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
“____” _____ 2021 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи віртуалізованої серверної інфраструктури на базі HCI”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-20М-1,4
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Джевлах М.Р.
«____» _____ 2021 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Наталія ЯКИМЕНКО
«____» _____ 2021 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

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
« 6 » вересня 2021 року

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

Джевлаху Микиті Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи віртуалізованої серверної інфраструктури на базі HCI*

2. Керівник роботи *Якименко Наталія Миколаївна, канд. фіз.-мат. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 42-13 від 02.08.2021 року

3. Строк подання студентом роботи до захисту *10.12.2021 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи віртуалізованої серверної інфраструктури на базі HCI*

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

1. Призначення та область використання.	*7. Економічна ефективність розробленої*
2.Перегляд аналогічних існуючих систем.	*програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	
6. Наукова новизна	

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

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2021	14.11.2021
Охорона праці	Оришака О.В.	06.10.2021	16.11.2021

7. Дата видачі завдання « 6 » вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2021 р.	
3.	Розробка моделі компонента	20.10.2021 р.	
4.	Розробка структур даних	25.10.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2021 р.	
6.	Програмування алгоритмів	10.11.2021 р.	
7.	Розрахунок економічної ефективності	13.11.2021 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2021 р.	
9.	Оформлення ПЗ	17.11.2021 р.	
10.	Попередній захист роботи	10.12.2021 р.	

Дата видачі завдання
« 6 » вересня 2021 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2021 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Джевлах М.Р. Дослідження та програмна реалізація системи віртуалізованої серверної інфраструктури на базі HCI. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи віртуалізованої серверної інфраструктури на базі HCI.

Метою розробки є дослідження та програмна реалізація системи віртуалізованої серверної інфраструктури на базі HCI.

Об'єктом дослідження є процес віртуалізованої серверної інфраструктури на базі HCI.

Предметом дослідження є методи віртуалізованої серверної інфраструктури на базі HCI.

Методи дослідження базуються на методах теорії комп/ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи віртуалізованої серверної інфраструктури на базі HCI.

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

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

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows XP/Vista/7/8/10.

Програму розроблено в середовищі Embarcadero RAD Studio Delphi 10.3.2 Rio Architect.

Ключові слова: комп'ютерна інженерія, віртуалізована серверна інфраструктура, HCI

ABSTRACT

Dzhevlakh M.R. Research and software implementation of HCI-based virtualized server infrastructure system. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this final qualification work on the second (master's) level of higher education the software which is intended for system of the virtualized server infrastructure on the basis of HCI is developed.

The purpose of development is research and software implementation of a system of virtualized server infrastructure based on HCI.

The object of research is the process of virtualized server infrastructure based on HCI.

The subject of research is the methods of virtualized server infrastructure based on HCI.

Research methods are based on the methods of computer network theory, methods of mathematical statistics, methods of software development.

The result is a software implementation of a system of virtualized server infrastructure based on HCI.

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

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

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

The program is developed in Embarcadero RAD Studio Delphi 10.3.2 Rio Architect.

Keywords: computer engineering, virtualized server infrastructure, HCI

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ВМ	–	віртуальні машини
ЛОМ	–	локальна обчислювальна мережа
ПК	–	персональний комп’ютер
ПЗ	–	програмне забезпечення
СЗД	–	система зберігання даних
ЦОД	–	центр обробки даних
DNS	–	Domain Name System
HCI	–	гіперконвергентна інфраструктура
http	–	HyperText Transfer Protocol
IP	–	Internet Protocol
VDI	–	технологія віртуалізації робочих місць

ВСТУП

Актуальність теми. Основними перевагами технології віртуалізації робочих місць (VDI), у порівнянні зі звичайними десктопами, є істотне зниження витрат на обслуговування парку настільних комп'ютерів і підвищення інформаційної безпеки. Незважаючи на ці важливі переваги, впровадження VDI йде повільно. Як же можна його прискорити?

VDI припускає використання віртуалізованої серверної інфраструктури для запуску віртуальних машин з копіями операційної системи й застосунків ПК. Замість настільних ПК на робочому місці користувача встановлюється тонкий або нульовий термінал з мінімальною процесорною потужністю, а настільні застосунки виконуються на віртуалізованому сервері, на якому розгорнута VDI.

Основними перевагами VDI, у порівнянні зі звичайними десктопами, є істотне зниження витрат на обслуговування парку настільних комп'ютерів (насамперед за рахунок економії робочого часу системних адміністраторів) і підвищення інформаційної безпеки при використанні конфіденційних даних. Крім того, оскільки компонентів, які можуть вийти з ладу, таких як жорсткі диски, у термінала менше, ніж у звичайних настільних ПК, витрати на ремонт клієнтських комп'ютерів скорочуються.

Однак, незважаючи на ці важливі переваги, впровадження VDI іде повільно, особливо якщо зрівняти з технологіями віртуалізації серверної інфраструктури, які давно стали фактичним стандартом для IT-інфраструктури сучасної компанії. В Україні темпи впровадження рішень VDI значно нижче, ніж у Європі й США. Почасти це можна пояснити тим, що праця системних адміністраторів і інших спеціалістів, відповідальних за обслуговування IT-інфраструктури, в українських компаніях оцінюється не так високо, як на Заході.

У результаті при заміні настільних ПК на віртуальні столи потенційна економія витрат на зарплату IT-персоналу не настільки значна й не може

виправдати інвестиції на придбання потужних серверів і систем зберігання, необхідних для побудови віртуалізованого середовища. Тому в нашій країні VDI найчастіше впроваджується в організаціях, де вимоги до інформаційної безпеки робочих місць співробітників досить високі: насамперед у банках і інших організаціях фінансового сектора.

