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

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи моніторингу терміналів банківських установ”

Виконав здобувач вищої освіти
IV курсу, групи КІ-20
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Дворцов Н.Р.
«____» _____ 2024 р.

Керівник проекту
доктор технічних наук, професор
_____ Коваленко О.В.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2024 року

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

Дворцову Нікіті Романовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи моніторингу терміналів банківських установ*

2. Керівник роботи *Коваленко Олександр Володимирович, докт. техн. наук, професор*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 131-02 від 01.04.2024 року

3. Строк подання студентом роботи до захисту *23.05.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи моніторингу терміналів банківських установ*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

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

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

7. Дата видачі завдання « 17 » січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	23.05.2024 р.	

Дата видачі завдання
« 17 » січня 2024 р.

Підпис керівника
_____ Коваленко О.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Дворцов Н.Р.
(прізвище та ініціали)

АНОТАЦІЯ

Дворцов Н.Р. Програмне забезпечення системи моніторингу терміналів банківських установ. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи моніторингу терміналів банківських установ.

Метою розробки є програмне забезпечення системи моніторингу терміналів банківських установ.

Результат роботи – програмна реалізація системи моніторингу терміналів банківських установ.

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

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

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

Програму розроблено в середовищі Delphi 10.4.1.

Ключові слова: комп'ютерна інженерія, моніторинг, термінал банківської установи

ABSTRACT

Dvortsov N.R. Software for the monitoring system of banking institutions' terminals. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this final qualification work for the first (bachelor) level of higher education, software was developed, which is intended for the monitoring system of banking institutions' terminals.

The purpose of the development is software for the monitoring system of banking institutions' terminals.

The result of the work is the software implementation of the monitoring system of banking institutions' terminals.

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

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

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

The program was developed in the Delphi 10.4.1 environment.

Keywords: computer engineering, monitoring, banking institution terminal

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ОС	–	Операційна система;
ПК	–	Персональний комп’ютер;
ГМ	–	Глобальна мережа Інтернет;
ЛМ	–	Локальна мережа;
КСП	–	Клієнт-серверне програмне забезпечення;
ПЗ	–	Програмне забезпечення;
БД	–	База даних;
ПОРТ	–	Порт протоколу TCP/IP;
Адреса	–	Адреса протоколу TCP/IP;
SMS	–	Short message service – служба коротких повідомлень;
ЦО	–	цивільна оборона;
КС	–	комп’ютерна система;
ПЕОМ	–	персональна електронно-обчислювальна машина;
HTML	–	HyperText Murkup Language – мова розмітки гіпертекстових документів.

ВСТУП

Актуальність теми. Грамотна оптимізація поповнення банкоматів готівкою здатна істотно знизити експлуатаційні витрати, які сильно впливають на загальну вартість володіння парком банкоматів.

Управління та планування готівки в банкоматах базується на моніторингу стану готівки в апаратах самообслуговування в реальному часі. Цей інструмент є основним і найбільш затребуваним функціоналом систем управління ланцюгом поставок готівки. Це дозволяє отримувати найактуальнішу інформацію про наявність готівки у віддалених терміналах і запобігати можливим випадкам простою банкоматів через відсутність або переповнення готівки.

Програмне забезпечення для управління готівкою, яке розробляється у даній роботі, також має можливість завчасно реагувати на потенційні інциденти, пов'язані з готівкою, і проблеми з грошовими потоками. Різні функції нашої системи управління готівкою в банкоматах є важливою частиною ефективного управління грошовими потоками фінансової установи.

Пристрій банкомату має бути завжди готовим до використання різними типами користувачів (окремими особами, власниками бізнесу , малими підприємствами тощо). Таким чином, впровадження рішення з управління готівкою допоможе підтримувати рівень готівки на достатньому рівні, щоб задовольнити користувачів:

– Зворотний відлік вичерпання готівки (для визначення пріоритетності касових пунктів).

– Миттєві сповіщення при досягненні ліміту рівня готівки.

– Негайне повідомлення про технічні інциденти.

Моніторинг готівки можна використовувати не лише для банкоматів, але й для платіжних кіосків, депозитних автоматів, а також банківських відділень і

сховищ, щоб мати повну картину управління ланцюгом постачання готівки у всій мережі.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи моніторингу терміналів банківських установ.

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

– Огляд існуючих систем моніторингу терміналів банківських установ.

– Дослідження системи моніторингу терміналів банківських установ.

– Програмна реалізація системи моніторингу терміналів банківських установ.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі моніторингу терміналів банківських установ.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи моніторингу терміналів банківських установ, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Програмне забезпечення, яке розробляється у даному бакалаврському проекті, призначено для реалізації системи моніторингу терміналів банківських установ.

В останні роки темпи розвитку технологій, які використовуються в банківській галузі, досягли свого піку. Банки, які традиційно займають помірковану або навіть консервативну позицію, дедалі більше дотримуються принципів цифрової трансформації, оскільки вони перебувають у висококонкурентному середовищі, коли пропонують сучасні банківські продукти. Ефективність операційної діяльності фінансової установи напряму залежить від того, наскільки сучасні основні інструменти та IT-рішення, які вона використовує.

Програмне забезпечення, яке розробляється у даній роботі може використовуватися банками, CIT-сервісними компаніями та іншими установами для оптимізації готівкового обігу, планування розподілу готівки в мережі, оптимізації процесів поповнення касових пунктів та автоматизація балансу каси в кінці робочого дня.

Досвід розробки рішення та побажання фінансових установ, які використовують систему, дозволили фахівцям максимально ефективно змінити інтерфейс користувача та впровадити новітні стандарти UI/UX.

Які можливості пропонує своїм користувачам система, яка розробляється у даній роботі:

1. Більш ергономічний і гнучкий інтерфейс користувача. Новий інтерфейс користувача, вбудований у React, став ще простішим та інтуїтивніше зрозумілим. Впровадження нових стандартів дозволило підвищити рівень ергономічності

системи, щоб спеціалісти касових відділень могли виконувати свою роботу максимально ефективно, навіть при зростаючих навантаженнях. Також тепер інтерфейс рішення можна легко адаптувати під конкретні вимоги та бізнес-процеси компанії для організації оптимальної роботи системного оператора.

2. Комплексні інструменти оперативної аналітики. Розширено можливості функціонального модуля. Цей модуль дозволяє користувачам отримувати різноманітні оперативні звіти та графіки, які відображають ефективність розподілу готівки в інфраструктурі установи. Додано можливість впровадження нової графіки, необхідної конкретному клієнту.

3. Більш зручні процеси організації доставки цінностей. Удосконалено частину системи, яка відповідає за управління доставкою цінностей. Тепер при веденні замовлення на доставку готівки або цінностей оператор також може керувати такими об'єктами, як сумки, комплекти касет тощо. Це спрощує процес контролю та обліку матеріально-технічних цінностей, які використовуються під час інкасації КІН.

1.2 Область застосування

Областю застосування програмного забезпечення, яке розробляється у даному бакалаврському проекті, є банківські послуги для фізичних осіб.

У процесі трансформації бізнес-процесів, пов’язаних із забезпеченням готівкою, банки часто вдаються до використання сторонніх СІТ або послуг безпеки.

Аутсорсинг інкасації дозволяє банкам вирішити проблему надмірних витрат на транспортування готівки та цінностей, а не відволікати оперативні ресурси на різноманітні рутинні питання, такі як пошук та найм спеціалістів, їх навчання та сертифікація, оснащення та закупівля автотранспорту та багато іншого. При розробці системи стояло завдання забезпечення безперебійної взаємодії між працівниками банку, які безпосередньо розміщують замовлення на

інкасацію, та представниками сторонніх організацій, які здійснюють інкасацію або надають супровід та охорона. Реалізовано проект інтеграції зі спеціальним банківським сервісом, який автоматизує процес замовлення супроводу для перевезення цінностей національною поліцією.

Етапи бізнес-процесу в рамках впровадженого рішення:

1. Моніторинг об'єктів, що потребують поповнення або вивантаження готівки.

2. Автоматичний розрахунок поповнення з оптимальним розподілом валюти та номіналу.

3. Формування заявки на перевезення цінностей з автоматичним запитом в поліцію.

4. Запит та отримання даних про агентів по збору, які були призначені національною поліцією для кожного замовлення.

5. Здійснення інкасації згідно сформованого замовлення.

Таким чином, вдалося прискорити процес узгодження замовлень на супровід перевезення цінностей та усунути «сірі зони» (частини бізнес-процесів зі спірною відповідальністю) між співробітниками різних організацій. Це рішення можна використовувати для синхронізації роботи з будь-якими сторонніми постачальниками інкасації або охоронних послуг, послуги яких потрібні при виконанні замовлень на інкасацію та поповнення. Платформа дозволяє комплексно підходити до оптимізації грошових потоків на стороні банку, забезпечуючи зменшення кількості коштів у користуванні та зменшення кількості необхідних інкасацій. Водночас система дозволяє контролювати всі супутні процеси та позбавляє від необхідності паперової роботи.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи моніторингу терміналів банківських установ, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

ProClassic

ProClasssic – це сукупність клієнт-орієнтованих архітектур програмного забезпечення для система самообслуговування, які базуються на стандартних операційних системах та відкритих стандартах.

ProDevice/SSP

ProDevice/SSP забезпечує доступ до периферії самообслуговування і спеціальної банківської периферії у відповідності до стандарту WOSA/XFS (Windows Open Service Architecture/ Extensions for Financial Services).

ProTopas

ProTopas – (The OPen Application Software for Self Service) – це гнучка платформа розробки додатків для пристроїв самообслуговування в середовищі відкритих систем.

ProTopas/Web extention

ProTopas/Web extention – продукт, що базується на технології Windows NT, WOSA/XFS і ProTopas frameworks. Це рішення полягає у використанні Інтернет-технологій для розробки інтерфейсу самообслуговування і керуванням потоком задач. Сфера використання: підключення до серверних та хостових систем, інтеграція в середовище домашнього банківського обслуговування, підтримка існуючих мереж.

ProSOP

ProSOP – програма для локального обслуговування і правління. Не залежить від додатків системи самообслуговування, має WOSA/XFS-інтерфейс.

ProView

ProView – програма для адміністрування і моніторингу мереж самообслуговування, незалежно від: постачальника hardware, програмного забезпечення, протокола, host. Є можливість створення груп користувачів з правами: системний адміністратор в центрі даних, банківські працівники, інкасатори, служба безпеки та IT-сервіс.

Програма допомагає здійснювати превентивне обслуговування систем самообслуговування через запити, наприклад:

– кількість банкнот в касеті;

– запас паперу;

– стан стрічки принтеру.

ProCash NDC/DDC:

– Підтримка протоколів Diebold 911, 912, NDC, NDC+.

– ProTopas додаток.

– Додаткові розділи ProTopas registry LINXCI, LINXPAR.

– Розширення додатків:

– розширені стани;

– розширені можливості графічного клієнтського інтерфейсу (BMP, PCX, AVI, MPEG);

– підтримка мультимовності чеків та журналів.

– конструювання Host-повідомлень через registry;

– механізм «мапування» помилок.

Observium

Observium – інструмент моніторингу мережі, створений на базі PHP/MySQL/SNMP. Він призначений для Linux, UNIX, Cisco, Juniper, Brocade, Foundry, HP, і ін. За допомогою цього інструмента ви можете одержувати досить наочні графіки, а також користуватися його досить зручним інтерфейсом. Він може контролювати велику кількість процесів і систем. Єдиний недолік – відсутність автоматичної сигналізації про проблеми. Але це можна компенсувати, використовуючи Observium поряд з такою системою, як Nagios.

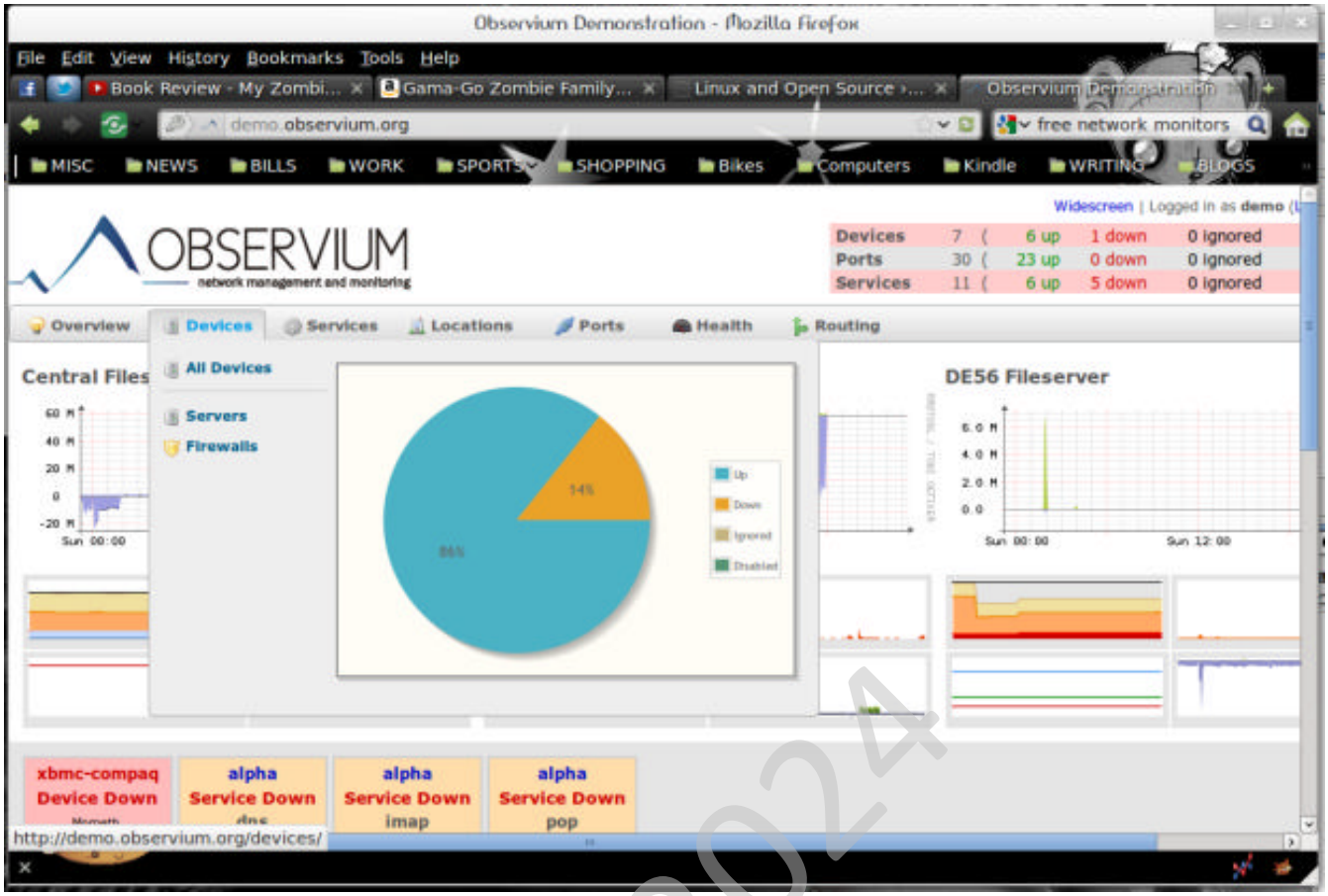

Рисунок 2.1 – Інтерфейс користувача Observium

Ganglia

Ganglia – масштабована розподілена система моніторингу, що фокусується на кластерах і грідах. Вона надає швидкий і огляд, що читається легко, всієї кластерної системи. Цей інструмент уже портован на багато платформ і використовується в тисячах кластерів по усьому світі. Кожний, хто впроваджує в себе серверні кластери, повинен використовувати Ganglia для моніторингу цієї системи. Ganglia може масштабуватися для використання в системах до 2000 вузлів.

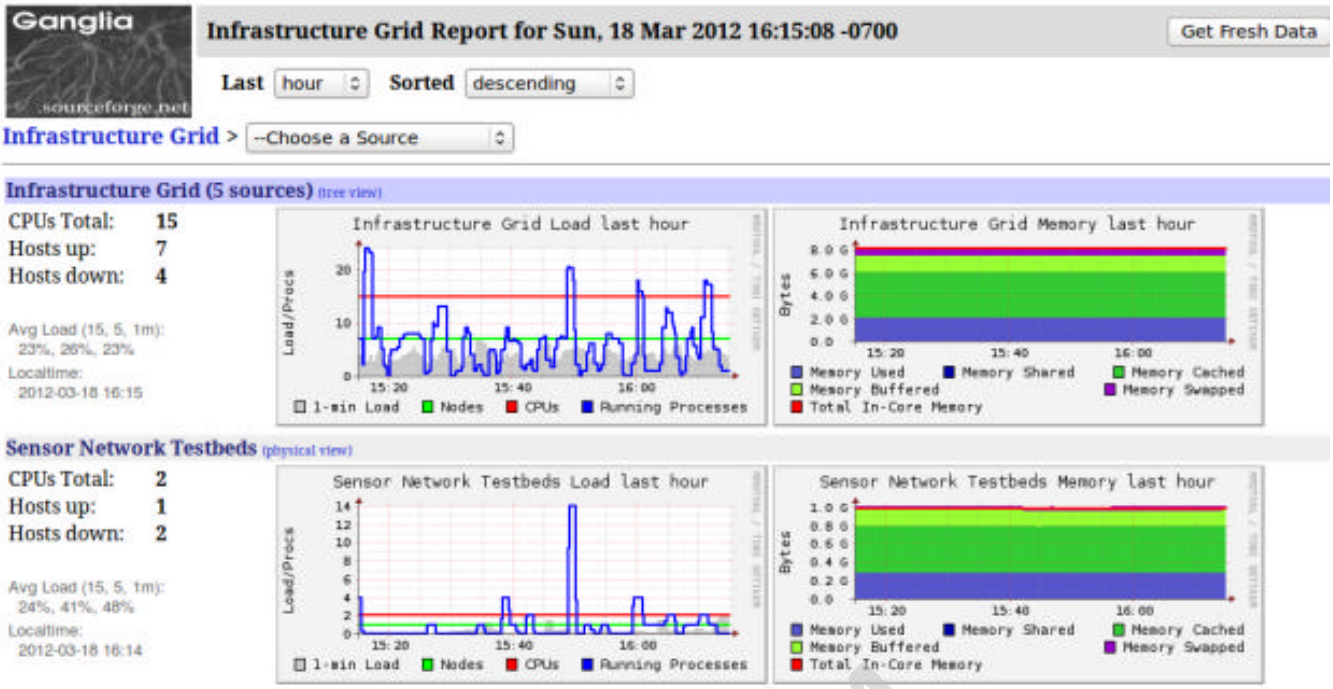

Рисунок 2.2 – Інтерфейс користувача Ganglia

Spiceworks

Spiceworks – стає одним з галузевих стандартів для безкоштовних інструментів моніторингу мережі/систем. Хоча ви його одержуєте в комплекті з деякою кількістю реклами, його функціонал і веб-інтерфейс просто чудові. Spiceworks контролює ваші системи, інформує вас, якщо щось несправне й пропонує до використання відмінні топографічні інструменти. Він також надає вам можливість спілкуватися з колегами через співтовариство Spiceworks, що убудована прямо в систему.

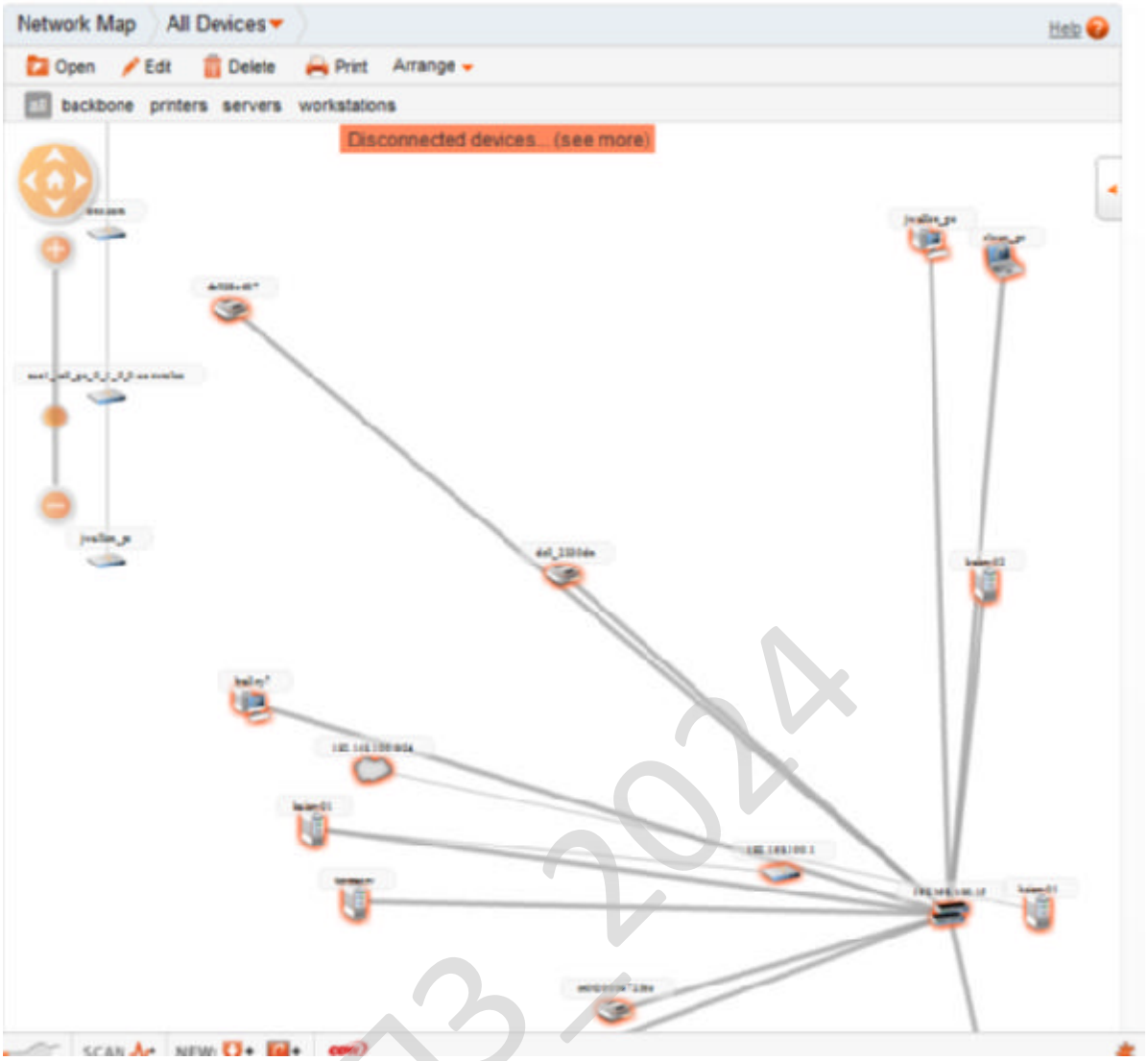

Рисунок 2.3 – Інтерфейс користувача Spiceworks

Nagios

Nagios – багатьма вважається "королем" систем моніторингу мережі класу open source. Хоча це не найпростіша система для установки й конфігурування (вам треба буде вручну редагувати конфігураційні файли), Nagios є потужним інструментом. Хоча не всім подобається ручна конфігурація, але саме ця можливість робить Nagios одним з найбільш гнучких моніторів мережі. І, нарешті, ряд функціональних можливостей Nagios є просто відмінними. Ви можете, наприклад, установити можливість повідомлення через email, SMS, або принтер.

