // об’ява класу
  TU0 = class(TForm)
    Button1: TButton;
    rgScope: TRadioGroup;
    GroupBox1: TGroupBox;
```

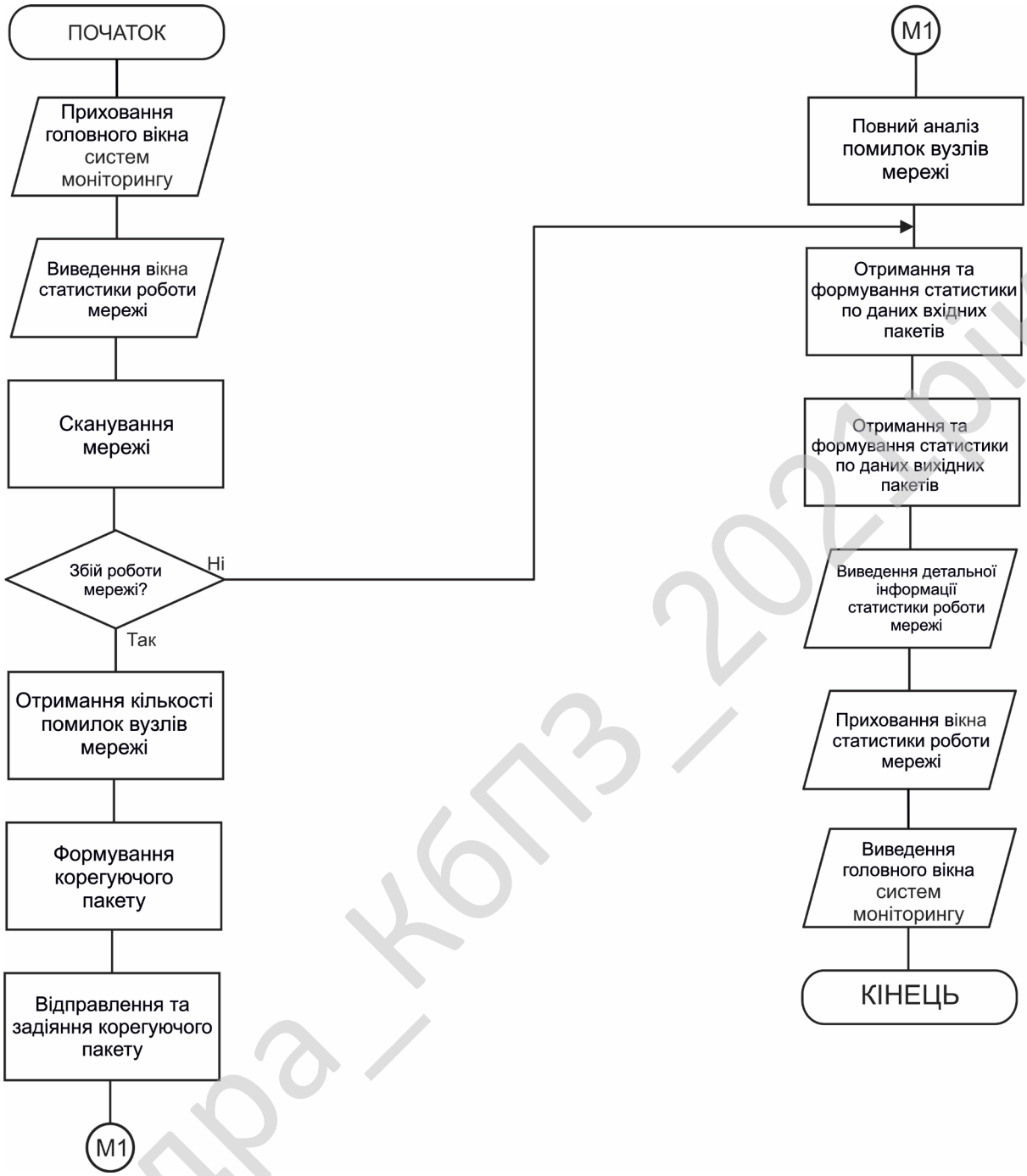

Рисунок 4.2 – Блок-схема роботи підпрограми

```
GroupBox2: TGroupBox;
cbTypeAny: TCheckBox;
cbTypeDisk: TCheckBox;
cbTypePrint: TCheckBox;
cbUsageAll: TCheckBox;
cbUsageConnectable: TCheckBox;
cbUsageContainer: TCheckBox;
```

```
    NetTree: TTreeView;
    ImageList1: TImageList;
    procedure Button1Click(Sender: TObject);
    procedure NetTreeGetImageIndex(Sender: TObject; Node: TTreeNode);
    procedure NetTreeDblClick(Sender: TObject);
    procedure NetTreeCustomDrawItem(Sender: TCustomTreeView;
      Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
  private
    { Private declarations }
  public
    procedure Open_Do_Close_Enum(const ParentNode: TTreeNode;
         ResScope, ResType, ResUsage: DWORD; const NetContainerToOpen:
         PNetResource);
    function OpenEnum(const NetContainerToOpen: PNetResource;
         ResScope, ResType, ResUsage: DWORD): THandle;
    function EnumResources(const ParentNode: TTreeNode;
         ResScope, ResType, ResUsage: DWORD; hNetEnum: THandle): UINT;
  end;
var // екземпляр класу створюємо
  Form1: TU0;

implementation
// реалізаційна частина модулю
{$R *.DFM}
// сканування
procedure TU0.Button1Click(Sender: TObject);
var
 ResScope, ResType, ResUsage: dword;
begin
 Button1.Caption:= 'Пошук мережевих ресурсів. Чекайте ...';
 Button1.Enabled:=false;
 NetTree.Items.Clear;
 case rgScope.ItemIndex of
  1: ResScope:=RESOURCE_GLOBALNET;
  2: ResScope:=RESOURCE_REMEMBERED;
  else ResScope:=RESOURCE_CONNECTED;
 end;
 ResType:=0;
 if cbTypeAny.Checked
 then ResType:=ResType or RESOURCETYPE_ANY;
 if cbTypeDisk.Checked
 then ResType:=ResType or RESOURCETYPE_DISK;
```

```
        if cbTypePrint.Checked
        then ResType:=ResType or RESOURCETYPE_PRINT;
        ResUsage:=0;
        if cbUsageConnectable.Checked
        then ResUsage:=ResUsage or RESOURCEUSAGE_CONNECTABLE;
        if cbUsageContainer.Checked
        then ResUsage:=ResUsage or RESOURCEUSAGE_CONTAINER;
        Open_Do_Close_Enum(NetTree.Items.Add(nil, 'Network Resources'),
                                    ResScope, ResType, ResUsage, nil);
         Button1.Caption:='Обновить список ресурсов';
        Button1.Enabled:=true;
       end;

       // запит
       procedure TU0.Open_Do_Close_Enum(const ParentNode: TTreeNode;
       ResScope, ResType, ResUsage: DWORD; const NetContainerToOpen:
PNetResource);
       var
        hNetEnum: THandle;
       begin
        hNetEnum:=OpenEnum(NetContainerToOpen, ResScope, ResType, ResUsage);
        if (hNetEnum=0)
        then Exit;
        EnumResources(ParentNode, ResScope, ResType, ResUsage, hNetEnum);
        if (NO_ERROR<>WNetCloseEnum(hNetEnum))
        then ShowMessage('WNetCloseEnum Error');
       end;

       // відкриття ресурсу
       function TU0.OpenEnum(const NetContainerToOpen: PNetResource;
                          ResScope, ResType, ResUsage: DWORD): THandle;
       var
        hNetEnum: THandle;
       begin
        Result:=0;
        if (NO_ERROR<>WNetOpenEnum(ResScope, ResType, ResUsage,
                                   NetContainerToOpen, hNetEnum))
        then ShowMessage('Error!')
        else Result:=hNetEnum;
       end;
```

```
// отримання списку
function TU0.EnumResources(const ParentNode: TTreeNode;
ResScope, ResType, ResUsage: DWORD; hNetEnum: THandle): UINT;
function ShowResource(const ParentNode: TTreeNode; Res: TNetResource):
                    TTreeNode;
begin
   Result:=NetTree.Items.AddChild(ParentNode, string(Res.lpRemoteName));
end;
const
 RESOURCE_BUF_ENTRIES = 2000;
var
 ResourceBuffer: array[1..RESOURCE_BUF_ENTRIES] of TNetResource;
 i, ResourceBuf, EntriesToGet: dword;
 NewNode: TTreeNode;
begin
 Result:=0;
 while true do
  begin
   ResourceBuf:=sizeof(ResourceBuffer);
   EntriesToGet:=RESOURCE_BUF_ENTRIES;
   if (NO_ERROR<>WNetEnumResource(hNetEnum, EntriesToGet,
                                @ResourceBuffer, ResourceBuf))
   then
    begin
     case GetLastError() of
      NO_ERROR:
// Кидаємо з комутатора, проходимося по буферу
      Break;
      ERROR_NO_MORE_ITEMS:
// Return with 0 code because this only happens when we got
// RESOURCE_BUF_ENTRIES entries on the previous call to
// WNetEnumResource, and there were coincidentally exactly
// RESOURCE_BUF_ENTRIES entries total in the enum at the time of
// that previous call
      Exit;
      else ShowMessage('Помилка!');
      Result:=1;
      Exit;
     end;
    end;
   for i:=1 to EntriesToGet do
    begin
```

```
      NewNode:=ShowResource(ParentNode, ResourceBuffer[i]);
      if (ResourceBuffer[i].dwUsage and RESOURCEUSAGE_CONTAINER)<>0
      then Open_Do_Close_Enum(NewNode, ResScope, ResType, ResUsage,
                              @ResourceBuffer[i]);
      Application.ProcessMessages;
     end;
   end;
end;

// робота з ресурсом
function TU0.EnumNetResources(const ParentNode: TTreeNode;
ResScope, ResType, ResUsage: DWORD;
const NetResource: PNetResource): boolean;

function AddNetResource(const Res: TNetResource): TTreeNode;
  begin
   Result:=NetTree.Items.AddChild(ParentNode, Res.lpRemoteName);
  end;
var
 EnumHandle: THandle;
 Res, ResEnum: DWORD;
 LocalNetResource: array[1..2000] of TNetResource;
 i, NumEntries, BufSize: DWORD;
 NewNode: TTreeNode;
begin
 Result:=false;
 Res:=WNetOpenEnum(ResScope, ResType, ResUsage, NetResource, EnumHandle);
 if Res=NO_ERROR then begin
  repeat
   BufSize:=SizeOf(LocalNetResource);
   NumEntries:=2000;
// якомога більше.
   ResEnum:=WNetEnumResource(EnumHandle, NumEntries, @LocalNetResource,
            BufSize);
   if ResEnum=NO_ERROR
   then
    begin
   for i:=1 to NumEntries do
    begin
// Range checking
   NewNode:=AddNetResource(LocalNetResource[i]);
   if ((LocalNetResource[i].dwUsage and RESOURCEUSAGE_CONTAINER)<>0) then
```

```
           begin
            EnumNetResources(NewNode, ResScope, ResType, ResUsage,
@LocalNetResource[i]);
           end;
          end;
         end
        else if ResEnum<>ERROR_NO_MORE_ITEMS then
         begin
          ShowMessage(Format('WNetEnumResource Error: %d', [Res]));
          break;
         end;
         until ResEnum = ERROR_NO_MORE_ITEMS;
         end
         else
          begin
           ShowMessage(Format('WNetOpenEnum Error: %d', [Res]));
          end;
         Res:=WNetCloseEnum(EnumHandle);
         if(Res<>NO_ERROR)
         then ShowMessage(Format('WNetCloseEnum Error: %d', [Res]))
         else Result:=true;
       end;
       // отримання дерева ресурсу
       procedure TU0.NetTreeGetImageIndex(Sender: TObject; Node: TTreeNode);
       begin
        if Node.HasChildren
        then Node.ImageIndex:=1
        else Node.ImageIndex:=0;
       end;

       procedure TU0.NetTreeDblClick(Sender: TObject);
       begin
        try
         ShellExecute(0,'open',PChar(NetTree.Selected.Text),'','',SW_SHOW);
        except