Гіперконвергентна інфраструктура (HCI) – це ПЗ-центрична архітектура із твердою зв'язаністю елементів зберігання, мереж і обчислень, інтегрованих у стандартне устаткування одного постачальника.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи віртуалізованої серверної інфраструктури на базі HCI.

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

– Огляд існуючих систем віртуалізованої серверної інфраструктури на базі HCI.

– Дослідження системи віртуалізованої серверної інфраструктури на базі HCI.

– Програмна реалізація системи віртуалізованої серверної інфраструктури на базі HCI.

Об'єктом дослідження є процес віртуалізованої серверної інфраструктури на базі HCI.

Предметом дослідження є методи віртуалізованої серверної інфраструктури на базі HCI.

Методи дослідження базуються на методах теорії комп/ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод віртуалізованої серверної інфраструктури на базі HCI.

– Розроблено вітчизняний продукт віртуалізованої серверної інфраструктури на базі HCI, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі віртуалізованої серверної інфраструктури на базі HCI.

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

Робота апробована на LV Науково-технічна конференція здобувачів вищої освіти «Наука – виробництву», 2021, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №12.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи віртуалізованої серверної інфраструктури на базі HCI, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Технології віртуалізації робочих місць (Virtual Desktop Infrastructure, VDI) і термінального доступу дозволяють значно зменшити витрати на обслуговування користувачів і персональних комп'ютерів. Централізоване зберігання даних підвищує інформаційну безпеку компанії й скорочує ризики витоків/розкрадання важливих даних.

Всі застосунки й дані користувача зберігаються й обробляються в ЦОДі замовника, при цьому для співробітника нічого не міняється – він працює через звичний інтерфейс на своєму екрані. Подібні рішення застосовуються як у рамках окремих офісів, так і в територіально розподілених філіальних структурах з великою кількістю співробітників:

– банки;

– ритейл;

– державні організації;

– аеропорти й т.д.

Інфраструктура VDI набагато простіше й швидше в розгортанні в порівнянні із традиційними робочими станціями. Розміщення обчислювальної інфраструктури в одному ЦОД і централізоване відновлення ПЗ скорочує затримки в роботі бізнес-систем і підвищує якість роботи IT.

Підключення користувача до ЦОД забезпечується з використанням звичайних ПК або спеціалізованих технічних засобів – тонких клієнтів. Вони споживають набагато менше електроенергії й в окремих випадках практично не сприйнятливі до агресивних умов навколишнього середовища (підвищена температура, вологість, тривалий час роботи без перезавантаження й т.д.). Такі моделі успішно використовуються на промислових і виробничих підприємствах.

1.2 Область застосування

Гіперконвергентна інфраструктура (HCI) – це ПЗ-центрична архітектура із твердою зв'язаністю елементів зберігання, мереж і обчислень, інтегрованих у стандартне устаткування одного постачальника.

Організації розгортають гіперконвергентну інфраструктуру для своїх робітничих середовищ рівня 2 і задоволені моделлю устаткування HCI, оскільки таке рішення легко запустити. Можна почати з малого розміру й при необхідності виконати вертикальне масштабування. Можна зменшити обсяг обчислень і СЗД, щоб консолідувати робітничі середовища й знизити операційні витрати на периферії за рахунок зняття з IT-фахівців завдань по керуванню.

Проте HCI пов'язана з певним компромісом відносно продуктивності через втрату дискретних обчислювальних компонентів і доменів на випадок збоїв СЗД. Крім того, керованість із масштабуванням являє собою окрему проблему при вертикальному масштабуванні робітничих середовищ рівня 2 і додаванні устаткування. Можна привести такий аргумент: інфраструктура, необхідна для роботи тисяч ВМ і керування ними, повинна ставитися до робітничого середовища рівня 1, а управляти декількома великими об'єктами значно легше, ніж тисячею малих об'єктів.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи віртуалізованої серверної інфраструктури на базі HCI, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Як ілюстрація розглянемо кілька рішень HCI, пропонованих як платформа для розгортання інфраструктури VDI.

Компанія Dell EMC розробила два комплексних рішення VDI Complete для розгортання інфраструктури віртуальних десктопів на базі технологій віртуалізації VMware. Гіперконвергентні комплекси VxRail використовують для програмно-визначаємого зберігання пакет програм Virtual SANEnterprise, інструменти захисту даних у віртуалізованому середовищу Recovery Point for VM і vSphere Replication / Data Protection, а також утиліту для швидкого розгортання приставки VxRail Manager.

VxRail – це повністю готове до роботи рішення (за заявою вендора, розгортання всієї інфраструктури VDI за допомогою VxRail виконується менше ніж за годину). Хоча сама лінійка VxRail складається з п'яти типів вузлів, Dell EMC рекомендує для підтримки VDI вузли?V Series на базі серверів 2U з повністю твердотільною (all flash) або гібридною дисковою підсистемою ємністю до 46 Тбайт на один вузол, хоча для обес -печения високої продуктивності деск-топів переважно використовувати вузли в конфігурації all flash.

Друге рішення являє собою пакет для програмно-визначаємого зберігання Virtual SAN Ready Node, що поставляється разом із двосокетними серверами Dell PowerEdge R630/R730. Замовник сам повинен виконати інсталяцію на PowerEdge гіпервізора vSphere і побудувати за допомогою Virtual SAN Ready Node кластер серверів і програмно-визначаєме сховище, що складається з дисків серверів. У порівнянні з VxRail такий варіант впровадження інфраструктури VDI вимагає

досить високої кваліфікації IT-персоналу, але в той же час на устаткування буде витрачено менше засобів, а крім того, покупець зможе самостійно настроїти параметри вузлів відповідно до вимог своїх застосунків.