Рисунок 2.4 – Інтерфейс користувача Nagios

Zabbix

Zabbix – потужний інструмент моніторингу, що пропонує можливість конфігурації виду, наближення й маппінгу на своїй веб-консолі. Zabbix пропонує моніторинг без використання програм-агентів, збирає майже будь-який вид даних, які ви хочете контролювати, забезпечує звітність по SLA і доступності, і може контролювати 10 000 пристроїв. Ви можете навіть одержати підтримку (правда, уже за гроші) для цього продукту open source. Однієї з унікальних характеристик Zabbix є можливість установки звукових сигналів. Якщо щось відключається, то ви можете почути звуковий файл (наприклад, звук клаксона).

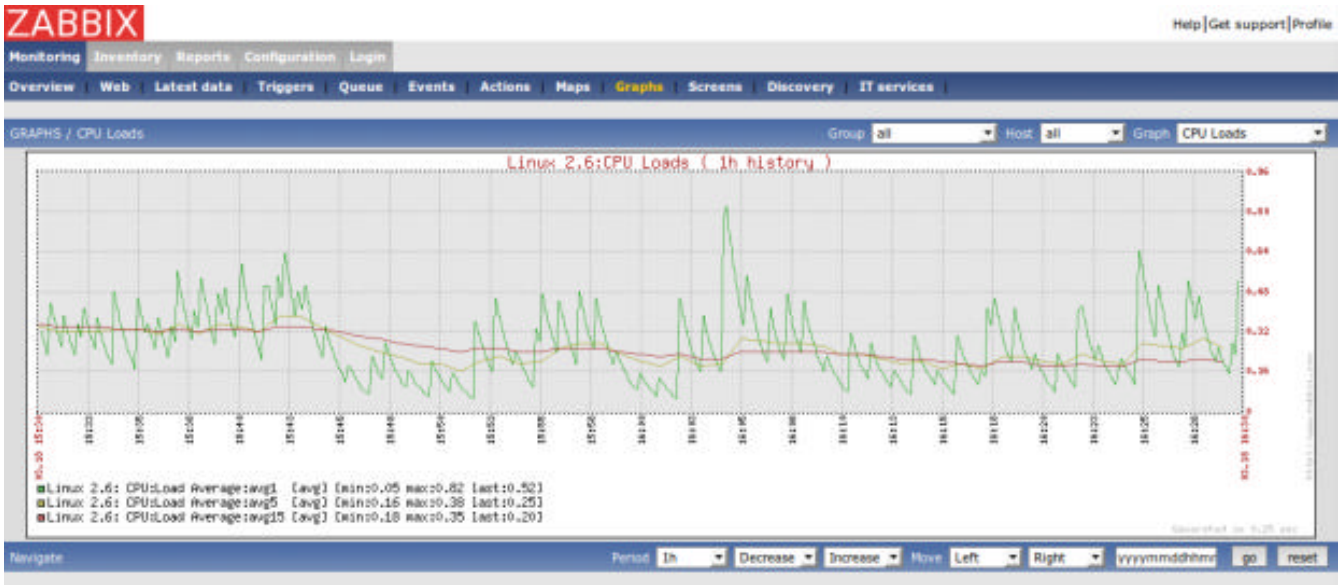

Рисунок 2.5 – Інтерфейс користувача Zabbix

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 випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи моніторингу терміналів банківських установ.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

У контексті розвитку мережі автоматизованих пунктів обслуговування перед банком стоять дві глобальні стратегічні цілі. По-перше, це розвиток функціональних можливостей пристроїв, які дозволять надавати через банкомати якомога більше послуг з портфеля фінансової організації. Другий – підвищення операційної ефективності, тобто оптимізація витрат на володіння інфраструктурою та зниження необхідних витрат на неї за рахунок більш грамотного управління пристроями. Це може включати витрати на обслуговування пристроїв, витрати на інкасацію готівки, вартість обробки транзакцій, вартість збору готівки тощо.

Зазвичай для зменшення операційних витрат, пов'язаних з автоматизованими пунктами обслуговування, банки вдаються до аутсорсингу послуг непрофільних видів діяльності: обслуговування банкоматів, інкасації готівки і навіть аутсорсинг процесингу.

Збір готівки для парку пристроїв самообслуговування є однією з найбільш витратних статей ефективності роботи банкоматів. Витрати банків лише на залучення готівки досить високі. Якщо до цього додати витрати на страхування готівки та витрати на її обробку та доставку, то цифра ще більша. Чим більше готівки потрібно банку для банкоматів, тобто чим більше готівки в обігу, тим вищі витрати фінансової установи на утримання термінальної мережі. Завдяки сучасним інструментам моніторингу та планування грошових потоків власники термінальних мереж можуть, не впливаючи на роботу всієї інфраструктури, зменшити кількість готівки, необхідної для банкоматів. А це, в свою чергу, також може значно знизити витрати на його вирощування та переробку.

Ще одна стаття витрат, яку можна скоротити за рахунок використання

спеціалізованих рішень, це кількість інкасацій. Оптимальне планування готівки для банкоматів може значно скоротити кількість необхідних відвідувань CIT-сервісу, одночасно забезпечуючи автоматизовані пункти обслуговування достатньою кількістю готівки, необхідної для їх безперервної роботи.

Система допомагає візуалізувати грошові потоки, оптимізувати робочі процеси грошових потоків і отримати технічну платформу для розробки касових операцій. Інтеграція надала операторам нові можливості. Тепер, увійшовши в систему, оператор може підключитися до потрібної сортувальної машини, вибрати автоматичне отримання результатів перерахунку безпосередньо з касового сортувальника та використовувати отримані дані для різних цілей.

Найближчим часом буде реалізована можливість отримання звіту про роботу кожної одиниці лічильно-сортувального обладнання: кількість обробленої готівки в розрізі валют і номіналів, а також інші показники.

Система використовується для автоматизації процесів, пов’язаних з готівковим обігом, управління інкасовими ордерами, планування та моніторингу готівки. Програмне забезпечення забезпечує можливість всебічного обліку операцій надходження та відтоку готівки в розрахунково-касових центрах і з’єднує грошові потоки від центрального банку до центрів обслуговування клієнтів і навпаки. Завдяки інтеграції програмного забезпечення із сортувальним обладнанням оператори можуть отримувати результати підрахунку банкнот у режимі реального часу та використовувати ці дані для автоматизації балансування.

Проблеми з обліком збору готівки вирішуються за допомогою нових функцій.

Ця функція дозволяє:

1. Впровадити автоматизовану звірку готівки з можливістю відстежувати запаси готівки в касетах і сумках банкоматів.

2. Увімкніть можливість розміщувати замовлення на поповнення готівки в банкоматах за допомогою попередньо підготовлених мішків і касет.

3. Безпроблемно створюйте супровідні документи та звіти.

4. Зменшити потенційні ризики надлишку та нестачі готівки в місцях зберігання та видачі готівки.

Серед останніх доповнень сімейства програмного забезпечення – «сумки для банкоматів», «касети для банкоматів» і можливість моніторингу авансів готівки, наданих касирам.

Функція **«Касети банкомату»** фіксує завантажені касети з готівкою, які використовуються службами доставки готівки.

Відображена інформація включає:

– Завантажена сума з урахуванням номіналу.

– Кількість заповнених касет.

– Дата заповнення та номер касети.

– Номер пломби.

– Ім'я касира.

Функція **«Сумки для банкоматів»** реєструє мішки з готівкою, сформовані з комплектів касет, із записом детальних облікових даних для кожної касети в сумці. Підготовлені сумки призначаються для завантаження в банкомати та відстежуються протягом усього циклу передачі готівки.

Ця функція успішно використовується відомим азербайджанським банком з найбільшою мережею банкоматів країни вже більше року. Завдяки впровадженню оновлень у фінансовій установі ефективно автоматизовано завдання з управління касовою логістикою під час завантаження готівки в мішки та касети для транзитних операцій.

Зменшилась кількість помилок під час підрахунку готівки на касетному балансі сховища

Комерційні банки, які займаються підготовкою готівки до завантаження в банкомат, зазвичай заздалегідь заповнюють касети та сумки з готівкою, зберігають їх у сховищі та наступного дня передають персоналу каси.

Типова ситуація в банку №1

Уявіть гамірний вечір у банку, де касири готують готівку для завантаження в банкомат. Серед них і Марія, старший касир із 15-річним стажем. Вона старанно заповнює кожну касету, двічі перевіряє кількість і запечатує мішки, перш ніж помістити їх у сховище.

Однак вечірня поспіх іноді призводить до ненавмисних помилок. На жаль, одного напруженого дня Марія помилково написала неправильну адресу сховища в касовому ордері, вказавши не центральне сховище, а місцеве, куди мали бути розміщені касети. У результаті наступного дня ці сумки з касетами не дійшли до банкоматів, оскільки касова служба забрала сумки з центрального сховища за графіком.

Водночас банку довелося перенаправляти сумки з готівкою з одного сховища в інше та провести внутрішнє розслідування. Ця плутанина вплинула на логістику та фінансову звітність, спричинивши затримки для персоналу та ускладнивши відстеження руху готівки.

Впровадження нового функціоналу «Сумки для банкоматів» і «Касети для банкоматів» дозволяє автоматизувати облік завантаженої готівки в електронному форматі та забезпечує повний контроль за балансом кожної одиниці зберігання. Відповідальному касиру потрібно лише ввести необхідні дані в систему, і касети з готівкою будуть автоматично обліковані у відповідних сумках і сховищах.

Для кожного сховища система зберігає та відображає таку інформацію:

– Залишок готівки та матеріальних цінностей на початок дня.

– Загальний залишок з урахуванням готівки, що зберігається в касетах і сумках.

– Зміни балансу за вхідними та вихідними операціями.

– Залишок готівки та цінностей на кінець дня.

Процес завантаження готівки в грошові касети та сумки став доступним для всіх учасників.

Автоматизована система обліку готівкових касет і сумок, інтегрована в баланси сховищ «Касети банкоматів» і «Сумки банкоматів», забезпечує централізований і одночасний доступ до даних. Рівні доступу залежать від повноважень та відповідальності співробітників.

Типова ситуація в банку №2

Бухгалтери банку помітили невідповідність у наявності готівки в сховищах. Під час перевірки вони виявили помилку в ручному записі мішків з готівкою під час процесу передачі готівки.

Щоб виявити джерело помилки та усунути її, спеціалісти провели ретельний огляд облікових справ, щоб визначити, чи були достовірні записи щодо завантаження касет і мішків у кожному сховищі. Вони зіставили цю інформацію із запланованим залишком готівки для кожного мішка та сховища та перерахували кількість мішків. Згодом вони звернулися до старших касирів і транзитних команд, щоб з'ясувати, які касети були завантажені в банкомати та з якого сховища.

На підставі зібраної інформації бухгалтери встановили невірну адресу сховища, зазначену в ордері про передачу готівки. Вони внесли необхідні правки в документи та звернулися до каси, щоб повернути касети з одного банкомату та завантажити їх в інший. Весь процес зайняв загалом 18 робочих годин із залученням 10 банківських працівників.

Якби в банку, де працює Марія (з прикладу вище), був встановлений функціонал «Касети для банкоматів» і «Сумки для банкоматів», фахівець би сам помітив її помилку і вчасно її виправив.

Більше немає дефіциту або надлишку готівки: кошти ефективно розподіляються по місцях для зберігання та видачі

Коли бухгалтерський облік ведеться через стандартну програму електронних таблиць, виникає проблема централізованого доступу до даних, що перешкоджає синхронізації та обміну інформацією між співробітниками, особливо коли вони працюють віддалено або в різних відділах.

Типова ситуація в банку №3

Повертаючись зі зміни, електрик Антон хотів пригостити свою семирічну доньку морозивом, яке продавали на кіоску по дорозі. Частування можна було отримати лише готівкою, а у Антона при собі була лише його картка. Він сподівався зняти гроші в найближчому до дому банкоматі, але його не було. Така ситуація сталася вже втретє.

Розчарований Антон вирішив скористатися банкоматом іншого банку, зручно розташованим неподалік від його офісу. Цього разу йому пощастило, і він успішно зняв необхідну суму для покупки частування. В результаті цього інциденту Антон вирішив більше не покладатися на послуги попереднього банку і замість цього звернувся до іншого, чиї послуги виявилися більш надійними.

Функції «Касети банкоматів» і «Сумки банкоматів» дозволяють співробітникам банку в реальному часі переглядати точні дані про запаси готівки в касетах і сумках, що зберігаються в сховищах.

Замість громіздких файлів є простий і зручний інструмент.

Управління балансом грошових касет і сумок в електронній електронній таблиці, такій як Excel, представляє додаткові складності разом із помилками введення даних і обмеженим доступом до інформації. Фінансові спеціалісти повинні вручну оновлювати дані для кожної операції інкасації готівки, що призводить до тривалих процесів і затримок у контролі балансу.

Крім того, електронні таблиці вразливі з точки зору безпеки даних. Несанкціонований доступ, видалення файлів або пошкодження можуть призвести до втрати або витоку конфіденційної інформації, створюючи додаткові проблеми для відновлення даних.

Стандартні електронні таблиці надають основні інструменти для аналітики. Створення звіту вручну забирає час співробітників і покладається на компетентність і логічні висновки спеціалістів, що впливає на точність результатів. Зі збільшенням обсягу даних керування інформацією в таких файлах стає ще складнішим.

Не кожен фахівець вміє виконувати ті самі завдання Excel. Потрібне постійне навчання співробітників. Іноді банки починають покладатися на компетенцію персоналу, який володіє знаннями процесу та досвідом у форматуванні звітів у банку.

Типова ситуація в банку №4

В універсальному комерційному банку старанно працював досвідчений касир на ім'я Нікіта. Він обслуговував клієнтів протягом 8 років і керував записами грошових потоків. Завдяки своїм знанням Excel, Нікіта якісно справлявся із завданнями, пов’язаними з моніторингом балансу сумок та касет банкоматів.

Під час відпустки Нікіти його обов'язки тимчасово взяла на себе молода та енергійна колега Вікторія. Її завданням було актуалізувати дані в таблиці із залишками готівкових касет і сумок для забезпечення своєчасного поповнення банкоматів готівкою. У суєті та захопленні новим досвідом Вікторія випадково поміняла колонки на сумки та касети.

Наступного ранку касири почали заправляти касети за даними таблиці Excel, абсолютно не підозрюючи про помилку. Клієнти приходили знімати гроші зі своїх карток після отримання зарплати, але виявляли, що банкомати не можуть видати запитані суми. Деякі клієнти обурилися і вимагали пояснень.

Коли Нікіта повернувся з відпустки, він виявив, що в таблицю були внесені неправильні дані. Йому довелося прискіпливо перевірити всі 20 пакетів з касетами, щоб внести достовірну інформацію. Цей процес зайняв понад вісім годин, і Олексію довелося залишатися на роботі до пізнього вечора.

Функціональність можна налаштовувати відповідно до бізнес-процесів банку (ролі та рівні доступу, назви сховищ, позначення груп транспортування готівки тощо). Завдяки інтуїтивно зрозумілому інтерфейсу та зрозумілій технічній документації будь-який співробітник може легко керувати інструментом.

Супровідні документи можна створити лише кількома клацаннями миші

та роздрукувати, усуваючи потребу в переписуванні у разі плям на папері чи друкарських помилок у імені члена команди з доставки готівки. Електронні записи зберігаються для історичних цілей, що полегшує підготовку аналітичних звітів.

Впровадження нового функціоналу програмного забезпечення – «Касети для банкоматів» і «Сумки для банкоматів» і опцій контролю видачі готівки в касу – значно покращує процес відстеження касет і сумок готівки в сховищах банку.

Ця функція дозволяє:

1. Впровадити автоматизовану звірку готівки з можливістю відстежувати запаси готівки в касетах і сумках банкоматів.

2. Увімкніть можливість розміщувати замовлення на поповнення готівки в банкоматах за допомогою попередньо підготовлених мішків і касет.

3. Безпроблемно створюйте супровідні документи та звіти.

4. Зменшити потенційні ризики надлишку та нестачі готівки в місцях зберігання та видачі готівки.

Модуль касової звітності – це модуль, який відповідає за створення, відображення та експорт звітів із кожної точки обслуговування, а також за досягнення ключових показників ефективності та інших аспектів інтегрованої грошової логістики.

Необхідні звіти, пов'язані з готівкою та процесами

– Знята готівка за валютами та номіналами.
– Сума знятої готівки.
– Залишки готівки на апаратах самообслуговування.
– Поточні залишки апаратів самообслуговування.

Наскрізна аналітика розподілу готівки. При впровадженні інших касових модулів системи інструмент формування звітів використовується для отримання аналітики щодо доручень СІТ та операцій Касового центру.

3.2 Розробка структурної схеми

Розглянемо розроблену у бакалаврському проекті структурну схему роботи системи яка зображена на рисунку 3.1. Програма призначена для системи моніторингу терміналів банківських установ.

Призначення розробленого програмного забезпечення – реалізація системи моніторингу терміналів банківських установ. Необхідне для виявлення дій користувача за кожним терміналом банківської установи. Має наступні можливості:

– встановлювати час проведення користувачів за терміналом банківської установи;

– які використовуються ПЗ;

– скільки часу витрачено на роботі з браузером та іншими інстальованими програмами;

– перегляд в любий момент часу терміналу банківської установи;

– у прихованому режимі відстежувати роботу заборонених програм (ICQ, Skype) та сайтів (*.ru).

Для одержання інформації на віддалених машинах необхідно мати відповідні права.

Спочатку адміністратор встановлює на клієнтські термінали банківської установи у локальній чи глобальній мережі терміналів банківських установ розроблене клієнтське ПЗ.

При встановленні ПЗ треба мати права Адміністратора для повного доступу до програмних ресурсів користувача терміналом банківської установи.

Після налагодження розробленого ПЗ на клієнтських терміналом банківської установи необхідно налаштувати сервер БД FireBird та на сервері встановити розроблене в дипломі серверне ПЗ яке буде контролювати роботу мережі терміналів банківських установ та проводити сповіщення адміністратора при виявлені небезпечних моментів таких як противоправні дії користувачів при

роботі за терміналом банківської установи.

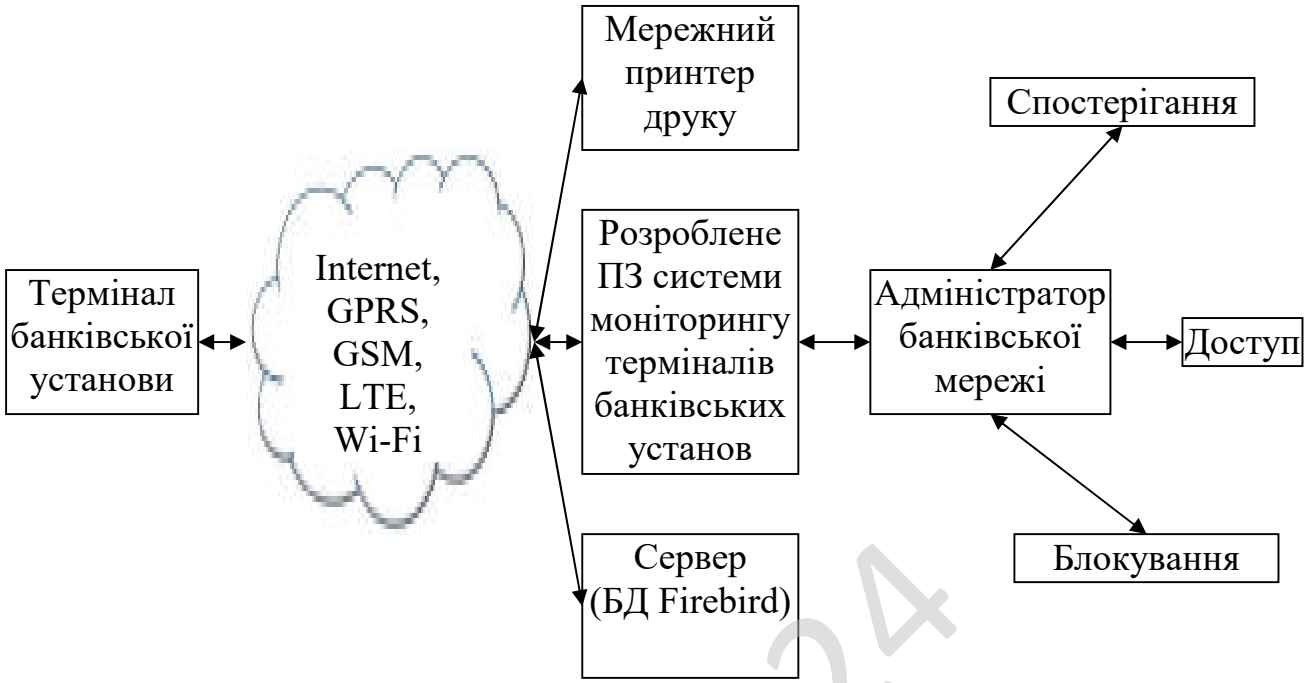

Рисунок 3.1 – Структурна схема роботи системи

При налагодженні серверного ПЗ адміністратор вибирає як буде проходити сповіщення чи це буде локальне сповіщення чи SMS, TWITTER повідомлення якщо адміністратор не знаходиться у терміналом банківської установи (сервера).

Розглянемо технології, які використовувались у бакалаврському проекті.

Для підвищення продуктивності мережі терміналів банківських установ для системи моніторингу терміналів банківських установ пропонується:

– Виявляти мережні проблеми на всіх семи рівнях моделі OSI з наданням детального опису проблеми й можливих шляхів її рішення на базі експертного аналізу.

– Проводити автоматичний збір даних для аналізу за розкладом.

– Декодувати мережні протоколи.

– Проводити оптимізацію продуктивності комутаторів, концентраторів, маршрутизаторів і іншого мережного встаткування на базі зібраної інформації.

– Спостерігати за змінами в мережній конфігурації в режимі on-line.

Програма системи моніторингу терміналів банківських установ повинна забезпечувати можливість моделювання мережного трафіку (пакети від клієнтської частини).

Адміністратор, що здійснює планування мережі терміналів банківських установ, може легко створити мережний трафік, що співвідноситься з очікуваним після установки нового мережного обладнання або ПЗ. Крім того, можна спрогнозувати реакцію мережних пристроїв на зміни мережної конфігурації або на збільшення трафіку. При цьому генератор трафіку повинен працювати у двох режимах: повторюване відправлення того самого кадру й прокручування відладочного файлу з різними параметрами. Різні системи відображення й наявність фільтрів, дозволяють адміністраторові бачити мережний трафік у різноманітних форматах.

ПЗ накопичують мережну статистику в міру захвата мережного трафіку, з відображенням накопиченої інформації в цифровому й графічному виді. Генеруються наступні види звітів:

– Мережна статистика, включаючи кількість станцій і кадрів, а так само відсоток завантаження мережі терміналів банківських установ.

– Статистика мережних помилок.

– Статистика по протоколах.

– Статистика по робочих станціях, включаючи переданий і отриманий трафік, статистику розмірів кадрів.

Портативні аналізатори трафіку призначені для дослідження одного сегмента мережі терміналів банківських установ або каналу передачі даних.

Програма системи моніторингу терміналів банківських установ ПК здійснює захват кадрів, що проходять через один або більше мережних інтерфейсів локального комп'ютера, а також декодування відомих протоколів і відображення отриманої інформації в різному виді.

Області застосування:

– Моніторинг мережі терміналів банківських установ в реальному часі та збір даних про стан мережі терміналів банківських установ.

– Визначення джерел і споживачів інформації в мережі терміналів банківських установ.

– Дозвіл проблем, пов'язаних з перевантаженням мережі терміналів банківських установ.

– Планування мережі терміналів банківських установ.

– Визначення помилок конфігурації.