        end;
       end;
       end.
```

Розглянемо концепцію архітектури клієнт-сервер – це один з архітектурних шаблонів програмного забезпечення та є домінуючою концепцією

у створенні розподілених мережних програм і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

– набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;

– набір клієнтів, які використовують сервіси, що надаються серверами;

– мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

Дуже важливо ясно уявляти, хто або що розглядається як «клієнт». Можна говорити про клієнтський комп'ютер, з якого відбувається звернення до інших комп'ютерів. Можна говорити про клієнтське та серверне програмне забезпечення. Нарешті, можна говорити про людей, які бажають за допомогою відповідного програмного та апаратного забезпечення отримати доступ до тієї чи іншої інформації.

Загальноприйнятим є положення, що клієнти та сервери – це перш за все програмні модулі. Найчастіше вони знаходяться на різних комп'ютерах, але бувають ситуації, коли обидві програми – і клієнтська, і серверна, фізично розміщуються на одній машині; в такій ситуації сервер часто називається локальним.

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

– рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;

– прикладний рівень, який реалізує основну логіку ПЗ і на якому здійснюється необхідна обробка інформації;

– рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дворівнева клієнт-серверна архітектура передбачає взаємодію двох програмних модулів – клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

– модель тонкого клієнта, в рамках якої вся логіка ПЗ та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;

– модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та ін.

Типовим прикладом клієнт-серверної взаємодії є WWW. Існує величезна кількість веб-серверів, на яких розміщується та чи інша інформація. У найпростішому випадку ця інформація являє собою набір веб-сторінок, які можуть зберігатися на сервері у вигляді файлів, розмічених за допомогою мови розмітки HTML. Але ситуація, як правило, є складнішою; значна частина веб-ресурсів на сучасному етапі є динамічними, тобто вони не існують в заздалегідь підготовленому вигляді, а створюються безпосередньо в процесі обробки запиту від користувача.

Для того, щоб людина, яка працює в Інтернеті, могла переглянути ту чи іншу сторінку, на її комп'ютері повинно бути встановлено відповідне програмне забезпечення. Програми для перегляду веб-сторінок називаються браузерами.

Але, крім браузерів, до серверів можуть звертатися і інші клієнти, а саме – автономні програми. Вони можуть передбачати взаємодію з людиною, а можуть працювати в цілком автоматичному режимі. Типовим класом таких програм є роботи, призначені для автоматичного перегляду веб-ресурсів. Зокрема, роботи є

важливим елементом пошукових систем і використовуються ними для перегляду сторінок і збору інформації про них.

Для запиту до веб-сервера клієнтська програма повинна задати місцезнаходження комп'ютера, на якому розміщується серверна програма, назву потрібного документа і, можливо, інші дані, які специфікують запит. Мережа забезпечує знаходження сервера і передачу йому клієнтського запиту. Серверні програми обробляють цей запит, відповідь пересилається по мережі клієнтові.

Трирівнева клієнт-серверна архітектура, яка почала розвиватися з середини 90-х років, передбачає відділення прикладного рівня від управління даними. Відокремлюється окремий програмний рівень, на якому зосереджується прикладна логіка ПЗ. Програми проміжного рівня можуть функціонувати під управлінням спеціальних серверів ПЗ, але запуск таких програм може здійснюватися і під управлінням звичайного веб-сервера. Нарешті, управління даними здійснюється сервером даних.

Для роботи з системою користувач використовує стандартне програмне забезпечення –звичайний браузер. Це позбавляє його необхідності завантажувати та інсталювати спеціальні програми (хоча інколи така необхідність все-таки виникає).

Але користувачеві слід надати в розпорядженні інтерфейс, який дозволяв би йому взаємодіяти з системою і формувати запити до неї. Форми, що визначають цей інтерфейс, розміщуються на веб-сторінках та завантажуються разом з ними.

Веб-оглядач формує запит та пересилає його до сервера, який здійснює обробку. При необхідності сервер викликає серверні програмні модулі, які забезпечують обробку запиту і в разі потреби звертаються до сервера даних. Сервер даних здійснює операції з даними, що зберігаються в системі та складають її інформаційну основу. Зокрема, він може здійснити вибірку з інформаційної бази відповідно до запиту та передати її модулю проміжного рівня для подальшої

обробки. Дані, з якими працює сервер даних, найчастіше організовані як реляційна база даних.

Найчастіше веб-сервер і серверні модулі проміжного рівня розміщуються на одному комп'ютері, хоч і являють собою окремі і логічно незалежні програмні модулі.

На сучасному етапі для програмування модулів проміжного рівня використовується мова серверних сценаріїв PHP, а для управління даними – СУБД MySQL. Таким чином, зв'язку PHP-MySQL слід розглядати як стандартний інструмент для створення порівняно простих інтерактивних веб-сайтів та систем електронної комерції; близько 90% комерційних систем сьогодні створюється саме на цій основі. Водночас як засоби управління даними, так і middleware-засоби можуть бути найрізноманітнішими. Так, для створення серверних програм, крім PHP, широко застосовуються Java, Perl, Python, Delphi.

Взагалі, технології створення розподілених, зокрема веб-програм, стрімко розвиваються. Слід згадати про технології EJB (Enterprise Java Beans), CORBA, а також про .NET – порівняно нову ініціативу компанії Microsoft. Для зберігання даних та їх передачі часто використовується так звана розширювана мова розмітки XML (Extensible Markup Language).

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.

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення захистимо за допомогою національного стандарту захисту інформації на основі алгоритму шифрування/дешифрування ДСТУ 4145-2002 з використанням еліптичних кривих над двійковим розширеним полем Галуа. У системі шифрування/дешифрування як параметри розглядається еліптична крива E_p (a,b) і точка G на ній.

Учасник В вибирає закритий ключ n і обчислює відкритий ключ $P_B = n \times G$. Щоб зашифрувати повідомлення P_m використовується відкритий ключ одержувача В P_B. Учасник А вибирає випадкове ціле позитивне число k і

обчислює зашифроване повідомлення C_m, що є точкою на еліптичній кривій.

$$C_m = \{k \times G,\ P_m + k \times P_B\}. \quad (4.1)$$

Щоб дешифрувати повідомлення, учасник В множить першу координату точки на свій закритий ключ і віднімає результат від другої координати:

$$P_m + k \times P_B - n_B \times (k \times G) = P_m + k \times (n_B \times G) - n_B \times (k \times G) = P_m. \quad (4.2)$$

Учасник А зашифрував повідомлення P_m додаванням до нього $k \times P_B$. Ніхто не знає значення k, тому, хоча P_B і є відкритим ключем, ніхто не знає $k \times P_B$. Супротивнику для відновлення повідомлення доведеться обчислити k. Зробити це буде нелегко. Одержувач також не знає k, але йому як підказку посилається $k \times G$. Помноживши $k \times G$ на свій закритий ключ, одержувач одержить значення, що було додано відправником до незашифрованого повідомлення. Тим самим одержувач, не знаючи k, але маючи свій закритий ключ, може відновити незашифроване повідомлення.

Нехай задано просте число $p > 4$. Тоді еліптичною кривою E, визначеною над розширеним двійковим полем полем F_{2^m}, називається безліч пар чисел (x, y), $x, y \in F$, що задовольняють тотожності:

$$y^2 \equiv x^3 + a \cdot x + b (\mathrm{mod}\, 2^m), \quad (4.3)$$

де $4 \cdot a^3 + 27 \cdot b^2$ не рівно з нулю по модулю 2^m.

Інваріантом еліптичної кривої називається величина $J(E)$, що задовольняє тотожності:

$$J(E) \equiv 1728 \frac{4a^3}{4a^3 + 27b^2} (\mathrm{mod}\, 2^m). \quad (4.4)$$

Коефіцієнти a, b еліптичної кривої E, по відомому інваріанту $J(E)$, визначаються таким чином:

$$\begin{cases} a \equiv 3k (\mathrm{mod}\, 2^m) \\ b \equiv 2k (\mathrm{mod}\, 2^m) \end{cases} \text{де } k \equiv \frac{J(E)}{1728 - J(E)} (\mathrm{mod}\, 2^m), J(E) \neq 0 \text{ або } 1728. \quad (4.5)$$

Пари (x,y), що задовольняють тотожності (4.1), називаються точками еліптичної кривої E, x та y – відповідно x– та y-координатами точки.

Точки еліптичної кривої позначатимемо $Q(x,y)$ або просто Q. Дві точки

еліптичної кривої рівні, якщо рівні їх відповідні x– і y-координати.

На безлічі всіх точок еліптичною кривою Е введемо операцію додавання, яку позначатимемо знаком "+". Для двох довільних точок $Q_1(x_1, y_1)$ та $Q_2(x_2, y_2)$ еліптичної кривої Е, розглянемо декілька варіантів.

Нехай координати точок Q_1 та Q_2 задовольняють умові $x_1 \neq x_2$. В цьому випадку їх сумою називатимемо точку $Q_3(x_3, y_3)$ координати якої визначаються порівняннями:

$$\begin{cases} x_3 \equiv \lambda^2 - x_1 - x_2 (\mathrm{mod}\, 2^m), \\ y_3 \equiv \lambda(x_1 - x_3) - y_1 (\mathrm{mod}\, 2^m), \end{cases} \text{де } \lambda \equiv \frac{y_2 - y_1}{x_2 - x_1} (\mathrm{mod}\, 2^m). \quad (4.6)$$

Якщо виконана рівність $x_1 = x_2$ та $y_1 = y_2 \neq 0$, то визначимо координати точки Q_3 таким чином:

$$\begin{cases} x_3 \equiv \lambda^2 - 2x_1 (\mathrm{mod}\, 2^m), \\ y_3 \equiv \lambda(x_1 - x_3) - y_1 (\mathrm{mod}\, 2^m), \end{cases} \text{де } \lambda \equiv \frac{3x_1^2 + a}{2y_1} (\mathrm{mod}\, 2^m). \quad (4.7)$$

У разі, коли виконана умова $x_1 = x_2$ та $y_1 = -y_2 (\mathrm{mod}\, p)$, суму точок Q_1та Q_2 називатимемо нульовою точкою О, не визначаючи її х– і у-координати. В цьому випадку, точка Q_2 називається запереченням точки Q_1. Для нульової точки О виконана рівність:

$$Q + 0 = 0 + Q = Q, \quad (4.8)$$

де Q – довільна точка еліптичної кривої Е.

Щодо введеної операції складання безліч всіх точок еліптичною кривою Е, разом з нульовою точкою, утворюють кінцеву абельову (комутативну) групу порядку т, для якого виконана нерівність:

$$p + 1 - 2\sqrt{p} \leq m \leq p + 1 + 2\sqrt{p} \quad (4.9)$$

Точка Q називається точкою кратності k, або просто – кратною точкою еліптичної кривої E, якщо для деякої точки P виконана рівність:

$$Q = P + \ldots + P = kP \quad (4.10)$$

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1. З рисунку головного вікна можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Верхнього меню: Формат; Налаштування; Функції; Довідка.

– Функціональних кнопок ПЗ: Пошук; Статистика; БД статистичної інформації; Налаштування фільтру; Вікно прогнозування завантаженості.

– Розділу виведення результату роботи системи.

– Розділу виведення поточного стану системи.

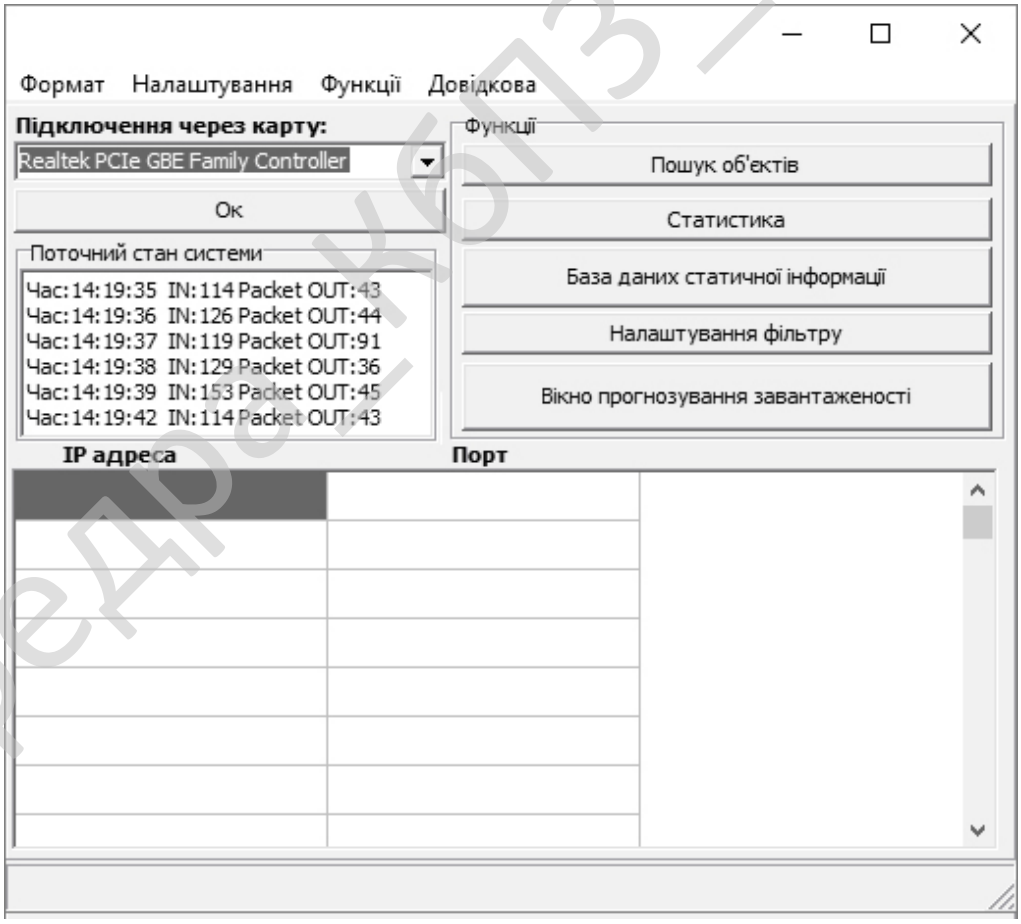

Рисунок 5.1 – Головне вікно ПЗ

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення. 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.

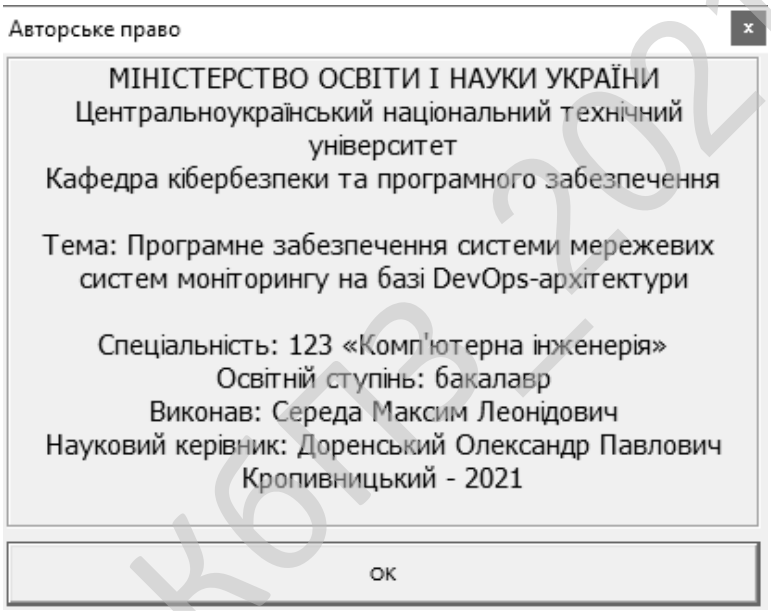

Рисунок 5.2 – Авторське право

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.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання кваліфікаційної бакалаврської роботи, призначено для системи мережевих систем моніторингу на базі DevOps-архітектури.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем мережевих систем моніторингу на базі DevOps-архітектури.

– Досліджена система мережевих систем моніторингу на базі DevOps-архітектури.

– На основі отриманих результатів досліджень створена програмна реалізація системи мережевих систем моніторингу на базі DevOps-архітектури.

Розроблені під час виконання кваліфікаційної бакалаврської роботи алгоритми дозволяють успішно вирішувати завдання мережевих систем моніторингу на базі DevOps-архітектури.

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 RAD Studio Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи мережевих систем моніторингу на базі DevOps-архітектури. 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.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ДСТУ 4145-2002.

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. Kovalenko O., Popereshnyak S., Grinenko S., Grinenko O., Radivilova T. «Methods for Assessing the Maturity Levels of Software Ecosystems». *CEUR Workshop Proceedings* Volume 2654, 2019, Pages 251-261. Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85083214331&origin=AuthorNamesList&txGid=5633fba897776a6e0f3d5633fbcd3fbe **(Scopus)**.

2. Kovalenko O., Drieieva H., Simakhin V., Bondar S., Drieiev O., Zhumadilova M. «Multifractal Properties of Traffic Generator Based on Markov Chains ». *CEUR Workshop Proceedings* Volume 2588, 2019, Pages 567-579. Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85083214331&origin=AuthorNamesList&txGid=176e2cada8976a6e0f3d5633fbcd3fbe **(Scopus)**.

3. Kovalenko Oleksandr Qualitative risk analysis of software development / Oleksandr Kovalenko, Jamil Al-Azzeh, Oleksii Smirnov, Anna Kovalenko, Serhii Smirnov // Asian Journal of Information Technology. – Volume 17 Issue 3. – Medwell Journals. – 2018. – P. 218-230. ISSN: 1682-3915. URL: http://medwelljournals.com/abstract/?doi=ajit.2018.218.230 Doi: ajit.2018.218.230

4. Kovalenko Oleksandr, The mathematical model of the testing technology for DOM XSS vulnerabilities / O. Kovalenko, O. Smirnov, A.Kovalenko, S. Smirnov, V. Vialkova // Scientific & practical cyber security journal (SPCSJ) Volume 2 Issue 1, P. 22-28. Georgia. Tbilisi. Scientific Cyber Security Association (SCSA), 2018 ISSN: 2587-4667. URL: https://journal.scsa.ge/wp-content/uploads/2018/12/04-3-o.kovalenko-a.kovalenko-o.smirnov-s.smirnov-v.vialkova.pdf

5. Коваленко А.В. Технология тестирования DOM XSS уязвимости / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов // Scientific & practical cyber security journal (SPCSJ) Volume 1. Issue 1. P. 38-45 Georgia. Tbilisi. Scientific Cyber Security Association (SCSA), 2017 ISSN: 2587-

4667. URL: https://journal.scsa.ge/wp-content/uploads/2018/12/8-dom-xss-testing-technology-vulnerabilities.pdf

6. Коваленко О.В. Моделі та методи розроблення програмного забезпечення комп'ютерних систем для підвищення безпеки даних: монографія / О.В. Коваленко // К.: Вид. «КОД» – 2019. – 305 с.

7. Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Информационные технологии в управлении, образовании, науке и промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Видавець Рожко С.Г., 2016. – 566 с.

8. Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Інформаційні технології: проблеми та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: Видавець Рожко С.Г., 2017. – 447 с.

9. Коваленко А.В. Комплекс математических моделей технологии тестирования web-приложений / А.В. Коваленко, А.А. Смирнов // Інформаційні технології: сучасний стан та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: ТОВ «ДІСА ПЛЮС», 2018. – 461 с.

10. Коваленко А.В. Задачи распознавания ситуаций в ERP системах / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник наукових праць "Системи обробки інформації". – Випуск 4(120). – Х.: ХУПС – 2014. – С. 161-164.

11. Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник наукових праць "Системи обробки інформації". – Випуск 5(142). – Х.: ХУПС – 2016. – С. 153-157.

12. Коваленко А.В. Проблемы анализа и оценки рисков информационной деятельности / А.В. Коваленко, А.А. Смирнов, Н.Н. Якименко, А.П. Доренский // Збірник наукових праць "Системи обробки інформації". –

Випуск 3(140). – Х.: ХУПС – 2016. – С. 40-42.

13. Коваленко А.В. Метод качественного анализа рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, Н.Н. Якименко, А.П. Доренский // Наука і техніка Повітряних Сил Збройних Сил України. – Випуск 2(23). – Харків: ХУПС. – 2016. – С. 150-158.

14. Коваленко А.В. Метод количественной оценки рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, Н.Н. Якименко, А.П. Доренский // Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. – 2016. – С. 128-133.

15. Коваленко А.В. Использование псевдобулевых методов бивалентного программирования для управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Системи управління, навігації та зв'язку. – Випуск 1 (37). – Полтава: ПолтНТУ. – 2016. – С. 98-103.

16. Коваленко А.В. Метод управления рисками разработки программного обеспечения / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 2 (38). – Полтава: ПолтНТУ. – 2016. – С. 93-100.

17. Коваленко А.В. Технология тестирования уязвимости к SQL инъекциям / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 5 (45). – Полтава: ПолтНТУ. – 2017. – С. 66-71.

18. Коваленко А.В. Масштабирование имитационной модели технологии тестирования безопасности / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 6 (46). – Полтава: ПолтНТУ. – 2017. – С. 181-184.

19. Коваленко А.В. Имитационная модель технологии тестирования безопасности Web-приложений / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 1 (47). – Полтава: ПолтНТУ. – 2018. – С. 114-123.

20. Коваленко О.В. Методи якісного аналізу та кількісної оцінки ризиків розроблення програмного забезпечення/ О.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 3 (49). – Полтава: ПолтНТУ. – 2018. – С. 116-125.

21. Коваленко О.В. Управління ризиками розроблення програмного

забезпечення за умови обмеженості коштів виділених на усунення помилок безпеки/ О.В. Коваленко // Техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. – Випуск 31. – Кропивницький: ЦНТУ. – 2018. – С. 128-140.

22. Коваленко О.В. GERT-мережевий синтез технології тестування на вразливість WEB-додатків/ О.В. Коваленко // Захист інформації. – Випуск 20(2) – К.: НАУ. – 2018. – С. 89-94.

23. Коваленко О.В. Імітаційна модель технології тестування безпеки на основі положень теорії масштабування / О.В. Коваленко // Безпека інформації. – Випуск 24 (2). – К.: НАУ. – 2018. – С. 110-117.

24. Коваленко О.В. Оцінка ефективності технології тестування безпеки / О.В. Коваленко // Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки. Том 29 (68) № 2, 2018. – С. 137-141

25. Коваленко О.В. Методи та засоби управління безпекою додатків / О.В. Коваленко // Інформаційно-керуючі системи на залізничному транспорті. №4, 2018. – С. 41-44.

26. Коваленко О.В. Розробка інформаційної технології передтестової компіляції та розподілу доступу / О.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 4 (50). – Полтава: ПолтНТУ. – 2018. – С. 115-119.

27. Коваленко О.В. Аналіз та дослідження інформаційних технологій розробки програмного забезпечення/ О.В. Коваленко // Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки. Том 29 (68) № 5, 2018. – С. 131-137.

28. Коваленко О.В. Удосконалений метод управління ризиками розроблення програмного забезпечення на основі напівмарковської моделі прийняття рішень/ О.В. Коваленко // Сучасні інформаційні системи. – Випуск 2(3). – Харків. – 2018. – С. 41-48.

29. Коваленко О.В. Математичні моделі технології тестування DOM XSS

вразливості та вразливості до SQL ін'єкцій / О.В. Коваленко // Вісник Черкаського державного технологічного університету. Серія : Технічні науки №4, 2018. – С. 29-36.

30. Коваленко О.В. Математична модель технології тестування вразливості до SQL ін’єкцій / О.В. Коваленко // Системи управління, навігації та зв’язку. – Випуск 6 (58). – Полтава: ПолтНТУ. – 2019. – С. 43-47.

31. Коваленко О.В. Математична модель технології тестування комплексу DOM XSS вразливостей для аналітичної оцінки часових витрат / О.В. Коваленко // Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 173-180, 2019.

32. Коваленко А.В. Проблемы анализа и оценки рисков информационной деятельности / А.В. Коваленко, А.А. Смирнов // Збірник наукових праць II міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 24-27 лютого 2016 р. – Київ: Європейський університет. – 2016. – С. 138-139.

33. Коваленко А.В. Анализ и оценка рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез «Securitea informationala 2015-2016». Conferenta internationala (editia a XII-a). Chisinau. Moldova. 3 martie 2016. – Chisinau: ADSEM. – 2016. – P. 96-102.

34. Коваленко А.В. Исследование источников и причин риска разработки программного обеспечения, этапов и работ, при выполнении которых возникает риск / А.В. Коваленко, А.А. Смирнов // Збірник тез VII всеукраїнської науково-практичної конференції "Інформатика та системні науки (ІСН-2016)". м. Полтава. 10-12 березня 2016 р. – Полтава.: ПУЕТ – 2016. – С. 264-266.

35. Коваленко А.В. Оценка показателя чистой приведенной стоимости для количественной оценки рисков проекта разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез науково-практичної конференції “Проблеми кібербезпеки інформаційно-телекомунікаційних

систем”. м. Київ. 10-11 березня 2016 р. – Київ: КНУ ім. Тараса Шевченко – 2016. – С. 81-82.

36. Коваленко А.В. Методика структурной идентификации рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез Міжнародної науково-практичної конференції «Інформаційна безпека та комп'ютерні технології» (IS&CT). м. Кіровоград. 24-25 березня 2016 р. – Кіровоград: КНТУ. – 2016. – С. 71-72.

37. Коваленко А.В. Методы качественного анализа рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез першої міжнародної науково-практичної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК - 2016). м. Харків. 30 березня – 1 квітня 2016 р. – Харків: НТУ «ХПІ». – 2016. – С. 6-7.

38. Коваленко А.В. Структурная идентификация рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез XVIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 15-16 квітня 2016 р. – Кіровоград: КНТУ. – 2016. – С. 175-182.

39. Коваленко А.В. Исследование разработанной методики структурной идентификации рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез VIII міжнародної науково-практичної конференції “Проблеми і перспективи розвитку IT-індустрії”. м. Харків. 28-29 квітня 2016 р. – Харків: ХНЕУ. – 2016. – С. 49.

40. Коваленко А.В. Исследование дерева рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез III міжнародної науково-практичної конференції «Інформаційна та економічна безпека» (INFECO-2016)». м. Харків. 28-30 квітня 2016 р. – Харків: ХННІ ДВНЗ «УБС». – 2016. – С. 174-178.

41. Коваленко А.В. Методы качественного анализа и количественной

оценки рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Сборник тезисов XII международной конференции "Стратегия качества в промышленности и образовании". г. Варна. Болгария. 30 мая – 02 июня 2016 г – Варна. ТУВ. – 2016. – С. 585-589.

42. Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Матеріали Всеукраїнської науково-практичної конференції «Кібербезпека в Україні: правові та організаційні питання». м. Одеса, 21 жовтня 2016 р. – Одеса : ОДУВС, 2016. – С.146-148.

43. Коваленко А.В. Метод управления рисками разработки программного обеспечения с использованием псевдобулевых методов бивалентного программирования / А.В. Коваленко, А.А. Смирнов // Матеріали Всеукраїнської науково-практичної конференції «Актуальні задачі та досягнення у галузі кібербезпеки». м. Кропивницький, 23-25 листопада 2016 року – Кропивницький: ЦНТУ, 2016. – С. 162.

44. Коваленко А.В. Псевдобулевые методы бивалентного программирования для управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко, С.А. Смирнов // Збірник наукових праць III міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 22-25 лютого 2017 р. – Київ: Європейський університет. – 2017. – С. 158-162.

45. Коваленко А.В. Метод управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез II науково-практичної конференції “Проблеми кібербезпеки інформаційно-телекомунікаційних систем”. м. Київ. 23-24 березня 2017 р. – Київ: КНУ ім. Тараса Шевченко – 2017. – С. 203-205.

46. Коваленко А.В. Алгоритмы анализа уязвимостей при управлении рисками разработки программного обеспечения / А.В. Коваленко,

А.А. Смирнов, А.С. Коваленко // Conferenta internationala (editia a XIII-a). «Securitea informationala 2017». Chisinau. Republic of Moldova. 4-5 aprilie 2017. – Chisinau: ADSEM. – 2017. – P. 19-22.

47. Коваленко А.В. Алгоритм анализа DOM XSS уязвимости при управлении рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник тез дев'ятнадцятого міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кропивницький 7-8 квітня 2017 р. – Кропивницький: ГЛА НАУ. – 2017. – С. 125-127.

48. Коваленко А.В. Алгоритм анализа уязвимости SQL Injection для управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник тез другої міжнародної науково-технічної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2017). м. Харків.10-12 квітня 2017 р. – Харків: НТУ «ХПІ». – 2017. – С. 27.

49. Коваленко А.В. Метод управления рисками разработки программного обеспечения на основе алгоритмов анализа уязвимостей / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник тез Міжнародної науково-практичної конференції «Інформаційна безпека та комп'ютерні технології» (IS&CT). м. Кіровоград. 20-22 квітня 2017 р. Кіровоград: КНТУ. – 2017. – С. 92.

50. Коваленко А.В. Алгоритмы анализа DOM XSS уязвимости и уязвимости SQL Injection при управлении рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник тез IX міжнародної науково-практичної конференції "Проблеми і перспективи розвитку IT-індустрії". м. Харків. 20-21 квітня 2017 р. – Харків: ХНЕУ. – 2017. – С. 61.

51. Kovalenko O.V. Method of testing the dom xss vulnerability / Kovalenko Oleksandr, Kovalenko Anna, Smirnov Oleksii, Smirnov Serhii // International

Conference «information technologies, systems and networks ITSN-2017». Chisinau, Republic of Moldova. 17 – 18 October 2017. – Chisinau: Academy of Sciences of Moldova, Military Academy of Armed Forces “Alexandru cel Bun”. – 2017. – P. 7.

52. Коваленко О.В. Метод тестування DOM XSS уразливості / О.В. Коваленко, О.А. Смірнов, А.С. Коваленко, С.А. Смірнов // Збірник тез всеукраїнської науково-практичної інтернет-конференції «Автоматика та комп’ютерно-інтегровані технології у промисловості, телекомунікаціях, енергетиці та транспорті». м. Кропивницький. 16-17 листопада 2017 р. – Кропивницький: ЦНТУ. – 2017. – С. 198-199.

53. Коваленко О.В. GERT-модель технології тестування DOM XSS уразливості / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник наукових праць IV міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 21-24 лютого 2018 р. – Київ: Європейський університет. – 2018. – С. 65-70.

54. Коваленко О.В. Технології тестування уразливостей Web-застосунків з використанням GERT-моделі / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник тез всеукраїнської науково-практичної конференції "Комп'ютерні інтелектуальні системи та мережі (КІСМ-2018)". м. Кривий Ріг. 21-23 березня 2018 р. – Кривий Ріг.: ДВНЗ КНУ – 2018. – С. 227-230.

55. Коваленко А.В. Тестирование уязвимости Web-приложений к атаке вида межсайтовый скриптинг / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов // Збірник тез «Securitea informationala 2018». Conferenta internationala (editia a XIV-a). Chisinau. Moldova. 20-21 martie 2018. – Chisinau: ADSEM. – 2018. – P. 54-56.

56. Коваленко А.В. Комплекс математических моделей технологии тестирования web-приложений / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов // Збірник тез X міжнародної науково-практичної

конференції “Проблеми і перспективи розвитку IT-індустрії”. м. Харків. 19-20 квітня 2018 р. – Харків: ХНЕУ. – 2018. – С. 38.

57. Коваленко О.В. Розробка методу передтестової компіляції й розподілу доступу / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник наукових праць III міжнародної науково-практичної конференції “Інформаційна безпека та комп’ютерні технології”, м. Кропивницькй. 19-20 квітня 2018 р. – Кропивницький: ЦНТУ. – 2018. – С. 214-215.

58. Коваленко О.В. Оцінка ефективності технологій тестування безпеки уразливостей DOM XSS й SQL-ін'єкцій / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Сборник тезисов XIV международной конференции "Стратегия качества в промышленности и образовании", Варна, Болгария. 04-07 июня 2018 г – Варна. ТУВ. – 2018. – С. 271-274.

59. Коваленко О.В. Аналіз основних підходів математичного моделювання та методологій для забезпечення максимальних показників безпеки програмного забезпечення / О.В. Коваленко, А.С. Коваленко // Збірник наукових праць всеукр. наук.-практ. конф. здобувачів вищої освіти й молодих учених «Комп’ютерна інженерія і кібербезпека : досягнення та інновації, м. Кропивницькй. 27-29 листопада

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					КБР-123.21.0040.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Середа М.Л.			Програмне забезпечення системи мережевих систем моніторингу на базі DevOps-архітектури	Літ. Б	Аркуш 1 / Аркушів 6
Перевірив		Доренський О.П.					
Н. Контр.		Гермак В.С.				ЦНТУ КІ-18-3СК	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи мережевих систем моніторингу на базі DevOps-архітектури.

2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 204-02 від 28.12.2020 року).

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи мережевих систем моніторингу на базі DevOps-архітектури.

4 Джерела розробки

Джерелом цієї кваліфікаційної бакалаврської роботи є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи мережевих систем моніторингу на базі DevOps-архітектури;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows XP/Vista/7/8/10.

5.8.1 Обладнання

Комп'ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14" 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище RAD Studio Delphi 10.4.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 71 аркуш.

8 Етапи розробки

8.1 Збір і обробка інформації по темі кваліфікаційної бакалаврської роботи. Постановка задачі на виконання кваліфікаційної бакалаврської роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень кваліфікаційної бакалаврської роботи.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання кваліфікаційної бакалаврської роботи на попередній захист 22.05.2020 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист 3.06.2020 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник кваліфікаційної бакалаврської роботи

_____ Доренський О.П.

Програмне забезпечення системи мережевих систем моніторингу на базі DevOps-архітектури

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 48

Літера: РП

Кропивницький – 2021 року

Файл Main.pas – основне тіло розробленого ПЗ