У якості ПЗ віртуальних робочих столів обоє рішення підтримують пакети VMware Horizon 7 і Horizon Apps, а для розгортання на віртуальних десктопах застосунків з важкою графікою можна встановити у вузлах графічні прискорювачі NVIDIA GPU (надається як опція).

Варто згадати, що Dell EMC, крім рішень для побудови а паратної платформи VDI, пропонує великий пакет тонких терміналів із серії Dell Wyse, тому замовники компанії можуть одержати повний стек рішень для впровадження інфраструктури віртуальних клієнтів.

Серед гіперконвергентних комплексів інших виробників, що просувають дане рішення у вигляді апаратної платформи для впровадження VDI, можна згадати продукти компанії Nutanix, що спеціалізується на технологіях VDI (на основі її систем випускають свої комплекси Dell EMC (серія XC) і Lenovo), а також комплекси HyperFlex від Cisco і SimpliVity від HPE. Всі ці гіперконвергентні рішення підтримують використання ПЗ віртуальних десктопів Citrix XenDesktop і VMware Horizon.

Компанія Huawei у якості готової апаратної платформи для розгортання VDI пропонує гіперконвергентний комплекс FusionCloud Appliance на базі стієчних серверів або блейд-серверів FusionCube 2000/6000/9000, що масштабується до 5000 віртуальних десктопів. Для більших інсталяцій VDI (до 20 тис. віртуальних десктопів) китайський вендор рекомендує використовувати інтегровану систему FusionCloud Standard, що складається із блейд-серверів E9000 і окремого дискового масиву середнього класу S5500T.

Інший шлях до скорочення первісних витрат на побудову інфраструктури віртуальних десктопів вибрала компанія Parallels, що розробила рішення Parallels Containers for Windows (PCW) для контейнеризації ОС Windows у середовищі Windows Server. У порівнянні із традиційним підходом до реалізації VDI, у

випадку застосування контейнерів Windows щільність розгортання на сервері віртуальних десктопів істотно збільшується (до 200-250 штук на один сервер), що знижує вимоги до потужності серверної платформи VDI, а також дозволяє заощадити на ліцензуванні Microsoft Windows.

Український системний інтегратор IBS разом з Parallels розробив на базі PCW три додаткових компоненти, що спрощують розгортання інфраструктури віртуальних робочих столів: брокер з'єднань для керування інфраструктурою VDI і підключення до неї користувачів Odin Connection Broker, клієнт для пристроїв доступу під керуванням Linux і Windows, призначений для підключення до віртуальних робочих місць Odin VDI Client, а також програму-агент Odin Hardware Node Extention, що дозволяє брокерові з'єднань управляти хостами віртуалізації.

В основі платформи Odin VDI лежить стандартна с ерверна операційна система Windows Server, поверх якої встановлюється PCW і запускаються робітничі середовища, так звані контейнери, ізольовані на рівні адресного простору оперативної пам'яті й закритої (private) області на диску. Контейнери не мають потреби в індивідуальній інсталяції операційної системи й емуляції роботи устаткування, оскільки в усіх запускається екземпляр базової ОС. У кожного з них свої користувачі, у кожному виконуються індивідуальні процеси, є власний реєстр, мережні адаптери, виділений обсяг оперативної пам'яті, а також жорсткий диск. За заявою розроблювачів, у режимі очікування контейнер займає всього 200-300 Мбайт оперативної пам'яті.

Гіперконвергентні рішення Dell EMC

Вигода від розширення портфеля HCI компанії Dell EMC. Перешикуйте вашу роботу за допомогою інтегрованих систем під ключ, які прискорюють модернізацію IT, або продовжуйте використовувати існуючі операційні моделі, одержуючи при цьому переваги HCI за допомогою гнучких, попередньо перевірених будівельних блоків HCI.

Dell EMC VxRail

Створені для VMware, разом з VMware і для поліпшення VMware, системи VxRail надають найшвидший і простий шлях до трансформації й модернізації інфраструктури центра обробки даних. VxRail – єдина платформа гіперконвергентної інфраструктури для кожного робочого навантаження й сценарію використання VMware, включаючи VDI, застосунка з високою обчислювальною щільністю, а також для традиційних і хмарних застосунків у справжній гібридній хмарній інфраструктурі.

VxRail HCI System Software – перша і єдина ОС для гіперконвергентної інфраструктури, розроблена разом з VMware. Вона забезпечує ефективність експлуатації, цілісний стік і автоматизоване комплексне керування життєвим циклом, а також прогнозну аналітикові на основі ШІ.

Dell EMC доставляє систему VxRail попередньо налаштовану й протестовану в обраній вами конфігурації. Починаючи від пристроїв до інтегрованих стійок, з мережним устаткуванням або без, ви зможете підібрати рішення VxRail, що оптимально відповідає вашим потребам.

Основні характеристики:

– Програмно-визначаєма архітектура, що консолідує ресурси обчислень, зберігання, віртуалізації й керування.

– Інтеграція з усім стеком технологій VMware, включаючи VMware Cloud Foundation.

– Надає сервіси керування даними, забезпечує відказостійкість і якість обслуговування корпоративного рівня.

– Автоматизація налаштування мережі за допомогою функції SmartFabric Services for VxRail, що дозволяє значно спростити й прискорити розгортання.

– Єдина крапка підтримки для програмного й апаратного забезпечення.

– Analytics Consulting Engine (ACE) забезпечує практично значиму аналітику й керування декількома кластерами з машинним навчанням.

Dell EMC Cloud for Microsoft Azure Stack

Dell EMC Cloud for Microsoft Azure Stack – це локальна гібридна хмарна платформа для надання інфраструктури й моделі «платформа як послуга» (PaaS) з погодженими умовами роботи в хмарі Azure на локальній площадці або у віддалених середовищах.