– Генерація мережного трафіку з метою налагодження.

Повинні бути реалізовані наступні можливості:

– Перехоплення мережних пакетів у реальному часі й відображення раніше захоплених пакетів.

– Автоматичне виявлення мережних вузлів.

– Аналіз помилок фізичного рівня.

– Генерація попереджень про перевищення заданих порогів.

– Відображення статистичної інформації в реальному часі.

Програма системи моніторингу терміналів банківських установ ПК повинна дозволяти аналізувати мережний трафік у реальному часі. Після запуску переводить мережний інтерфейс у режим прослуховування й накопичує статистичну інформацію про трафік, дозволяючи відображати її в табличному або графічному виді.

Всі діаграми обновляються в реальному часі, тобто містять саму останню інформацію. Програма декодує протоколи мережної взаємодії, починаючи з канального й закінчуючи рівнем додатків. Декодер відображає повну інкапсуляцію й розшифровує всі заголовки.

Генерація трафіку незамінна при налагодженні мережних додатків розроблювачем. Програма системи моніторингу терміналів банківських установ ПК повинна відтворювати записані пакети, по одному або пачкою, із заданим

часовим інтервалом для кращого контролю завантаження мережі терміналів банківських установ. Пакети можуть бути попередньо відкоректовані редактором. Можлива також одночасна генерація трафіку з моніторингом.

Попередження генеруються в будь-який час, коли перевищуються заздалегідь певні граничні значення показників. Інформація про події, що вимагають термінового втручання, може бути послана на стільниковий телефон, по електронній пошті.

Важливою особливістю пакета є можливість аналізу мережного трафіку за тривалий період часу з поданням даних у графічному форматі.

Традиційним інструментом мережного інженера є декодер протоколів, за допомогою якого можна переглядати заголовки й вміст мережних пакетів у вигляді придатному для читання. Таким способом можна виявляти несправності в порівняно невеликих мережах – декодер протоколів ефективний тоді, коли проблема вже практично локалізована.

Експертний аналізатор перехоплює пакети з одночасною побудовою бази мережних об'єктів, виявляючи мережні аномалії: симптоми й діагнози. Симптоми – некритичні події, наприклад, однократний повтор передачі даних. Діагнози – критичні збої, що вимагають негайного втручання. Після локалізації й аналізу проблеми видається попередження адміністраторові з описом несправності й рекомендуються дії по її усуненню – це відбувається автоматично й у реальному масштабі часу.

Важлива перевага програми системи моніторингу терміналів банківських установ ПК – робота в локальних мережах, що комутуються (Switched). На відміну від концентраторів фізичного рівня (Hubs), останнім часом мережі терміналів банківських установ всі частіше використовують комутацію пакетів на другому рівні – канальному. Концентратори другого рівня (Switched Hubs) дозволяють різко знизити число т.зв. колізій, що є характерним явищем мереж CSMA/CD (Ethernet).

У більших мережах колізії різко знижують загальну продуктивність мережі терміналів банківських установ. Мережі терміналів банківських установ, що комутуються, неможливо аналізувати звичайними способами перехоплення пакетів. Концентратор другого рівня передає пакети тільки їхньому безпосередньому одержувачеві, а не всім вузлам мережі терміналів банківських установ.

Важливою особливістю є можливість аналізу мережного трафіку за тривалий період часу. Збирається й накопичується статистика, що може бути відображена в реальному часі у вигляді "моментального знімку" мережної активності, збережена для подальшого використання.

Аналізатор переглядає перехоплені в мережі терміналів банківських установ кадри й сигналізує адміністраторові про виявлені аномалії. Він автоматично виявляє проблеми на всіх семи рівнях моделі OSI у реальному часі, починаючи з канального й закінчуючи рівнем додатків. Інформація про виниклу проблему відображається разом з рівнем її важливості. При цьому оперує двома основними поняттями: симптоми (менш критичні проблеми) і діагнози (критичні).

Адміністратор може викликати вікно детального опису, у якому утримується інформація про симптоми проблеми й можливі шляхи її рішення. Виявлені симптоми й діагнози розташовуються системою на відповідних рівнях моделі мережної взаємодії. Зведена таблиця відображає поточний стан, що для кожного рівня включає діагнози, симптоми й всі виявлені мережні об'єкти.

Обґрунтування вибору бази даних. Жоден інший SQL-сервер не порівняється з FireBird в легкості установки, використовування і управління. Реалізація промислових стандартів ANSI SQL-92 і ODBC дозволяє використовувати дані FireBird практично з будь-якого існуючого сьогодні інструменту для створення клієнтських додатків.

1. Справжня масштабованість.

2. Легкість конкурентної роботи з даними.

3. Відповідність індустріальним стандартам.

4. Ідеально підходить для мультимедіа-додатків з використанням BLOB багато розмірних масивів.

5. Підтримує UNICODE і національні набори символів.

Якщо продуктивність бази даних критична для програмного забезпечення, то FireBird є якнайкращим вибором. FireBird забезпечує високопродуктивну функціональність, що підтримує складних операції для таких різних галузей, як біржова торгівля, фармацевтична промисловість, аерокосмічна промисловість, мережеве управління і т.п.

Масштабованість Windows-систем, NetWare, і платформ UNIX, вибір FireBird є дійсно незалежним. Якщо програма робочої групи одержав визнання, можливо легко перемкнути його на продуктивніший сервер рівня підприємства. Всі розроблені об'єкти бази даних будуть негайно доступні після переміщення бази даних на могутнішу платформу, підтримувану FireBird.

Опис архітектури бази даних FireBird

InterBase SQL Server був створений, розроблявся і продавався фірмою Interbase Software Corporation (ISC). При створенні і розвитку InterBase SQL Server була використана велика кількість нетрадиційних рішень і нових технологій.

Архітектура поколінь записів SQL-серверу FireBird побудована на архітектурі множинних поколінь записів (Multi-Generational Architecture – MGA). Ця архітектура використовує унікальний версіоніруючий механізм, який володіє високою продуктивністю при обробці коротких транзакцій і транзакцій ухвалення рішень.

Традиційно сервери баз даних підтримують модель On-Line Transaction Processing (OLTP), що характеризується великою кількістю коротких, одиночних транзакцій. Тоді як FireBird підтримує такий режим, додатково підтримуються тривалі транзакції підтримки ухвалення рішень.

Механізм версіонування дозволяє транзакціям позбавитися зайвих блокувань використовуваних даних, і використовує принцип читання даних не приводить до блокування їх зміни. На відміну від інших баз даних, не блокуючі

транзакції FireBird не вимагають додаткового програмування щоб забезпечити постійний, відтворний результат для кожного запиту. Таким чином, механізм версіонування дозволяє співіснувати коротким і довгим транзакціям і забезпечує максимальну продуктивність для обох.

База даних FireBird є першим, що запропонував концепцію активної бази даних. Активна база даних містить повідомлення про події, процедури, що зберігаються, тригери, визначувані користувачем функції і фільтри полів типа BLOb для автоматизації процесів, що відбуваються на сервері. Окрім цього, для реалізації логіки бази даних на сервері, FireBird забезпечує цілісність даних підтримкою чотирьох типів декларативної посилальної цілісності.

Тригери забезпечують логіку обробки даних зберігають і виконують логіку обробки даних на сервері, таким чином кожна програма, що використовує корпоративні дані, автоматично використовує цю логіку. Тригери FireBird автоматизують відгук на подіях на сервері, і часто використовуються для перевірки даних при вставці, зміні і видаленні записів в таблицях

Повідомлення про події автоматизують додатки роблять базу даних дійсно активної, автоматично повідомляючи зацікавлені клієнтські додатки в подіях, що відбулися в базі даних. Наприклад, коли кількість товару на складі зменшується нижче певної межі, додаток менеджера по закупівлях буде негайно повідомлений про це. Це відбувається без необхідності постійного опиту бази даних, тому не погіршує продуктивність системи, разом з тим гарантуючи доставку повідомлення програмному забезпеченню.

Процедури, що зберігаються, забезпечують продуктивність в FireBird значно збільшують продуктивність обробки даних, виконуючи її на сервері. Процедури, що зберігаються, можуть бути використані ПЗ, приєднаним до бази даних. Це дозволяє використовувати модульну розробку бази даних, забезпечує легкість супроводу і повторного використовування.

Визначувані користувачем функції (UDF) додають функціональність

UDF забезпечують можливість розширення SQL-серверу FireBird зовнішніми функціями будь-якої складності, написаними на компілюючи мовах програмування.. Наприклад, UDF можуть виробляти спеціальну обробку дати і часу, чисел, рядків, вмісту BLOb-полів, і навіть викликати ПЗ операційної системи.

Декларативна посилальна цілісність дозволяє FireBird ефективно управляти відносинами між таблицями бази даних (відповідно до entry level ANSI SQL92).

Підтримуючи наступні категорії обмежень:

– Унікальний і первинний ключ: Гарантує відсутність в таблиці двох рядків, що мають однакові значення ключових полів;

– Посилальна цілісність: Контролює відносини між таблицями, для гарантії цілісності даних в таблицях;

– Контроль даних: на рівні поля, полів запису, і зовнішніх таблиць по відношенню до поточної;

– Домени: Дозволяють створювати нові "типи" даних на основі базових типів;

– Домени можуть бути використані для оголошення часто використовуваних правил контролю полів, наприклад діапазону допустимих значень, списку допустимих значень, маски введення, значень за умовчанням і т.п.

Легкість управління і обслуговування SQL-серверів вимагають цілих підрозділів, де співробітники знімаються тільки обслуговуванням SQL-серверу, його настройкою і управлінням. FireBird розроблений не тільки для забезпечення високої продуктивності, але і для розробників баз даних будь-якого рівня, і не вимагає зусиль для підтримки працездатності. Це відмінне рішення як з погляду вартості супроводу, так і при перенесенні баз даних з настільних додатків на SQL-сервер. Додатково, FireBird підтримує 16– і 32-розрядний GUI Microsoft

Windows для спрощення адміністрування, моніторингу і відладки з клієнтського комп'ютера.

FireBird встановлюється дуже просто запуском setup. Після відповідей на питання про каталог установки і встановлюваних компонент, переписування файлів з дистрибутива на вінчестер, FireBird SQL Server готовий до роботи. Інші SQL-сервери вимагають ретельного прочитання інструкцій по установці, іноді модифікують ядро операційної системи, встановлюються протягом півгодини і більш навіть на комп'ютерах високої продуктивності, і після установки знову-же вимагають прочитання маси документації всього-лише для того, щоб почати працювати.

FireBird динамічно настроюється на кількість дискової і оперативної пам'яті або на кількість працюючих користувачів. Не потрібно настроювати сервер для отримання максимальної продуктивності.

Архітектура FireBird ефективно використовує ресурси системи.

Для установки потрібен мінімально 10Мб на диску (велику частину займають довідкові файли і приклади програмування) і мінімальна кількість оперативної пам'яті, достатнє для роботи операційної системи. Більшість інших продуктів вимагає велику кількість пам'яті і серверних ресурсів, збільшуючи вартість рішення.

FireBird дозволяє легко працювати з багато серверними транзакціями, і є першою комерційною СУБД, що реалізувала протокол 2PC. При цьому обробка транзакцій виконується по схемі двофазного підтвердження транзакцій, що гарантує цілісність даних без написання додаткового коду.

Як тільки транзакція виконується над двома і більш серверами баз даних, FireBird спочатку перевіряє готовність всіх серверів до завершення транзакції, і потім відправляє команду остаточного завершення транзакцій. Відновлення транзакцій, що не завершилися після першої фази, виконується легко дякуючи механізму множинних поколінь записів. Необхідність обробляти неструктуровані дані виникає в більшості додатків. FireBird підтримує як BLOB-поля так і

багатовимірні масиви. Це робить FireBird кращим вибором для мультимедійних і наукових додатків.

FireBird встановила промисловий стандарт в першій версії, віпущенной у 1986, дозволяючи зберігати звук, образи, графіку і двійкову інформацію прямо в базі даних використовуючи поля типа BLOb.

Окрім цього, сервер може використовувати фільтри полів BLOb. Це дозволяє стискати дані, що зберігаються, або перекодувати їх для потреб додатків. FireBird підтримує багатовимірні масиви для наукових або фінансових додатків. Зберігання масивів з розмірністю до 16 в одному полі бази даних спрощує програмування складних додатків і збільшує продуктивність.

FireBird забезпечує сумісність із стандартом SQL-92 (повна відповідність entry level). Це знижує витрати на навчання при освоєнні FireBird SQL Server, оскільки мова програмування баз даних відповідає відкритому індустріальному стандарту. Використовуючи стандартний SQL для визначення процедур, що зберігаються, трігерів, обмежень цілісності і декларативної посилальної цілісності, ви прискорюєте розробку і захищаєте свої вкладення в створені програмного забезпечення.

FireBird забезпечує зберігання обробку даних в різних національних кодуваннях. Підтримуються як одно байтові так і багато байтові набори символів для всіх операцій з рядками. Підтримувані набори символів включають UNICODE, ASCII, кодові сторінки DOS (включаючи 866 – CYRL), кодові сторінки Windows (включаючи WIN1251), і EUC. Для бази даних може бути вказаний набір символів за умовчанням, і перевизначена для будь-якого рядкового підлога таблиці. Спеціальні набори сортувань дозволяють враховувати особливості сортування національного алфавіту (наприклад PXW_CYRL). Ніяких додаткових інструментів для підтримки національних кодувань не вимагається.

FireBird підтримується великою кількістю популярних настільних баз даних і інструментами розробки, такими як Delphi і Delphi Client/Server, Borland C++, Borland C++ Builder, Corel Paradox, Visual dBASE, ReportSmith, PowerBuilder,

Cognos Impromptu, Novell InForms і додатками, що використовують JDBC. Окрім цього, драйвер ODBC дозволяє використовувати дані FireBird SQL Server з величезної кількості інших призначених для користувача програм і інструментальних засобів.

Загальні технічні характеристики і підтримувані платформи:

– Цілісність: Декларативний первинний ключ ; Декларативний вторинний ключ; Домени і контроль полів; Тригери – Необмежене число трігерів на одну дію із записом, спрацьовування при вставці, зміні і видаленні запису, включення і виключення трігерів під час роботи, виконання у випадковому або вказаному порядку, каскадне спрацьовування тригерів.

– Процедури, що зберігаються: підтримка повного синтаксису SQL;обробка помилок і виняткових ситуацій; можливість видачі результату у вигляді набору записів (select); рекурсивні виклики до 1000 рівнів вкладеності.

– Конкурентний доступ до даних: Оптимістична схема блокувань на рівні запису; Відсутність блокувань по читанню; Рівні ізоляції даних: читання підтверджених даних, відтворне читання і стабільність таблиць; Блокування для забезпечення, що розділяються, захищені і монопольні, рівні ізоляції "стабільність таблиць"

– Загальні можливості: Резервування даних без зупинки серверу (backup); Негайне автоматичне відновлення бази даних при збоях; Необмежене число під'єднуваних баз даних; Автоматичне управління двофазним завершенням транзакцій; Оптимальне зберігання символьної інформації (упаковка і стиснення) і BLOb-полів.

– Типи даних: Символьні (фіксованої і довільної довжини): до 32Кб на полі; Цілочисельні (коротке і довге ціле); Речовинні: простої і подвійної точності; Багатовимірні масиви: до 16 вимірювань на одне поле; BLOb – необмежена довжина, можливість визначення призначених для користувача підтипів; Імпорт і експорт ASCII даних; Фільтри BLOb для стиснення або перетворення даних BLOb.

– Стандарти: Відповідність ANSI SQL-92 Entry-Level, ODBC версії 2.0 (16-bit) і ODBC версії 2.5 (32-bit).

– Вимоги до системи: Мінімум оперативної пам'яті (необхідний мінімум для операційної системи) і дискового простору залежить від операційної системи конкретної платформи. Мережевий протокол – для всіх платформ TCP/IP, інші протоколи залежно від конкретної платформи.

Опис режимів транзакцій бази даних

При роботі з базою даних навіть якщо проект використовує базу даних локально, а не на віддаленому комп'ютері завжди є вірогідність втрати транзакції на сервер. Для запобігання таким діям, програмний продукт, що розробляється, повинен бути складений з урахуванням певних правил, таких як:

– режими транзакції;

– взаємодія транзакцій різного рівня;

– підтвердження транзакції;

– режими очікування;

– рівні ізоляції;

– режими доступу;

– компоненти бази даних.

Режими транзакцій бази даних підтримуються наступні рівні ізоляції транзакцій:

– READ COMMITTED RECORD_VERSION;

– READ COMMITTED NO RECORD_VERSION;

– SNAPSHOT;

– SNAPSHOT TABLE STABILITY.

Рівень ізоляції SNAPSHOT відповідає рівню REPEATABLE READ ANSI/ISO, за винятком того, що SNAPSHOT не забезпечує видимість фантомів, тобто дійсно забезпечує "відтворне читання" (на відміну від стандартного рівня ізольованості).

Транзакція у момент старту одержує "знімок" БД, який є незмінним до кінця транзакції. Читання даних, змінених конкуруючою транзакцією, дозволене, проте внесені нею зміни не доступні. Спроба зміни даних, що обновляються іншою транзакцією, приводить до конфлікту (Deadlock, SQLCODE = -913). Після завершення конкуруючих транзакцій оновлені ними дані все одно змінювати не можна, оскільки одержаний "знімок" вже не відображає поточного стану БД (Deadlock. Update conflicts with concurrent update. SQLCODE = -913).

Рівень SNAPSHOT забезпечує щонайвищий рівень ізоляції, проте може перешкоджати внесенню змін в БД при великому числі конкуруючих транзакцій. Необхідно застосовувати для забезпечення ідентичності результатів однакових запитів до БД в рамках однієї транзакції.

Встановлений в IB за умовчанням. Рівень ізоляції READ COMMITTED RECORD_VERSION майже повністю відповідає рівню READ COMMITTED ANSI/ISO.

Транзакції доступні зміни, вироблені іншими завершеними транзакціями. Спроба зміни даних, змінних іншою незавершеною транзакцією, приводить до конфлікту (Deadlock, SQLCODE = -913). На відміну від рівня READ COMMITTED ANSI/ISO дозволяється, не завершуючи транзакції, обновляти дані, змінені і підтверджені іншою транзакцією.

У цьому режимі можлива поява фантомних рядків і не узгоджено даних. Можна застосовувати для запитів до БД, що виконуються один раз протягом транзакції і не накопичуючих результатів, що підсумовують. Рівень ізоляції READ COMMITTED NO RECORD_VERSION аналогічний режиму READ COMMITTED RECORD_VERSION, проте спроба навіть простого читання даних, змінних іншою транзакцією, приводить до конфлікту (Deadlock, SQLCODE = -913). Після завершення конкуруючих транзакцій можна читати і обновляти змінені ними дані.

Таким чином, режим NO RECORD_VERSION непридатний для здійснення масових вибірок: записи прочитуються тільки до тих пір, поки не виявиться конфлікт.

Рівень ізоляції SNAPSHOT TABLE STABILITY фактично, блокування всієї таблиці на запис. Інші транзакції можуть тільки читати.

Режими доступу, всі рівні ізоляції мають додаткові опції доступу: тільки на читання (READ ONLY), або на читання і запис (READ WRITE).

Транзакція тільки на читання повинна обходитися "дешевше", тому що серверу не вимагається відстежувати можливі зміни в БД для виявлення конфліктів з іншими транзакціями. Режими очікування WAIT і NO WAIT дозволяють транзакції у разі конфлікту почекати, поки завершаться конкуруючі транзакції і провести свої зміни в базі даних або повернути код помилки зразу ж при виявленні конфлікту (NO WAIT).

Режим WAIT має сенс тільки якщо рівень ізоляції дозволяє змінювати раніше заблоковані рядки, тобто є рівнем READ COMMITTED. У режимі SNAPSHOT, встановленому за умовчанням, очікування даремне. Крім того, додаток, що виконує запит в такому режимі, підвисає на час очікування. Рекомендується використовувати NO WAIT з обробкою коду помилки.

За умовчанням встановлений режим WAIT. Підтвердження і відкіт транзакцій.

Всі операції над БД (включаючи команди DDL) в IB виконуються в контексті якої-небудь транзакції. Транзакції можуть бути явними і неявними. Неявна транзакція має параметри READ WRITE WAIT SNAPSHOT, починається при виконанні будь-якої команди і триває до команди явного завершення транзакції (COMMIT, ROLLBACK). Для виконання транзакції з іншими параметрами, а також для одночасного виконання декількох транзакцій з одного клієнта сервер IB дозволяє стартувати явні транзакції.

Для підтвердження транзакції використовуються команди COMMIT (підтвердження транзакції і її завершення), ROLLBACK (відмова від змін і

завершення транзакції) і COMMIT RETAINING (підтвердження транзакції із збереженням контексту). COMMIT RETAINING фіксує транзакцію, але зразу ж після цього стартує нову з тими ж параметрами, що і у завершеної транзакції і зберігаючи той же курсор. Таким чином, клієнтській програмі не вимагається наново створювати курсор і виконувати FETCH.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Комплекс бізнес-додатків програмного забезпечення, що розроблено призначений для проведення інформаційних, платіжних і сервісних операцій, здійснюваних на банкоматах (у т.ч. з функцією прийому готівки) і інформаційно-платіжних терміналах.

Переваги програмного забезпечення, що розроблено:

– Обслуговування на одному пристрої міжнародних банківських карт із магнітною смугою, мікропроцесорних карт, включаючи EMV-карти

– Сертифіковане ядро EMV (EMV 4.0 Level 2 Approved Application Kernel).

– Взаємодія з апаратним забезпеченням відповідно до архітектури, описуваної широко розповсюдженим стандартом CEN/XFS.

– Підтримка протоколів ISO8583, NDC/NDC+, DDC, у тому числі функції прийому готівки.

– Високий рівень безпеки.

– Взаємодія із системою прийому платежів для здійснення платежів на адресу третіх осіб.

– Ведення статистики про роботу термінала самообслуговування, журналу по всіх фінансових транзакціях, клієнтським чекам, збійним ситуаціям і відмовам у роботі.

Рисунок 3.2 – Функціональна схема системи

Функціональні можливості:

– Підтримка всіх стандартних функцій терміналів самообслуговування: списання коштів з банківського рахунку клієнта на користь постачальника послуг, поповнення банківського рахунку клієнта, видача й прийом наявних коштів, запит і печатка балансу, одержання історії операцій по карті, забезпечення роботи в службовому режимі (інкасація, настроювання й технічне обслуговування термінала).

– Забезпечення уведення клієнтом пін-коду, формування пін-блоку на ключах безпеки на основі уведеного пін-коду.

– Здійснення транзакцій у режимі онлайн-взаємодії із зовнішніми системами при використанні банківських карт. Здійснення транзакцій у режимі оффлайн (у режимі автономної роботи термінала самообслуговування) при використанні мікропроцесорних карт стандарту DUET.

– Забезпечення функції моніторингу стану апаратних і програмних компонентів термінала самообслуговування.

– Можливість гнучкого настроювання клієнтського інтерфейсу, екранів, друкованих форм (чеків, звітів, виписок).

– Можливість локалізації під національну мову як інтерфейсу, так і друкованих форм.

– Ведення розгорнутого технічного журналу з можливістю настроювання ступеня деталізації журналювання.

– Надання довідкової й рекламної інформації банку за допомогою внутрішньої корпоративної мережі.

– Підтримка сканера штрих-кодів.

Надійність:

– Стійкість до відмов і збоїв.

– Наявність механізмів запобігання помилок користувача.

– Забезпечення схоронності даних при відмовах і збоях апаратно-програмного комплексу.