```
unit Main; // назва модулю
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення
Тема бакалаврської дипломної роботи: Програмне забезпечення системи мережевих систем
моніторингу на базі DevOps-архітектури

Виконав: студент 4 курсу, групи КІ-18-3СК
Середа Максим Леонідович
2021 рік
Керівник: Доренський О.П
}
interface

// опис бібліотек

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, ComCtrls, Stat,
  ShellAPI, ShlObj, ImgList, TCP_IP, About;

//опис типів

type
  TMainForm = class(TForm)
    gbxShares: TGroupBox;
    lbxShares: TListBox;
    gbxSessions: TGroupBox;
    lvSessions: TListView;
    bvlSessions: TBevel;
    gbxFiles: TGroupBox;
    btnGetShares: TButton;
    btnCloseShares: TButton;
    btnAddShares: TButton;
    btnCloseSession: TButton;
    btnGetSessions: TButton;
    bvlTopSessions: TBevel;
    plButtonFiles: TPanel;
    btnGetFiles: TButton;
    btnCloseFile: TButton;
    bvlLeftFiles: TBevel;
    plFiles: TPanel;
    lvFiles: TListView;
    bvlTopFiles: TBevel;
    gbxTraffic: TGroupBox;
    lvTraffic: TListView;
    bvlTraffic: TBevel;
    tmrTraffic: TTimer;
    Button1: TButton;
    rgScope: TRadioGroup;
    GroupBox1: TGroupBox;
    cbUsageAll: TCheckBox;
    cbUsageConnectable: TCheckBox;
    cbUsageContainer: TCheckBox;
    GroupBox2: TGroupBox;
    cbTypeAny: TCheckBox;
    cbTypeDisk: TCheckBox;
    cbTypePrint: TCheckBox;
    NetTree: TTreeView;
    ImageList1: TImageList;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    function  IsNT(var Value: Boolean): Boolean;
    procedure btnGetSharesClick(Sender: TObject);
    procedure btnCloseSharesClick(Sender: TObject);
```

```
    function  SelectDirectory: String;
    procedure btnAddSharesClick(Sender: TObject);
    function  CardinalToTimeStr(Value:Cardinal):String;
    procedure btnGetSessionsClick(Sender: TObject);
    procedure btnCloseSessionClick(Sender: TObject);
    procedure btnGetFilesClick(Sender: TObject);
    procedure btnCloseFileClick(Sender: TObject);
    procedure tmrTrafficTimer(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure NetTreeCustomDrawItem(Sender: TCustomTreeView;
      Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
    procedure NetTreeDblClick(Sender: TObject);
    procedure NetTreeGetImageIndex(Sender: TObject; Node: TTreeNode);
    procedure Button4Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);

//опис типів та записів


  private
    { Private declarations }
  public
    { Public declarations }
    SessionCloseKey: array [0..512] of SmallInt;
    procedure Open_Do_Close_Enum(const ParentNode: TTreeNode;
         ResScope, ResType, ResUsage: DWORD; const NetContainerToOpen:
PNetResource);
   function OpenEnum(const NetContainerToOpen: PNetResource;
         ResScope, ResType, ResUsage: DWORD): THandle;
    function EnumResources(const ParentNode: TTreeNode;
        ResScope, ResType, ResUsage: DWORD; hNetEnum: THandle): UINT;
  end;

type
 TShareInfo2 = packed record
  shi2_netname : PWChar;
  shi2_type: DWORD;
  shi2_remark :PWChar;
  shi2_permissions: DWORD;
  shi2_max_uses : DWORD;
  shi2_current_uses : DWORD;
  shi2_path : PWChar;
  shi2_passwd : PWChar;
 end;
 PShareInfo2 = ^ TShareInfo2;
 TShareInfo2Array = array [0..512] of TShareInfo2;
 PShareInfo2Array = ^ TShareInfo2Array;

type
 TShareInfo50 = packed record
  shi50_netname : array [0..12] of Char;
  shi50_type : Byte;
  shi50_flags : Word;
  shi50_remark : PChar;
  shi50_path : PChar;
  shi50_rw_password : array [0..8] of Char;
  shi50_ro_password : array [0..8] of Char;
 end;

type
 TSessionInfo502 = packed record
   Sesi502_cname: PWideChar;
   Sesi502_username: PWideChar;
   Sesi502_num_opens: DWORD;
   Sesi502_time: DWORD;
   Sesi502_idle_time: DWORD;
   Sesi502_user_flags: DWORD;
   Sesi502_cltype_name: PWideChar;
```

```
    Sesi502_transport: PWideChar;
  End;
  PSessionInfo502 = ^TSessionInfo502;
  TSessionInfo502Array = array[0..512] of TSessionInfo502;
  PSessionInfo502Array = ^TSessionInfo502Array;

type
  TSessionInfo50 = packed record
    Sesi50_cname       : PChar;
    Sesi50_username    : PChar;
    sesi50_key         : Cardinal;
    sesi50_num_conns   : Word;
    sesi50_num_opens   : Word;
    sesi50_time        : Cardinal;
    sesi50_idle_time   : Cardinal;
    sesi50_protocol    : Byte;
    pad1               : Byte;
  end;

type
  TFileInfo3 = packed record
    fi3_id          : DWORD;
    fi3_permissions : DWORD;
    fi3_num_locks   : DWORD;
    fi3_pathname    : PWChar;
    fi3_username    : PWChar;
  end;
  PFileInfo3 = ^TFileInfo3;
  TFileInfo3Array = array[0..512] of TFileInfo3;
  PFileInfo3Array = ^TFileInfo3Array;

type
  TFileInfo50 = packed record
    fi50_id          : Cardinal;
    fi50_permissions : WORD;
    fi50_num_locks   : WORD;
    fi50_pathname    : PChar;
    fi50_username    : PChar;
    fi50_sharename   : PChar;
  end;

type
   TMibIfRow = packed record
     wszName           : array[0..255] of WideChar;
     dwIndex           : DWORD;
     dwType            : DWORD;
     dwMtu             : DWORD;
     dwSpeed           : DWORD;
     dwPhysAddrLen     : DWORD;
     bPhysAddr         : array[0..7] of Byte;
     dwAdminStatus     : DWORD;
     dwOperStatus      : DWORD;
     dwLastChange      : DWORD;
     dwInOctets        : DWORD;
     dwInUcastPkts     : DWORD;
     dwInNUCastPkts    : DWORD;
     dwInDiscards      : DWORD;
     dwInErrors        : DWORD;
     dwInUnknownProtos : DWORD;
     dwOutOctets       : DWORD;
     dwOutUCastPkts    : DWORD;
     dwOutNUCastPkts   : DWORD;
     dwOutDiscards     : DWORD;
     dwOutErrors       : DWORD;
     dwOutQLen         : DWORD;
     dwDescrLen        : DWORD;
     bDescr            : array[0..255] of Char;
   end;
   TMibIfArray = array [0..512] of TMibIfRow;
```

```
  PMibIfRow = ^TMibIfRow;
  PMibIfArray = ^TMibIfArray;

type
 TMibIfTable = packed record
   dwNumEntries     : DWORD;
   Table            : TMibIfArray;
 end;
 PMibIfTable = ^TMibIfTable;
var
NetShareEnumNT:function (     servername:PWChar;
                          level:DWORD;
                          bufptr:Pointer;
                          prefmaxlen:DWORD;
                          entriesread,
                          totalentries,
                          resume_handle:LPDWORD): DWORD; stdcall;

var
NetShareEnum:function ( pszServer   : PChar;
                        sLevel           : Cardinal;
                        pbBuffer    : Pchar;
                        cbBuffer    : Cardinal;
                        pcEntriesRead,
                        pcTotalAvail: Pointer):DWORD; stdcall;

var
NetShareDelNT:function (servername: PWideChar;
                        netname: PWideChar;
                        reserved: DWORD): LongInt; stdcall;

var
NetShareDel:function (  pszServer,
                        pszNetName:PChar;
                        usReserved:Word): DWORD; stdcall;

var
NetShareAddNT: function(servername: PWideChar;
                        level: DWORD;
                        buf: Pointer;
                        parm_err: LPDWORD): DWORD; stdcall;

var
NetShareAdd: function ( pszServer:Pchar;
                        sLevel:Cardinal;
                        pbBuffer:PChar;
                        cbBuffer:Word):DWORD; stdcall;