Виконайте цифрову трансформацію бізнесу, використовуючи погоджені з Azure сервіси, хмарні застосунки й ресурси для самообслуговування. Забезпечте швидке, просте й упевнене впровадження гібридної хмари.

Основні характеристики:

– Локальна й периферійна платформа гібридної хмари.

– Автоматизоване надання IT-послуг.

– Спеціалізовані хмарні застосунки.

– Прискорене впровадження гібридної хмари.

– Ресурси для самообслуговування на вимогу.

– Погоджені умови роботи на платформі Azure на локальній площадці або в периферійних середовищах.

Ефективно надавайте застосунки й послуги завдяки платформі з погодженим інтерфейсом Azure для локального середовища:

– Забезпечте більше швидке надання застосунків завдяки ресурсам, доступним на вимогу в режимі самообслуговування.

– Впровадите інновації за допомогою хмарних застосунків, щоб виконати цифрову трансформацію вашого бізнесу.

– Забезпечте швидке, просте й упевнене впровадження гібридної хмари.

– Гнучкі варіанти фінансування, доступні в рамках моделі Flex on Demand.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero RAD Studio Delphi 10.3.2 Rio Architect – це найшвидший спосіб створювати й обновляти інтенсивно працюючі з даними, сильно взаємодіючі застосунки з візуально насиченим користувальницьким інтерфейсом для Windows 10, Mac, мобільних пристроїв, IoT і інших платформ за допомогою Object Pascal і C++. Широкий вибір функцій підтримки Windows 10, у тому числі нові компоненти VCL для Windows 10, стилі для VCL і FMX, а також служби UWP (універсальної платформи Windows), наприклад повідомлення, дозволяють легко й швидко перенести застосунки в Windows 10, зберігши користувачів. Нова платформа дозволяє підтримувати великі проекти на більшому числі платформ із подвоєним обсягом пам'яті в середовищі розробки й удвічі більшим розміром підтримуваних проектів. Крім того, підтримка декількох моніторів і десятки нових функцій середовища розробки, призначених для прискорення створення коду, зроблять роботу як ніколи ефективною. За допомогою RAD Studio 10 розроблювачі зможуть створювати застосунки в 5 разів швидше в порівнянні з іншими інструментами, а розробка застосунків для декількох настільних, мобільних, хмарних платформ і платформ баз даних, включаючи 32і 64 -розрядні версії Windows 10, Mac OS X, iOS і Android, стане ще швидше.

Зміни у версії 10.3 Rio:

– Створюйте міжплатформені застосунки. 80% інтернет користувачів мають смартфони й застосунки доступу, а також дані з мобільного пристрою й ноутбука / настільного комп'ютера, саме тому так важливо в цей час, щоб застосунки працювали в будь-якому пристрої.

– У всіх версіях Professional, Enterprise і Architect RAD Studio 10.3 надається підтримка процесу розробки застосунків для мобільних пристроїв. Розроблювачі RAD Studio кодують лише один раз, компілюють споконвічно для

кожної платформи, що скорочує час і трудозатрати на вивчення декількох мов і дозволяє паралельно управляти циклами розробки.

– Підтримка Android API26, відповідність вимогам Google Play Store відносно нових застосунків із серпня 2018 року й відновлення застосунків з листопада 2018 року.

– Власні елементи керування Android і стилізовані елементи керування FMX в одній і тій же формі Android, включаючи тему матеріального дизайну для Android 5.0 або вище.

– Підтримка iOS 12 (32і 64-біт) для створення App Store і корпоративних застосунків.

– Підтримка смайликів Юнікод.

– Програмуйте по-своєму. Завдяки двом новим темам самостійне налаштування інтегрованого середовища розробки для відповідності вашому стилю кодування ще ніколи не була настільки простій.

– Темне й світле оформлення Незалежно від того чи волієте ви кодувати вночі або у світлий час доби, завдяки темному й світлому оформленню RAD Studio ви можете вибрати потрібний вам стиль. Було доведено, що темне оформлення допомагає знизити зорову напругу в умовах низького освітлення, дозволяючи вам працювати більш продуктивно вночі. Немає нічого простіше, ніж перейти від темного до світлого оформлення й навпаки за допомогою меню панелі інструментів.

– Виконаєте користувальницьке налаштування свого середовища розробки Поліпшена програма установки інтерфейсу користувача й менеджера ліцензій інтерфейсу користувача дозволяє визначити ті можливості, які необхідні й опустити непотрібні, незалежно від того чи розробляєте ви застосунки для декількох платформ або всього однієї.

– Чистий, оновлений інтерфейс користувача інтегрованого середовища розробки Знайдіть потрібні можливості. Швидко. Головне вікно інтегрованого середовища розробки відцентровано й відрізняється високим ступенем читаності.

Ви з легкістю визначите, де перебуває область фокусування клавіатури з оновленими змінами фонових квітів фокуса. Вкладки редактора більше, що полегшує читання шрифтів, тому ви можете швидко внести зміни й зберегти кодування.

– Чудові застосунки Windows з VCL. Бібліотека візуальних компонентів (Visual Component Library, VCL) пропонує просту й візуальну розробку користувальницького інтерфейсу застосунки, у версії 10.3 представлені нові відновлення, які дозволять вашим застосункам виглядати сучасними й свіжими.

– Розширена підтримка HighDPI. Завдяки новому елементу керування VCL High DPI ImageList у версії 10.3 розроблювачі, що створюють нові застосунки VCL для Windows або он овлюючі існуючі застосунки для High DPI дисплеїв, можуть повністю підтримувати зроблені до рівня пікселів зображення зі змінною розв'язною здатністю на всіх елементах керування, а також будь-яке користувальницьке креслення, що вимагає масштабованих зображень для моніторів з різною розв'язною здатністю.