– Перевірка наявності й коректності всіх вихідних даних, необхідних для виконання будь-якої операції.

Безпека

У комплексі програмного забезпечення, що розроблено розмежовані права доступу обслуговуючого персоналу до функцій пристрою на основі рольової моделі. Підтримуються алгоритми шифрування DES/3DES. При відсутності зовнішнього електроживлення програмного забезпечення, що розроблено

відслідковує рівень заряду батарей і при падінні його до критичного рівня переводить термінал у режим «не обслуговує». програмного забезпечення, що розроблено дозволяє відслідковувати спроби несанкціонованого доступу. Можливе настроювання функції затримки карти (забуті, розмагнічені й ушкоджені, затримані через уведення невірних варіантів пін-коду, зазначені в стоп-аркуші карти й т.д.).

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврського проектування, наведена на рисунку 3.3.

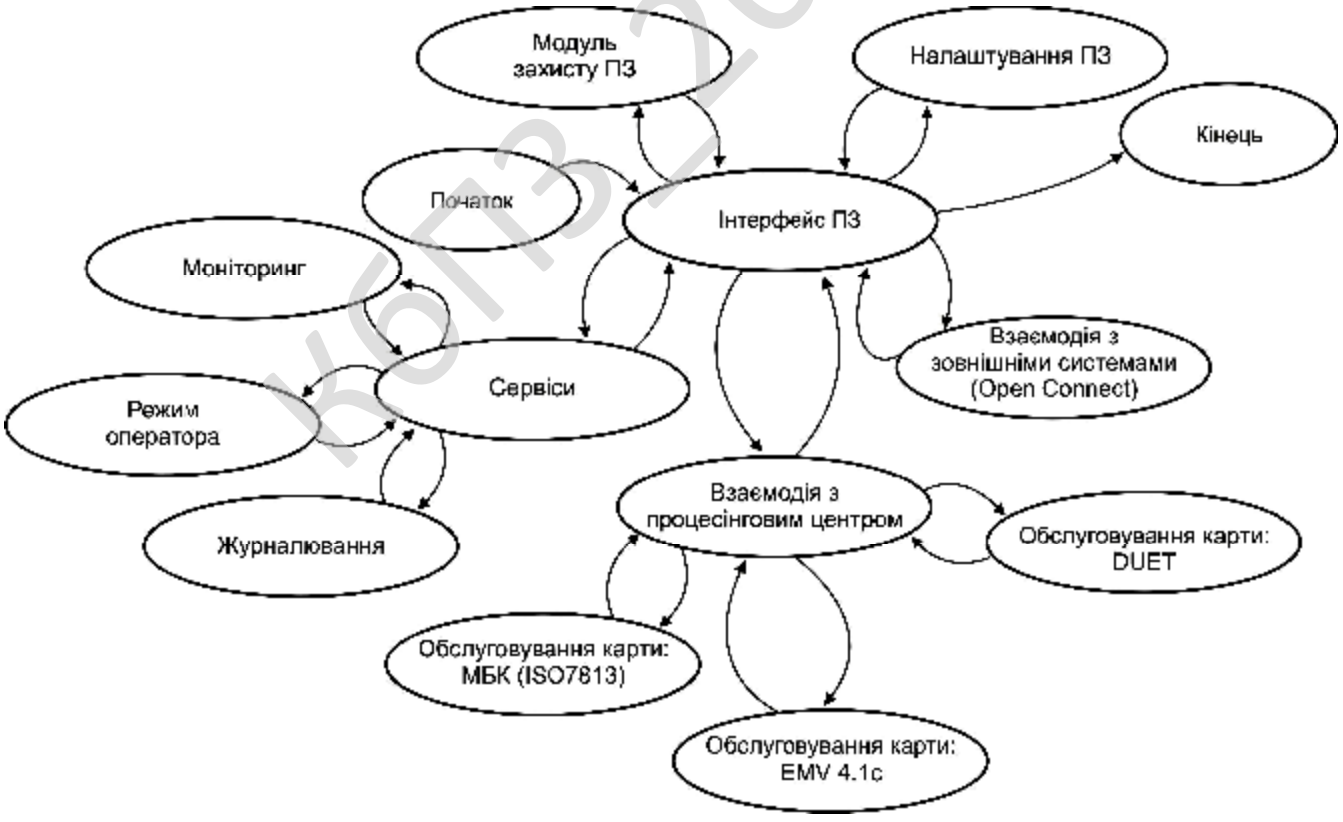

Рисунок 3.3 – Діаграма взаємодії процесів

Після початку роботи розробленого ПЗ ми потрапляємо до інтерфейсу ПЗ, звідки після модулю захисту ПЗ та налаштування ПЗ та блоку взаємодії з зовнішніми системами (Open Connect) проводяться роботи з сервісами з моніторингом, роботою у режимі оператора та дії журналювання. Далі проводиться взаємодія з процесінговим центром з перевіркою на обслуговування карт форматів: ISO7813, EMV 4.1c, DUET.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 наведено блок-схему основної програми. Її робота складається з виконання наступних кроків.

Спершу відбувається виведення основного вікна програми. Після цього:

– Сканування наявності необхідних ресурсів банківського терміналу.

– Ініціалізація ресурсів ПЗ.

– Читання файлу налаштувань підключення до БД.

– Перевірка ПЗ та файлів БД.

– Перевірка ПЗ пройдена?

– Сканування таблиць БД.

– Є доступ до мережі Інтернет?

– Є доступ до серверу БД?

– Отримання файлу поточних дій.

– Отримання пакету даних?

– Ідентифікація відправника.

– Ідентифікація пройдена?

– Розпакування пакету даних.

– Обробка даних.

– Відправлення пакету даних?

– Формування скриптів дії.

– Пакування даних, додавання ідентифікаційних даних.

– Відправлення даних.

– Підпрограма моніторингу системи.

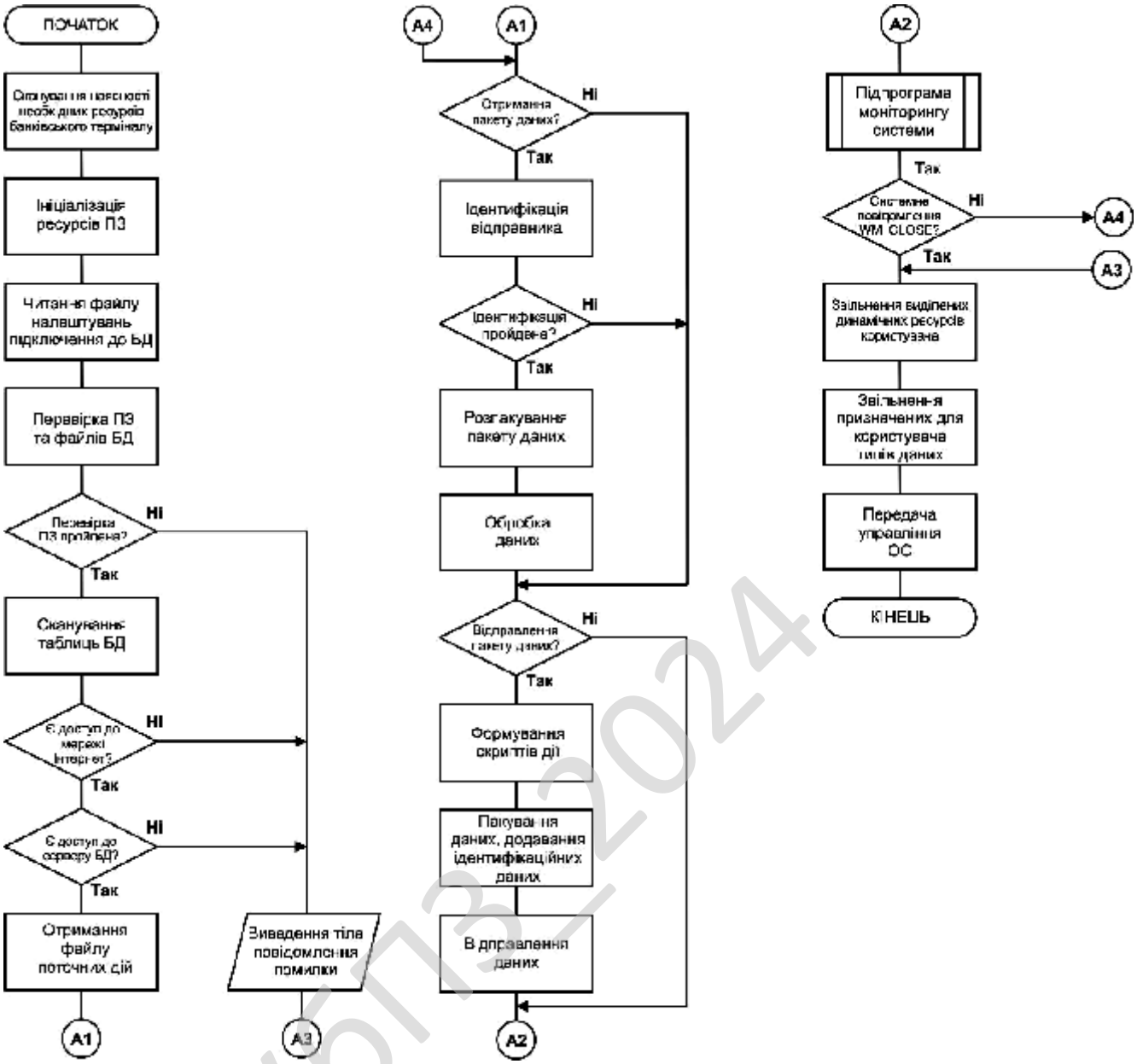

Рисунок 4.1 – Блок-схема основної програми

– Системне повідомлення WM_CLOSE?

– Звільнення виділених динамічних ресурсів користувача.

– Звільнення призначених для користувача типів даних.

– Передача управління ОС.

На рисунку 4.2 зображена робота підпрограми моніторингу системи, де відбувається наступне:

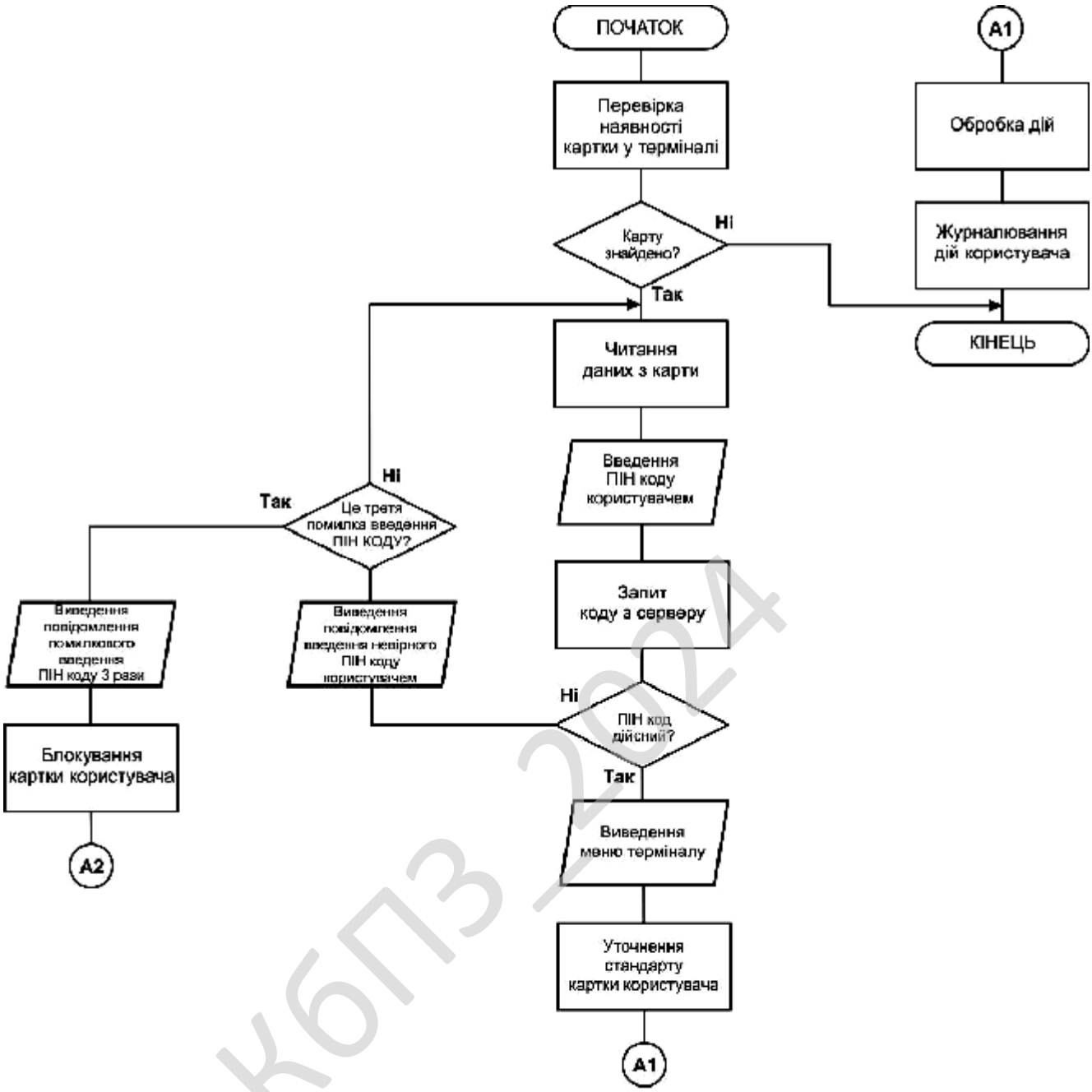

Рисунок 4.2 – Блок-схема підпрограми моніторингу системи

– Перевірка наявності картки у терміналі.

– Карту знайдено?

– Читання даних з карти.

– Введення ПІН коду користувачем.

– Запит коду з серверу.

– ПІН код дійсний?

– Виведення повідомлення введення невірного ПІН коду.

– Виведення повідомлення помилкового введення ПІН коду 3 рази.

– Блокування картки користувача.

– Виведення меню терміналу.

– Уточнення стандарту картки користувача.

– Обробка дій.

– Журналювання дій користувача.

наступне

Опис алгоритмів функціонування системи

Опис розроблених функцій. Опис роботи розроблених функцій. Для визначення режиму роботи розробленої в дипломі програми потрібно визначити ОС.

Розглянемо розроблену функцію, яка визначає тип системи.

```
function Tmainform.OS(var Value: Boolean): Boolean;
var Ver: Tosversioninfo;
    Bres: Boolean;
begin
  Ver.dwosversioninfosize := Sizeof(Tosversioninfo);
  Bres := Getversionex(Ver);
  if not Bres then
  begin
    Result := False;
    Exit;
  end else
    Result := True;
  case Ver.dwplatformid of
    VER_PLATFORM_WIN32_XP: Value := True;
//Windows 8  -  підходить
    VER_PLATFORM_WIN32_WINDOWS : Value := False;
//Windows 9x - Me - не підходить
    VER_PLATFORM_WIN32s : Result := False;
//Windows 3.x  -  не підходить
  end;
end;
```

Якщо дана функція повернула результат True, виходить, визначення версії системи пройшло успішно, і тип системи буде зберігатися в змінній Value, а якщо ні, то визначення типу системи, у цьому випадку прийде завершити виконання програми.

Дана функція буде практично найголовнішою, тому що вона буде вказувати бакалаврській програмі яку частину коду виконувати.

Розглянемо розроблену функцію, що надає інформацію про локальні ресурси й можливості їх контролю.

Terminal_enum – за допомогою цієї функції отримаємо дані про всі спільні ресурси. Оголошення функції для Windows 8:

```
Terminal_enum :function (Servername:Pwchar; Level :DWORD; Bufptr:Pointer;
Prefmaxlen: DWORD; Entriesread, Totalentries, resume_handle:LPDWORD): DWORD;
```

Параметри:

Servername – повинен містити ім'я віддаленого комп'ютера, на якому повинна виконаться функція, якщо виконуємо у себе, то даному параметрові можна привласнити NIL.

Level – повинен містити ідентифікатор структури.

Bufptr – повинен містити адресу покажчика на масив структур.

Prefmaxlen – повинен містити максимальну довжину повернених даних у байтах, якщо не ставити обмеження, то даному параметрові потрібно привласнити DWORD(– 1).

Entriesread – повинен містити покажчик на змінну, у яку запишеться кількість спільних ресурсів доступних на даний момент.

Результати виконання будуть збережені в масиві структур переданих функції при її виклику. Існує 6 типів структур переданих функції Terminal_enum

```
SHARE_INFO_0  –  тільки Windows 8 SP0
SHARE_INFO_1  –  тільки Windows 8 SP1
share_info_1  –  тільки Windows 9x – Me
SHARE_INFO_2  –  тільки Windows 8 SP2
share_info_50  –  тільки Windows 9x – Me
SHARE_INFO_502  –  тільки Windows 8 SP3
Структура share_info_50, оголошення структури:
```

```
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;
```

Розроблені поля:

shi50_netname – містить рядок з мережним ім'ям ресурсу;

shi50_type – визначає тип ресурсу;

shi50_flags – містить інформацію про права доступу до ресурсу;

shi50_remark – покажчик на рядок необов'язкових коментарів;

shi50_path – містить локальне розташування ресурсу;

shi50_rw_password – містить пароль на запис – читання;

shi50_ro_password – містить пароль на читання.

Одержати значення двох останні полів можна тільки при одержанні інформації про свій комп'ютер, в інших випадках вони залишаються порожніми. Структура SHARE_INFO_2, оголошення структури:

```
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;
```

Розроблені поля:

– shi2_netname містить покажчик на рядок утримуючу ім'я ресурсу;

– shi2_type визначає тип ресурсу;

– shi2_remark покажчик на рядок коментарів;

– shi2_permissions містить інформацію про права доступу до ресурсу;

– shi2_max_uses визначає максимальне кіл – у підключень до ресурсу;

– shi2_current_uses визначає кіл – у поточних підключень;

– shi2_path містить покажчик на рядок локального розташування ресурсу;

– shi2_passwd містить покажчик на рядок пароля.

Визначаємо, тип системи, завантажуємо необхідну бібліотеку, одержуємо адреси функцій і виконуємо функцію.

```
procedure Tmainform.btngetsharesclick(Sender: Tobject);
var
  i:Integer;
  Flibhandle : Thandle;
  Sharent : Pshareinfo2Array;
//<= Змінні
  entriesread,totalentries:DWORD;
// <= для Windows 8
  Share : array [0..512] of Tshareinfo50;
  pcentriesread,pctotalavail:Word;
  OS: Boolean;
begin
  lbxshares.Items.Clear;
  if not OS(OS) then Close;
//  Визначаємо тип системи
  if OS then begin
// Код для XP
    Flibhandle := Loadlibrary('My_PZ1.DLL');
//  Завантажуємо бібліотеку
    if Flibhandle = 0 then Exit;
    //Зв'язуємо функцію
    @Terminal_enumnt := Getprocaddress(Flibhandle,'Terminal_enum');
    if not Assigned(Terminal_enumnt) then //
// Перевірка
    begin
      Freelibrary(Flibhandle);
      Exit;
    end;
    Sharent := nil;
// Очищаємо покажчик на масив структур
    //Виклик функції
    if Terminal_enumnt(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
    Flibhandle := Loadlibrary('My_PZ2.DLL');
//  Завантажуємо бібліотеку
    if Flibhandle = 0 then Exit;
    //Зв'язуємо функцію
    @Terminal_enum := Getprocaddress(Flibhandle,'Terminal_enum');
    if not Assigned(Terminal_enum) then
// Перевірка
    begin
      Freelibrary(Flibhandle);
      Exit;
    end;
    if Terminal_enum(nil,50,@Share,Sizeof(Share),
            @pcentriesread,@pctotalavail) <> 0 then
// Виклик функції
    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);
 вивантажити бібліотеку
end;
```

При виконанні цього коду Listbox заповниться назвами загальних ресурсів, які беруться з масиву структур.

Масив заповнюється в результаті виконання функції.

Звичайно в XP використовуються функції Netapibufferallocate і Netapibufferfree – за допомогою їх виділяється й звільняється пам'ять під масив структур.

Я реалізував у дипломі по іншому – оголошення структури Tshareinfo2Array = array [0..512] of Tshareinfo2; Тут пам'ять уже виділена.

Я волію користуватися саме таким оголошенням структури. І у вихідному коді розробленої бакалаврської програми так і реалізовано.

Тепер розглянемо розроблену функцію Terminal_sharedel яка дозволить закрити обраний загальний ресурс.

```
var
Terminal_sharedel:function (pszserver,psznetname:
                                Pchar;usreserved:Word):DWORD;
```

Параметри:

Servername – повинен містити ім'я віддаленого комп'ютера, якщо закриваємо свої ресурси то даному параметру потрібно присвоїти NIL;

Netname – покажчик на рядок утримуючу ім'я ресурсу, що закривається.

Обидві функції не використовують ніяких структур. У якості імені передається не шлях до ресурсу, а саме ім'я ресурсу яке я визначив за допомогою коду (розглянутого вище). У випадку успішного виконання функцій, їх результат буде дорівнює нулю.

```
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 OS(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
//Якщо не знайдений завершення роботи  ПЗ
  if not lbxshares.Selected[i] then Exit;
  Sharename := lbxshares.Items.Strings[i];
  if OS then begin
```

```
// Код для XP
    Flibhandle := Loadlibrary('MY_PZ1.DLL');
    if Flibhandle = 0 then Exit;
    @Terminal_sharedelnt := Getprocaddress(Flibhandle,'Terminal_sharedel');
    if not Assigned(Terminal_sharedelnt) 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
    Terminal_sharedelnt(nil,Nament,0);
// Видаляємо ресурс
    Freemem(Nament);
// Звільняємо пам'ять
  end else begin
    Flibhandle := Loadlibrary('MY_PZ2.DLL');
    if Flibhandle = 0 then Exit;
    @Terminal_sharedel := Getprocaddress(Flibhandle,'Terminal_sharedel');
    if not Assigned(Terminal_sharedel) then
// Перевірка
    begin
      Freelibrary(Flibhandle);
      Exit;
    end;
    Fillchar(Name9x, Sizeof(Name9x), #0);
// Очищаємо масив
    move(Sharename[1],Name9x[0],Length(Sharename));
// Заповнюємо масив
    Terminal_sharedel(nil,@Name9x,0);
// Видаляємо ресурс
  end;
  Freelibrary(Flibhandle);
end;
```

Відкриття локального ресурсу. Розглянемо розроблену функцію Terminal_shareadd. Оголошення функції для Windows 8:

```
var
```

```
Terminal_shareadd: function (servername: Pwidechar; level: DWORD;
buf:Pointer; parm_err: LPDWORD): DWORD;
```

Параметри:

servername – повинен містити ім'я віддаленого комп'ютера на якім повинна виконається функція, якщо відкриваємо локальний ресурс то даному параметру потрібно привласнити NIL;

level – повинен містити ідентифікатор структури;

buf – повинен містити покажчик на структуру;

parm_err – містить покажчик помилки

У функції для XP також використовується покажчик а не адреса покажчика.