Var
NetSessionEnumNT:function(servername,
                        UncClientName,
                        username:PWChar;
                        level:DWORD;
                        bufptr:Pointer;
                        prefmaxlen:DWORD;
                        entriesread,
                        totalentries,
                        resume_handle:LPDWORD):DWORD; stdcall;

var
NetSessionEnum:function(pszServer:PChar;
                        sLevel: DWORD;
                        pbBuffer:Pointer;
                        cbBuffer:DWORD;
                        pcEntriesRead,
                        pcTotalAvial:Pointer):integer; stdcall;

var
NetSessionDelNT:function(ServerName,
```

```
                        UncClientName,
                        username:PWChar):DWORD; stdcall;

var
NetSessionDel:function( pszServer:PChar;
                        pszClientName: PChar;
                        sReserved: SmallInt):DWORD; stdcall;

var
NetFileEnumNT:function( servername,
                        basepath,
                        username:PWChar;
                        level:DWORD;
                        bufptr:Pointer;
                        prefmaxlen:DWORD;
                        entriesread,
                        totalentries,
                        resume_handle:LPDWORD):DWORD; stdcall;

var
NetFileEnum:function(     pszServer,
                        pszBasePath:PChar;
                        sLevel:DWORD;
                        pbBuffer:Pointer;
                        cbBuffer:DWORD;
                        pcEntriesRead,
                        pcTotalAvail:pointer):integer; stdcall;

var
NetFileClose:function(  ServerName:PWideChar;
                        FileId:DWORD):DWORD; stdcall;

var
NetFileClose2:function( pszServer:PChar;
                        ulFileId:LongWord):DWORD; stdcall;

var
GetIfTable:function(    pIfTable    : PMibIfTable;
                        pdwSize     : PULONG;
                        bOrder      : Boolean ): DWORD; stdcall;

var
  MainForm: TMainForm;

implementation

{$R *.dfm}

{ TMainForm }

//////////////////////////
//
//  Спочатку нам потрібно визначиться, під якою системою ми працюємо,
//  щоб довідатися яку частину коду (для NT чи ні) використовувати в цей момент.
//  Для цього напишемо невелику функцію, що і буде визначати тип системи.
//

function TMainForm.IsNT(var Value: Boolean): Boolean;
var Ver: TOSVersionInfo;
    BRes: Boolean;
begin
  Ver.dwOSVersionInfoSize := SizeOf(TOSVersionInfo);
  BRes := GetVersionEx(Ver);
  if not BRes then   //Перевірка
  begin
    Result := False; //Інформація не отримана
    Exit;            //ідемо
  end else
    Result := True;  //Інформація отримана
```

```
  case Ver.dwPlatformId of //визначаємося
    VER_PLATFORM_WIN32_NT       : Value := True;  //Windows NT тьа вище -
підходить
    VER_PLATFORM_WIN32_WINDOWS  : Value := False; //Windows 9 x-Me- підходить
    VER_PLATFORM_WIN32s         : Result := False //Windows 3.x- не підходить
  end;
end;

////////////////////////////
//
//  Одержання всіх відкритих загальних ресурсів досліджуємої мережевої системи
моніторингу на базі DevOps-архітектури
//

procedure TMainForm.btnGetSharesClick(Sender: TObject);
var
  i:Integer;
  FLibHandle : THandle;
  ShareNT : PShareInfo2Array;     //<= Змінні
  entriesread,totalentries:DWORD; //<= для Windows NT
  Share : array [0..512] of TShareInfo50; //<= Змінні
  pcEntriesRead,pcTotalAvail:Word;        //<= для Windows 9 x-Me
  OS: Boolean;
begin
  lbxShares.Items.Clear;
  if not IsNT(OS) then Close; //Визначаємо тип системи

  if OS then begin
//Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
//Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
//Зв' язуємо функцію
    @NetShareEnumNT := GetProcAddress(FLibHandle,' NetShareEnum' );
    if not Assigned(NetShareEnumNT) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    ShareNT := nil;
//Очищаємо покажчик на масив структур
//Виклик функції
    if NetShareEnumNT(nil,2,@ShareNT,DWORD(-1),
                      @entriesread,@totalentries,nil) <> 0 then
    begin
//Якщо виклик невдалий вивантажуємо бібліотеку
      FreeLibrary(FLibHandle);
      Exit;
    end;
    if entriesread > 0 then
//Обробка результатів
    for i:= 0 to entriesread- 1 do
      lbxShares.Items.Add(String(ShareNT^[i].shi2_netname));
  end else begin //Код для 9 x-me
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
//Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
    //Зв' язуємо функцію
    @NetShareEnum := GetProcAddress(FLibHandle,' NetShareEnum' );
    if not Assigned(NetShareEnum) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
//Виклик функції
    if NetShareEnum(nil,50,@Share,SizeOf(Share),
                 @pcEntriesRead,@pcTotalAvail)<> 0 then
    begin  //Якщо виклик невдалий вивантажуємо бібліотеку
```

```
      FreeLibrary(FLibHandle);
      Exit;
    end;
    if pcEntriesRead > 0 then //Обробка результатів
    for i:= 0 to pcEntriesRead- 1 do
      lbxShares.Items.Add(String(Share[i].shi50_netname));
  end;
  FreeLibrary(FLibHandle);
//Не забуваємо вивантажити бібліотеку
end;

//  Закриття загального ресурсу

procedure TMainForm.btnCloseSharesClick(Sender: TObject);
var
  OS:Boolean;
  FLibHandle : THandle;
  Name9x:array [0..12] of Char;
  NameNT:PWChar;
  i:Integer;
  ShareName: String;
begin
  if not IsNT(OS) then Close;
//Визначаємо тип системи

  if lbxShares.Items.Count = 0 then Exit;
  for i:= 0 to lbxShares.Items.Count-1 do
    if lbxShares.Selected[i] then Break;
//Шукаємо обраний елемент
  if not lbxShares.Selected[i] then Exit;
//Якщо не знайдений ідемо
  ShareName := lbxShares.Items.Strings[i];

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareDelNT := GetProcAddress(FLibHandle,' NetShareDel' );
    if not Assigned(NetShareDelNT) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    i:= SizeOf(WideChar)*256;
    GetMem(NameNT,i);
//Виділяємо пам' ять під змінну
    StringToWideChar(ShareName,NameNT,i);
//Перетворимо в PWideChar
    NetShareDelNT(nil,NameNT,0);
//Видаляємо ресурс
    FreeMem(NameNT);
//Звільняємо пам' ять
  end else begin
//Код для 9 x-ме
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareDel := GetProcAddress(FLibHandle,' NetShareDel' );
    if not Assigned(NetShareDel) then
//Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FillChar(Name9x, SizeOf(Name9x), #0);
//Очищаємо масив
    move(ShareName[1],Name9x[0],Length(ShareName));
//Заповнюємо масив
    NetShareDel(nil,@Name9x,0);
//Видаляємо ресурс
  end;
```

```
  FreeLibrary(FLibHandle);
end;
//
//  Показ діалогу вибору директорії
//

function TMainForm.SelectDirectory: String;
var
  lpItemID : PItemIDList;
  BrowseInfo : TBrowseInfo;
  DisplayName : array[0..MAX_PATH] of Char;
  TempPath : array[0..MAX_PATH] of Char;
begin
  FillChar(BrowseInfo, sizeof(TBrowseInfo), #0);
  BrowseInfo.hwndOwner := Handle;
  BrowseInfo.pszDisplayName := @DisplayName;
  BrowseInfo.lpszTitle := ' Specify a directory' ;
  BrowseInfo.ulFlags := BIF_RETURNONLYFSDIRS;
  lpItemID := SHBrowseForFolder(BrowseInfo);
  if Assigned(lpItemId) then begin
    SHGetPathFromIDList(lpItemID, TempPath);
    GlobalFreePtr(lpItemID);
  end else Result := ' ' ;
  Result := String(TempPath);
end;

//
//  Додавання загального ресурсу
//

procedure TMainForm.btnAddSharesClick(Sender: TObject);
const
  STYPE_DISKTREE = 0;
  ACCESS_ALL =  258;
  SHI50F_FULL = 258;
var
  FLibHandle : THandle;
  Share9x : TShareInfo50;
  ShareNT : TShareInfo2;
  TmpDir, TmpName: String;
  TmpDirNT, TmpNameNT: PWChar;
  OS: Boolean;
  TmpLength: Integer;
begin
  TmpDir := SelectDirectory;
//Визначаємо шлях до наступного ресурсу
  TmpName := InputBox(' Share name' ,' Enter name' ,' Test' );
//Визначаємо ім' я під яким він буде видний у мережевої системи моніторингу на
базі DevOps-архітектури
  if TmpDir = ' '  then Exit;

  if not IsNT(OS) then Close;
//З'ясовуємо тип системи

  if OS then begin
// Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareAddNT := GetProcAddress(FLibHandle,' NetShareAdd' );
    if not Assigned(NetShareAddNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    TmpLength := SizeOF(WideChar)*256;
//Визначаємо необхідний розмір

    GetMem(TmpNameNT, TmpLength);
//Конвертуємо в PWChar
```

```
    StringToWideChar(TmpName, TmpNameNT, TmpLength);
    ShareNT.shi2_netname := TmpNameNT;
//Ім' я

    ShareNT.shi2_type := STYPE_DISKTREE;
//Тип ресурсу
    ShareNT.shi2_remark := ' ' ;
//Коментар
    ShareNT.shi2_permissions := ACCESS_ALL;
//Доступ
    ShareNT.shi2_max_uses := DWORD(-1);
// Кіл-ть максим. підключ.
    ShareNT.shi2_current_uses := 0;
// Кіл-ть поточних підкл.

    GetMem(TmpDirNT, TmpLength);
    StringToWideChar(TmpDir, TmpDirNT, TmpLength);
    ShareNT.shi2_path := TmpDirNT;
//Шлях до ресурсу

    ShareNT.shi2_passwd := ' ' ; //Пароль

    NetShareAddNT(nil,2,@ShareNT, nil); //Додаємо ресурс
    FreeMem (TmpNameNT); //звільняємо пам' ять
    FreeMem (TmpDirNT);
  end else begin
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareAdd := GetProcAddress(FLibHandle,' NetShareAdd' );
    if not Assigned(NetShareAdd) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FillChar(Share9x.shi50_netname, SizeOf(Share9x.shi50_netname), #0);
    move(TmpName[1],Share9x.shi50_netname[0],Length(TmpName)); //Ім' я
    Share9x.shi50_type := STYPE_DISKTREE;
//Тип ресурсу
    Share9x.shi50_flags := SHI50F_FULL;
//Доступ
    FillChar(Share9x.shi50_remark,
      SizeOf(Share9x.shi50_remark), #0);
//Коментар
    FillChar(Share9x.shi50_path,
      SizeOf(Share9x.shi50_path), #0);
    Share9x.shi50_path := PAnsiChar(TmpDir); //Шлях до ресурсу
    FillChar(Share9x.shi50_rw_password,
      SizeOf(Share9x.shi50_rw_password), #0); //Пароль повного доступу
    FillChar(Share9x.shi50_ro_password,
      SizeOf(Share9x.shi50_ro_password), #0);
//Пароль для читання
    NetShareAdd(nil,50,@Share9x,SizeOf(Share9x));
  end;
  FreeLibrary(FLibHandle);
end;

//
//  активний і неактивний час сесій буде даватися нам
//  у вигляді кіл-ті секунд (тип Cardinal). Напишемо невелику
//  функцію, задача якої буде перетворювати Кіл-ть секунд у більше
//  звичну форму відображення.
//

function TMainForm.CardinalToTimeStr(Value: Cardinal): String;
var d,h,m,s: Real;
begin
    d:=0;
    h:=0;
    m:=0;
```

```
    s:=Value;
    if s > 59 then begin
        m:=int(s / 60);
        s:= s-s-(m*60);
    end;
    if m > 59 then begin
        h:=int(m/60);
        m:= m-m-(h*60);
    end;
    if h > 23 then begin
        d:=int(h/24);
        h:= h-h-(d*24);
    end;
    Result:=' ' ;
    if (d>0) then Result:=Result+floattostr(d)+'  d. ' ;
    if (h<9) then Result:=Result+' 0' +floattostr(h)+' :'  else
Result:=Result+floattostr(h)+' :' ;
    if (m<9) then Result:=Result+' 0' +floattostr(m)+' :'  else
Result:=Result+floattostr(m)+' :' ;
    if (s<9) then Result:=Result+' 0' +floattostr(s)     else
Result:=Result+floattostr(s);
end;

{
  Одержання списку сесій досліджуємої мережевої системи моніторингу на базі
DevOps-архітектури
}

procedure TMainForm.btnGetSessionsClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  SessionInfo50: array [0..512] of TSessionInfo50;
  SessionInfo502 : PSessionInfo502Array;
  TotalEntries,EntriesReadNT: DWORD;
  EntriesRead,TotalAvial: Word;
  i:integer;
begin
  lvSessions.Items.Clear;

  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionEnumNT := GetProcAddress(FLibHandle, ' NetSessionEnum' );
    if not Assigned(NetSessionEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    SessionInfo502 := nil;
    if NetSessionEnumNT(nil,nil,nil,502,@SessionInfo502,DWORD(-
1),@entriesreadNT, @totalentries, nil)=0 then
    for i:=0 to EntriesReadNT-1 do
    begin
      with lvSessions.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(SessionInfo502^[i].sesi502_cname);
//Ім' я комп' ютера
        SubItems.Add(SessionInfo502^[i].sesi502_username);
//Ім' я користувача
        SubItems.Add(IntToStr(SessionInfo502^[i].sesi502_num_opens));
//Відкритих ресурсів
        SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].Sesi502_Time)); //
//Час активний
        SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].sesi502_idle_time));
//Час не активний
      end;
```

```
    end;
  end else begin
//Код для Windows 9 x-Me
   FLibHandle := LoadLibrary(' SVRAPI.DLL' );
   if FLibHandle = 0 then Exit;
   @NetSessionEnum := GetProcAddress(FLibHandle, ' NetSessionEnum' );
    if not Assigned(NetSessionEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
   if NetSessionEnum
(nil,50,@SessionInfo50,SizeOf(SessionInfo50),@EntriesRead,@TotalAvial) = 0 then
   for i:=0 to EntriesRead-1 do
   begin
      with lvSessions.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(SessionInfo50[i].Sesi50_cname);
//Ім' я комп' ютера досліджуємої мережевої системи моніторингу на базі DevOps-
архітектури
        SubItems.Add(SessionInfo50[i].Sesi50_username);   //
//Ім' я користувача
        SubItems.Add(IntToStr(SessionInfo50[i].sesi50_num_opens));
//Відкритих ресурсів
        SubItems.Add(CardinalToTimeStr(SessionInfo50[i].Sesi50_Time));
//Час активний
        SubItems.Add(CardinalToTimeStr(SessionInfo50[i].sesi50_idle_time));
//Час не активний
        SessionCloseKey[i]:= SessionInfo50[i].sesi50_key;
//Унікальний ідентифікатор для закриття
      end;
   end;
 end;
 FreeLibrary(FLibHandle);
end;

//
//  Завершення обраної сесії
//

procedure TMainForm.btnCloseSessionClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  CNameNT: PWideChar;
  CName9x: PAnsiChar;
  Key:SmallInt;
  i: Integer;
begin
  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if not Assigned(lvSessions.Selected) then Exit;
  i:= lvSessions.Selected.Index; //Визначаємо номер обраної сесії

  if OS then begin
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionDelNT := GetProcAddress(FLibHandle, ' NetSessionDel' );
    if not Assigned(NetSessionDelNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CNameNT := PWChar(WideString(' \\' +lvSessions.Items.Item[i].Caption));
    NetSessionDelNT(nil,CNameNT,nil);
  end else begin
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
```

```
    @NetSessionDel := GetProcAddress(FLibHandle, ' NetSessionDel' );
    if not Assigned(NetSessionDel) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CName9x := PAnsiChar(lvSessions.Items.Item[i].Caption);
    key := SessionCloseKey[i]; //Беремо ключ із масиву
    NetSessionDel(nil,CName9x,Key);
  end;
  FreeLibrary(FLibHandle);
end;

{
  Одержання списку відкритих файлів досліджуємої мережевої системи моніторингу
на базі DevOps-архітектури
}

procedure TMainForm.btnGetFilesClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  FileInfoNT: PFileInfo3Array;
  FileInfo9x: array [0..512] of TFileInfo50;
  TotalEntries,EntriesReadNT: DWORD;
  EntriesRead,TotalAvial: Word;
  i:integer;
begin
  lvfiles.Items.Clear;

  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileEnumNT := GetProcAddress(FLibHandle, ' NetFileEnum' );
    if not Assigned(NetFileEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FileInfoNT := nil;
    if NetFileEnumNT(nil,nil,nil,3,@FileInfoNT,DWORD(-1),@entriesreadNT,
@totalentries, nil)=0 then
    for i:=0 to EntriesReadNT-1 do
    begin
      with lvFiles.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(IntToStr(FileInfoNT^[i].fi3_id)); //Ідентифікатор
        SubItems.Add(FileInfoNT^[i].fi3_pathname);   //Шлях до файлу
        SubItems.Add(FileInfoNT^[i].fi3_username);   //Ім' я користувача
      end;
    end;
  end else begin //Код для Windows 9 x-Me
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileEnum := GetProcAddress(FLibHandle, ' NetFileEnum' );
    if not Assigned(NetFileEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    if NetFileEnum (nil,
nil,50,@FileInfo9x,SizeOf(FileInfo9x),@EntriesRead,@TotalAvial)= 0 then
    for i:=0 to EntriesRead-1 do
    begin
      with lvFiles.Items.Add do  //Заповнення даними зі структури
      begin
```

```
        Caption := string(IntToStr(FileInfo9x[i].fi50_id)); //Ідентифікатор
        SubItems.Add(FileInfo9x[i].fi50_pathname); //Шлях до файлу
        SubItems.Add(FileInfo9x[i].fi50_username); //Ім' я користувача
      end;
   end;
 end;
 FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Закриття файлу
//

procedure TMainForm.btnCloseFileClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  i: Integer;
begin
  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if not Assigned(lvFiles.Selected) then Exit;
  i:= lvFiles.Selected.Index; //Визначаємо номер обраного файлу

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileClose := GetProcAddress(FLibHandle, ' NetFileClose' );
    if not Assigned(NetFileClose) then
    begin
      FreeLibrary(FLibHandle);
      Close;
    end;
    NetFileClose(nil,StrToInt(lvFiles.Items.Item[i].Caption));//Закриваємо файл
  end else begin //Код для Windows 9 x-Me
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileClose2 := GetProcAddress(FLibHandle, ' NetFileClose2' );
    if not Assigned(NetFileClose2) then
    begin
      FreeLibrary(FLibHandle);
      Close;
    end;
    NetFileClose2(nil,StrToInt(lvFiles.Items.Item[i].Caption));
  end;
  FreeLibrary(FLibHandle);
end;

{
  Визначаємо вхідний / вихідний трафік досліджуємої мережевої системи
моніторингу на базі DevOps-архітектури
}

procedure TMainForm.tmrTrafficTimer(Sender: TObject);
  // Допоміжна функція, що перетворить MAC адресу до "нормального" виду
  //Визначаємо спеціальний тип, щоб можна було передати у функцію масив
  type TMAC = array [0..7] of Byte;
  //Як перше значення масив, друге значення, розмір даних у масиві
  function GetMAC(Value: TMAC; Length: DWORD): String;
  var
    i: Integer;
  begin
    if Length = 0 then Result := '  00-00-00'  else
    begin
      Result := ' ' ;
      for i:= 0 to Length-2 do
        Result := Result + IntToHex(Value[i],2)+' -' ;
      Result := Result + IntToHex(Value[ Length-1],2);
```