– Підтримка Per Monitor V2. Переконаєтеся, що ваш додаток масштабується правильно для всіх типів масштабування в Windows, реагуючи на зміни масштабування DPI на різних екранах під час виконання.

– Розширена підтримка Windows 10 і WinRT API. Сюди відноситься ряд ключових API-інтерфейсів WinRT і останні API-інтерфейси Windows 10, включаючи готові до використання компонента для убудованих у застосунки покупок і випробувань у магазині Windows 10 Store.

– Розгортання застосунків на основі служб за допомогою RAD Server. Продуктивність RAD Server була значно поліпшена завдяки десятикратному збільшенню потужності відносно простих операцій.

– Нові компоненти обробки JSON допоміжного засобу.

– Розширена підтримка RAD Server для клієнта Ext JS. Об'єднайте зовнішній інтерфейс javascript і веб-службу, підтримувану REST Server REST. (У версії Architect тепер включена ліцензія ExtJS Professional).

– Версії Enterprise включають ліцензію для одиничного розгортання RAD Server.

– Версії Architect включають ліцензію для розподіленого розгортання RAD Server.

– Що нового в C++? Підтримка C++17 Win32 збільшує продуктивність, поліпшує роботу компілятора й прискорює процес кодування. Були оновлені RTL і STL.

– Нова версія STL/Dinkumware 2018 для Win32 і Win64.

– Поліпшене автодоповнення коду Автодоповнення коду для даного компілятора тепер асинхронне, швидше й із кращими результатами, чим у попередньому автодоповненні коду C++. Уведення тексту не буде припинятися, поки виконується розрахунок.

– Тепер є підтримка налагодження для оптимізації компонувань.

– 2X швидкість математичної продуктивності для Win64.

– Нові додаткові лабораторії C++ в GetIt.

– Нові й поліпшені можливості роботи з базами даних. InterBase 2017 / IBToGo 2017 в RAD Studio. Версії Professional включають ліцензію розроблювача InterBase 2017, у той час як версії Enterprise і Architect містять у собі ліцензії InterBase ToGo. InterBase ToGo доповнена можливістю шифрування, функціями зміни подань, призначених для простої синхронізації даних застосунку по підписці без обмежень за розміром файлу бази даних.

– Поліпшена й оновлена підтримка для популярних баз даних, включаючи mySQL v8.0, MariaDB 10.3, SQL Server 2017, PostgreSQL v10, Firebird v3.0, MongoDB, InterBase, SQLite 3.23.1, SQL Anywhere і багатьох інших.

– Удосконалення DataSnap.

– Поліпшення REST. Підтримка додаткових родинних REST методів, типів і властивостей.

– Повністю оновлений модуль живлення версії Architect. Одержіть більше від версії Architect, включаючи ці ліцензії сімейства Idera.

– Ліцензія Sencha ExtJS Professional: Створіть свій ідеальний мережний вхідний інтерфейс за допомогою javascript і ExtJS.

– Ліцензія на розгортання InterBase ToGo. Додайте сховище даних у свої застосунки за допомогою цієї гнучкої, зашифрованої бази даних, що вбудовується.

– Ліцензія для розподіленого розгортання RAD Server. Ідеально підходить для серверного застосунку архітектури мікросервісів.

– Ліцензія AquaData Studio. Вражаючий аналіз бази даних.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи віртуалізованої серверної інфраструктури на базі HCI.

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

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

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

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

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

д) розробити рекомендації по організаційних та методичних заходах, які

забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Технологія VDI

Кожний співробітник, приходячи на роботу, повинен одержати доступ до своїх програм. IT відділ повинен забезпечити безперервну якісну роботу всіх застосунків. Класичний спосіб – це виділений кожному користувачеві комп'ютер, що вимагає первісного налаштування й подальшого обслуговування. Альтернативний спосіб рішення завдання – VDI, служби віртуальних робочих столів.

У сучасному бізнесі доступ до корпоративних сервісів повинен бути доступний співробітникам з будь-яких доступних пристроїв і не обмежується стаціонарним комп'ютером. Тому VDI у реалізації зараз став складніше, ніж був ще кілька років назад.

Проблематика (ПК)

Нема сенсу впроваджувати нову технологію заради технології. Якщо в компанії немає проблем з експлуатацією парку комп'ютерів, користувачі задоволені, керівництво досить, то потрібно серйозно задуматися потрібно чи щось міняти. Інша справа якщо є проблеми, які не вдається вирішити протягом довгого років. Нижче перераховані класичні проблеми, з якими зіштовхуються IT відділи при обслуговуванні інфраструктури персональних комп'ютерів:

– простій у роботі, співробітник не може виконувати свої безпосередні обов'язки під час виконання процедур по обслуговуванню ПК.

– персонал, для обслуговування ПК необхідний штат співробітників. Чим більше комп'ютерів у компанії, тим більше штат.

– програмне забезпечення, установлене на ПК жадає від IT співробітників своєчасного відновлення, налаштування, оптимізації.

– безпека, особистих даних користувачів змішуються з корпоративною інформацією. Співробітники мають фізичний доступ до інформації, що зберігається на дисках ПК.

– застарілий парк ПК, у деяких компаніях комп'ютери не міняються десятиліттями, швидкість роботи на таких пристроях украй низька.

– філіальна мережа, викликає складності із часом реагування на інциденти, якщо потрібне особиста присутність адміністратора для його рішення.

– і багато чого іншого.

Керівники IT з більшим інтересом ставляться до нових технологій, відвідують виставки, семінари, слухають представників вендорів. Якщо розглядати VDI, то реальну картину до впровадження, під час впровадження й після впровадження представити по маркетингових слайдах неможливо. Потрібно співробітничати з командою, що вже робила подібні проекти й зможе попередити про прийдешні труднощі.