Розглянемо вихідний код виклику функції:

```
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;
```

Для виконання функції необхідно додати в розділ Uses модулі Shelldipapi і Shldipobj. Ось сам код відкриття 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 майбутнього ресурсу
//Визначаємо мережне ім'я
  Tmpname := Inputbox('Share name','Enter name','Test');
  if Tmpdir = '' then Exit;
  if not OS(OS) then Close;
// З'ясовуємо тип системи
  if OS then begin //Код для XP
    Flibhandle := Loadlibrary('MY_PZ1.DLL');
    if Flibhandle = 0 then Exit;
    @Terminal_shareaddnt := Getprocaddress(Flibhandle,'Terminal_shareadd');
    if not Assigned(Terminal_shareaddnt) 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_READ;
//Доступ
    //Кількість максимальних підключень
    Sharent.shi2_max_uses := DWORD( - 1);
//Кількість поточних підключень
    Sharent.shi2_current_uses := 0;
    Getmem(Tmpdirnt, Tmplength);
    Stringtowidechar(Tmpdir, Tmpdirnt, Tmplength);
    Sharent.shi2_path := Tmpdirnt;
```

```
//Шлях до ресурсу
    Sharent.shi2_passwd := nil;
// Пароль
    Terminal_shareaddnt(nil,2,@Sharent,nil);
// Додаємо ресурс
    Freemem (Tmpnament);
// звільняємо пам'ять
    Freemem (Tmpdirnt);
  end else begin
// Код для 9x
    Flibhandle := Loadlibrary('MY_PZ2.DLL');
    if Flibhandle = 0 then Exit;
    @Terminal_shareadd := Getprocaddress(Flibhandle,'Terminal_shareadd');
    if not Assigned(Terminal_shareadd) then
    begin
      Freelibrary(Flibhandle);
      Close;
    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_RDONLY;
// 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
    Terminal_shareadd(nil,50,@Share9x,Sizeof(Share9x));
  end;
  Freelibrary(Flibhandle);
end;
```

Змінні Tmpnament і Tmpdirnt звільняються тільки після виконання функції. Це критично, а якщо ні, то структура передана функції буде із двома "незаповненими" полями.

Даний код відкриває новий ресурс на повний доступ.

Одержання списку поточних сесій. Для визначення користувачів підключених до комп'ютера була розроблена функція Mydip_Netsession. Оголошення функції для Windows 8:

```
Mydip_Netsession: function(Servername, Uncclientname, Username: Pwchar; Level: DWORD; bufptr: Pointer; prefmaxlen: DWORD;entriesread, totalentries, resume_handle: LPDWORD):DWORD;
```

Параметри:

Servername – повинен містити ім'я віддаленого комп'ютера на якому повинна виконається функція, якщо дивимося в себе то даному параметрові потрібно привласнити NIL.

Uncclientname – містить покажчик на рядок утримуючу ім'я сесії.

Username – містить покажчик на рядок утримуючу ім'я користувача.

Level – повинен містити ідентифікатор структури.

Bufptr – повинен містити адреса покажчика на масив структур.

Prefmaxlen – повинен містити максимальну довжину повернутих даних у байтах

Entriesread – повинен містити покажчик на змінну в яку запишеться кількість загальних ресурсів доступних на даний момент.

Для чіткого представлення часу роботи в бакалаврській програмі була розроблена функція, завдання якої перетворювати кількість секунд у більш 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 - (m*60);
        end;
        if m > 59 then begin
            h:=int(m/60);
            m:=m - (h*60);
        end;
        if h > 23 then begin
            d:=int(h/24);
            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;
```

Завершення сесій. Для завершення відкритих сесій розроблена функція Mydip_Netsessiondel. Оголошення функції для Windows 8:

```
Var Mydip_Netsessiondel: function(Servername,
    Uncclientname,Username:Pwchar):DWORD;
```

Servername – повинен містити ім'я віддаленого комп'ютера на якому повинна виконається функція;

uncclientname – повинен містити ім'я клієнта чия сесія завершується;

username – повинен містити ім'я користувача чия сесія завершується, якщо параметр NIL, завершаться всі сесії указані в параметрі uncclientname.

Розглянемо вихідний код виклику функції:

```
procedure Tmainform.btnclosesessionclick(Sender: Tobject);
var
OS: Boolean; Flibhandle : Thandle;
Cnament: Pwidechar; Cname9x: Pansichar;
  Key:Smallint; i: Integer;
begin
  if not OS(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('MY_PZ1.DLL');
    if Flibhandle = 0 then Exit;
     @Mydip_Netsessiondelnt := Getprocaddress(Flibhandle,
                                              'Mydip_Netsessiondel');
    if not Assigned(Mydip_Netsessiondelnt) then
    begin
      Freelibrary(Flibhandle); Exit;
    end;
    //Перетворимо дані в необхідний вигляд
    Cnament := Pwchar(Widestring('\\'+lvsessions.Items.Item[i].Caption));
    Mydip_Netsessiondelnt(nil,Cnament,nil);
  end else begin
    Flibhandle := Loadlibrary('MY_PZ2.DLL');
    if Flibhandle = 0 then Exit;
@Mydip_Netsessiondel:=Getprocaddress(Flibhandle, 'Mydip_Netsessiondel');
    if not Assigned(Mydip_Netsessiondel) then
    begin Freelibrary(Flibhandle); Exit;  end;
    //Перетворимо дані в необхідний вигляд
    Cname9x := Pansichar(lvsessions.Items.Item[i].Caption);
    key := Sessionclosekey[i]; //Беремо ключ із масиву
    Mydip_Netsessiondel(nil,Cname9x,Key);
  end;
  Freelibrary(Flibhandle);
end;
```


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_1 та 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:

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

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено головне вікно програми. З нього видно, що інтерфейс користувача програми складається з таких логічних блоків:

– Меню: Довідкова; Налаштування.

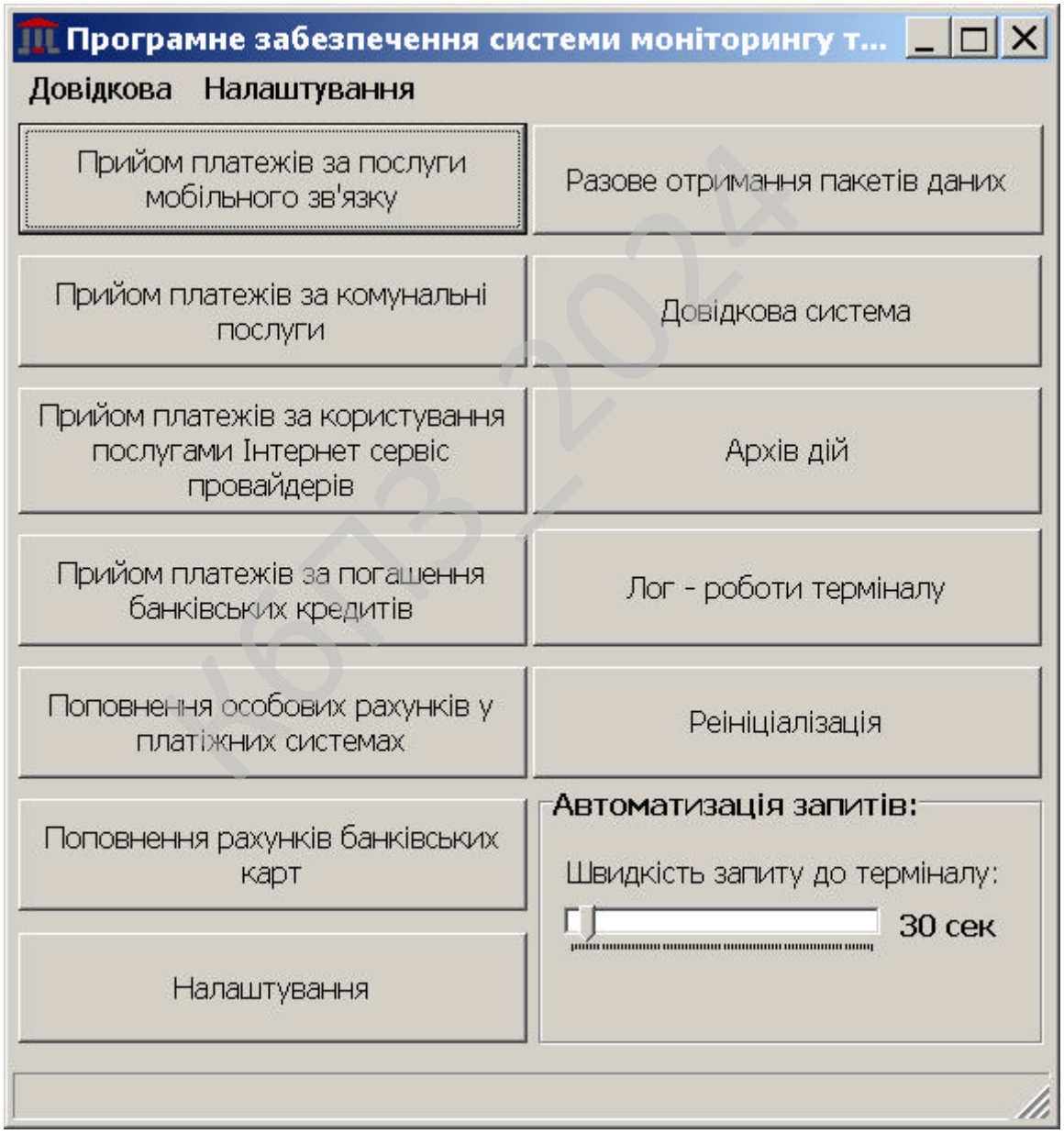

Рисунок 5.1 – Головне вікно програми

– Функціонал: Прийом платежів за послуги мобільного зв'язку; Прийом платежів за комунальні послуги; Прийом платежів за користування послугами Інтернет сервіс провайдерів; Прийом платежів за погашення банківських кредитів; Поповнення особових рахунків у платіжних системах; Поповнення рахунків банківських карт; Налаштування; Разове отримання пакетів даних; Довідкова система; Архів дій; Автоматизація запитів; Швидкість запиту до терміналу.

На рисунку 5.2 зображено форму авторського права. Початок тексту ліцензії для розробленого продукту. Ця програма – вільне програмне забезпечення; Ви можете розповсюджувати її та/або вносити зміни відповідно до умов Загальної Публічної Ліцензії GNU у тому вигляді, у якому вона була опублікована Фондацією Вільного Програмного Забезпечення; або 2ї версії Ліцензії, або (на Ваш розсуд) будь-якої більш пізньої версії. Розроблене ПЗ має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows 10/11 без особливих складностей освоїть і цю програму, оскільки її інтерфейс повністю розроблений під дане операційне середовище.

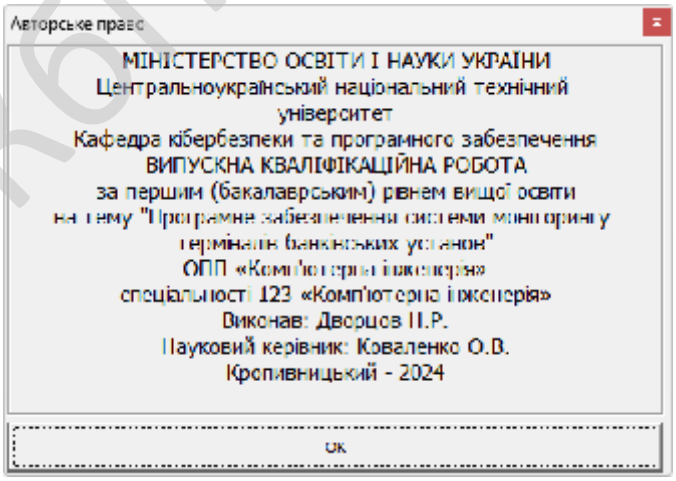

Рисунок 5.2 – Довідка

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи моніторингу терміналів банківських установ.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем моніторингу терміналів банківських установ.

– Досліджена система моніторингу терміналів банківських установ.

– На основі отриманих результатів досліджень створена програмна реалізація системи моніторингу терміналів банківських установ.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання моніторингу терміналів банківських установ.

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 Delphi 10.4.1. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи моніторингу терміналів банківських установ. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне

програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ДСТУ 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. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 1». Cisco Press. 2020. – 848 p.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

23. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

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

25. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation

Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

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

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

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

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

30. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

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

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

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

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

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

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

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

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

39. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в

комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

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

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

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

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

44. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

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

46. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу

відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

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

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

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

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

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

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

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

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи моніторингу терміналів банківських установ.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 131-02 від 01.04.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи моніторингу терміналів банківських установ.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти 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 моніторингу терміналів банківських установ;

– 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, працювати в ОС Windows 10/11 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 10/11.

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 Delphi 10.4.1.

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 – 78 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

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.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2024 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 13.06.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Коваленко О.В.

Програмне забезпечення системи моніторингу терміналів банківських установ

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 40

Літера: РП

Кропивницький – 2024 року

Отримання інформації з локальної мережі (Form5.pas)

```
unit Unit5; // модуль 5

// Copyriht @ Дворцов Нікіта Романович, 2024

interface

uses
SysUtils, Windows, Classes, WinSock, DIP_Common, MiTeC_IpHlpAPI, MiTeC_NetBIOS;

type
TDIPLOMsock = class(TPersistent)
private
FDesc: string;
FStat: string;
FMajVer: word;
FMinVer: word;
FModes: TExceptionModes;
procedure SetMode(const Value: TExceptionModes);
public
constructor Create;
procedure GetInfo;
published
property ExceptionModes: TExceptionModes read FModes Write SetMode;
property Description: string read FDesc stored False;
property MajorVersion: word read FMajVer stored False;
property MinorVersion: word read FMinVer stored False;
property Status: string read FStat stored False;
end;

TAdapterType = (atOther, atEthernet, atTokenRing, atFDDI, atPPP, atLoopback,
atSlip);

PAdapter = ^TAdapter;
TAdapter = record
Name,
Address: string;
Typ: TAdapterType;
EnableDHCP,
HaveWINS: boolean;
IPAddress,
IPAddressMask,
Gateway_IPAddress,
Gateway_IPAddressMask,
DHCP_IPAddress,
DHCP_IPAddressMask,
PrimaryWINS_IPAddress,
PrimaryWINS_IPAddressMask,
SecondaryWINS_IPAddress,
SecondaryWINS_IPAddressMask: TStringList;
end;

TNodeType = (ntUnknown, ntBroadcast, ntPeerToPeer, ntMixed, ntHybrid);

TTCPIP = class(TPersistent)
private
FAdapters: TStringList;
FProxy: boolean;
FRouting: boolean;
FDNS: boolean;
FHost: string;
FDomain: string;
FDNSPrefix: string;
FDNSList: TStrings;
FDNSSuffix: TStrings;
FNode: TNodeType;
```

```
FDHCPScope: string;
FModes: TExceptionModes;

procedure ClearList;
function GetAdapter(Index: Word): TAdapter;
function GetAdapterCount: Word;
procedure SetMode(const Value: TExceptionModes);
public
constructor Create;
destructor Destroy; override;
procedure GetInfo;
procedure Report(var sl :TStringList; Standalone: Boolean = True); virtual;
function FindAdapter(AName: string): Integer;
property Adapter[Index: Word]: TAdapter read GetAdapter;
end;

TNetwork = class(TPersistent) // Об’ва класу TNetwork
private
FVirtAdapter,FPhysAdapter: TStrings;
FWinsock: TDIPLOMsock;
FIPAddress: TStrings;
FMACAddress: TStrings;
FCli: TStrings;
FServ: TStrings;
FProto: TStrings;
FModes: TExceptionModes;
FTCPIP: TTCPIP;
function GetLocalIP :string;
procedure SetMode(const Value: TExceptionModes);
public
constructor Create;
destructor Destroy; override;
procedure GetInfo;

procedure Report(var sl :TStringList; Standalone: Boolean = True); virtual;
end;

const
NodeTypes: array[TNodeType] of string = ('Unknown','Broadcast','Peer-To-
Peer','Mixed','Hybrid');
AdapterTypes: array[TAdapterType] of string = ('Other', 'Ethernet', 'Token
Ring', 'FDDI', 'PPP', 'Loopback', 'Slip');

implementation

uses Registry, MiTeC_Routines, DIP_Devices;
{ TDIPLOMsock }
constructor TDIPLOMsock.Create;
begin
ExceptionModes:=[emExceptionStack];
end;

procedure TDIPLOMsock.GetInfo; // Інфо сокету
var
GInitData :TWSADATA;
begin

if wsastartup($101,GInitData)=0 then begin
FDesc:=GInitData.szDescription;
FStat:=GInitData.szSystemStatus;
FMajVer:=Hi(GInitData.wHighVersion);
FMinVer:=Lo(GInitData.wHighVersion);
wsacleanup;
end else
FStat:='Winsock cannot be initialized.';
end;

procedure TDIPLOMsock.SetMode(const Value: TExceptionModes); // Режим сокету
begin
```

```
FModes := Value;
end;

{ TNetwork }

function TNetwork.GetLocalIP: string;
type
TaPInAddr = array [0..255] of PInAddr;
PaPInAddr = ^TaPInAddr;
var
phe  :PHostEnt;
pptr :PaPInAddr;
Buffer :array [0..63] of char;
i :integer;
GInitData :TWSADATA;
begin
wsastartup($101,GInitData);
result:='';
GetHostName(Buffer,SizeOf(Buffer));
phe:=GetHostByName(buffer);
if not assigned(phe) then
exit;
pptr:=PaPInAddr(Phe^.h_addr_list);
i:=0;
while pptr^[I]<>nil do begin
result:=Result+inet_ntoa(pptr^[I]^)+',';
inc(i);
end;
Delete(Result,Length(Result),1);
wsacleanup;
end;

procedure TNetwork.GetInfo; // Інфо мережне
var
i: integer;
s,ck,dv: string;
sl: TStringList;
const
rkNetworkNT = {HKEY_LOCAL_MACHINE}'\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\NetworkCards';
rkNetwork2K = {HKEY_LOCAL_MACHINE}'\SYSTEM\CurrentControlSet\Control\Network';

rvNetworkNT = 'Description';

rvProtoClass = 'NetTrans';
rvServClass = 'NetService';
rvCliClass = 'NetClient';
begin
Winsock.GetInfo;
TCPIP.GetInfo;
if IsNT then
with TRegistry.Create do begin
sl:=TStringList.Create;
try
 RootKey:=HKEY_LOCAL_MACHINE;
 if OpenKeyReadOnly(rkNetworkNT) then begin
   GetKeyNames(sl);
   CloseKey;
   for i:=0 to sl.Count-1 do
     if OpenKeyReadOnly(rkNetworkNT+'\'+sl[i]) then begin
       s:=ReadString(rvNetworkNT);
       if FPhysAdapter.IndexOf(s)=-1 then
         FPhysAdapter.Add(s);
       Closekey;
     end;
 end;
finally
 sl.Free;
 Free;
```

```
end;
end;
FIPAddress.CommaText:=GetLocalIP;
if IsNT5 then begin
ck:=rkNetwork2K;
dv:=rvNetworkNT;
end else begin
ck:=ClassKey;
dv:=DescValue;
end;
GetClassDevices(ck,rvProtoClass,dv,FProto);
GetClassDevices(ck,rvServClass,dv,FServ);
GetClassDevices(ck,rvCliClass,dv,FCli);
NB_GetMACAddresses(Machinename,FMACAddress);
end;

constructor TNetwork.Create;
var
i: Integer;
s: string;
begin
inherited;
FWinsock:=TDIPLOMsock.Create;
FTCPIP:=TTCPIP.Create;
ExceptionModes:=[emExceptionStack];
s:='';
with TDevices.Create do begin
GetInfo;
for i:=0 to DeviceCount-1 do begin
if Devices[i].FriendlyName='' then
 s:=Devices[i].Description
else
 s:=Devices[i].FriendlyName;
if Devices[i].DeviceClass=dcNet then begin
 if (Devices[i].ResourceListKey<>'') and (Devices[i].Location<>'') then
   FPhysAdapter.Add(s)
 else
   FVirtAdapter.Add(s)
end;
end;
Free;
end;
end;

destructor TNetwork.Destroy; // Виклик деструктора
begin
FWinsock.Destroy;
FTCPIP.Destroy;
FVirtAdapter.Destroy;
FPhysAdapter.Destroy;
FMACAddress.Destroy;
FIPAddress.Destroy;
FProto.Destroy;
FCli.Destroy;
FServ.Destroy;
inherited;
end;

procedure TNetwork.Report; // Мережній звіт
begin
with sl do begin
if Standalone then ReportHeader(sl);
Add('<Network classname="TNetwork">');
Add('<section name="Physical Adapters">');
StringsToRep(PhysicalAdapters,'Count','Adapter',sl);
Add('</section>');
Add('<section name="Virtual Adapters">');
StringsToRep(VirtualAdapters,'Count','Adapter',sl);
Add('</section>');
```

```
Add('<section name="Protocols">');
StringsToRep(Protocols,'Count','Protocol',sl);
Add('</section>');
Add('<section name="Services">');
StringsToRep(Services,'Count','Service',sl);
Add('</section>');
Add('<section name="Clients">');
StringsToRep(Clients,'Count','Client',sl);
Add('</section>');
Add('<section name="IPAddresses">');
StringsToRep(IPAddresses,'Count','IPAddress',sl);
Add('</section>');
Add('<section name="MACAdresses">');
StringsToRep(MACAddresses,'Count','MACAddress',sl);
Add('</section>');
Add('</Network>');
Add('<Winsock classname="TDIPLOMsock">');
Add(Format('<data name="Description"
type="string">%s</data>',[CheckXMLValue(Winsock.Description)]));
Add(Format('<data name="Status"
type="string">%s</data>',[CheckXMLValue(Winsock.Status)]));
Add('</Winsock>');
TCPIP.Report(sl,False);
if Standalone then ReportFooter(sl);
end;
end;

procedure TNetwork.SetMode(const Value: TExceptionModes); // Встановлення режиму
begin
FModes := Value;
if Assigned(TCPIP) then
TCPIP.ExceptionModes:=FModes;
if Assigned(Winsock) then
Winsock.ExceptionModes:=FModes;
end;

{ TTCPIP }

procedure TTCPIP.ClearList; // Очищення списку
var
i: Integer;
p: PAdapter;
begin
for i:=0 to FAdapters.Count-1 do begin
p:=PAdapter(FAdapters.Objects[i]);
p^.IPAddress.Free;
p^.IPAddressMask.Free;
p^.Gateway_IPAddress.Free;
p^.Gateway_IPAddressMask.Free;
p^.DHCP_IPAddress.Free;
p^.DHCP_IPAddressMask.Free;
p^.PrimaryWINS_IPAddress.Free;
p^.PrimaryWINS_IPAddressMask.Free;
p^.SecondaryWINS_IPAddress.Free;
p^.SecondaryWINS_IPAddressMask.Free;
Dispose(p);
end;
FAdapters.Clear;
end;

constructor TTCPIP.Create;
begin
inherited;
FAdapters:=TStringList.Create;
FDNSList:=TStringList.Create;
FDNSSuffix:=TStringList.Create;
ExceptionModes:=[emExceptionStack];
end;
```

```
destructor TTCPIP.Destroy;
begin
ClearList;
FDNSList.Destroy;
FAdapters.Destroy;
FDNSSuffix.Destroy;
inherited;
end;

function TTCPIP.FindAdapter(AName: string): Integer;
begin
Result:=FAdapters.IndexOf(AName);
end;

function TTCPIP.GetAdapter(Index: Word): TAdapter;
begin
Result:=PAdapter(FAdapters.Objects[Index])^;
end;

function TTCPIP.GetAdapterCount: Word;
begin
Result:=FAdapters.Count;
end;

procedure TTCPIP.GetInfo; // Інфо
var
ai, aiInitPtr: PIP_ADAPTER_INFO;
lastip,ip: PIP_ADDR_STRING;
np: PFixedInfo;
r,j: dword;
Size: ulong;
A: PAdapter;
s: string;
Temp: string;
Suffix: string;
const
rkTCPIP =
{HKEY_LOCAL_MACHINE\}'SYSTEM\CurrentControlSet\Services\Tcpip\Parameters';
rvSL = 'SearchList';
begin
try
try
ClearList;
if InitIpHlpAPI then begin
size:=SizeOf(IP_ADAPTER_INFO);
aiInitPtr:=AllocMem(size);
try
r:=GetAdaptersInfo(aiInitPtr,size);
while(r=ERROR_BUFFER_OVERFLOW) do begin
 size:=Size+SizeOf(IP_ADAPTER_INFO);
 ReallocMem(aiInitPtr,size);
 r:=GetAdaptersInfo(aiInitPtr,size);
end;
ai:=aiInitPtr;
if(r=ERROR_SUCCESS) then begin
 while assigned(ai) do begin
   New(A);
   A^.IPAddress:=TStringList.Create;
   A^.IPAddressMask:=TStringList.Create;
   A^.Gateway_IPAddress:=TStringList.Create;
   A^.Gateway_IPAddressMask:=TStringList.Create;
   A^.DHCP_IPAddress:=TStringList.Create;
   A^.DHCP_IPAddressMask:=TStringList.Create;
   A^.PrimaryWINS_IPAddress:=TStringList.Create;
   A^.PrimaryWINS_IPAddressMask:=TStringList.Create;
   A^.SecondaryWINS_IPAddress:=TStringList.Create;
   A^.SecondaryWINS_IPAddressMask:=TStringList.Create;
   A^.Name:=Trim(string(ai^.Description));
   case ai^.Type_ of
```

```
      MIB_IF_OTHER_ADAPTERTYPE: A^.Typ:=atOther;
      MIB_IF_ETHERNET_ADAPTERTYPE: A^.Typ:=atEthernet;
      MIB_IF_TOKEN_RING_ADAPTERTYPE: A^.Typ:=atTokenRing;
      MIB_IF_FDDI_ADAPTERTYPE: A^.Typ:=atFDDI;
      MIB_IF_PPP_ADAPTERTYPE: A^.Typ:=atPPP;
      MIB_IF_LOOPBACK_ADAPTERTYPE: A^.Typ:=atLoopback;
      MIB_IF_SLIP_ADAPTERTYPE: A^.Typ:=atSlip;
    end;
    s:='';
    if ai^.AddressLength>0 then begin
      for j:=0 to ai^.AddressLength-1 do
        s:=s+Format('%2.2x:',[ai^.Address[j]]);
      SetLength(s,Length(s)-1);
    end;
    A^.Address:=s;
    A^.EnableDHCP:=Boolean(ai^.DhcpEnabled);
    A^.HaveWINS:=Boolean(ai^.HaveWins);

    ip:=@(ai^.IpAddressList);
    repeat
      lastip:=ip;
      A^.IPAddress.Add(string(ip^.IpAddress.s));
      A^.IPAddressMask.Add(string(ip^.IpMask.s));
      ip:=ip.Next;
    until not Assigned(ip) or (lastip=ip);
    ip:=@(ai^.GatewayList);
    repeat
      lastip:=ip;
      A^.Gateway_IPAddress.Add(string(ip^.IpAddress.s));
      A^.Gateway_IPAddressMask.Add(string(ip^.IpMask.s));
      ip:=ip.Next;
    until not Assigned(ip) or (lastip=ip);
    ip:=@(ai^.DhcpServer);
    repeat
      lastip:=ip;
      A^.DHCP_IPAddress.Add(string(ip^.IpAddress.s));
      A^.DHCP_IPAddressMask.Add(string(ip^.IpMask.s));
      ip:=ip.Next;
    until not Assigned(ip) or (lastip=ip);
    ip:=@(ai^.PrimaryWinsServer);
    repeat
      lastip:=ip;
      A^.PrimaryWINS_IPAddress.Add(string(ip^.IpAddress.s));
      A^.PrimaryWINS_IPAddressMask.Add(string(ip^.IpMask.s));
      ip:=ip.Next;
    until not Assigned(ip) or (lastip=ip);
    ip:=@(ai^.SecondaryWinsServer);
    repeat
      lastip:=ip;
      A^.SecondaryWINS_IPAddress.Add(string(ip^.IpAddress.s));
      A^.SecondaryWINS_IPAddressMask.Add(string(ip^.IpMask.s));
      ip:=ip.Next;
    until not Assigned(ip) or (lastip=ip);
    FAdapters.AddObject(A^.Name,TObject(@A^));
    ai:=ai.Next;
  end;
end;
finally
if Assigned(aiInitPtr) then
  FreeMem(aiInitPtr);
end;
Size:=SizeOf(TFixedInfo);
np:=Allocmem(size);
try
r:=GetNetworkparams(np,Size);
while r=ERROR_BUFFER_OVERFLOW do begin
  Reallocmem(np,size);
  r:=GetNetworkparams(np,Size);
end;
```

```
if r=ERROR_SUCCESS then begin
 case np^.NodeType of
   BROADCAST_NODETYPE: FNode:=ntBroadcast;
   PEER_TO_PEER_NODETYPE: FNode:=ntPeerToPeer;
   MIXED_NODETYPE: FNode:=ntMixed;
   HYBRID_NODETYPE: FNode:=ntHybrid;
   else FNode:=ntUnknown;
 end;
 FDHCPScope:=string(np^.ScopeId);
 if Assigned(np^.CurrentDnsServer) then
   FDNSPrefix:=string(np^.CurrentDnsServer.IpAddress.S)
 else
   FDNSPrefix:='';
 ip:=@(np^.DnsServerList);
 FDNSList.Clear;
 repeat
   Temp:=string(ip^.IpAddress.s);
   FDNSList.Add(Temp);
   ip:=ip.Next;
 until (not Assigned(ip)) or (Temp=string(ip^.IpAddress.s));
 FDNSSuffix.Clear;
 Suffix:=ReadRegistryString(HKEY_LOCAL_MACHINE,rkTCPIP,rvSL);
 while Pos(',',Suffix) <> 0 do begin
  FDNSSuffix.Add(Copy(Suffix,1,Pos(',',Suffix)-1));
  Delete(Suffix,1,Pos(',',Suffix));
 end;
 FDNSSuffix.Add(Suffix);
 FHost:=np^.HostName;
 FDomain:=np^.DomainName;
 FProxy:=Boolean(np^.EnableProxy);
 FRouting:=Boolean(np^.EnableRouting);
 FDNS:=Boolean(np^.EnableDns);
end;
finally
if Assigned(np) then
 FreeMem(np);
end;
end;
except
end;
finally
end;
end;

procedure TTCPIP.Report; // Звіт
var
j: Integer;
begin
with sl do begin
if Standalone then ReportHeader(sl);
Add('<TCPIP classname="TTCPIP">');
Add(Format('<data name="HostName"
type="string">%s</data>',[CheckXMLValue(HostName)]));
Add(Format('<data name="DomainName"
type="string">%s</data>',[CheckXMLValue(DomainName)]));
Add(Format('<data name="NodeType"
type="string">%s</data>',[NodeTypes[NodeType]]));
Add(Format('<data name="PrimaryDNSSuffix"
type="string">%s</data>',[CheckXMLValue(PrimaryDNSSuffix)]));
Add(Format('<data name="DNSServers"
type="string">%s</data>',[CheckXMLValue(DNSServers.CommaText)]));
for j:=0 to AdapterCount-1 do begin
Add(Format('<section name="Adapter_%d">',[j]));
with Adapter[j] do begin
 Add(Format('<data name="Name"
type="string">%s</data>',[CheckXMLValue(Adapter[j].Name)]));
 Add(Format('<data name="PhysicalAddress"
type="string">%s</data>',[Adapter[j].Address]));
```

```
 Add(Format('<data name="Type"
type="string">%s</data>',[AdapterTypes[Adapter[j].Typ]]));
 Add(Format('<data name="IPAddress"
type="string">%s</data>',[Adapter[j].IPAddress.CommaText]));
 Add(Format('<data name="IPMask"
type="string">%s</data>',[Adapter[j].IPAddressMask.CommaText]));
 Add(Format('<data name="Gateway_IPAddress"
type="string">%s</data>',[Adapter[j].Gateway_IPAddress.CommaText]));
 Add(Format('<data name="Gateway_IPMask"
type="string">%s</data>',[Adapter[j].Gateway_IPAddressMask.CommaText]));
 Add(Format('<data name="DHCP_IPAddress"
type="string">%s</data>',[Adapter[j].DHCP_IPAddress.CommaText]));
 Add(Format('<data name="DHCP_IPMask"
type="string">%s</data>',[Adapter[j].DHCP_IPAddressMask.CommaText]));
 Add(Format('<data name="PrimaryWINS_IPAddress"
type="string">%s</data>',[Adapter[j].PrimaryWINS_IPAddress.CommaText]));
 Add(Format('<data name="PrimaryWINS_IPMask"
type="string">%s</data>',[Adapter[j].PrimaryWINS_IPAddressMask.CommaText]));
 Add(Format('<data name="SecondaryWINS_IPAddress"
type="string">%s</data>',[Adapter[j].SecondaryWINS_IPAddress.CommaText]));
 Add(Format('<data name="SecondaryWINS_IPMask"
type="string">%s</data>',[Adapter[j].SecondaryWINS_IPAddressMask.CommaText]));
end;
Add('</section>');
end;
Add('</TCPIP>');
if Standalone then ReportFooter(sl);
end;
end;
procedure TTCPIP.SetMode(const Value: TExceptionModes);
begin
FModes := Value;
end;
end.
```