```
    end;
  end;

//Сама процедура
var
  FLibHandle : THandle;
  Table: TMibIfTable;
  i : integer;
  Size   : integer;
begin
  tmrTraffic.Enabled := false; //Припиняємо таймер
  lvTraffic.Items.BeginUpdate;
  lvTraffic.Items.Clear;  //Очищаємо список
  FLibHandle := LoadLibrary(' DAPI.DLL' ); //Завантажуємо бібліотеку
  if FLibHandle = 0 then Exit;
  @GetIfTable := GetProcAddress(FLibHandle, ' GetIfTable' );
  if not Assigned(GetIfTable) then
  begin
    FreeLibrary(FLibHandle);
    Close;
  end;

  Size := SizeOf(Table);
  if GetIfTable(@Table, @Size, false ) = 0 then //Виконуємо функцію
    for i:= 0 to Table.dwNumEntries-1 do begin
      with lvTraffic.Items.Add do begin //Виводимо результати
        Caption := String(Table.Table[i].bDescr); //Найменування інтерфейсу
        SubItems.Add(GetMAC(TMAC(Table.Table[i].bPhysAddr),
          Table.Table[i].dwPhysAddrLen)); //MAC адреса
        SubItems.Add(IntToStr(Table.Table[i].dwInOctets)); //
// Усього прийнято байт з досліджуємої мережевої системи моніторингу на базі
DevOps-архітектури  (DATA)
        SubItems.Add(IntToStr(Table.Table[i].dwOutOctets));
//Усього відправлено байт у досліджуєму мережу
      end;
    end;
  lvTraffic.Items.EndUpdate;
  FreeLibrary(FLibHandle);
  tmrTraffic.Enabled := true;
//Не забуваємо активувати таймер
end;


 function OpenEnum(const NetContainerToOpen: PNetResource; ResScope, ResType,
ResUsage: DWORD): THandle;
var
 hNetEnum: THandle;
begin
 Result:=0;
 if (NO_ERROR<>WNetOpenEnum(ResScope, ResType, ResUsage,
                            NetContainerToOpen, hNetEnum))
 then ShowMessage(' Помилка!' )
 else Result:=hNetEnum;
end;

function EnumResources(const ParentNode: TTreeNode;
ResScope, ResType, ResUsage: DWORD; hNetEnum: THandle): UINT;
 function ShowResource(const ParentNode: TTreeNode; Res: TNetResource):
TTreeNode;
  begin
   Result:=MainForm.NetTree.Items.AddChild(ParentNode,
string(Res.lpRemoteName));
  end;

const
 RESOURCE_BUF_ENTRIES = 2000;

var
 ResourceBuffer: array[1..RESOURCE_BUF_ENTRIES] of TNetResource;
```

```
 i, ResourceBuf, EntriesToGet: dword;
 NewNode: TTreeNode;
begin
 Result:=0;
 while true do
  begin
   ResourceBuf:=sizeof(ResourceBuffer);
   EntriesToGet:=RESOURCE_BUF_ENTRIES;
   if (NO_ERROR<>WNetEnumResource(hNetEnum, EntriesToGet,
                                  @ResourceBuffer, ResourceBuf))
   then
    begin
     case GetLastError() of
      NO_ERROR:  // проход буферу без перемикання
      Break;
      ERROR_NO_MORE_ITEMS:
          // Повертає 0 у тому випадку, коли останов
          // RESOURCE_BUF_ENTRIES дані на попередньому виклику, щоб
          // WNetEnumResource, та були точно
          // RESOURCE_BUF_ENTRIES дані в запису на момент
          // попереднього виклику
      Exit;
      else ShowMessage(Помилка!' );
      Result:=1;
      Exit;
     end;
    end;
   for i:=1 to EntriesToGet do
    begin
     NewNode:=ShowResource(ParentNode, ResourceBuffer[i]);
     if (ResourceBuffer[i].dwUsage and RESOURCEUSAGE_CONTAINER)<>0
     then MainForm.Open_Do_Close_Enum(NewNode, ResScope, ResType, ResUsage,
@ResourceBuffer[i]);
     Application.ProcessMessages;
    end;
  end;
end;

procedure TMainForm.Open_Do_Close_Enum(const ParentNode: TTreeNode; ResScope,
ResType, ResUsage: DWORD; const NetContainerToOpen: PNetResource);
var
 hNetEnum: THandle;
begin
 hNetEnum:=OpenEnum(NetContainerToOpen, ResScope, ResType, ResUsage);
 if (hNetEnum=0)
 then Exit;
 EnumResources(ParentNode, ResScope, ResType, ResUsage, hNetEnum);
 if (NO_ERROR<>WNetCloseEnum(hNetEnum))
 then ShowMessage(' WNetCloseEnum Помилка' );
end;

procedure TMainForm.Button1Click(Sender: TObject);
var
 ResScope, ResType, ResUsage: dword;
begin
 Button1.Caption:=' Пошук мережних ресурсів. Чекайте...' ;
 Button1.Enabled:=false;
 //
 NetTree.Items.Clear;
 case rgScope.ItemIndex of
  1: ResScope:=RESOURCE_GLOBALNET;
  2: ResScope:=RESOURCE_REMEMBERED;
  else ResScope:=RESOURCE_CONNECTED;
 end;
 ResType:=0;
 if cbTypeAny.Checked
 then ResType:=ResType or RESOURCETYPE_ANY;
 if cbTypeDisk.Checked
 then ResType:=ResType or RESOURCETYPE_DISK;
```

```
 if cbTypePrint.Checked
 then ResType:=ResType or RESOURCETYPE_PRINT;
 ResUsage:=0;
 if cbUsageConnectable.Checked
 then ResUsage:=ResUsage or RESOURCEUSAGE_CONNECTABLE;
 if cbUsageContainer.Checked
 then ResUsage:=ResUsage or RESOURCEUSAGE_CONTAINER;
 Open_Do_Close_Enum(NetTree.Items.Add(nil, ' Network Resources' ),
                                ResScope, ResType, ResUsage, nil);
 //
 Button1.Caption:=' Обновити список ресурсів' ;
 Button1.Enabled:=true;

end;

procedure TMainForm.NetTreeCustomDrawItem(Sender: TCustomTreeView;
  Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
begin
if cdsSelected in State
 then Sender.Canvas.Font.Style:=Sender.Canvas.Font.Style+[fsUnderline];
end;

procedure TMainForm.NetTreeDblClick(Sender: TObject);
begin
ShellExecute(0,' open' ,PChar(NetTree.Selected.Text),' ',' ' ,SW_SHOW);
end;

procedure TMainForm.NetTreeGetImageIndex(Sender: TObject; Node: TTreeNode);
begin
 if Node.HasChildren
 then Node.ImageIndex:=1
 else Node.ImageIndex:=0;
end;

procedure TMainForm.Button4Click(Sender: TObject);
begin
Form1.Show;
end;

procedure TMainForm.Button2Click(Sender: TObject);
begin
Form2.Show;
end;

procedure TMainForm.Button3Click(Sender: TObject);
begin
Form3.Show;
end;

end.
```

Файл About.pas – довідкової інформації

```
unit About;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Button1: TButton;
    Image2: TImage;
    Image1: TImage;
    Image3: TImage;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
Form1.Close;
end;

end.
```

```
unit Stat;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, DataIP, DApi, Buttons;

type
  TForm3 = class(TForm)
    StaticText7: TStaticText;
    TCPStatMemo: TMemo;
    StaticText5: TStaticText;
    IPStatsMemo: TMemo;
    StaticText12: TStaticText;
    ICMPInMemo: TMemo;
    ICMPOutMemo: TMemo;
    StaticText4: TStaticText;
    UDPStatsMemo: TMemo;
    Timer1: TTimer;
    cbTimer: TCheckBox;
    btRTTI: TSpeedButton;
    edtRTTI: TEdit;
    procedure Timer1Timer(Sender: TObject);
    procedure btRTTIClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
     procedure DOIpStuff;
  public
    { Public declarations }
  end;

var
  Form3: TForm3;

implementation

{$R *.dfm}

procedure TForm3.DOIpStuff;
begin

  Get_TCPStatistics( TCPStatMemo.Lines );
  Get_IPStatistics( IPStatsMemo.Lines );
  Get_UDPStatistics( UDPStatsMemo.Lines );
  Get_ICMPStats( ICMPInMemo.Lines, ICMPOutMemo.Lines );

end;

procedure TForm3.Timer1Timer(Sender: TObject);
begin
  if cbTimer.State = cbCHECKED then
 begin
   Timer1.Enabled := false;
   DoIPStuff;
   Timer1.Enabled := true;
 end;
end;

procedure TForm3.btRTTIClick(Sender: TObject);
var
  IPadr         : dword;
  Rtt, HopCount : longint;
  Res           : integer;
```

```
begin
  btRTTI.Enabled := false;
  Screen.Cursor := crHOURGLASS;
  IPadr := Str2IPAddr( edtRTTI.Text );
  Res := Get_RTTAndHopCount( IPadr, 128, RTT, HopCount );
  if Res = NO_ERROR then
    ShowMessage( ' Час запиту '
      + inttostr( rtt ) + ' ms, '
      + inttostr( HopCount )
      + ' hops to : ' + edtRTTI.Text
      )
  else
    ShowMessage( ' Помилка:' + #13
                 + ICMPErr2Str( Res ) ) ;
  btRTTI.Enabled := true;
  Screen.Cursor := crDEFAULT;

end;

procedure TForm3.FormCreate(Sender: TObject);
begin

  if LoadD then
  begin
      DOIpStuff;
      Timer1.Enabled := true;
  end
  else
      ShowMessage( 'Інтернет помічник DLL не є доступним, або не підтримується'
) ;
end;

end.
```

Файл DAPI.pas - обробка API функцій розробленої системи

```
unit DAPI;

interface
uses
  Windows, winsock;

const
  VERSION       = ' 1.5' ;

//------- Заголовок з Microsoft IPTYPES.H-H-----------------

const
  ANY_SIZE      = 1;
  MAX_ADAPTER_DESCRIPTION_LENGTH = 128; // arb.
  MAX_ADAPTER_NAME_LENGTH = 256; // змінна
  MAX_ADAPTER_ADDRESS_LENGTH = 8; // змінна
  DEFAULT_MINIMUM_ENTITIES = 32; // змінна
  MAX_HOSTNAME_LEN = 128; // змінна
  MAX_DOMAIN_NAME_LEN = 128; // змінна
  MAX_SCOPE_ID_LEN = 256; // змінна

 // Вузлові типи ( NETBIOS)
  BROADCAST_NODETYPE = 1;
  PEER_TO_PEER_NODETYPE = 2;
  MIXED_NODETYPE = 4;
  HYBRID_NODETYPE = 8;

  NETBIOSTypes  : array[0..8] of string[20] = ( ' Невизначений' , ' Передача' ,
' Рівень до рівня' , ' ' , ' Змішаний' , ' ' , ' ' , ' ' , ' Гібрід'    );

 // Типи адаптеру
{ v1.4-> 1.5
  IF_OTHER_ADAPTERTYPE = 0;
  IF_ETHERNET_ADAPTERTYPE = 1;
  IF_TOKEN_RING_ADAPTERTYPE = 2;
  IF_FDDI_ADAPTERTYPE = 3;
  IF_PPP_ADAPTERTYPE = 4;
  IF_LOOPBACK_ADAPTERTYPE = 5;
  IF_SLIP_ADAPTERTYPE = 6;

Знайдено у ipifcons.h :
#define MIB_IF_TYPE_OTHER               1
#define MIB_IF_TYPE_ETHERNET            6
#define MIB_IF_TYPE_TOKENRING           9
#define MIB_IF_TYPE_FDDI                15
#define MIB_IF_TYPE_PPP                 23
#define MIB_IF_TYPE_LOOPBACK            24
#define MIB_IF_TYPE_SLIP                28
}
  IF_OTHER_ADAPTERTYPE = 1;
  IF_ETHERNET_ADAPTERTYPE = 6;
  IF_TOKEN_RING_ADAPTERTYPE = 9;
  IF_FDDI_ADAPTERTYPE = 15;
  IF_PPP_ADAPTERTYPE = 23;
  IF_LOOPBACK_ADAPTERTYPE = 24;
  IF_SLIP_ADAPTERTYPE = 28;

//  AdaptTypes    : array[0..6] of string[10] =
//    ( ' інший' , ' ethernet' , ' tokenring' , ' FDDI' , ' PPP' , ' loopback' ,
' SLIP'  );
  AdaptTypes    : array[1..28] of string[10] =
    ( ' інший' , ' ' , ' ' , ' ' , ' ' , ' ethernet' , ' ' , ' ' , ' tokenring'
,
    ' ' , ' ' , ' ' , ' ' , ' ' , ' FDDI' , ' ' , ' ' , ' ' , ' ' , ' ' , ' ' ,
' ' , ' PPP' ,     ' loopback' , ' ' , ' ' , ' ' , ' SLIP'  );
// Кінець змін в типі адаптерів

//-------для інших MS заготовочних файлів------------------
```

```
  MAX_INTERFACE_NAME_LEN = 256; { mrapi.h }
  MAXLEN_PHYSADDR = 8; { iprtrmib.h }
  MAXLEN_IFDESCR = 256; {-"--     }

//-----------------------------------

type
  TMacAddress = array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;

//---IP адресні структури-------------------------

  PTIP_ADDRESS_STRING = ^TIP_ADDRESS_STRING;
  TIP_ADDRESS_STRING = array[0..15] of char; //  IP рядок
  //
  PTIP_ADDR_STRING = ^TIP_ADDR_STRING;
  TIP_ADDR_STRING = packed record // для використання у зв' язних списках
    Next: PTIP_ADDR_STRING;
    IpAddress: TIP_ADDRESS_STRING;
    IpMask: TIP_ADDRESS_STRING;
    Context: DWORD;
  end;

//-----Fixed Info структура----------------------

  PTFixedInfo = ^TFixedInfo;
  TFixedInfo = packed record
    HostName: array[1..MAX_HOSTNAME_LEN + 4] of char;    // дані
    DomainName: array[1..MAX_DOMAIN_NAME_LEN + 4] of char;   // дані
    CurrentDNSServer: PTIP_ADDR_STRING;
    DNSServerList: TIP_ADDR_STRING;
    NodeType: UINT;
    ScopeID: array[1..MAX_SCOPE_ID_LEN + 4] of char;   // дані
    EnableRouting: UINT;
    EnableProxy: UINT;
    EnableDNS: UINT;
  end;

{ структура мережного інтерфейсу досліджуємої мережевої системи моніторингу на
базі DevOps-архітектури
(DATA)
}

//////////////////////
//                                                                    //
// Наступне є діючими станами для WAN да LAN інтерфейсів. //
// Порядок станів створений для визначення. Для      //
// стану >= CONNECTED можливо передавати дані зразу. Стан >= DISCONNECTED //
// може передавати деякі дані. Стан < DISCONNECTED може   //
// не передавати дані.                                                 //
// карта з поміткою UNREACHABLE якщо DIM викликає InterfaceUnreachable для
//
// причин. Крім невдачі з'єднання.                                  //
//                                                                    //
// NON_OPERATIONAL- Перевірка для LAN інтерфейсу. Позначає карту що не працює
//
//                      або не з'єднується з картою.        //
// UNREACHABLE- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт      //
//                      не з'єднується за потрібний час.
//
// DISCONNECTED- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт
//
//                      не з'єднується.
//
// CONNECTING- Перевірка WAN інтерфейсів . Означає спробу з'єднання     //
//                      з сайтом, якого немає.              //
// CONNECTED- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт      //
//                      з'єднується.                                    //
// OPERATIONAL- Перевірка LAN Interfaces. Позначає карту підключену //
```

```
//                      в праці.                                    //
//                                                                  //
// Усі дії користувачів записуються до  MIB-II значення    //
// можуть бути використовані                                        //
//                                                                  //
//////////////////////////

const
// дані додані до ipifcons.h
  IF_OPER_STATUS_NON_OPERATIONAL = 0 ;
  IF_OPER_STATUS_UNREACHABLE = 1 ;
  IF_OPER_STATUS_DISCONNECTED = 2 ;
  IF_OPER_STATUS_CONNECTING = 3 ;
  IF_OPER_STATUS_CONNECTED = 4 ;
  IF_OPER_STATUS_OPERATIONAL = 5 ;

  MIB_IF_TYPE_OTHER = 1 ;
  MIB_IF_TYPE_ETHERNET = 6 ;
  MIB_IF_TYPE_TOKENRING = 9 ;
  MIB_IF_TYPE_FDDI = 15 ;
  MIB_IF_TYPE_PPP = 23 ;
  MIB_IF_TYPE_LOOPBACK = 24 ;
  MIB_IF_TYPE_SLIP = 28 ;

  MIB_IF_ADMIN_STATUS_UP = 1 ;
  MIB_IF_ADMIN_STATUS_DOWN = 2 ;
  MIB_IF_ADMIN_STATUS_TESTING = 3 ;

  MIB_IF_OPER_STATUS_NON_OPERATIONAL = 0 ;
  MIB_IF_OPER_STATUS_UNREACHABLE = 1 ;
  MIB_IF_OPER_STATUS_DISCONNECTED = 2 ;
  MIB_IF_OPER_STATUS_CONNECTING = 3 ;
  MIB_IF_OPER_STATUS_CONNECTED = 4 ;
  MIB_IF_OPER_STATUS_OPERATIONAL = 5 ;

type
  PTMibIfRow = ^TMibIfRow;
  TMibIfRow = packed record
    wszName: array[1..MAX_INTERFACE_NAME_LEN] of WCHAR;
    dwIndex: DWORD;
    dwType: DWORD;       // дивись MIB_IF_TYPE
    dwMTU: DWORD;
    dwSpeed: DWORD;
    dwPhysAddrLen: DWORD;
    bPhysAddr: array[1..MAXLEN_PHYSADDR] of byte;
    dwAdminStatus: DWORD;    // дивись MIB_IF_ADMIN_STATUS
    dwOperStatus: DWORD;     // дивись MIB_IF_OPER_STATUS
    dwLastChange: DWORD;
    dwInOctets: DWORD;
    dwInUcastPkts: DWORD;
    dwInNUCastPkts: DWORD;
    dwInDiscards: DWORD;
    dwInErrors: DWORD;
    dwInUnknownProtos: DWORD;
    dwOutOctets: DWORD;
    dwOutUCastPkts: DWORD;
    dwOutNUCastPkts: DWORD;
    dwOutDiscards: DWORD;
    dwOutErrors: DWORD;
    dwOutQLen: DWORD;
    dwDescrLen: DWORD;
    bDescr: array[1..MAXLEN_IFDESCR] of char; //byte;
  end;