Основні принципи VDI

Якщо раніше VDI був дивиною для IT, то зараз складно знайти велику компанію, де його немає зовсім. Часто це пілотні проекти, які були зроблені безкоштовно й працюють роками після впровадження. Нижче перераховані основні концепції які описують робочий процес, якої він повинен бути після впровадження VDI. Що повинна принести нова технологія і як вона спростить роботу користувачам і підніме на новий щабель знань адміністраторів:

– Співробітник зі свого робочого місця, ноутбука або мобільного пристрою підключається до свого віртуального робочого місця, що перебуває на серверах організації.

– Користувач одержує доступ тільки до дозволеним адміністратором застосункам. Всі користувачі компанії діляться на групи, у кожної групи є список доступних застосунків.

– Керівництво розуміє, скільки і які ліцензії потрібно для роботи компанії і яких співробітників використовують ці застосунки.

– Програми налаштовуються й обновляються централізовано адміністратором і не вимагають переривання робочого процесу співробітників.

– Кожний користувач одержує доступ до виділеної віртуальної машини, запущеної на сервері. Віртуальна машина створюється й запускається автоматично. Якщо співробітник не працює, то серверні ресурси не використовуються.

– Всі віртуальні машини ізольовані друг від друга, тому сесії користувачів не перетинаються й не впливають один на одного.

– Всі дані користувача зберігаються на серверах компанії й надійно захищені від втрати у випадку збоїти або поломки пристрою користувача

– Після впровадження проекту VDI значно знижується навантаження на службу технічної підтримки, так як не потрібно особистої присутності для рішення інцидентів. Всі процедури уніфіковані, стандартизовані й надійні. Один адміністратор може управляти 1000 і більше робочих місць.

– Саме робоче місце користувача теж змінюється й, в остаточному підсумку, перетворюється в тонкий клієнт, пристрій, що лише транслює зображення на монітор і не може зберігати яку-небудь інформацію на своїх дисках.

– Користувач може підключитися до свого віртуального робочого місця VDI з будь-якого тонкого клієнта, що перебуває в корпоративній мережі.

– Масштабування кількості робочих місць або філій відбувається в рази швидше.

– Це, звичайно, не все. Кожна компанія може знайти свої плюси від впровадження VDI.

Склад проекту VDI

Проект VDI «під ключ» – це програмно-апаратний комплекс, що складається із трьох основних елементів: устаткування, ліцензії, роботи з гарантійною підтримкою.

Склад устаткування**,** виробник програмного забезпечення для VDI можуть варіюватися залежно від переваг замовника, результатів тестування, корпоративних політик. Можна умовно розділити необхідне для проекту устаткування на серверні потужності, систему зберігання даних, мережа передачі даних.

Склад ліцензій залежить від необхідного замовникові функціонала, результатів тестування продуктів, набору вже наявних у замовника ліцензій. Для реалізації проекту VDI потрібні ліцензії двох типів: ліцензії на продукт VDI і ліцензії Microsoft.

Склад робіт із впровадження й подальшій технічній підтримці обговорюється із замовником і, по можливості, частину дій, які замовник може зробити самотужки залишається за ним.

Комбінації трьох елементів складаються в різні комерційні пропозиції (КП), які відрізняються друг від друга за ціною й функціональними можливостями.

Можливості VDI продуктів

Так як 30% вартості проекту VDI становлять ліцензії VDI, всім замовникам цікаво, чим саме друг від друга (крім ціни) відрізняються функціональні можливості продуктів і редакцій продуктів. На це питання єдино правильної відповіді немає. Але можна перелічити загальний функціонал, що є в сучасних продуктах VDI лідируючі позиції серед яких займають: VMware Horizon, Citrix XenDesktop, MS RDS. Про правила ліцензування кожного продукту буде розказано в окремих статтях, а поки подивимося, який загального функціонала містять у собі ліцензії VDI.

1. Центральна консоль керування всією інфраструктурою VDI, – через консоль ви створюєте віртуальні машини VDI з «золотого образа», через консоль установлюєте правила включення, вимикання ВМ. Тут адміністратор призначає робочі станції VDI, які будуть доступні користувачам і багато чого іншого.

2. Керування застосунками за допомогою шарів.

3. Можливість створення для користувачів наступних видів віртуальних робочих місць:

– виділена віртуальна машина, що не буде прив'язана до «золотого образа»;

– віртуальна машина з «золотого образа», з можливістю збереження інформації;

– організація доступу до фізичного комп'ютера (крім Microsoft);

– обнуляєма після виходу віртуальна машини з «золотого образа»;

– віртуальна машина зі збереженням змін.

4. Керування профілями користувачів, відділення даних від операційної системи. Користувачі з однієї групи одержують доступ до однакових віртуальних машин, але після входу користувача в сесію відбувається персоналізація. Завантажується профіль користувача, підключаються переспрямовані папки й мережні диски, підключаються мережні й локальні принтери, стають доступними для запуску пакетованого застосунка. Базовий функціонал надає Microsoft в Roaming Profile, більше розширений в VMware – Persona management, і в Citrix – Profile management.

5. Убудовані механізми підключення користувачам принтерів.

6. Існують клієнти для доступу до VDI для операційних систем: Windows, Linux, iOS, Android і через браузер з підтримкою HTML5 (крім Microsoft).

7. Підтримка роботи в локальній мережі LAN, і віддалений доступ WAN.

8. Підтримка аудіо й відеоконференцій в Microsoft Lync, Skype, Cisco, приблизно, з однаковим набором обмежень.

9. Можливість роботи USB накопичувачами, підтримка смарт карт, ключів E-token.

10. Проброс у сесію локальних дисків пристрою, з якого відбувається підключення для обміну файлами.