Модуль обробки даних (Form3.pas)

```
unit Unit3; // модуль 3

// Copyriht @ Дворцов Нікіта Романович, 2024

interface

uses
  Windows, Messages, Sysutils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Stdctrls, Winsock, Extctrls, Math;

type
  Tform1 = class(Tform)
    Button1: Tbutton;
    Label1: Tlabel;
    Edit1: Tedit;
    Label2: Tlabel;
    Button2: Tbutton;
    Bevel1: Tbevel;
    Edit2: Tedit;
    Label3: Tlabel;
    Label4: Tlabel;
    Bevel2: Tbevel;
    Button3: Tbutton;
    Edit3: Tedit;
    Label5: Tlabel;
    Label6: Tlabel;
    Bevel3: Tbevel;
    Label7: Tlabel;
    Bevel4: Tbevel;
    Button4: Tbutton;
    Edit4: Tedit;
    Label8: Tlabel;
    Label9: Tlabel;
    Button5: Tbutton;
    Edit5: Tedit;
    Label10: Tlabel;
    Label11: Tlabel;
    Bevel5: Tbevel;
    Timer1: Ttimer;
    Listbox1: Tlistbox;
    Button6: Tbutton;
    Label12: Tlabel;
    Label13: Tlabel;
    procedure Button1Click(Sender: Tobject);
    procedure Button2Click(Sender: Tobject);
    procedure Label4Dblclick(Sender: Tobject);
    procedure Label1Dblclick(Sender: Tobject);
    procedure Button3Click(Sender: Tobject);
    procedure Formcreate(Sender: Tobject);
    procedure Button4Click(Sender: Tobject);
    procedure Label6Dblclick(Sender: Tobject);
    procedure Label9Dblclick(Sender: Tobject);
    procedure Button5Click(Sender: Tobject);
    procedure Label11Dblclick(Sender: Tobject);
    procedure Timer1Timer(Sender: Tobject);
    procedure Button6Click(Sender: Tobject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: Tform1;
  stop_traf: boolean;
  count,trafbitin,trafbitout,trafbitold: integer;
```

```
implementation

{$R *.dfm}


function Ipaddrtoname(Ipaddr: string): string;
var
 Sockaddrin: Tsockaddrin;
 Hostent: Phostent;
 Wsadata: Twsadata;
begin
 Wsastartup($101, Wsadata);
 Sockaddrin.sin_addr.s_addr:=inet_addr(Pchar(Ipaddr));
 Hostent:=Gethostbyaddr(@Sockaddrin.sin_addr.S_addr, 4, AF_INET);
 if Hostent<>nil
 then Result:=Strpas(Hostent.h_name)
 else Result:='';
end;

procedure Tform1.Button1Click(Sender: Tobject);
// Обробник 1 кнопки
begin
 Label1.Caption:='Name: '+Ipaddrtoname(Edit1.Text);
end;


procedure Tform1.Button2Click(Sender: Tobject);
// Обробник 2 кнопки
var
 // Зберігаємо оригінальне значення IP адреси
 Orgval: string;
 // частини оригінального IP
 O1,O2,O3,O4: string;
H1,H2,H3,H4: string;
 // Тут будуть зібрані всі 16х частини
 Hexip: string;
 XN: array[1..8] of Extended;
 Flt1: Extended;
 Xc: Integer;
begin
 // Зберігаємо у зворотному порядку
 Xn[8]:=Intpower(16,0);Xn[7]:=Intpower(16,1);
Xn[6]:=Intpower(16,2);Xn[5]:=Intpower(16,3);
 Xn[4]:=Intpower(16,4);Xn[3]:=Intpower(16,5);
Xn[2]:=Intpower(16,6);Xn[1]:=Intpower(16,7);
 // Зберігаємо оригінальний IP адрес
 Orgval:=Edit2.Text;
 O1:=Copy(Orgval,1,Pos('.',Orgval)-1);Delete(Orgval,1,Pos('.',Orgval));
 O2:=Copy(Orgval,1,Pos('.',Orgval)-1);Delete(Orgval,1,Pos('.',Orgval));
 O3:=Copy(Orgval,1,Pos('.',Orgval)-1);Delete(Orgval,1,Pos('.',Orgval));
 O4:=Orgval;
 H1:=Inttohex(Strtoint(O1),2);H2:=Inttohex(Strtoint(O2),2);
 H3:=Inttohex(Strtoint(O3),2);H4:=Inttohex(Strtoint(O4),2);
 // Одержуємо 16х значення IP адреси
 Hexip:=H1+H2+H3+H4;
 // Перетворимо це велике 16х значення в змінну Float
 Flt1:=0;
 for Xc:=1 to 8 do
  begin
    case Hexip[Xc] of
     '0'..'9': Flt1:=Flt1+(Strtoint(Hexip[XC])*Xn[Xc]);
     'A': Flt1:=Flt1+(10*Xn[Xc]);
     'B': Flt1:=Flt1+(11*Xn[Xc]);
     'C': Flt1:=Flt1+(12*Xn[Xc]);
     'D': Flt1:=Flt1+(13*Xn[Xc]);
     'E': Flt1:=Flt1+(14*Xn[Xc]);
     'F': Flt1:=Flt1+(15*Xn[Xc]);
    end;
```

```
  end;
 Label4.Caption:='Number: '+Floattostr(Flt1);
end;


procedure Tform1.Label4Dblclick(Sender: Tobject);
// Обробник 4 кнопки
begin
 Edit2.Text:=Label4.Caption;
end;

procedure Tform1.Label1Dblclick(Sender: Tobject);
// Обробник 1 кнопки
begin
 Edit1.Text:=Label1.Caption;
end;

const
  WINSOCK_VERSION=$0101;

procedure Tform1.Button3Click(Sender: Tobject);
// Обробник 3 кнопки
var
 Wsadata: Twsadata;
 p: Phostent;
begin
 Wsastartup(WINSOCK_VERSION, Wsadata);
 p:=Gethostbyname(Pchar(Edit3.Text));
 Label6.Caption:='IP: '+inet_ntoa(Pinaddr(p.h_addr_list));
 Wsacleanup;
end;

// повертає IP адресу
function Localip: string;
type
 Tapinaddr=array [0..10] of Pinaddr;
 Papinaddr=Tapinaddr;
var
 phe:Phostent;
 pptr:Papinaddr;
 Buffer:array [0..63] of char;
 i:Integer;
 Ginitdata:TWSADATA;
begin
 Wsastartup($101, Ginitdata);
 Result:='';
 Gethostname(Buffer, Sizeof(Buffer));
 phe:=Gethostbyname(buffer);
 if phe=nil then Exit;
 pptr:=Papinaddr(Phe.h_addr_list);
 i:=0;
 while pptr[i]<>nil do
  begin
   result:=Strpas(inet_ntoa(pptr[i]));
   Inc(i);
  end;
 Wsacleanup;
end;

procedure Tform1.Formcreate(Sender: Tobject);
// Обробник створення форми
begin
 Label7.Caption:='Local IP: '+Localip;
end;

procedure Tform1.Button4Click(Sender: Tobject);
// Обробник 4 кнопки
var
 wsdata: Twsadata;
```

```
 hostname: array [0..255] of char;
 hostent: Phostent;
 addr: Pchar;
begin
 Wsastartup ($0101, wsdata);
  try
   Gethostname(hostname, sizeof (hostname));
   Strpcopy(hostname,Edit4.Text);
   hostent:=Gethostbyname(hostname);
   if Assigned(hostent)
   then
    if Assigned(hostent.h_addr_list)
    then
     begin
      addr:=hostent.h_addr_list;
      if Assigned(addr)
      then
       begin
        Label9.Caption:=Format('%d.% d.% d.% d',[byte(addr[0]),
        byte(addr[1]),byte(addr[2]),byte(addr[3])]);
       end;
     end;
  finally
   Wsacleanup;
  end;
end;

procedure Tform1.Label6Dblclick(Sender: Tobject);
// Обробник 6 кнопки
begin
 Edit3.Text:=Label6.Caption;
end;

procedure Tform1.Label9Dblclick(Sender: Tobject);
// Обробник 9 кнопки
begin
 Edit4.Text:=Label9.Caption;
end;

function Ipaddrtocompname(Ipaddr: string): string;
var
 Sockaddrin: Tsockaddrin;
 Hostent: Phostent;
 Wsadata: Twsadata;
begin
 Wsastartup($101, Wsadata);
 Sockaddrin.sin_addr.s_addr:=inet_addr(Pchar(Ipaddr));
 Hostent:=gethostbyaddr(@Sockaddrin.sin_addr.S_addr, 4, AF_INET);
 if Hostent<>nil
 then Result:=Strpas(Hostent.h_name)
 else Result:='';
end;

procedure Tform1.Button5Click(Sender: Tobject);
// Обробник 5 кнопки
begin
 Label11.Caption:='Name: '+Ipaddrtocompname(Edit5.Text);
end;

procedure Tform1.Label11Dblclick(Sender: Tobject);\
// Обробник 11 кнопки
begin
 Edit5.Text:=Label11.Caption;
end;

type
 Tmibifrow = packed record
  wszname           : array[0..255] of Widechar;
  dwindex           : DWORD;
```

```
  dwtype           : DWORD;
  dwmtu            : DWORD;
  dwspeed          : DWORD;
// визначає поточну швидкість передачі в бітах у секунду
  dwphysaddrlen    : DWORD;
  bphysaddr        : array[0..7] of Byte;
// містить фізичну адресу інтерфейсу
  dwadminstatus    : DWORD;
  dwoperstatus     : DWORD;
  dwlastchange     : DWORD;
  dwinoctets       : DWORD;
// містить кількість байт прийнятих через інтерфейс
  dwinucastpkts    : DWORD;
  dwinnucastpkts   : DWORD;
  dwindiscards     : DWORD;
  dwinerrors       : DWORD;
  dwinunknownprotos: DWORD;
  dwoutoctets      : DWORD;
// містить кількість байт відправлених інтерфейсом
  dwoutucastpkts   : DWORD;
  dwoutnucastpkts  : DWORD;
  dwoutdiscards    : DWORD;
  dwouterrors      : DWORD;
  dwoutqlen        : DWORD;
  dwdescrlen       : DWORD;
  bdescr           : array[0..255] of Char;
// містить опис інтерфейсу
 end;
 Tmibifarray = array [0..512] of Tmibifrow;
 Pmibifrow = Tmibifrow;
 Pmibifarray = Tmibifarray;

type
 Tmibiftable = packed record
   dwnumentries: DWORD;
   Table       : Tmibifarray;
 end;
 Pmibiftable = Tmibiftable;

var
 Getiftable:function(piftable: Pmibiftable; pdwsize: PULONG;
                     border: Boolean): DWORD; stdcall;

// Інтерфейси

function Wsaioctl(s: Tsocket; cmd: DWORD; lpinbuffer: PCHAR; dwinbufferlen:
 DWORD;
 lpoutbuffer: PCHAR; dwoutbufferlen: DWORD;
 lpdwoutbytesreturned: LPDWORD;
 lpoverlapped: POINTER;
 lpoverlappedroutine: POINTER): integer; stdcall; external 'WS2_32.DLL';

const
 SIO_GET_INTERFACE_LIST = $4004747F;
 IFF_UP = $00000001;
 IFF_BROADCAST = $00000002;
 IFF_LOOPBACK = $00000004;
 IFF_POINTTOPOINT = $00000008;
 IFF_MULTICAST = $00000010;

type
 sockaddr_gen = packed record
  Addressin: sockaddr_in;
  filler: packed array [0..7] of char;
end;

type
 INTERFACE_INFO = packed record
  iiflags: u_long; // Прапори інтерфейсу
```

```
  iiaddress: sockaddr_gen; // Адреса інтерфейсу
  iibroadcastaddress: sockaddr_gen; // Broadcast адреса
  iinetmask: sockaddr_gen; // Маска подмережі
end;

function Enuminterfaces(var sint: string): Boolean;
var
 s: Tsocket;
 wsad: WSADATA;
 Numinterfaces: Integer;
 Bytesreturned: u_long;
 paddrinet: SOCKADDR_IN;
 paddrstring: Pchar;
 Ptra: pointer;
 Buffer: array[0..20] of INTERFACE_INFO;
 i: integer;
begin
 result:=true; // ініціалізуємо змінну
 sint:='';
 Wsastartup($0101, wsad); // запускаємо Winsock
s:=Socket(AF_INET, SOCK_STREAM, 0); // відкриваємо сокет
 if (s=INVALID_SOCKET)
 then Exit;
 try // викликаємо Wsaioctl
  Ptra:=@bytesreturned;
  if (Wsaioctl(s, SIO_GET_INTERFACE_LIST, nil, 0, @Buffer,
                          1024, Ptra, nil, nil)<>SOCKET_ERROR)
  then
   begin // якщо OK, те визначаємо кількість існуючих інтерфейсів
     Numinterfaces:=Bytesreturned div Sizeof(INTERFACE_INFO);
     for i:=0 to Numinterfaces-1 do // для кожного інтерфейсу
      begin
       paddrinet:=Buffer[i].iiaddress.Addressin; // IP адреса
       paddrstring:=inet_ntoa(paddrinet.sin_addr);
       if paddrstring<>'127.0.0.1'
       then
        begin
         sint:=sint+'IP = '+paddrstring+', '+#10#13;
         paddrinet:=Buffer[i].iinetmask.Addressin; // маска подмережі
         paddrstring:=inet_ntoa(paddrinet.sin_addr);
         sint:=sint+' Mask='+paddrstring+',';
        end
       else sint:='IP = "localhost"';
      end;
  end;
 except
end;
 // закриваємо сокети
 Closesocket(s);
 Wsacleanup;
 result:=false;
end;

function Bytestostring(Value: integer): string;
const
 Onekb=1024;
 Onemb=Onekb*1024;
 Onegb=Onemb*1024;
begin
 if Value<Onekb
 then Result:=Formatfloat('#,##0.00 B',Value)
 else
  if Value<Onemb
  then Result:=Formatfloat('#,##0.00 KB', Value/Onekb)
  else
   if Value<Onegb
   then Result:=Formatfloat('#,##0.00 MB', Value/Onemb)
end;
```

```
procedure Tform1.Timer1Timer(Sender: Tobject);
// Обробник спрацювання 1 таймеру
type
 TMAC=array [0..7] of Byte;

function Getmac(Value: TMAC; Length: DWORD): string;
var
 i: integer;
begin
 if Length=0
 then Result:='00-00-00-00-00-00'
 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;

var
 Flibhandle: Thandle;
 Table: Tmibiftable;
 i, Size: integer;
 s,trafnormin,trafnormout: string;
begin
 Timer1.Enabled:=false;
 Listbox1.Items.Beginupdate;
 Listbox1.Items.Clear; // очищаємо список
 Flibhandle:=Loadlibrary('IPHLPAPI.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 Listbox1.Items do
     begin // виводимо результати
 if string(Getmac(TMAC(Table.Table[i].bphysaddr),
Table.Table[i].dwphysaddrlen)) <>'00-00-00-00-00-00'
// порівняння MAC адреси
      then
       begin
Add('Description: '+string(Table.Table[i].bdescr));
// найменування інтерфейсу
Add('Mac-adress:'+string(Getmac(TMAC(Table.Table[i].bphysaddr),
Table.Table[i].dwphysaddrlen))); // MAC адреса
        trafbitin:=Table.Table[i].dwinoctets;
        trafnormin:=Bytestostring(trafbitin);
        trafbitout:=Table.Table[i].dwoutoctets;
// усього відправлено байт
        trafnormout:=Bytestostring(trafbitout);
        if stop_traf=true
        then
         begin
          trafbitold:=trafbitin;
          trafnormin:='0,00 B';
          trafnormout:='0,00 B';
         end;
```

```
        if trafbitin>=trafbitold
        then
         begin
          trafbitin:=trafbitin-trafbitold;
          trafnormin:=Bytestostring(trafbitin);
         end
        else
         begin
          trafbitin:=trafbitold;
          trafnormin:=Bytestostring(trafbitin);
         end;
        Add('In (Byte): '+trafnormin); // усього прийнято байт
        Add('Out (Byte): '+trafnormout);
// усього відправлено байт
        Add('-------------------------------------------------');
       end;
     end;
   end;
 //
 Enuminterfaces(s);
 Listbox1.Items.Add(s);
 Listbox1.Items.Endupdate;
 Freelibrary(Flibhandle);
 Timer1.Enabled:=true;
end;

procedure Tform1.Button6Click(Sender: Tobject);
begin
 if stop_traf=false then stop_traf:=true
 else stop_traf:=false;
end;

end.
```