 //
  PTMibIfTable = ^TMIBIfTable;
  TMibIfTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIfRow;
```

```
  end;

//---ADAPTER INFO структура-----------------------

  PTIP_ADAPTER_INFO = ^TIP_ADAPTER_INFO;
  TIP_ADAPTER_INFO = packed record
    Next: PTIP_ADAPTER_INFO;
    ComboIndex: DWORD;
    AdapterName: array[1..MAX_ADAPTER_NAME_LENGTH + 4] of char;       // дані
    Description: array[1..MAX_ADAPTER_DESCRIPTION_LENGTH + 4] of char;    //
дані
    AddressLength: UINT;
    Address: array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;      // дані
    Index: DWORD;
    aType: UINT;
    DHCPEnabled: UINT;
    CurrentIPAddress: PTIP_ADDR_STRING;
    IPAddressList: TIP_ADDR_STRING;
    GatewayList: TIP_ADDR_STRING;
    DHCPServer: TIP_ADDR_STRING;
    HaveWINS: BOOL;
    PrimaryWINSServer: TIP_ADDR_STRING;
    SecondaryWINSServer: TIP_ADDR_STRING;
    LeaseObtained: LongInt ; // UNIX час, секунди з 1970
    LeaseExpires: LongInt;   // UNIX час, секунди з 1970
    SpareStuff: array [1..200] of char ;   // дані- простір для списку IP адрес
досліджуємої мережевої системи моніторингу на базі DevOps-архітектури (DATA)
  end;

//--------TCP структура--------------------

  PTMibTCPRow = ^TMibTCPRow;
  TMibTCPRow = packed record
    dwState: DWORD;
    dwLocalAddr: DWORD;
    dwLocalPort: DWORD;
    dwRemoteAddr: DWORD;
    dwRemotePort: DWORD;
  end;
  //
  PTMibTCPTable = ^TMibTCPTable;
  TMibTCPTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..0] of TMibTCPRow;
  end;
  //
  PTMibTCPStats = ^TMibTCPStats;
  TMibTCPStats = packed record
    dwRTOAlgorithm: DWORD;
    dwRTOMin: DWORD;
    dwRTOMax: DWORD;
    dwMaxConn: DWORD;
    dwActiveOpens: DWORD;
    dwPassiveOpens: DWORD;
    dwAttemptFails: DWORD;
    dwEstabResets: DWORD;
    dwCurrEstab: DWORD;
    dwInSegs: DWORD;
    dwOutSegs: DWORD;
    dwRetransSegs: DWORD;
    dwInErrs: DWORD;
    dwOutRsts: DWORD;
    dwNumConns: DWORD;
  end;

//-----UDP СТРУКТУРА --------------------------

  PTMibUDPRow = ^TMibUDPRow;
  TMibUDPRow = packed record
```

```
    dwLocalAddr: DWORD;
    dwLocalPort: DWORD;
  end;
 //
  PTMibUDPTable = ^TMIBUDPTable;
  TMIBUDPTable = packed record
    dwNumEntries: DWORD;
    UDPTable: array[0..ANY_SIZE- 1] of TMibUDPRow;
  end;
 //
  PTMibUdpStats = ^TMIBUdpStats;
  TMIBUdpStats = packed record
    dwInDatagrams: DWORD;
    dwNoPorts: DWORD;
    dwInErrors: DWORD;
    dwOutDatagrams: DWORD;
    dwNumAddrs: DWORD;
  end;

//------IP СТРУКТУРА ------------------------

 //
  PTMibIPNetRow = ^TMibIPNetRow;
  TMibIPNetRow = packed record
    dwIndex: DWord;
    dwPhysAddrLen: DWord;
    bPhysAddr: TMACAddress;
    dwAddr: DWord;
    dwType: DWord;
  end;
  //
  PTMibIPNetTable = ^TMibIPNetTable;
  TMibIPNetTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPNetRow;
  end;
  //
  PTMibIPStats = ^TMibIPStats;
  TMibIPStats = packed record
    dwForwarding: DWORD;
    dwDefaultTTL: DWORD;
    dwInReceives: DWORD;
    dwInHdrErrors: DWORD;
    dwInAddrErrors: DWORD;
    dwForwDatagrams: DWORD;
    dwInUnknownProtos: DWORD;
    dwInDiscards: DWORD;
    dwInDelivers: DWORD;
    dwOutRequests: DWORD;
    dwRoutingDiscards: DWORD;
    dwOutDiscards: DWORD;
    dwOutNoRoutes: DWORD;
    dwReasmTimeOut: DWORD;
    dwReasmReqds: DWORD;
    dwReasmOKs: DWORD;
    dwReasmFails: DWORD;
    dwFragOKs: DWORD;
    dwFragFails: DWORD;
    dwFragCreates: DWORD;
    dwNumIf: DWORD;
    dwNumAddr: DWORD;
    dwNumRoutes: DWORD;
  end;
  //
  PTMibIPAddrRow = ^TMibIPAddrRow;
  TMibIPAddrRow = packed record
    dwAddr: DWORD;
    dwIndex: DWORD;
    dwMask: DWORD;
```

```
    dwBCastAddr: DWORD;
    dwReasmSize: DWORD;
    Unused1,
    Unused2: WORD;
  end;
  //
  PTMibIPAddrTable = ^TMibIPAddrTable;
  TMibIPAddrTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPAddrRow;
  end;

  //
  PTMibIPForwardRow = ^TMibIPForwardRow;
  TMibIPForwardRow = packed record
    dwForwardDest: DWORD;
    dwForwardMask: DWORD;
    dwForwardPolicy: DWORD;
    dwForwardNextHop: DWORD;
    dwForwardIFIndex: DWORD;
    dwForwardType: DWORD;
    dwForwardProto: DWORD;
    dwForwardAge: DWORD;
    dwForwardNextHopAS: DWORD;
    dwForwardMetric1: DWORD;
    dwForwardMetric2: DWORD;
    dwForwardMetric3: DWORD;
    dwForwardMetric4: DWORD;
    dwForwardMetric5: DWORD;
  end;
  //
  PTMibIPForwardTable = ^TMibIPForwardTable;
  TMibIPForwardTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPForwardRow;
  end;

//---- ICMP-СТРУКТУРА ----------------------

  PTMibICMPStats = ^TMibICMPStats;
  TMibICMPStats = packed record
    dwMsgs: DWORD;
    dwErrors: DWORD;
    dwDestUnreachs: DWORD;
    dwTimeEcxcds: DWORD;
    dwParmProbs: DWORD;
    dwSrcQuenchs: DWORD;
    dwRedirects: DWORD;
    dwEchos: DWORD;
    dwEchoReps: DWORD;
    dwTimeStamps: DWORD;
    dwTimeStampReps: DWORD;
    dwAddrMasks: DWORD;
    dwAddrReps: DWORD;
  end;

  PTMibICMPInfo = ^TMibICMPInfo;
  TMibICMPInfo = packed record
    InStats: TMibICMPStats;
    OutStats: TMibICMPStats;
  end;

//---------імпорт до DAPI.DLL------------------

var

GetAdaptersInfo: function ( pAdapterInfo: PTIP_ADAPTER_INFO;
  pOutBufLen: PULONG ): DWORD; stdcall;
```

```
GetNetworkParams: function ( FixedInfo: PTFixedInfo; pOutPutLen: PULONG ):
        DWORD; stdcall;

GetTcpTable: function ( pTCPTable: PTMibTCPTable; pDWSize: PDWORD;
  bOrder: BOOL ): DWORD; stdcall;

GetTcpStatistics: function ( pStats: PTMibTCPStats ): DWORD; stdcall;

GetUdpTable: function ( pUdpTable: PTMibUDPTable; pDWSize: PDWORD;
 bOrder: BOOL ): DWORD; stdcall;

GetUdpStatistics: function ( pStats: PTMibUdpStats ): DWORD; stdcall;

GetIpStatistics: function ( pStats: PTMibIPStats ): DWORD; stdcall;

GetIpNetTable: function ( pIpNetTable: PTMibIPNetTable;
  pdwSize: PULONG;  bOrder: BOOL ): DWORD; stdcall;

GetIpAddrTable: function ( pIpAddrTable: PTMibIPAddrTable;
  pdwSize: PULONG; bOrder: BOOL ): DWORD; stdcall;

GetIpForwardTable: function ( pIPForwardTable: PTMibIPForwardTable;
  pdwSize: PULONG; bOrder: BOOL ): DWORD; stdCall;

GetIcmpStatistics: function ( pStats: PTMibICMPInfo ): DWORD; stdCall;

GetRTTAndHopCount: function ( DestIPAddress: DWORD; HopCount: PULONG;
  MaxHops: ULONG; RTT: PULONG ): BOOL; stdCall;

GetIfTable: function ( pIfTable: PTMibIfTable; pdwSize: PULONG;
  bOrder: boolean ): DWORD; stdCall;

GetIfEntry: function ( pIfRow: PTMibIfRow ): DWORD; stdCall;

// попередження – недокументована функція, можливі помилки при використанні
GetFriendlyIfIndex: function (var IfIndex: DWORD): DWORD; stdcall;


const
    DDLL = ' DAPI.DLL' ;
var
    DModule: THandle;

    function LoadD: Boolean;

implementation

function LoadD: Boolean;
begin
    Result := True;
    if DModule <> 0 then Exit;

// відкрити DLL
    DModule := LoadLibrary (DDLL);
    if DModule = 0 then
    begin
        Result := false;
        exit ;
    end ;
    GetAdaptersInfo := GetProcAddress (DModule, ' GetAdaptersInfo' ) ;
    GetNetworkParams := GetProcAddress (DModule, ' GetNetworkParams' ) ;
    GetTcpTable := GetProcAddress (DModule, ' GetTcpTable' ) ;
    GetTcpStatistics := GetProcAddress (DModule, ' GetTcpStatistics' ) ;
    GetUdpTable := GetProcAddress (DModule, ' GetUdpTable' ) ;
    GetUdpStatistics := GetProcAddress (DModule, ' GetUdpStatistics' ) ;
    GetIpStatistics := GetProcAddress (DModule, ' GetIpStatistics' ) ;
    GetIpNetTable := GetProcAddress (DModule, ' GetIpNetTable' ) ;
    GetIpAddrTable := GetProcAddress (DModule, ' GetIpAddrTable' ) ;
    GetIpForwardTable := GetProcAddress (DModule, ' GetIpForwardTable' ) ;
```

```
    GetIcmpStatistics := GetProcAddress (DModule, ' GetIcmpStatistics' ) ;
    GetRTTAndHopCount := GetProcAddress (DModule, ' GetRTTAndHopCount' ) ;
    GetIfTable := GetProcAddress (DModule, ' GetIfTable' ) ;
    GetIfEntry := GetProcAddress (DModule, ' GetIfEntry' ) ;
    GetFriendlyIfIndex := GetProcAddress (DModule, ' GetFriendlyIfIndex' ) ;
end;

initialization
    DModule := 0 ;
finalization
    if DModule <> 0 then
    begin
        FreeLibrary (DModule) ;
        DModule := 0 ;
    end ;

end.
```

Файл DataIP.pas- функції роботи з IP-протоколом

```
unit DataIP;


interface

uses
  Windows, Messages, SysUtils, Classes, Dialogs, DApi;

const
  NULL_IP       = '   0.0.  0.0' ;

//---перетворення добре відомих номерів портів до імен сервісів--------

type
  TWellKnownPort = record
    Prt: DWORD;
    Srv: string[20];
  end;


const
    // тільки найбільш популярні сервіси...
  WellKnownPorts: array[1..32] of TWellKnownPort
  = (
//    ( Prt: 0; Srv:   ' RESRVED'  ),     {Зарезервовано}
    ( Prt: 7; Srv:   ' ECHO   '  ),     {Ping    }
    ( Prt: 9; Srv:   ' DISCARD'  ),
    ( Prt: 13; Srv:  ' DAYTIME'  ),
    ( Prt: 17; Srv:  ' QOTD   '  ),     {Показчик на день}
    ( Prt: 19; Srv:  ' CHARGEN'  ),     {Генератор символів}
    ( Prt: 20; Srv:  ' FTPDATA'  ),     { File Transfer Protocol- дані}
    ( Prt: 21; Srv:  ' FTPCTRL'  ),     { File Transfer Protocol- управління}
    ( Prt: 22; Srv:  ' SSH    '  ),
    ( Prt: 23; Srv:  ' TELNET '  ),
    ( Prt: 25; Srv:  ' SMTP   '  ),     { Simple Mail Transfer Protocol}
    ( Prt: 37; Srv:  ' TIME   '  ),     { Часовий протокол }
    ( Prt: 43; Srv:  ' WHOIS  '  ),     { Сервіс – Кто це  }
    ( Prt: 53; Srv:  ' DNS    '  ),     { Domain Name Service }
    ( Prt: 67; Srv:  ' BOOTPS '  ),     { BOOTP Сервер }
    ( Prt: 68; Srv:  ' BOOTPC '  ),     { BOOTP Кієнт }
    ( Prt: 69; Srv:  ' TFTP   '  ),     { стандартний  FTP  }
    ( Prt: 70; Srv:  ' GOPHER '  ),     { Протокол Gopher       }
    ( Prt: 79; Srv:  ' FINGER '  ),     { Протокол Finger       }
    ( Prt: 80; Srv:  ' HTTP   '  ),     { Протокол HTTP         }
    ( Prt: 88; Srv:  ' KERBROS'  ),     { Протокол Kerberos     }
    ( Prt: 109; Srv: ' POP2   '  ),     { Протокол Post Office Protocol Version
2 }
    ( Prt: 110; Srv: ' POP3   '  ),     { Протокол Post Office Protocol Version
3 }
    ( Prt: 111; Srv: ' SUN_RPC'  ),     { Протокол SUN Remote Procedure Call }
    ( Prt: 119; Srv: ' NNTP   '  ),     { Протокол Network News Transfer
Protocol }
    ( Prt: 123; Srv: ' NTP    '  ),     { Протокол Network Time protocol       }
    ( Prt: 135; Srv: ' DCOMRPC'  ),     { Протокол Location Service           }
    ( Prt: 137; Srv: ' NBNAME '  ),     { NETBIOS сервіс імен          }
    ( Prt: 138; Srv: ' NBDGRAM'  ),     { NETBIOS сервіс датаграм     }
    ( Prt: 139; Srv: ' NBSESS '  ),     { NETBIOS сервіс сесій        }
    ( Prt: 143; Srv: ' IMAP   '  ),     { Протокол Internet Message Access
Protocol }
    ( Prt: 161; Srv: ' SNMP   '  ),
{ Протокол Simple Netw. Management Protocol }
    ( Prt: 169; Srv: ' SEND   '  )
    );
```

```
//------перетворення ICMP кодів помилок до рядків-------------

const
  ICMP_ERROR_BASE = 11000;
  IcmpErr : array[1..22] of string =
  (
   ' IP_BUFFER_TOO_SMALL' ,' IP_DEST_NET_UNREACHABLE' , '
IP_DEST_HOST_UNREACHABLE' ,
   ' IP_PROTOCOL_UNREACHABLE' , ' IP_DEST_PORT_UNREACHABLE' , ' IP_NO_RESOURCES'
,
   ' IP_BAD_OPTION' ,' IP_HARDWARE_ПОМИЛКА' , ' IP_PACKET_TOO_BIG' , '
IP_REQUEST_TIMED_OUT' ,
   ' IP_BAD_REQUEST' ,' IP_BAD_ROUTE' , ' IP_TTL_EXPIRED_TRANSIT' ,
   ' IP_TTL_EXPIRED_REASSEM' ,' IP_PARAMETER_PROBLEM' , ' IP_SOURCE_QUENCH' ,
   ' IP_OPTION_TOO_BIG' , ' IP_BAD_DESTINATION' ,' IP_ADDRESS_DELETED' ,
   ' IP_SPEC_MTU_CHANGE' , ' IP_MTU_CHANGE' , ' IP_UNLOAD'
  );


//-----Перетворення різних перерахованих величин у рядки---------

  ARPEntryType  : array[1..4] of string = ( ' інший' , ' неправильний' ,
    ' динамічний' , ' статичний'
    );
  TCPConnState  :
    array[1..12] of string =
    ( ' closed' , ' listening' , ' syn_sent' ,
    ' syn_rcvd' , ' established' , ' fin_wait1' ,
    ' fin_wait2' , ' close_wait' , ' closing' ,
    ' last_ack' , ' time_wait' , ' delete_tcb'
    );

  TCPToAlgo     : array[1..4] of string =
    ( ' Const.Timeout' , '  MIL-STD-1778' ,
    ' Van Jacobson' , ' інший' );

  IPForwTypes   : array[1..4] of string =
    ( ' інший' , ' invalid' , ' local' , ' remote'  );

  IPForwProtos  : array[1..18] of string =
    ( ' інший' , ' LOCAL' , ' NETMGMT' , ' ICMP' , ' EGP' ,
    ' GGP' , ' HELO' , ' RIP' , ' IS_IS' , ' ES_IS' ,
    ' CISCO' , ' BBN' , ' OSPF' , ' BGP' , ' BOOTP' ,
    ' AUTO_STAT' , ' STATIC' , ' NOT_DOD'  );

type

// для DNetworkParams
  TNetworkParams = record
    HostName: string ;
    DomainName: string ;
    CurrentDnsServer: string ;
    DnsServerTot: integer ;
    DnsServerNames: array [0..9] of string ;
    NodeType: UINT;
    ScopeID: string ;
    EnableRouting: UINT;
    EnableProxy: UINT;
    EnableDNS: UINT;
  end;

  TIfRows = array of TMibIfRow ; // динамічний масив колонок

// для DAdaptersInfo
  TAdaptorInfo = record
    AdapterName: string ;
    Description: string ;
    MacAddress: string ;
    Index: DWORD;
```