11. Підтримка локально підключених сканерів, принтерів і інших USB пристроїв.

Різниця, як завжди, криється в дріб'язках, інновація нових версій і саме вони в проектах часто схиляє чашу вибору убік того або іншого виробника. Досить, складно складе таблицю порівняння, із вказівкою параметрів, що розрізняються, продуктів, так як вона занадто швидко застаріє з виходом нових версій, update-тів, патчів.

У цілому, технологія VDI зараз розвилася до дуже високого рівня й, якщо говорити про прості офісні комп'ютери, які не виконують специфічних завдань, то переводити їх у віртуальне середовище можна не побоюючись за результат. Звичайно, якщо робити проект відповідно до рекомендацій виробників. Для заміни комп'ютерів, що використовують 3D графіку потрібне тестування й усвідомлений вибір.

Обґрунтування проекту VDI

Найважливішу роль у проекті VDI грає керівник IT відділу замовника, що повинен довести бізнесу (своєму керівництву), що цей проект для бізнесу буде корисний в економічному плані. Для порівняння поточних витрат на зміст парку ПК і, як альтернативи, проекту VDI складається таблиця техніко-економічного обґрунтування. У неї максимально докладно вносяться по категоріях всі існуючі капітальні й операційні витрати, після чого можна зрівняти два вектори розвитку.

У спрощеному варіанті всі розрахунки зводяться до вартості робочого місця в рік. Ціна одного робочого місця ПК зараз становить 900$ доларів у рік, а витрати на робоче місце VDI складуть 800$.

Нижче буде наведений простий розрахунок, що покаже із чого складається вартість робочого місця VDI і як вона виглядає в порівнянні з вартістю звичайного ПК. У розрахунки заставляються витрати компанії на електроенергію, обслуговування, хоча це, безсумнівно, могло б зіграти на руку VDI, але й створити ґрунт для непотрібної дискусії, так як у всіх ці витрати різні.

Отже, вартість робочого місця VDI складається із серверного устаткування, системи зберігання даних і мережі зберігання даних, ліцензій на VDI, ліцензій на Microsoft і вартості тонкого клієнта. Уже після основних витрат щороку потрібно оплачувати технічну підтримку з підпискою або тільки підпискою, щоб мати можливість обновлятися.

Для VMware і Citrix вважалася архітектура, коли файли віртуальних машин зберігаються на локальних SSD дисках, а для Microsoft RDS, що не вміє так працювати, архітектуру зі зберіганням файлів на SSD дисках системи зберігання, тому вона вийшла дорожче. Я не привожу тут докладні розрахунки, щоб не захаращувати текст.

У перший рік, незважаючи ні на що, нижче всього виявляється вартість VDI від Microsoft , потім іде Citrix, а VMware через високу ціну на ліцензії й технічну підтримку виявляється найдорожчим. Різниця в ціні між варіантом покупки ліцензії Windows server Datacenter і варіантом придбання річної підписки MS VDA мінімальна, але на другому році використання ліцензії не зажадають додаткових вкладень, а підписку прийдеться продовжувати.

Щороку потрібно буде оплачувати підписки, усього виходить 5 варіантів для VDI і один простій для персональних комп'ютерів, які до кінця п'ятого року потрібно буде замінити. Найменше вкладень зажадає Citrix у варіанті з Windows server Datacenter, так як передплата на ліцензії Citrix коштує мізерно мало (у порівнянні з конкурентами), а серверні ліцензії не вимагають вкладень. VMware у варіанті з VDA найдорожче зв'язування через сполучення високої вартості технічної підтримки VMware і високої вартості підписки MS VDA.

Переваги й недоліки

Напевно, не варто говорити, що у всього є свої пе реваги й недоліки. Такі є й у технології віртуалізації робочих місць. Щоб зрозуміти, чи потрібна вона вам, потрібно розуміти переваги й недоліки цієї технології. Багато про що вужі було написано в цій статті, але спробуємо узагальнити.

До переваг технології можна віднести:

– Віртуальні декстопи набагато безпечніші, ніж фізичні. Дані зберігаються в дата-центрі – їх набагато складніше украсти або ушкодити. Дата-Центр апріорі краще захищений, чим робоче місце співробітника. Тут і фізична охорона, і резервне копіювання, і можливість у будь-який момент заблокувати певне робоче місце, якщо на ньому замічена підозріла активність.

– Простота керування – віртуальними комп'ютерами простіше управляти, чим звичайними.

– Економія засобів – простота керування знижує витрати на адміністрування комп'ютерного парку. Тим більше, що тепер не потрібно витрачатися на заміну компонентів звичайних комп'ютерів, які періодично виходять із ладу.

– Скорочення часу простою – віртуалізація дозволяє скоротити час простою при збої до мінімуму. У випадку з фізичними ПК необхідно час на покупку (вибір, оплата, доставка) і заміну комплектуючих. Типовий приклад – вихід з ладу жорсткого диска локального ПК. Жорсткі диски настільних комп'ютерів, на жаль, не так надійні, як нам би цього хотілося й через 3 роки збільшується ймовірність їхнього виходу з ладу. Скільки часу знадобиться на усунення збоїти? Покупка нового диска, переустановка операційної системи, відновлення даних з резервної копії – мінімум полудня або хоча б півтора-друга година, якщо жорсткий диск був у наявності. Відновлення ВМ зі снапшота – справа декількох хвилин. От і вся різниця.