Проект розробленого ПЗ (Dvortsov_Diplom_Project.dpr)

```
program Dvortsov_Diplom_Project; // Назва програми

// Copyriht @ Дворцов Нікіта Романович, 2024

Uses // Підключаємо модулі та бібліотеки
  Forms,
  Unit1 in 'Unit1.pas' {Form1},
  Unit2 in 'Unit2.pas' {Form2},
  Unit3 in 'Unit3.pas' {Form3};
  Unit4 in 'Unit4.pas' {Form4};
  Unit5 in 'Unit5.pas' {Form4};

{$R *.RES} // Використовуємо стандартний файл ресурсів

begin
  Application.Initialize; // Ініціалізація мого проекту
   // Фігуруєма назва у верхнюому правому кутку головної форми
  Application.Title := 'Dvortsov_Diplom_Project';
  Application.CreateForm(TForm1, Form1); // Створення форми 1
  Application.CreateForm(TForm2, Form2); // Створення форми 2
  Application.CreateForm(TForm3, Form3); // Створення форми 3
  Application.Run; // Початок роботи
end.
```

Взаємодія серверної частини з клієнтською (Form2.pas)

```
unit Unit2; // модуль 2

// Copyriht @ Дворцов Нікіта Романович, 2024

{$DEFINE UA}

interface

uses
Windows, Messages, Sysutils, Classes, Graphics, Controls, Forms,
Dialogs, Stdctrls, Comctrls, Commctrl, Winsock;

const
{$IFDEF UA}
// Якщо при компіляції встановлено мову Українську
// використовуємо наступні ресурси
  RES_UNKNOWN = 'Невідомо';
  RES_IP      = 'IP адреса: ';
  RES_CMP     = 'Ім'я комп'ютера: ';
  RES_USR     = 'Ім'я користувача: ';
  RES_DOM     = 'Домен: ';
  RES_SER     = 'Сервер домена: ';
  RES_COM     = 'Коментар:';
  RES_GRP     = 'Групи: ';
  RES_MAC     = 'MAC адреса: ';
  RES_SHARES  = 'Доступні ресурси: ';
  RES_TIME    = 'Час: ';
  RES_COM_NO  = 'Відсутній';
{$ELSE}
// Якщо Англійську то наступні
  RES_UNKNOWN = 'Unknown';
  RES_IP      = 'IP adress: ';
  RES_CMP     = 'Computer name: ';
  RES_USR     = 'User name: ';
  RES_DOM     = 'Domen: ';
  RES_SER     = 'Domen server: ';
  RES_COM     = 'Comment: ';
   RES_GRP     = 'Groups: ';
  RES_MAC     = 'MAC adress: ';
  RES_SHARES  = 'Available shares: ';
  RES_TIME    = 'Expended time: ';
  RES_COM_NO  = 'Absent';
{$ENDIF}

WSA_TYPE = $101; //чи можливо встановлення $202;

// Для роботи з ARP (Address Resolution Protocol) таблицею
IPHLPAPI = 'IPHLPAPI.DLL';
MAX_ADAPTER_ADDRESS_LENGTH = 7;

type

LMSTR = LPWSTR;
NET_API_STATUS = DWORD;

// Наступні три типи використовуються для роботи з Iphlpapi.dll

// Так буде виглядати MAC
Tmacaddress = array[0..MAX_ADAPTER_ADDRESS_LENGTH] of byte;

// Це структура для одиничного запиту
Tmibipnetrow = packed record
  dwindex         : DWORD;
  dwphysaddrlen   : DWORD;
  bphysaddr       : Tmacaddress;  // тут  лежить MAC адрес
  dwaddr          : DWORD;
  dwtype          : DWORD;
```

```
end;

// виділення пам'яті динамічно

Tmibipnetrowarray = array [0..512] of Tmibipnetrow;

// запитувана структура
Ptmibipnettable = Tmibipnettable;
Tmibipnettable = packed record
  dwnumentries    : DWORD;
  Table: Tmibipnetrowarray;
end;

// Структура для перерахування користувачів
_WKSTA_USER_INFO_1 = record
  wkui1_username: LPWSTR;
  wkui1_logon_domain: LPWSTR;
  wkui1_oth_domains: LPWSTR;
  wkui1_logon_server: LPWSTR;
end;
WKSTA_USER_INFO_1 = _WKSTA_USER_INFO_1;
PWKSTA_USER_INFO_1 = _WKSTA_USER_INFO_1;
LPWKSTA_USER_INFO_1 = _WKSTA_USER_INFO_1;

// Структура для визначення приналежності користувача до груп
Pgroupusersinfo0 = _GROUP_USERS_INFO_0;
_GROUP_USERS_INFO_0 = packed record
  grui0_name: LPWSTR;
end;
Tgroupusersinfo0 = _GROUP_USERS_INFO_0;
GROUP_USERS_INFO_0 = _GROUP_USERS_INFO_0;

// Структура для визначення доступних мережних ресурсів
PSHARE_INFO_1 = SHARE_INFO_1;
_SHARE_INFO_1 = record
  shi1_netname: LMSTR;
  shi1_type: DWORD;
  shi1_remark: LMSTR;
end;
SHARE_INFO_1 = _SHARE_INFO_1;
Tshareinfo1 = SHARE_INFO_1;
Pshareinfo1 = PSHARE_INFO_1;

TF_Lanform = class(TF_LAN)
  btngetinfo: Tbutton;
  meminfo: Tmemo;
  Label1: Tlabel;
  procedure btngetinfoclick(Sender: Tobject);
  procedure Formcreate(Sender: Tobject);
private
  IP, Font: Integer;     // змінні для роботи з
  edip: HWND;          // WC_IPADDRESS класом
  function Getnamefromip(const IP: String): String;
  function Getusers(const Compname: String): String;
  function Getdomain(const Compname, Provider: String): String;
  function Getcomment(Compname, Provider: String): String;
  function Getprovider(const Compname: String): String;
  function Getmacfromip(const IP: String): String;
  function Getdomainserver(const Domainname: String): String;
  function Getgroups(Domainserver: String; Username: String): String;
  function Getshares(const Compname: String): String;
end;

// Тут іде статичне завантаження бібліотек

{$EXTERNALSYM Wnetgetresourceinformation}
function Wnetgetresourceinformation(lpnetresource: Pnetresource;
  lpbuffer: Pointer; var lpcbbuffer: DWORD; lplpsystem: Pointer): DWORD;
stdcall;
```

```
{$EXTERNALSYM Getipnettable}
function Getipnettable(pipnettable: Ptmibipnettable;
  pdwsize: PULONG; border: Boolean): DWORD; stdcall;

function Wnetgetresourceinformation; external mpr name
'Wnetgetresourceinformationa';
function Getipnettable; external IPHLPAPI name 'Getipnettable';

function Netgetanydcname(servername: LPCWSTR;  domainname: LPCWSTR;
  bufptr: Pointer): Cardinal;
  stdcall; external 'netapi32.dll';

function Netshareenum(servername: LMSTR; level: DWORD; var bufptr: Pointer;
  prefmaxlen: DWORD; entriesread, totalentries,
  resume_handle: LPDWORD): NET_API_STATUS; stdcall; external 'Netapi32.dll';

function Netapibufferfree(buffer: Pointer): Cardinal;
  stdcall; external 'netapi32.dll';

function Netwkstauserenum(Servername: LPCWSTR;
                        Level: DWORD;
                        Bufptr: Pointer;
                        Prefmaxlen: DWORD;
                        Entriesread: LPDWORD;
                        Totalentries: LPDWORD;
                        Resumehandle: LPDWORD): Longint; stdcall; external
                        'netapi32.dll';

function Netusergetgroups(Servername: LPCWSTR;
                        Username: LPCWSTR;
                        level: DWORD;
                        bufptr: Pointer;
                        prefmaxlen: DWORD;
                        var entriesread: DWORD;
                        var totalentries: DWORD): Longint; stdcall;
                        external 'netapi32.dll';

var
Mainform: TF_Lanform;

implementation

{$R *.dfm}

// Для введення IP адреси будемо використовувати клас WC_IPADDRESS
procedure TF_Lanform.Formcreate(Sender: Tobject);
begin
// Задамо IP адресу
IP := MAKEIPADDRESS(192, 168, 2, 108);
// Ініціалізуємо додаткові класи бібліотеки Comctl32.dll.
Initcommoncontrol(ICC_INTERNET_CLASSES);
// Створимо віконце
edip:= Createwindow(WC_IPADDRESS, nil, WS_CHILD or WS_VISIBLE,
  6, 16, Mainform.Width-22, 21, Mainform.Handle, 0, hinstance, nil);
// Укажемо йому який IP показувати
Sendmessage(edip, IPM_SETADDRESS, 0, IP);
Font := Createfont(-11, 0, 0, 0, 400, 0, 0, 0, DEFAULT_CHARSET,
  OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
  DEFAULT_PITCH or FF_DONTCARE, 'MS Sans Serif');
Sendmessage(edip, WM_SETFONT, Font, 0);
end;

procedure TF_Lanform.btngetinfoclick(Sender: Tobject);
var
Tmpcompname, Tmpprovider, Tmpgroup, Tmpuser, Tmpserver: String;
Time: Cardinal;
Ipstr: String;
begin
Time := Gettickcount;  // Засікаємо час
```

```
Sendmessage(edip, IPM_GETADDRESS, 0, Longint(PDWORD(@IP)));

// Перетворення типів "Dotted IP"
Ipstr := Inttostr(FIRST_IPADDRESS(IP));
Ipstr := Ipstr + '.' + Inttostr(SECOND_IPADDRESS(IP));
Ipstr := Ipstr + '.' + Inttostr(THIRD_IPADDRESS(IP));
Ipstr := Ipstr + '.' + Inttostr(FOURTH_IPADDRESS(IP));

with meminfo, meminfo.Lines do
// Виведення інформації
begin
  Clear;
// Очищаємо екран
  Refresh;
// обновляємо
  Add(RES_IP + Ipstr);
// Виводимо IP адресу
  Tmpcompname := Getnamefromip(Ipstr);
  if Tmpcompname = RES_UNKNOWN then Exit;
  Add(RES_CMP + Tmpcompname);
// Виведення ім'я комп'ютера
  Tmpuser := Getusers(Ipstr);
  Add(RES_USR + Tmpuser);
// Виведення ім'я користувача
  Tmpprovider := Getprovider(Tmpcompname);
  Add(RES_PROV + Tmpprovider);
  Add(RES_COM + Getcomment(Tmpcompname,
    Tmpprovider));
// Виводимо коментар до ресурсу
  Tmpgroup := Getdomain(Tmpcompname, Tmpprovider);
  Add(RES_DOM + Tmpgroup);
// Виводимо групу
  Tmpserver := Getdomainserver(Tmpgroup);
  if Tmpserver <> '' then
  begin
    Add(RES_SER + Tmpserver);
// Виведення ім'я сервера
    Add(RES_GRP + Getgroups(Tmpserver, Tmpuser));
// Виведення групи домена в які входить користувач
  end;
  Add(RES_SHARES + Getshares(Tmpcompname));
// Виведення список доступних ресурсів
  Add(RES_MAC + Getmacfromip(Ipstr));
// Виводимо MAC адресу
  Add(RES_TIME + Inttostr(Gettickcount - Time));
 // Скільки часу витрачене
end;
end;

function TF_Lanform.Getnamefromip(const IP: String): String;
var
WSA: Twsadata;
Host: Phostent;
Addr: Integer;
Err: Integer;
begin
Result := RES_UNKNOWN;
Err := Wsastartup(WSA_TYPE, WSA);
if Err <> 0 then
begin
  Showmessage(Syserrormessage(Getlasterror));
  Exit;
end;
try
  Addr := inet_addr(Pchar(IP));
  if Addr = INADDR_NONE then
  begin
    Showmessage(Syserrormessage(Getlasterror));
```

```
    Wsacleanup;
    Exit;
  end;
  Host := gethostbyaddr(@Addr, Sizeof(Addr), PF_INET);
  if Assigned(Host) then  // перевірка
    Result := Host.h_name
  else
    Showmessage(Syserrormessage(Getlasterror));
finally
  Wsacleanup;
end;
end;

// Перераховуємо всіх на машині користувачів
function TF_Lanform.Getusers(const Compname: String): String;
var
Buffer, tmpbuffer: Pointer;
Prefmaxlen       : DWORD;
Resume_Handle    : DWORD;
Entriesread      : DWORD;
Totalentries     : DWORD;
I, Size          : Integer;
Psrvr            : Pwidechar;
begin
Psrvr := nil;
try
  // Переводимо ім'я комп'ютера типу Pwidechar
  Size := Length(Compname);
  Getmem(Psrvr, Size * Sizeof(Widechar) + 1);
  Stringtowidechar(Compname, Psrvr, Size + 1);

  Prefmaxlen := DWORD(-1);
  Entriesread := 0;
  Totalentries := 0;
  Resume_Handle := 0;
  Buffer := nil;

  // Одержуємо список користувачів на комп'ютері з Psrvr
  if Netwkstauserenum( Psrvr, 1, @Buffer, Prefmaxlen, @Entriesread,
@Totalentries, @Resume_Handle) = S_OK then
  begin
    tmpbuffer := Pointer(DWORD(Buffer) + Sizeof(WKSTA_USER_INFO_1));
    for I := 1 to Totalentries - 1 do
    begin
Result := Result + WKSTA_USER_INFO_1(tmpbuffer).wkui1_username + ', ';
tmpbuffer := Pointer(DWORD(tmpbuffer) + Sizeof(WKSTA_USER_INFO_1));
    end;
    Result := Copy(Result, 1, Length(Result) - 2);
end
else
  Showmessage(Syserrormessage(Getlasterror));
finally
  Netapibufferfree(Buffer);
  Freemem(Psrvr);
end;
end;

// робимо рекурсивне сканування ресурсів

function TF_Lanform.Getcomment(Compname, Provider: String): String;
var
Stopscan: Boolean;
Tmpres: Tnetresource;

// Процедура сканування
procedure Scan(Res: Tnetresource; Root: boolean);
var
  Enum, I: Cardinal;
  Scanres: array [0..512] of Tnetresource;
```

```
  Size, Entries, Err: DWORD;
begin

  if Stopscan then Exit; // Використовуємо прапор для виходу з рекурсії

  if Root = True then
    Err := Wnetopenenum(RESOURCE_GLOBALNET, RESOURCETYPE_DISK,
      0, nil, Enum)
  else
    Err := Wnetopenenum(RESOURCE_GLOBALNET, RESOURCETYPE_DISK,
      0, @Res, Enum);

  if Err = NO_ERROR then
  begin
    Size := Sizeof(Scanres);
    Entries := DWORD(-1);
    Err := Wnetenumresource(Enum, Entries, @Scanres, Size);
    if Err = NO_ERROR then
    try
      for I := 0 to Entries - 1 do
      begin
        if Stopscan then Exit;
        with Scanres[i] do
        begin
          if dwdisplaytype = RESOURCEDISPLAYTYPE_SERVER then
            if lpremotename = Compname then
            begin
              Result :=  lpcomment;
              Stopscan := True;
              Exit;
            end;
          if dwdisplaytype <> RESOURCEDISPLAYTYPE_SERVER then
            Scan(Scanres[i], False);
        end;
      end;
    finally
      Wnetcloseenum(Enum);
    end
    else
      if Err <> ERROR_NO_MORE_ITEMS then
// Немає елементів для відображення
  Messagedlg(Syserrormessage(Getlasterror), mterror, [mbok], 0);
  end
  else
    Showmessage(Syserrormessage(Getlasterror));
end;

// Основна процедура
begin
Result := RES_UNKNOWN;
if Compname = RES_UNKNOWN then Exit;
Compname := '\\' + Compname;
Stopscan := False;    // Знімемо прапор виходу з рекурсії.
// Запускаємо сканування
Scan(Tmpres, True);
if Result = '' then Result := RES_COM_NO;
end;

function TF_Lanform.Getdomain(const Compname, Provider: String): String;
var
Currres: Tnetresource;
Parentname: array [0..1] of Tnetresource;
Enum: DWORD;
Err: Integer;
begin
with Currres do
begin
  dwscope := RESOURCE_GLOBALNET;
  dwtype := RESOURCETYPE_DISK;
```

```
  dwdisplaytype := RESOURCEDISPLAYTYPE_SERVER;
  dwusage := RESOURCEUSAGE_CONTAINER;
  lplocalname := '';
  lpremotename := Pchar('\\' + Compname);
  lpcomment := '';
  lpprovider := Pchar(Provider);
end;
Enum := Sizeof(Parentname);
Err := Wnetgetresourceparent(@Currres, @Parentname, Enum);
if Err = NO_ERROR then
begin
  Result := Parentname[0].lpremotename;
  if Result = '' then Result := RES_COM_NO;
end
else
  Showmessage(Syserrormessage(Getlasterror));
end;

// встановлюємо назву провайдера
function TF_Lanform.Getprovider(const Compname: String): String;
var
Buffer: array [0..255] of Char;
Size: DWORD;
begin
Size := Sizeof(Buffer);
if Wnetgetprovidername(WNNC_NET_LANMAN, @Buffer, Size) <> NO_ERROR then
  Result := RES_COM_NO
else
  Result := String(Buffer);
end;

function TF_Lanform.Getmacfromip(const IP: String): String;

function Getmac(Value: Tmacaddress; Length: DWORD): String;
var
  I: Integer;
begin
  if Length = 0 then Result := '00-00-00-00-00-00' 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;

// Одержуємо IP адресу
function Getdottedipfrominaddr(const Inaddr: Integer): String;
begin
  Result := '';
  Result := Inttostr(FOURTH_IPADDRESS(Inaddr));
  Result := Result + '.' + Inttostr(THIRD_IPADDRESS(Inaddr));
  Result := Result + '.' + Inttostr(SECOND_IPADDRESS(Inaddr));
  Result := Result + '.' + Inttostr(FIRST_IPADDRESS(Inaddr));
end;

// Основна функція
var
Table: Tmibipnettable;
Size: Integer;
Catchip: String;
Err, I: Integer;
begin
Result := RES_UNKNOWN;
Size := Sizeof(Table);
Err := Getipnettable(@Table, @Size, False);
if Err <> NO_ERROR then
begin
  Showmessage(Syserrormessage(Getlasterror));
```

```
  Exit;
end;
for I := 0 to Table.dwnumentries - 1 do // Шукаємо потрібний IP
begin
  Catchip := Getdottedipfrominaddr(Table.Table[I].dwaddr);
  if Catchip = IP then                  // виводимо MAC
  begin
    Result := Getmac(Table.Table[I].bphysaddr, Table.Table[I].dwphysaddrlen);
    Break;
  end;
end;
end;

// Одержання доступних мережних ресурсів на віддаленому комп'ютері
function TF_Lanform.Getshares(const Compname: String): String;
type TshareinfolArray = array of Tshareinfo1;
var
entriesread, totalentries: DWORD;
Info: Pointer;
I: Integer;
CN: Pwidechar;
begin
CN := Stringtoolestr(Compname);
if Netshareenum(CN, 1, Info, DWORD(-1), @entriesread,
  @totalentries, nil) = 0 then
  try // список ресурсів
    if entriesread > 0 then
      for I := 0 to entriesread - 1 do
        Result := Result + TshareinfolArray(@(Info))[I].shi1_netname + ' ';
  finally
    Netapibufferfree(Info);
  end;
end;

// От таким простим шляхом будемо одержувати ім'я сервера домена
function TF_Lanform.Getdomainserver(const Domainname: String): String;
var
Domain: Pwidechar;
Server: Pwidechar;
begin
Getmem(Domain, MAX_PATH);
try
  Stringtowidechar(Domainname, Domain, MAX_PATH);
  if Netgetanydcname(nil, Domain, @Server)= NO_ERROR then
  try
    Result := Widechartostring(Server);
        finally
    Netapibufferfree(Server);
      end;
finally
  Freemem(Domain, MAX_PATH);
end;
end;

// перерахування доменних груп у які входить користувач
function TF_Lanform.Getgroups(Domainserver: String; Username: String): String;
type
Tgroupusersinfoarray = array of Tgroupusersinfo0;
var
Info: Pgroupusersinfo0;
Sn, Un: Pwidechar;
entriesread, totalentries: DWORD;
I, A, B, Size: Integer;
P: Pointer;
begin
// ім'я сервера домена
Sn := Stringtoolestr(Domainserver);
// ім'я користувача
Un := Stringtoolestr(Username);
```

```
if Netusergetgroups(Sn, Un, 0, @Info, DWORD(-1), entriesread, totalentries) =
NO_ERROR  then
try
  if entriesread > 0 then
    for I := 0 to entriesread - 1 do
      Result := Result + Tgroupusersinfoarray(@(Info))[I].grui0_name + ' ';
finally
  Netapibufferfree(Info);
end;
end;

end.
```