```
    aType: UINT;
    DHCPEnabled: UINT;
    CurrIPAddress: string ;
    CurrIPMask: string ;
    IPAddressTot: integer ;
    IPAddressList: array of string ;
    IPMaskList: array of string ;
    GatewayTot: integer ;
    GatewayList: array of string ;
    DHCPTot: integer ;
    DHCPServer: array of string ;
    HaveWINS: BOOL;
    PrimWINSTot: integer ;
    PrimWINSServer: array of string ;
    SecWINSTot: integer ;
    SecWINSServer: array of string ;
    LeaseObtained: LongInt ; // UNIX час, секунди з 1970
    LeaseExpires: LongInt;   // UNIX час, секунди з 1970
  end ;

  TAdaptorRows = array of TAdaptorInfo ;


//--------експортуємі дані---------------------

function DAdaptersInfo(var AdpTot: integer;var AdpRows: TAdaptorRows): integer ;
procedure Get_AdaptersInfo( List: TStrings );
function DNetworkParams (var NetworkParams: TNetworkParams): integer ;
procedure Get_NetworkParams( List: TStrings );
procedure Get_ARPTable( List: TStrings );
procedure Get_TCPTable( List: TStrings );
procedure Get_TCPStatistics( List: TStrings );
function DTCPStatistics (var TCPStats: TMibTCPStats): integer ;
procedure Get_UDPTable( List: TStrings );
procedure Get_UDPStatistics( List: TStrings );
function DUdpStatistics (UdpStats: TMibUDPStats): integer ;
procedure Get_IPAddrTable( List: TStrings );
procedure Get_IPForwardTable( List: TStrings );
procedure Get_IPStatistics( List: TStrings );
function DIPStatistics (var IPStats: TMibIPStats): integer ;
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint;
  var RTT: longint; var HopCount: longint ): integer;
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
function DIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
procedure Get_IfTable( List: TStrings );
function DIfEntry(Index: integer; var IfRow: TMibIfRow): integer ;
procedure Get_RecentDestIPs( List: TStrings );

// функції перетворення
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
function IpAddr2Str( IPAddr: DWORD ): string;
function Str2IpAddr( IPStr: string ): DWORD;
function Port2Str( nwoPort: DWORD ): string;
function Port2Wrd( nwoPort: DWORD ): DWORD;
function Port2Svc( Port: DWORD ): string;
function ICMPErr2Str( ICMPErrCode: DWORD) : string;

implementation

var
  RecentIPs     : TStringList;

//-------Основні функції----------------------

{ отримання наступного "токена" з рядка }
function NextToken( var s: string; Separator: char ): string;
var
  Sep_Pos       : byte;
begin
```

```
  Result := ' ' ;
  if length( s ) > 0 then begin
    Sep_Pos := pos( Separator, s );
    if Sep_Pos > 0 then begin
      Result := copy( s, 1, Pred( Sep_Pos ) );
      Delete( s, 1, Sep_Pos );
    end
    else begin
      Result := s;
      s := ' ' ;
    end;
  end;
end;

//-----------------------------------
{ перетворення числового MAC-адреса  до ww-xx-yy-zz рядка }
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
var
  i             : integer;
begin
  if Size = 0 then
  begin
    Result := '  00-00-00' ;
    EXIT;
  end
  else Result := ' ' ;
  //
  for i := 1 to Size do
    Result := Result + IntToHex( MacAddr[i], 2) + ' -' ;
  Delete( Result, Length( Result ), 1 );
end;

//-----------------------------------
{ перетворення IP-адреси в мережний байт типу DWORD }
function IpAddr2Str( IPAddr: DWORD ): string;
var
  i             : integer;
begin
  Result := ' ' ;
  for i := 1 to 4 do
  begin
    Result := Result + Format( ' %3d.' , [IPAddr and $FF] );
    IPAddr := IPAddr shr 8;
  end;
  Delete( Result, Length( Result ), 1 );
end;

//-----------------------------------
{ перетворення крапкової десяткової IP-адреси в мережний байт типу DWORD}
function Str2IpAddr( IPStr: string ): DWORD;
var
  i             : integer;
  Num           : DWORD;
begin
  Result := 0;
  for i := 1 to 4 do
  try
    Num := ( StrToInt( NextToken( IPStr, ' .'  ) ) ) shl 24;
    Result := ( Result shr 8 ) or Num;
  except
    Result := 0;
  end;

end;

//-----------------------------------
{ перетворення номеру порту в мережний байт типу DWORD }
function Port2Wrd( nwoPort: DWORD ): DWORD;
begin
```

```
  Result := Swap( WORD( nwoPort ) );
end;

//-----------------------------------
{ перетворення номеру порту в мережний байт типу string }
function Port2Str( nwoPort: DWORD ): string;
begin
  Result := IntToStr( Port2Wrd( nwoPort ) );
end;

//-----------------------------------
{ перетворення номеру порту в сервіс ID }
function Port2Svc( Port: DWORD ): string;
var
  i             : integer;
begin
  Result := Format( ' %4d' , [Port] ); // у випадку, якщо порт не знайдено
  for i := Low( WellKnownPorts ) to High( WellKnownPorts ) do
    if Port = WellKnownPorts[i].Prt then
    begin
      Result := WellKnownPorts[i].Srv;
      BREAK;
    end;
end;

//-----------------------------------
{ голона частина, фіксація мережний параметрів досліджуємої мережевої системи
моніторингу на базі DevOps-архітектури (DATA)}

procedure Get_NetworkParams( List: TStrings );
var
    NetworkParams: TNetworkParams ;
    I, ErrorCode: integer ;
begin
    if not Assigned( List ) then EXIT;
    List.Clear;
    ErrorCode := DNetworkParams (NetworkParams) ;
    if ErrorCode <> 0 then
    begin
        List.Add (SysErrorMessage (ErrorCode));
        exit;
    end ;
    with NetworkParams do
    begin
        List.Add( ' Ім'я хосту          : '  + HostName );
        List.Add( ' Домен            : '  + DomainName );
        List.Add( ' NETBIOS тип : '  + NETBIOSTypes[NodeType] );
        List.Add( ' DHCP область        : '  + ScopeID );
        List.Add( ' ROUTING визначено   : '  + IntToStr( EnableRouting ) );
        List.Add( ' PROXY   визначено   : '  + IntToStr( EnableProxy ) );
        List.Add( ' DNS     визначено   : '  + IntToStr( EnableDNS ) );
        if DnsServerTot <> 0 then
        begin
            for I := 0 to Pred (DnsServerTot) do
                List.Add( ' DNS адреса серверу   : '  + DnsServerNames [I] ) ;
        end ;
    end ;
end ;

//-------------------------------------//
function DNetworkParams (var NetworkParams: TNetworkParams): integer ;
var
  FixedInfo     : PTFixedInfo;         // дані
  InfoSize      : Longint;
  PDnsServer    : PTIP_ADDR_STRING ;   // дані
begin
    InfoSize := 0 ;   // дані
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadD then exit ;
```

```
    result := GetNetworkParams( Nil, @InfoSize );  // дані
    if result <> ERROR_BUFFER_OVERFLOW then exit ; // дані
    GetMem (FixedInfo, InfoSize) ;                    // дані
    try
    result := GetNetworkParams( FixedInfo, @InfoSize );   // дані
    if result <> ERROR_SUCCESS then exit ;
    NetworkParams.DnsServerTot := 0 ;
    with FixedInfo^ do
    begin
        NetworkParams.HostName := trim (HostName) ;
        NetworkParams.DomainName := trim (DomainName) ;
        NetworkParams.ScopeId := trim (ScopeID) ;
        NetworkParams.NodeType := NodeType ;
        NetworkParams.EnableRouting := EnableRouting ;
        NetworkParams.EnableProxy := EnableProxy ;
        NetworkParams.EnableDNS := EnableDNS ;
        NetworkParams.DnsServerNames [0] := DNSServerList.IPAddress ;  // дані
        if NetworkParams.DnsServerNames [0] <> ' '  then
                                        NetworkParams.DnsServerTot := 1 ;
        PDnsServer := DnsServerList.Next;
        while PDnsServer <> Nil do
        begin
            NetworkParams.DnsServerNames [NetworkParams.DnsServerTot] :=
                                        PDnsServer^.IPAddress ;  // дані
            inc (NetworkParams.DnsServerTot) ;
            if NetworkParams.DnsServerTot >=
                            Length (NetworkParams.DnsServerNames) then exit ;
            PDnsServer := PDnsServer.Next ;
        end;
    end ;
    finally
       FreeMem (FixedInfo) ;                    // дані
    end ;
end;

//---------------------------------

function ICMPErr2Str( ICMPErrCode: DWORD) : string;
begin
   Result := ' UnknownError : '  + IntToStr( ICMPErrCode );
   dec( ICMPErrCode, ICMP_ERROR_BASE );
   if ICMPErrCode in [Low(ICMpErr)..High(ICMPErr)] then
     Result := ICMPErr[ ICMPErrCode];
end;


//---------------------------------

// включення байтів у/з для кожного адаптера

function DIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
var
  I,
  TableSize   : integer;
  pBuf, pNext : PChar;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadD then exit ;
  SetLength (IfRows, 0) ;
  IfTot := 0 ; // дані
  TableSize := 0;
   // перший виклик: необхідно отримати розмір пам' яті
  result := GetIfTable (Nil, @TableSize, false) ;  // дані
  if result <> ERROR_INSUFFICIENT_BUFFER then exit ;
  GetMem( pBuf, TableSize );
  try
      FillChar (pBuf^, TableSize, #0);  // очищаємо буфер з  W98 не беремо
крапку таблиці
      result := GetIfTable (PTMibIfTable (pBuf), @TableSize, false) ;
```

```
      if result <> NO_ERROR then exit ;
      IfTot := PTMibIfTable (pBuf)^.dwNumEntries ;
      if IfTot = 0 then exit ;
      SetLength (IfRows, IfTot) ;
      pNext := pBuf + SizeOf(IfTot) ;
      for i := 0 to Pred (IfTot) do
      begin
         IfRows [i] := PTMibIfRow (pNext )^ ;
         inc (pNext, SizeOf (TMibIfRow)) ;
      end;
  finally
      FreeMem (pBuf) ;
  end ;
end;

procedure Get_IfTable( List: TStrings );
var
  IfRows        : TIfRows ;
  Error, I      : integer;
  NumEntries    : integer;
  sDescr, sIfName: string ;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  SetLength (IfRows, 0) ;
  Error := DIfTable (NumEntries, IfRows) ;
  if (Error <> 0) then
      List.Add( SysErrorMessage( GetLastError ) )
  else if NumEntries = 0 then
      List.Add( ' данних немає '  )
  else
  begin
      for I := 0 to Pred (NumEntries) do
      begin
          with IfRows [I] do
          begin
             if wszName [1] = #0 then
                 sIfName := ''
             else
                 sIfName := WideCharToString (@wszName) ;  // конвертуємо Юнікод
до рядка
             sIfName := trim (sIfName) ;
             sDescr := bDescr ;
             sDescr := trim (sDescr);
             List.Add (Format (
              ' %0.8x |%3d | %16s |%8d |%12d |%2d |%2d |%10d |%10d | %-s| %-s'
,
               [dwIndex, dwType, MacAddr2Str( TMacAddress( bPhysAddr ),
               dwPhysAddrLen ), dwMTU, dwSpeed, dwAdminStatus,
               dwOPerStatus, Int64 (dwInOctets), Int64 (dwOutOctets),  //
конвертуємо до 32-біт
                sIfName, sDescr] )  // дані, додані в/з
               );
          end;
      end ;
  end ;
  SetLength (IfRows, 0) ;  // вільна пам' ять
end ;

function DIfEntry(Index: integer; var IfRow: TMibIfRow): integer ;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadD then exit ;
  FillChar (IfRow, SizeOf (TMibIfRow), #0);  // очищаємо буфер з  W98 не беремо
  IfRow.dwIndex := Index ;
  result := GetIfEntry (@IfRow) ;
end ;

//---------------------------------
```

```
{ інформація про інстальовані адаптери }

function DAdaptersInfo(var AdpTot: integer; var AdpRows: TAdaptorRows): integer
;
var
  BufLen        : DWORD;
  AdapterInfo   : PTIP_ADAPTER_INFO;
  PIpAddr       : PTIP_ADDR_STRING;
  PBuf          : PCHAR ;
  I             : integer ;
begin
  SetLength (AdpRows, 4) ;
  AdpTot := 0 ;
  BufLen := 0 ;
  result := GetAdaptersInfo( Nil, @BufLen );
  if (result <> ERROR_INSUFFICIENT_BUFFER) and (result = NO_ERROR) then exit ;
  GetMem( pBuf, BufLen );
  try
      FillChar (pBuf^, BufLen, #0);  // очищуємо буфер
      result := GetAdaptersInfo( PTIP_ADAPTER_INFO (PBuf), @BufLen );
      if result = NO_ERROR then
      begin
         AdapterInfo := PTIP_ADAPTER_INFO (PBuf) ;
         while ( AdapterInfo <> nil ) do
         begin
            AdpRows [AdpTot].IPAddressTot := 0 ;
            SetLength (AdpRows [AdpTot].IPAddressList, 2) ;
            SetLength (AdpRows [AdpTot].IPMaskList, 2) ;
            AdpRows [AdpTot].GatewayTot := 0 ;
            SetLength (AdpRows [AdpTot].GatewayList, 2) ;
            AdpRows [AdpTot].DHCPTot := 0 ;
            SetLength (AdpRows [AdpTot].DHCPServer, 2) ;
            AdpRows [AdpTot].PrimWINSTot := 0 ;
            SetLength (AdpRows [AdpTot].PrimWINSServer, 2) ;
            AdpRows [AdpTot].SecWINSTot := 0 ;
            SetLength (AdpRows [AdpTot].SecWINSServer, 2) ;
            AdpRows [AdpTot].CurrIPAddress := NULL_IP;
            AdpRows [AdpTot].CurrIPMask := NULL_IP;
            AdpRows [AdpTot].AdapterName := Trim( string(
AdapterInfo^.AdapterName ) );
            AdpRows [AdpTot].Description := Trim( string(
AdapterInfo^.Description ) );
            AdpRows [AdpTot].MacAddress := MacAddr2Str( TMacAddress(
                                  AdapterInfo^.Address ),
AdapterInfo^.AddressLength ) ;
            AdpRows [AdpTot].Index := AdapterInfo^.Index ;
            AdpRows [AdpTot].aType := AdapterInfo^.aType ;
            AdpRows [AdpTot].DHCPEnabled := AdapterInfo^.DHCPEnabled ;
            if AdapterInfo^.CurrentIPAddress <> Nil then
            begin
                AdpRows [AdpTot].CurrIPAddress :=
AdapterInfo^.CurrentIPAddress.IpAddress ;
                AdpRows [AdpTot].CurrIPMask :=
AdapterInfo^.CurrentIPAddress.IpMask ;
            end ;

        // беремо список IP адрес та конвертуємо в IPAddressList
            I := 0 ;
            PIpAddr := @AdapterInfo^.IPAddressList ;
            while (PIpAddr <> Nil) do
            begin
                AdpRows [AdpTot].IPAddressList [I] := PIpAddr.IpAddress ;
                AdpRows [AdpTot].IPMaskList [I] := PIpAddr.IpMask ;
                PIpAddr := PIpAddr.Next ;
                inc (I) ;
                if Length (AdpRows [AdpTot].IPAddressList) <= I then
                begin
                     SetLength (AdpRows [AdpTot].IPAddressList, I -2) ;
                     SetLength (AdpRows [AdpTot].IPMaskList, I -2) ;
```

```
            end ;
        end ;
        AdpRows [AdpTot].IPAddressTot := I ;

    // беремо список IP адрес для GatewayList
        I := 0 ;
        PIpAddr := @AdapterInfo^.GatewayList ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].GatewayList [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].GatewayList) <= I then
                        SetLength (AdpRows [AdpTot].GatewayList, I -2) ;
        end ;
        AdpRows [AdpTot].GatewayTot := I ;

    // беремо список IP адрес для  GatewayList
        I := 0 ;
        PIpAddr := @AdapterInfo^.DHCPServer ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].DHCPServer [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].DHCPServer) <= I then
                        SetLength (AdpRows [AdpTot].DHCPServer, I -2) ;
        end ;
        AdpRows [AdpTot].DHCPTot := I ;

    // беремо список IP адрес для PrimaryWINSServer
        I := 0 ;
        PIpAddr := @AdapterInfo^.PrimaryWINSServer ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].PrimWINSServer [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].PrimWINSServer) <= I then
                        SetLength (AdpRows [AdpTot].PrimWINSServer, I -2) ;
        end ;
        AdpRows [AdpTot].PrimWINSTot := I ;

    // беремо список IP адрес для SecondaryWINSServer
        I := 0 ;
        PIpAddr := @AdapterInfo^.SecondaryWINSServer ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].SecWINSServer [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].SecWINSServer) <= I then
                        SetLength (AdpRows [AdpTot].SecWINSServer, I -2) ;
        end ;
        AdpRows [AdpTot].SecWINSTot := I ;

        AdpRows [AdpTot].LeaseObtained := AdapterInfo^.LeaseObtained ;
        AdpRows [AdpTot].LeaseExpires := AdapterInfo^.LeaseExpires ;

        inc (AdpTot) ;
        if Length (AdpRows) <= AdpTot then
                    SetLength (AdpRows, AdpTot -2) ;  // більше пам' яті
        AdapterInfo := AdapterInfo^.Next;
      end ;
      SetLength (AdpRows, AdpTot) ;
  end ;
finally
  FreeMem( pBuf );
end ;
```

```
end ;

procedure Get_AdaptersInfo( List: TStrings );
var
  AdpTot: integer;
  AdpRows: TAdaptorRows ;
  Error: DWORD ;
  I: integer ;
  //J: integer ;
  //S: string ;        id.
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  SetLength (AdpRows, 0) ;
  AdpTot := 0 ;
  Error := DAdaptersInfo(AdpTot, AdpRows) ;
  if (Error <> 0) then
      List.Add( SysErrorMessage( GetLastError ) )
  else if AdpTot = 0 then
      List.Add( ' данних немає ’  )
  else
  begin
      for I := 0 to Pred (AdpTot) do
      begin
        with AdpRows [I] do
        begin
            //List.Add(AdapterName + ' |’  + Description ); // jpt : не
використовується
            List.Add( Format(' %8.8x | %6s | %16s | %2d | %16s | %16s | %16s’ ,
                [Index, AdaptTypes[aType], MacAddress, DHCPEnabled,
                GatewayList [0], DHCPServer [0], PrimWINSServer [0]] ) );
            {if IPAddressTot <> 0 then    // jpt : не використовується
            begin
                S := ' '  ;
                for J := 0 to Pred (IPAddressTot) do
                        S := S + IPAddressList [J] + ' /’  + IPMaskList [J] + '
| '  ;
                List.Add(IntToStr (IPAddressTot) + '  IP Addresse(s): '  + S);
            end ;
            List.Add( '   '  ); }
        end ;
      end ;
  end ;
  SetLength (AdpRows, 0) ;
end ;