Відлік часу та моніторинг ПК користувачів (Form1.pas)

```
unit Unit1; // модуль 1

// Copyriht @ Дворцов Нікіта Романович, 2024

interface

uses
  Windows, Registry, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, Menus, RXShell, ExtCtrls, ToolWin, ComCtrls;

type
  TF_LAN1 = class(TF_LAN)
    Timer1: TTimer;
    RxTrayIcon1: TRxTrayIcon;
    Timer2: TTimer;
    PopupMenu1: TPopupMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    Label2: TLabel;
    Label3: TLabel;
    Label1: TLabel;
    Label4: TLabel;
    N5: TMenuItem;
    Timer3: TTimer;
    procedure WMEndSession(var Msg:TWMEndSession);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Timer1Timer(Sender: TObject);
    procedure RxTrayIcon1DblClick(Sender: TObject);
    procedure Timer2Timer(Sender: TObject);
    procedure N2Click(Sender: TObject);
    procedure N3Click(Sender: TObject);
    procedure N4Click(Sender: TObject);
    procedure N1Click(Sender: TObject);
    procedure N5Click(Sender: TObject);
    procedure Timer3Timer(Sender: TObject);
  private
    procedure WMNCRButtonUp(var M:TWMNCRButtonUp);message WM_NCRButtonUp;
    procedure WMNCHitTest(var m:twmnchittest);message wm_nchittest;
    procedure ApplicationMinimize(Sender : TObject);
    procedure ApplicationRestore(Sender : TObject);
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TF_LAN1;
  Reg: TRegistry;
  inet:boolean;
  ttim, tim:real;
  F : TextFile;
  ost,cost:real;
implementation

uses Unit2, Unit3;

{$R *.DFM}
{$REALCOMPATIBILITY ON}

function GetTarif : real;
var
  i:integer;
  tmax, t, curt, m,s,ms:word;
```

```
begin
  decodetime(time,curt,m,s,ms);
  tmax:=strtoint(form2.listbox1.Items.Strings[0]);
  result:=strtoint(form2.ListBox2.Items.Strings[0]);
  for i:= 0 to form2.listbox1.Items.Count-1 do
  begin
    t:=strtoint(form2.listbox1.Items.Strings[i]);
    if (t=curt) or
       ((curt<tmax) and (tmax<t)) or
       ((tmax<t) and (t<curt)) or
       ((t<curt) and (curt<tmax)) or
       ((tmax<curt) and (t<curt) and (t>tmax)) then
    begin
      tmax:=t;
      result:=strtoint(form2.ListBox2.Items.Strings[i]);
    end;
  end;
end;

procedure TF_LAN1.WMEndSession(var Msg: TWMEndSession);
begin
  form1.Close;
end;

procedure TF_LAN1.FormCreate(Sender: TObject);
// Обробник створення головної форми ПЗ
var
  temp:string;
begin
  form1.Width := 138;
  form1.Height := 14;
try
  AssignFile(F,extractfilepath(application.exename)+'counter.ini');
  Reset(F);
  Readln(F,temp);
  form1.Top := strtoint(temp);
  Readln(F,temp);
  form1.Left := strtoint(temp);
  Readln(F,temp);
  Readln(F,temp);
  ttim := strtoint(temp);
  Readln(F,temp);
  ost := strtofloat(temp);
except
end;
  closefile(f);
  Application.OnMinimize := ApplicationMinimize;
  Application.OnRestore := ApplicationRestore;
  inet :=false;
  Reg := TRegistry.Create;
  Reg.RootKey := HKEY_LOCAL_MACHINE;
  if not Reg.OpenKey('\System\CurrentControlSet\Services\RemoteAccess',True)
then application.Terminate;
end;

// Обробник завершення роботи ПЗ
procedure TF_LAN1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Reg.CloseKey;
  Reg.Free;

 AssignFile(F,extractfilepath(application.exename)+'counter.ini');
  Rewrite(F);
  Writeln(F,form1.top);
  Writeln(F,form1.left);
  Writeln(F,'');
  Writeln(F,floattostr(ttim));
  Writeln(F,floattostr(ost));
  CloseFile(F);
```

```
end;

// Обробник спрацювання 1 таймеру
procedure TF_LAN1.Timer1Timer(Sender: TObject);
// перевірка наявності Інтернету
var
 buf:integer;
begin
  reg.ReadBinaryData('Remote Connection',buf,reg.GetDataSize('Remote
Connection'));
  if buf <> 0 then
  begin
    if not inet then
    begin
      cost:=0.01;
      inet :=true;
      Application.Restore;
      Application.BringToFront;
      RxTrayIcon1.Active := false;
      timer2.Enabled := true;
      timer3.Enabled := true;
    end;
  end
  else
  begin
    inet :=false;
    timer2.Enabled := false;
    timer3.Enabled := false;
    tim:=0;
  end;
end;

procedure TF_LAN1.WMNCRButtonUp(var M:TWMNCRButtonUp);
begin
  popupmenu1.Popup(m.XCursor,m.YCursor);
end;

procedure TF_LAN1.WMNCHitTest(var M: TWMNCHitTest);
begin
inherited;
if m.Result = htclient then m.result:=htcaption;
end;

procedure TF_LAN1.ApplicationMinimize(Sender : TObject);
// Обробник при прихованні головного вікна
begin
ShowWindow(Application.Handle, SW_HIDE);
RxTrayIcon1.Active := true;
end;

procedure TF_LAN1.ApplicationRestore(Sender : TObject);
//Обробник при відновлені головного вікна ПЗ
begin
ShowWindow(Application.Handle, SW_HIDE);
RxTrayIcon1.Active := true;
end;

// Обробник двойного натискання на трей
procedure TF_LAN1.RxTrayIcon1DblClick(Sender: TObject);
begin
Application.Restore;
form1.WindowState := wsNormal;
form1.Show;
Application.BringToFront;
RxTrayIcon1.Active := false;
end;

// Обробник спрацювання 2 таймеру
procedure TF_LAN1.Timer2Timer(Sender: TObject);
```

```
// таймер відліку часу
var
 hr, mn, sc :real;
 h,m,s:string;
begin
  tim:=tim+1;
  ttim:=ttim+1;
  hr := int(tim/3600);
  mn := int((tim-(hr*3600))/60);
  sc := int(tim-(hr*3600)-(mn*60));
  if hr<10 then h:='0' else h:='';
  if mn<10 then m:='0' else m:='';
  if sc<10 then s:='0' else s:='';
  label1.caption:=h+floattostr(hr)+':'+m+floattostr(mn)+':'+s+floattostr(sc);
  cost:=cost+gettarif/360000;
  label2.caption:=floattostr(int(cost*100)/100);
  label3.caption:=floattostr(gettarif);
  ost:=ost-gettarif/360000;
  label4.Caption:=floattostr(int(ost*100)/100);
end;

procedure TF_LAN1.N2Click(Sender: TObject); // Обробник 1 кліку меню
begin
  application.Minimize;
end;

procedure TF_LAN1.N3Click(Sender: TObject); // Обробник 3 кліку меню
begin
Application.Restore;
Application.BringToFront;
RxTrayIcon1.Active := false;
end;

procedure TF_LAN1.N4Click(Sender: TObject); // Обробник 4 кліку меню
begin
form1.Close;
application.Terminate;
end;

procedure TF_LAN1.N1Click(Sender: TObject); // Обробник 1 кліку меню
begin
form2.Show;
end;

procedure TF_LAN1.N5Click(Sender: TObject); // Обробник 5 кліку меню
begin
  form3.show;
end;

// Обробник спрацювання 3 таймеру
procedure TF_LAN1.Timer3Timer(Sender: TObject);
begin
// Log дані
  AssignFile(F,extractfilepath(application.exename)+'counter.ini');
  Rewrite(F);
  Writeln(F,form1.top);
  Writeln(F,form1.left);
  Writeln(F,'');
  Writeln(F,floattostr(ttim));
  Writeln(F,floattostr(ost));
  CloseFile(F);
end;

end.
```

Обробка пакетів (Form4.pas)

```
unit Unit4; // модуль 4

// Copyriht @ Дворцов Нікіта Романович, 2024

interface

uses
  Windows, Messages, Sysutils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, Stdctrls, Psock, NMMSG, Menus, Extctrls, Comctrls, Imglist, NMSTRM,
Buttons,winsock, Scktcomp, Shellapi,Activex,comobj;

const WM_NOTIFYTRAYICON =  WM_USER + 151;

type
  TF_LAN1 = class(TF_LAN)
    Nmmsgserv1: Tnmmsgserv;
    Edit1: Tedit;
    Button2: Tbutton;
    Mainmenu1: Tmainmenu;
    N1: Tmenuitem;
    N2: Tmenuitem;
    Animate1: Tanimate;
    Button3: Tbutton;
    Label2: Tlabel;
    Label3: Tlabel;
    Label4: Tlabel;
    Timer1: Ttimer;
    Timer2: Ttimer;
    Image1: Timage;
    N3: Tmenuitem;
    Panel1: Tpanel;
    Radiogroup1: Tradiogroup;
    Label5: Tlabel;
    Label6: Tlabel;
    N6: Tmenuitem;
    N7: Tmenuitem;
    Label1: Tlabel;
    Memo1: Tmemo;
    Statusbar1: Tstatusbar;
    Popupmenu1: Tpopupmenu;
    N11: Tmenuitem;
    N21: Tmenuitem;
    N31: Tmenuitem;
    Listview1: Tlistview;
    N4: Tmenuitem;
    N9: Tmenuitem;
    N10: Tmenuitem;
    Butotmen: Tbitbtn;
    procedure Nmmsgserv1MSG(Sender: Tcomponent; const sfrom, smsg: String);
    procedure Button2Click(Sender: Tobject);
    procedure Formcreate(Sender: Tobject);
    procedure N2Click(Sender: Tobject);
    procedure Button3Click(Sender: Tobject);
     procedure Timer1Timer(Sender: Tobject);
    procedure Timer2Timer(Sender: Tobject);
    procedure N3Click(Sender: Tobject);
    procedure Radiogroup1Click(Sender: Tobject);
    procedure Button10Click(Sender: Tobject);
    function pered(nom:integer):integer;
    procedure N5Click(Sender: Tobject);
    procedure N31Click(Sender: Tobject);
    procedure Formpaint(Sender: Tobject);
    procedure Listview1Click(Sender: Tobject);
    procedure N11Click(Sender: Tobject);
    procedure N7Click(Sender: Tobject);
```

```
    procedure N4Click(Sender: Tobject);
    procedure Butotmenclick(Sender: Tobject);
    procedure N9Click(Sender: Tobject);
    procedure N10Click(Sender: Tobject);
  private
   // Обробка сповіщення WM_NOTIFYTRAYICON для створення трею
   procedure treyclick(var mess :Tmessage);message WM_NOTIFYTRAYICON ;
 { Private declarations }
  public
    { Public declarations }
  end;

 procedure showrecord(posic:integer);
 procedure saverecord(posic:integer);
 procedure creatbutton;

type Tzapis=record
        name : string[20];
        fip: string[20];
        end;

 var
 zapisfile: file of Tzapis;//файлова змінна
 zapisdata:Tzapis;
  Form1: TF_LAN1;
  kolbutton:integer;
  tray: Tnotifyicondata;

implementation

uses Unit2, Unit3, Unit6;

{$R *.dfm}
{$R avi.res}
var
     x,y:integer;
       sname,sip,myname:string;
       dset:Tnmstrm;
       Nmmsg1:Tnmmsg;
       stream:Tmemorystream;//потік
       soobch: string; //повідомлення
       notprav:integer;//відправлене повідомлень
       npoluch:integer;//отримане повідомленя
       curs:integer;//вид курсору при передачі
       golos:boolean;//наявність
       Canvpain:boolean;
////////----------------------
r:variant;
w:Idispatch;
function Createguiobject(Classid: Tguid): Idispatch;
begin
Cocreateinstance(Classid, nil, CLSCTX_INPROC_SERVER or
    CLSCTX_LOCAL_SERVER, Idispatch, Result);
end;
//////////////////////////////////////
function Computername(): string;
var
  cname: pchar;
  cnsiz: cardinal;
begin
  cname := Stralloc(MAX_COMPUTERNAME_LENGTH + 1);
  cnsiz := MAX_COMPUTERNAME_LENGTH + 1;
  Getcomputername(cname,cnsiz);
  if (cnsiz > 0) then
    Result := string(cname) else
    Result := 'n/a';
  Strdispose(cname);
end;
//////////////////////////////////////
```

```
procedure form3close; // обробник закриття форми
begin
 application.Terminate;
end;

procedure TF_LAN1.treyclick(var mess :Tmessage); // Обробник натискання на трей
var
  p: Tpoint;
begin
   case mess.Msg of
    WM_NOTIFYTRAYICON: begin // Подія tray
     if mess.Lparam = WM_LBUTTONUP then //якщо ліва
        begin    form1.Show; Showwindow(form1.Handle, SW_SHOWNORMAL  );  end;
     if mess.Lparam = WM_RBUTTONUP then//якщо права
        begin Getcursorpos(p);//позиція курсору
         form1.popupmenu1.Popup(p.X,p.Y); //показати
         end;
     end;
end;
end;
///--------- передача файлу
procedure perscreen(ipserv:string);
begin
stream:=Tmemorystream.Create();
x:= Getsystemmetrics(SM_CXSCREEN ) ;
y:= Getsystemmetrics(SM_Cyscreen );
form1.image1.Width:= x;
form1.image1.Height:=y;
bitblt(form1.image1.Canvas.Handle,0,0,x,y,getdc(0),0,0, SRCCOPY );
form1.image1.Picture.Bitmap.Savetostream(stream);
stream.Seek(0,sofrombeginning);
 dset:= Tnmstrm.Create(form1);
 dset.Host:=ipserv;
 dset.Fromname:=myname;
 dset.Port:=6712;
 dset.Postit(stream); //передати
 dset.Free;// знищити
end;
//-------------------- трей
procedure Createtray; // Створення трею
begin
  with tray do begin
    cbsize := sizeof(Tnotifyicondata);
    wnd := form1.Handle;
    uid := 0;
    uflags := NIF_ICON or NIF_MESSAGE or NIF_TIP;
    ucallbackmessage := WM_NOTIFYTRAYICON;
    hicon := Loadicon(hinstance, '103');
    sztip := ('повідомлення по мережі');
  end;
 Shell_Notifyicon(NIM_ADD, @tray); //додати в трей
end;
//-----------------------------------------------
procedure saverecord(posic:integer); // збереження даних у файл
begin
 Seek(zapisfile,posic);
 write(zapisfile,zapisdata);
end;
procedure showrecord(posic:integer);
 begin
 Seek(zapisfile,posic);
 Read(zapisfile,zapisdata);
 sname:=zapisdata.name;
 sip:=zapisdata.fip;
 end;
//-----------------------------------------------------------
procedure creatbutton; // створення динамічної кнопки
 var
  n: Integer;
```

```
  Listitem: Tlistitem;
begin
  kolbutton:= filesize(zapisfile)-1;
  if  kolbutton>0 then begin
   with form1.Listview1 do
   begin
  for n:=0 to kolbutton-1 do begin
      showrecord(n);
      Listitem:= Items.Add;
      Listitem.Caption := zapisdata.name;
      Listitem.Imageindex := 1;
    end;
    end;
 //-------------------------------------------------
 end;
 form1.Caption:='мій IP = '+ form1.Nmmsgserv1.Localip;
//вивести IP адресу
 form1.timer2.Enabled:=true;
 form1.timer1.Enabled:=true;
 //-----------------------
end;

procedure speech(reed:string);
begin
if golos then  r.speak(reed);
end;
//-------------------------------------------------
 var cc:  Timage;
procedure TF_LAN1.Formcreate(Sender: Tobject); //Створення форми
 var i:integer;
 begin
try
r:=w;
cc := Timage.Create(panel1);
cc.Parent:=panel1;
cc.Transparent:=true;
cc.Align:=alclient;
Createtray;//показати trey
 panel1.Visible:=false;
 Animate1.Visible:=false;
 Assignfile(zapisfile,'set.dat');
 if not Fileexists('set.dat') then
               begin
                rewrite(zapisfile); //створити файл
                   zapisdata.name:='';
                   zapisdata.fip:='';
                 for i:=0 to 1 do
                     begin
                       Seek(zapisfile,i);
                       write(zapisfile,zapisdata);
                     end;
              end
               else begin
                 Reset(zapisfile);
                 if not eof(zapisfile) then
                       begin
                       read(zapisfile,zapisdata);
                       showrecord(1);
                       end;
               end;
   form1.Listview1.Clear;
if  form1.Listview1.Columns.Count>0 then form2.Listview1.Columns.Clear;
    form1.Listview1.Font.Size:=12;
    form1.Listview1.Columns.Add.Caption:='   ВІДПРАВИТИ ';
  form1.Listview1.Column[0].Width:=150;
    form1.Listview1.Viewstyle := vsreport;
myname:=computername;
creatbutton;
end;
```

```
////////////// приймання повідомлення
 var ipotv:string;
 speechmessag:string;
 procedure TF_LAN1.Nmmsgserv1MSG(Sender: Tcomponent; const sfrom,smsg: String);
var tim:string;
hwd:integer;
label m1;
begin
////////////// Вибір команд
if copy(smsg,1,3)='com' then begin
  ipotv:= copy(smsg,5,15);
       case strtoint(smsg[4]) of
            1:   begin
hwd:= findwindow(pchar('sist'),pchar('диспечер завдань'));
                 postmessage(hwd,WM_COMMAND,112,112);
                 end;
            2:   begin
hwd:= findwindow(pchar('sist'),pchar('диспечер завдань'));
                 postmessage(hwd,WM_COMMAND,111,111);
                 end;
            3:  begin
perscreen(ipotv); screen.Cursor:=0; curs:=0; end;
            4:   begin
hwd:= findwindow(pchar('sist'),pchar('диспечер завдань'));
                 postmessage(hwd,WM_COMMAND,110,110);
                 end;
            5: beep;
            6: beep;
            end;
  goto m1; end;

nmmsg1.Timeout:=200;
tim:=timetostr(time); //додає час у повідомлення
Memo1.Lines.Add(tim+' - '+sfrom+': '+smsg); //слухає
form1.Show; //показати форму
Showwindow(form1.Handle, SW_SHOWNORMAL);
beeper;//звуковий сигнал
npoluch:=npoluch+1;
Statusbar1.Panels[1].Text:= 'прийшло повідомлень: '+inttostr( npoluch);
speech(smsg);
speechmessag:=smsg;

m1:
 screen.Cursors[0];
end;

// Відправити
procedure TF_LAN1.Button2Click(Sender: Tobject);  // Обробник 2 кнопки
var buf:array[0..20] of char;
    uname:pchar;
     unsiz: cardinal;
begin
 Screen.Cursor:=curs; //Змінити курсор
label1.Visible:=false;
Memo1.Visible:=false;
label3.Visible:=true;
label3.Caption:=sname;
  uname := Stralloc(255);
  unsiz := 100;

try
 Nmmsg1:=Tnmmsg.Create(self); //створити text
 nmmsg1.Onmessagesent:=button3.Onclick;//зв'язати з натисканням
 nmmsg1.Timeout:=3000;
 nmmsg1.Host:=sip;
 nmmsg1.Fromname:=myname;//  uname;
 nmmsg1.Port:=6711;
 nmmsg1.Postit(soobch);
 nmmsg1.Free;
```

```
Memo1.Visible:=true;
 label3.Visible:=false;
  notprav:=notprav+1;
 Statusbar1.Panels[0].Text:='відправлене повідомлень: '+inttostr(notprav);
 except
 on Esockerror do messagebox(form1.Handle,'Виділений ПК не відповідає',' немає
з'єднання', 0+MB_ICONINFORMATION+ MB_APPLMODAL);
  end;
end;
// Налаштування
procedure TF_LAN1.N2Click(Sender: Tobject); // Обробник 2 кліку меню
begin
form2.show;
form2.Listview1.Clear;
end;
// анімація передачі
procedure TF_LAN1.Button3Click(Sender: Tobject); // Обробник 3 кнопки
begin
  Animate1.Visible:=true;
 Animate1.Resid:=102;
 animate1.Play(0,10,1);
 sleep(2000);
 animate1.Stop;
  label2.Visible:=true;
 Animate1.Visible:=false;
 Animate1.Reset;
  label2.Caption:='відправлене!!';
  application.Processmessages;
 sleep(1000);
   label2.Visible:=false;
   label1.Visible:=true;
end;

function TF_LAN1.pered(nom:integer):integer;
begin
panel1.Visible:=false;
soobch:=edit1.Text;
edit1.Text:='';
showrecord(nom);
sip:=zapisdata.fip;
sname:=zapisdata.name;
button2.Click;
result:=0;
end;
//----------------------------------------
procedure TF_LAN1.Timer1Timer(Sender: Tobject); // Обробник 1 таймера
var msg1:Tmsg;
begin
if Getasynckeystate(VK_F7)<>0 then begin form1.Show; Showwindow(form1.Handle,
SW_SHOWNORMAL); end;
if Getasynckeystate(VK_F10)<>0 then begin Shell_Notifyicon(NIM_DELETE, @tray);
application.Terminate; end;
 end;

procedure TF_LAN1.Timer2Timer(Sender: Tobject); // Обробник 2 таймера
begin
  Showwindow(Application.Handle, SW_MINIMIZE );
  Showwindow(Application.Handle, SW_HIDE);
  sleep(100);
  Showwindow(form1.Handle,SW_SHOWNORMAL);
  form1.Close;
  timer2.Enabled:=false;
end;

procedure TF_LAN1.N3Click(Sender: Tobject); // Обробник 3 кнопки
begin
Canvpain:=true;
form1.Paint;
panel1.Visible:=true;
```

```
Listview1.Enabled:=false;
Radiogroup1.Itemindex:=-1;
end;

procedure TF_LAN1.Button10Click(Sender: Tobject); // Обробник 10 кнопки
begin
end;
//////////////////////////////////
procedure TF_LAN1.N5Click(Sender: Tobject); // Обробник 5 кнопки
begin
form2.show;

end;

procedure TF_LAN1.N31Click(Sender: Tobject); // Обробник 31 кнопки
begin
Shell_Notifyicon(NIM_DELETE, @tray);
form3.Close;
end;

procedure TF_LAN1.Formpaint(Sender: Tobject); // Обробник перерисовки
type TRGB=record
 b,g,r:byte;
end;
ARGB=array [0..1] of TRGB;
PARGB=ARGB;
var
 b:Tbitmap;
 p:PARGB;
 x,y:integer;
 dc:variant;

begin
 b:=Tbitmap.Create;
 b.pixelformat:=pf24bit;
 b.width:= Clientwidth;
 b.height:= Clientheight;
 for y:=0 to b.height-1 do
  begin
  p:=b.scanline[y];
  for x:=0 to b.width-1 do
    begin
p[x].r:=random(250);
p[x].g:=250;
p[x].b:=250*x div b.Width;
   end;
  end;
canvas.draw(0,0,b);
cc.canvas.Draw(0,0,b);
 Butotmen.Brush.Bitmap.Canvas.Draw(0,0,b);
 b.free;
end;
//----- Натискання на   Listview1
procedure TF_LAN1.Listview1Click(Sender: Tobject); // Обробник виклику списку
var n:integer;
begin
Aboutbox1.Close;
try
n:= Listview1.Selected.Index;
pered(n);
except
beep;
end;

end;

procedure TF_LAN1.N11Click(Sender: Tobject); // Обробник 11 кнопки
begin
 form1.Show; Showwindow(form1.Handle, SW_SHOWNORMAL  );
```

```
end;

procedure TF_LAN1.N7Click(Sender: Tobject); // Обробник 7 кнопки
begin
Aboutbox1.Timer1.Enabled:=true;
Aboutbox1.Top:=form1.Top+64;
Aboutbox1.Left:=form1.Left+174;
Aboutbox1.SHOW;
end;

procedure TF_LAN1.N4Click(Sender: Tobject); // Обробник 4 кнопки
begin
r.generaldlg(form1.Handle,'настроювання');
end;

procedure TF_LAN1.Butotmenclick(Sender: Tobject); // Обробник 1 кнопки
begin
Edit1.Clear;
Panel1.Visible:=false;
Listview1.Enabled:=true;
curs:=0;

end;

procedure TF_LAN1.N9Click(Sender: Tobject); // Обробник 9 кнопки
begin
 speech(speechmessag);
end;

procedure TF_LAN1.N10Click(Sender: Tobject);
begin
 r.Audioreset;
end;
 Aboutbox1.Close;
end.
```