//-----------------------------------
{ моніторимо час доступу до IP досліджуємої мережевої системи моніторингу на
базі DevOps-архітектури (DATA)}
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint; var RTT: Longint;
  var HopCount: Longint ): integer;
begin
  if not GetRTTAndHopCount( IPAddr, @HopCount, MaxHops, @RTT ) then
  begin
    Result := GetLastError;
    RTT :=-1; // Расположення BAD_HOST_NAME,etc...
    HopCount :=-1;
  end
  else
    Result := NO_ERROR;
end;

//-----------------------------------
{ ARP-таблиця включає відношення між віддаленим IP та віддаленим MAC-адресом.
}
procedure Get_ARPTable( List: TStrings );
var
  IPNetRow      : TMibIPNetRow;
  TableSize     : DWORD;
```

```
  NumEntries    : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  // перший виклик: беремо довжину таблиці
  TableSize := 0;
  ErrorCode := GetIPNetTable( Nil, @TableSize, false );   // дані
  //
  if ErrorCode = ERROR_NO_DATA then
  begin
    List.Add( '  ARP-кеш пустий.'  );
    EXIT;
  end;
  // беремо таблицю
  GetMem( pBuf, TableSize );
  NumEntries := 0 ;
  try
  ErrorCode := GetIpNetTable( PTMIBIPNetTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBIPNetTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        IPNetRow := PTMIBIPNetRow( PBuf )^;
        with IPNetRow do
          List.Add( Format( ' %8x | %12s | %16s | %10s' ,
                            [dwIndex, MacAddr2Str( bPhysAddr, dwPhysAddrLen ),
                            IPAddr2Str( dwAddr ), ARPEntryType[dwType]
                            ]));
        inc( pBuf, SizeOf( IPNetRow ) );
      end;
    end
    else
      List.Add( '  ARP-кеш пустий.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // необхідно відновити показник!
  finally
      dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( IPNetRow ) );
      FreeMem( pBuf );
  end ;
end;


//-----------------------------------
procedure Get_TCPTable( List: TStrings );
var
  TCPRow        : TMIBTCPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  DestIP        : string;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  RecentIPs.Clear;
  // перший виклик: беремо довжину таблиці
  TableSize := 0;
  NumEntries := 0 ;
```

```
  ErrorCode := GetTCPTable(Nil, @TableSize, false );  // дані
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо розмір пам' яті, викликаємо знову
  GetMem( pBuf, TableSize );
  // беремо таблицю
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin

    NumEntries := PTMIBTCPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        TCPRow := PTMIBTCPRow( pBuf )^; // беремо останій запис
        with TCPRow do
        begin
          if dwRemoteAddr = 0 then
            dwRemotePort := 0;
          DestIP := IPAddr2Str( dwRemoteAddr );
          List.Add(
            Format( ' %15s : %-7s | %15s : %-7s | %-16s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) ),
              DestIP,
              Port2Svc( Port2Wrd( dwRemotePort ) ),
              TCPConnState[dwState]
              ] ) );
         //
            if (not ( dwRemoteAddr = 0 ))
            and ( RecentIps.IndexOf(DestIP) =-1 ) then
               RecentIPs.Add( DestIP );
        end;
        inc( pBuf, SizeOf( TMIBTCPRow ) );
      end;
    end;
  end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( TMibTCPRow ) );
  FreeMem( pBuf );
end;

//---------------------------------
procedure Get_TCPStatistics( List: TStrings );
var
  TCPStats     : TMibTCPStats;
  ErrorCode    : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  if NOT LoadD then exit ;
  ErrorCode := GetTCPStatistics( @TCPStats );
  if ErrorCode = NO_ERROR then
    with TCPStats do
    begin
      List.Add( ' Алгоритм повторної передачі : '  + TCPToAlgo[dwRTOAlgorithm]
);
      List.Add( ' Мінімальний час виходу         : '  + IntToStr( dwRTOMin ) + '
ms'  );
      List.Add( ' Максимальний час виходу         : '  + IntToStr( dwRTOMax ) +
'  ms'  );
      List.Add( ' Максимальне число підключень : '  + IntToStr( dwRTOAlgorithm )
);
      List.Add( ' Активні підключення             : '  + IntToStr( dwActiveOpens
) );
```

```
      List.Add( ' пасивні підключення            : '  + IntToStr( dwPassiveOpens
) );
      List.Add( ' Невдала спроба відкриття     : '  + IntToStr( dwAttemptFails )
);
      List.Add( ' Скидання встановленого підключення  : '  + IntToStr(
dwEstabResets ) );
      List.Add( ' Поточне встановлене підключення.: '  + IntToStr( dwCurrEstab )
);
      List.Add( ' Отримані сегменти        : '  + IntToStr( dwInSegs ) );
      List.Add( ' Передані сегменти            : '  + IntToStr( dwOutSegs ) );
      List.Add( ' Перепідключені сегменти   : '  + IntToStr( dwReTransSegs ) );
      List.Add( ' помилка входу          : '  + IntToStr( dwInErrs ) );
      List.Add( ' Перезавантаження вихідних          : '  + IntToStr( dwOutRsts
) );
      List.Add( ' Совокупні зв' язки    : '  + IntToStr( dwNumConns ) );
    end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
end;

function DTCPStatistics (var TCPStats: TMibTCPStats): integer ;
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadD then exit ;
    result := GetTCPStatistics( @TCPStats );
end;

//-----------------------------------
procedure Get_UDPTable( List: TStrings );
var
  UDPRow        : TMIBUDPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;

  // перший виклик: беремо довжину таблиці
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetUDPTable(Nil, @TableSize, false );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // виділяємо пам' ять, викликаємо знову
  GetMem( pBuf, TableSize );

  // беремо таблицю
  ErrorCode := GetUDPTable( PTMIBUDPTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBUDPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        UDPRow := PTMIBUDPRow( pBuf )^; // беремо останій запис
        with UDPRow do
          List.Add( Format( ' %15s : %-6s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) )
              ] ) );
        inc( pBuf, SizeOf( TMIBUDPRow ) );
      end;
    end
```

```
    else
      List.Add( ' немає даних.'  );
  end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( TMibUDPRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------
procedure Get_IPAddrTable( List: TStrings );
var
  IPAddrRow     : TMibIPAddrRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0; ;
  NumEntries := 0 ;
  // перший виклик: беремо довжину таблиці
  ErrorCode := GetIpAddrTable(Nil, @TableSize, true );  // дані
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  GetMem( pBuf, TableSize );
  // беремо таблицю
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPAddrTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPAddrRow := PTMIBIPAddrRow( pBuf )^;
        with IPAddrRow do
          List.Add( Format( ' %8.8x | %15s | %15s | %15s | %8.8d' ,
            [dwIndex,
            IPAddr2Str( dwAddr ),
              IPAddr2Str( dwMask ),
              IPAddr2Str( dwBCastAddr ),
              dwReasmSize
            ] ) );
        inc( pBuf, SizeOf( TMIBIPAddrRow ) );
      end;
    end
    else
      List.Add( ' немає даних.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // відновлюємо показчик!
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( IPAddrRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------
{ отримуємо дані з таблиці маршрутизації досліджуємої мережевої системи
моніторингу на базі DevOps-архітектури (DATA); }
procedure Get_IPForwardTable( List: TStrings );
var
  IPForwRow     : TMibIPForwardRow;
  TableSize     : DWORD;
```

```
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin

  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0;

  // перший виклик: беремо довжину таблиці
  NumEntries := 0 ;
  ErrorCode := GetIpForwardTable(Nil, @TableSize, true);
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо таблицю
  GetMem( pBuf, TableSize );
  ErrorCode := GetIpForwardTable( PTMibIPForwardTable( pBuf ), @TableSize,
true);
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPForwardTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPForwRow := PTMibIPForwardRow( pBuf )^;
        with IPForwRow do
        begin
          if (dwForwardType < 1)
          or (dwForwardType > 4) then
                   dwForwardType := 1 ;   // дані
          List.Add( Format(
            ' %15s | %15s | %15s | %8.8x | %7s |   %5.5d |  %7s |   %2.2d' ,
            [IPAddr2Str( dwForwardDest ),
            IPAddr2Str( dwForwardMask ),
              IPAddr2Str( dwForwardNextHop ),
              dwForwardIFIndex,
              IPForwTypes[dwForwardType],
              dwForwardNextHopAS,
              IPForwProtos[dwForwardProto],
              dwForwardMetric1
              ] ) );
        end ;
        inc( pBuf, SizeOf( TMibIPForwardRow ) );
      end;
    end
    else
      List.Add( ' немає даних.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( TMibIPForwardRow ) );
  FreeMem( pBuf );
end;

//----------------------------------
procedure Get_IPStatistics( List: TStrings );
var
  IPStats       : TMibIPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  if NOT LoadD then exit ;
  ErrorCode := GetIPStatistics( @IPStats );
  if ErrorCode = NO_ERROR then
  begin
```

```
    List.Clear;
    with IPStats do
    begin
      if dwForwarding = 1 then
        List.add( ' Розблокована пересилка      : '  + ' так'  )
      else
        List.add(' Розблокована пересилка      : '  + ' ні'  );
      List.add( ' Любий TTL             : '  + inttostr( dwDefaultTTL ) );
      List.add( ' Датаграмма прийнята     : '  + inttostr( dwInReceives ) );
      List.add( ' Помилка заголовку      (In)  : '  + inttostr( dwInHdrErrors )
);
      List.add( ' Помилка адреси    (In)  : '  + inttostr( dwInAddrErrors ) );
      List.add( ' Датаграма переслана     : '  + inttostr( dwForwDatagrams ) );
// дані
      List.add( ' Невизначений протокол (In)  : '  + inttostr( dwInUnknownProtos
) );
      List.add( ' Датаграмма відмовлена     : '  + inttostr( dwInDiscards ) );
      List.add( ' Датаграма встановлена     : '  + inttostr( dwInDelivers ) );
      List.add( ' Зовнішній запрос            : '  + inttostr( dwOutRequests )
);
      List.add( ' Маршрутизація не виконана      : '  + inttostr(
dwRoutingDiscards ) );
      List.add( ' Немає маршрутів        (Out)  : '  + inttostr( dwOutNoRoutes )
);
      List.add( ' Перебраний час     : '  + inttostr( dwReasmTimeOut ) );
      List.add( ' Запит перебору     : '  + inttostr( dwReasmReqds ) );
      List.add( ' Повний перебор : '  + inttostr( dwReasmOKs ) );
      List.add( ' Помилка перебору     : '  + inttostr( dwReasmFails ) );
      List.add( ' Повна фрагментація: '  + inttostr( dwFragOKs ) );
      List.add( ' Помилка фрагментації   : '  + inttostr( dwFragFails ) );
      List.add( ' Датаграмма фрагментована    : '  + inttostr( dwFRagCreates )
);
      List.add( ' Кількість інтерфейсів    : '  + inttostr( dwNumIf ) );
      List.add( ' Кількість IP-адрес  : '  + inttostr( dwNumAddr ) );
      List.add( ' Маршрут в таблиці маршрутизатора  : '  + inttostr( dwNumRoutes
) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

function DIPStatistics (var IPStats: TMibIPStats): integer ;      // дані
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadD then exit ;
    result := GetIPStatistics( @IPStats );
end ;

//-----------------------------------
procedure Get_UdpStatistics( List: TStrings );
var
  UdpStats      : TMibUDPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  ErrorCode := GetUDPStatistics( @UdpStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with UDPStats do
    begin
      List.add( ' Датаграми (In)     : '  + inttostr( dwInDatagrams ) );
      List.add( '  Датаграми (Out)    : '  + inttostr( dwOutDatagrams ) );
      List.add( ' Немає портів          : '  + inttostr( dwNoPorts ) );
      List.add( ' Помилка    (In)     : '  + inttostr( dwInErrors ) );
      List.add( ' UDP список портів  : '  + inttostr( dwNumAddrs ) );
    end;
  end
```

```
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

//-------------------------------------*//
function DUdpStatistics (UdpStats: TMibUDPStats): integer ;     // дані
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadD then exit ;
    result := GetUDPStatistics (@UdpStats) ;
end ;

//-----------------------------------
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
var
  ErrorCode     : DWORD;
  ICMPStats     : PTMibICMPInfo;
begin
  if ( ICMPIn = nil ) or ( ICMPOut = nil ) then EXIT;
  ICMPIn.Clear;
  ICMPOut.Clear;
  New( ICMPStats );
  ErrorCode := GetICMPStatistics( ICMPStats );
  if ErrorCode = NO_ERROR then
  begin
    with ICMPStats.InStats do
    begin
      ICMPIn.Add( ' Прийнято повідомлень    : '  + IntToStr( dwMsgs ) );
      ICMPIn.Add( ' Помилка                 : '  + IntToStr( dwErrors ) );
  ICMPIn.Add( ' Розташування недосягенно    : '  + IntToStr( dwDestUnreachs ) );
      ICMPIn.Add( ' Час перевищений         : '  + IntToStr( dwTimeEcxcds ) );
      ICMPIn.Add( ' Проблеми з параметрами  : '  + IntToStr( dwParmProbs )
);
      ICMPIn.Add( ' Джерело відключено      : '  + IntToStr( dwSrcQuenchs ) );
      ICMPIn.Add( ' Переназначено           : '  + IntToStr( dwRedirects ) );
      ICMPIn.Add( ' Ехо запит       : '  + IntToStr( dwEchos ) );
      ICMPIn.Add( ' Ехо відповідь           : '  + IntToStr( dwEchoReps ) );
      ICMPIn.Add( ' Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
    ICMPIn.Add( ' Відповідь мітки часу    : '  + IntToStr( dwTimeStampReps ) );
      ICMPIn.Add( ' Запит маски адрес : '  + IntToStr( dwAddrMasks ) );
      ICMPIn.Add( ' Відповідь маски адрес   : '  + IntToStr( dwAddrReps ) );
    end;
     //
    with ICMPStats.OutStats do
    begin
      ICMPOut.Add( ' Повідомлення вдправлено        : '  + IntToStr( dwMsgs ) );
      ICMPOut.Add( 'Помилка               : '  + IntToStr( dwErrors ) );
      ICMPOut.Add( 'Розташування недосягенно    : '  + IntToStr( dwDestUnreachs
) );
      ICMPOut.Add( 'Час перевищений        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPOut.Add( 'Проблеми з параметрами      : '  + IntToStr( dwParmProbs )
);
      ICMPOut.Add( 'Джерело відключено        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPOut.Add( 'Переназначено            : '  + IntToStr( dwRedirects ) );
      ICMPOut.Add( 'Ехо запит        : '  + IntToStr( dwEchos ) );
      ICMPOut.Add( 'Ехо відповідь          : '  + IntToStr( dwEchoReps ) );
      ICMPOut.Add( 'Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
      ICMPOut.Add( 'Відповідь мітки часу    : '  + IntToStr( dwTimeStampReps )
);
      ICMPOut.Add( 'Запит маски адрес: '  + IntToStr( dwAddrMasks ) );
      ICMPOut.Add( 'Відповідь маски адрес   : '  + IntToStr( dwAddrReps ) );
    end;
  end
  else
    IcmpIn.Add( SysErrorMessage( ErrorCode ) );
  Dispose( ICMPStats );
end;

//-----------------------------------
```

```
procedure Get_RecentDestIPs( List: TStrings );
begin
  if Assigned( List ) then
    List.Assign( RecentIPs )
end;

initialization

  RecentIPs := TStringList.Create;

finalization

  RecentIPs.Free;

end.
```

Файл TCP_IP.pas- обробка з'єднань TCP/IP досліджуємої мережевої системи моніторингу на базі DevOps-архітектури

```
unit TCP_IP;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, DataIP, DApi, Buttons;

type
  TForm2 = class(TForm)
    StaticText2: TStaticText;
    StaticText3: TStaticText;
    TCPMemo: TMemo;
    UDPMemo: TMemo;
    Timer1: TTimer;
    cbTimer: TCheckBox;
    btRTTI: TSpeedButton;
    SpeedButton1: TSpeedButton;
    edtRTTI: TEdit;
    procedure Timer1Timer(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
    procedure btRTTIClick(Sender: TObject);
    procedure cbRecentIPsClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
      procedure DOIpStuff;
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}

procedure TForm2.Timer1Timer(Sender: TObject);
begin
  if cbTimer.State = cbCHECKED then
 begin
   Timer1.Enabled := false;
   DoIPStuff;
   Timer1.Enabled := true;
 end;
end;

procedure TForm2.DOIpStuff;
begin

  Get_TCPTable( TCPMemo.Lines );
  Get_UDPTable( UDPMemo.Lines );

end;

procedure TForm2.SpeedButton1Click(Sender: TObject);
begin
  Speedbutton1.Enabled := false;
  DoIPStuff;
  Speedbutton1.Enabled := true;
end;

procedure TForm2.btRTTIClick(Sender: TObject);
var
  IPadr         : dword;
```

```
  Rtt, HopCount : longint;
  Res           : integer;
begin
  btRTTI.Enabled := false;
  Screen.Cursor := crHOURGLASS;
  IPadr := Str2IPAddr( edtRTTI.Text );
  Res := Get_RTTAndHopCount( IPadr, 128, RTT, HopCount );
  if Res = NO_ERROR then
    ShowMessage( ' Час запиту '
      + inttostr( rtt ) + ' ms, '
      + inttostr( HopCount )
      + ' hops to : ' + edtRTTI.Text
      )
  else
    ShowMessage( ' Відбулася помилка:' + #13
                 + ICMPErr2Str( Res ) ) ;
  btRTTI.Enabled := true;
  Screen.Cursor := crDEFAULT;

end;

procedure TForm2.cbRecentIPsClick(Sender: TObject);
begin
//edtRTTI.Text := cbRecentIPs.Items[cbRecentIPs.ItemIndex];
end;

procedure TForm2.FormCreate(Sender: TObject);
begin

  if LoadD then
  begin
      DOIpStuff;
      Timer1.Enabled := true;
  end
  else
  ShowMessage( ' Інтернет помічник DLL не є доступним, або не підтримується' ) ;
end;

end.
```