Тепер про недоліки. Найбільший недолік віртуалізації робочих столів – необхідність серйозних вкладень. Тут діє правило: якщо хочеш гроші заощадити, спочатку прийде їх витратити. Хоча знов-таки багато чого залежить від поставлених цілей і розв'язуваних завдань. Якщо ви – починаюча компанія, який потрібні потужні комп'ютери і якої не хочеться (чи ні можливості) витрачатися на їхнє придбання (ще не зрозуміло – піде чи справа ні), тоді доцільно орендувати

VDI і серйозних вкладень не буде. А зекономлені засоби можна смів витратити на рекламу своєї компанії – так буде вигідніше.

Зовсім інша справа, якщо ви – велика компанія, що бажає забезпечити безпека своїх даних (зберігання яких ви навряд чи довірите сторонньому дата-центру) і скоротити витрати на IT, тоді прийде інвестувати серйозні засоби – вам знадобиться потужне "залізо", система зберігання даних, а також програмне забезпечення віртуалізації. Звичайно, можна використовувати й гібридну схему – наприклад, використовувати орендовані VDI для певних співробітників (співробітники call-центра, наприклад), що дозволить заощадити гроші на покупку комп'ютерних систем і їхню модернізацію, але половинчасті рішення для великого бізнесу не дуже привабливі, тому або прийде витратитися або VDI вам не підходить.

Якщо плануєте створювати інфраструктуру віртуалізації самотужки, зверніть увагу на вже готові системи. Застосування таких систем допоможе заощадити час, а в перспективі – гроші. Прикладом таких систем можуть бути HPE Apollo Systems, PureFlex System і деякі інші. Крім серверної системи ще знадобиться система зберігання даних. Можемо порекомендувати рішення HPE 3PAR, Dell EMC XtremIO, Fujitsu AF250 S2.

VDI на базі HCI

Як же підвищити ефективність впровадження VDI? Останні п'ять років ряд вендорів пропонують для цього так звані гіперконверговані комплекси (HyperConverged Appliance або HyperConverged Infrastructure, HCI), які просуваються як альтернатива класичному набору з інтегрованих між собою серверів і системи зберігання. Це рішення часто використовується для розгортання інфраструктури віртуальних десктопів.

Гіперконвергентна інфраструктура являє собою об'єднані в кластер стієчні або модульні сервери, на базі яких розгорнуте програмно-визначаєме сховище (Software Defined Storage, SDS). Замість окремого дискового масиву за допомогою спеціального програмного забезпечення SDS реалізується віртуальне

дискове сховище, що складається із внутрішніх дисків серверів HCI. Відмова від використання окремого дискового масиву дозволяє істотно знизити вартість як зберігання даних, так і всієї апаратної платформи, що застосовується для впровадження VDI.

Крім того, гіперконвергентний комплекс – це повністю готовий до роботи інтегрований продукт, що набагато простіше встановити, настроїти й обслуговувати, чим набір серверів і СЗД. Тому він краще підходить для проектів впровадження VDI у невеликих організаціях і віддалених філіях компаній, де немає свого IT-фахівця з обслуговування серверного устаткування.

Ще одна перевага гіперконвергентного підходу – масштабування. Покупець може приступитися до реалізації пиіотного проекту впровадження VDI з розгортання гіперконвергентного комплексу в мінімальній конфігурації (звичайно три або чотири вузли), а потім – у міру збільшення в компанії числа віртуальних робочих столів – докуповувати вузли й приєднувати їх до інфраструктури, нарощуючи процесорні потужності і ємність програмно-визначаємого сховища апаратної платформи VDI.

Як упоратися з BOOT STORM

Для ефективної підтримки віртуальних десктопів система зберігання даних у гіперконвергентній інфраструктурі повинна демонструвати високі показники продуктивності уведення-виводу (Input/Output Per Second, IOPS) на рівні дорогих дискових масивів середнього або старшого класу. Справа в тому, що при використанні в організації віртуальних десктопів більшість із них активуються користувачами майже одночасно на початку робочого дня, і для кожного екземпляра виконується зчитування даних його операційної системи й застосунків, які централізовано розміщаються на системі зберігання, що обслуговує VDI (такий процес одержав назву boot storm).

Коли мова йде про десятки, сотні й навіть тисячі віртуальних робочих столів, необхідно виконати це зчитування максимально швидко, інакше користувачам щодня ранком прийде довго чекати, поки їхнє робоче місце буде

готово до роботи. Крім того, навантаження на СЗД, що обслуговує інфраструктуру VDI, різко зростає під час сканування всіх віртуальних десктопів на віруси й установки відновлень або заплаток операційної системи й прикладного ПЗ, які в багатьох організаціях виконуються щодня (правда, в останньому випадку основне навантаження пов'язана з операціями не читання, а запису).

Застосування накопичувачів на основі флеш-пам'яті як в окремому дисковому масиві, що обслуговує VDI, так і в гіперконвергентних комплексах дозволяє істотно підвищити показники IOPS для системи зберігання й ефективно вирішити проблему boot storm. Коли для розміщення образів використовуються флеш-накопичувачі, завантаження навіть декількох сотень віртуальних робочих столів займає всього кілька секунд, а на підготовку робочого місця користувача йде менше часу, чим у випадку стандартних настільних ПК.

Крім того, технології SDS у сполученні із флеш-пам'яттю забезпечують істотне зниження часу доступу до даних, розташованим на іншому сервері (вузлі) інфраструктури, тому по швидкості читання даних програмно-визначаєме сховище на базі флеш-накопичувачів не уступає традиційному дисковому масиву з окремим RAID-контролером.

3.2 Розробка структурної схеми

Що являє собою гіперконвергентна інфраструктура?

Гіперконвергентна інфраструктура (HCI) поєднує обчислювальні, мережні ресурси й ресурси зберігання в єдину систему. Це оптимізоване рішення на базі ПЗ й серверів x86 заміняє спеціалізоване устаткування. Гіперконвергентна інфраструктура дає можливість спростити середовище ЦОД і поліпшити його масштабованість.

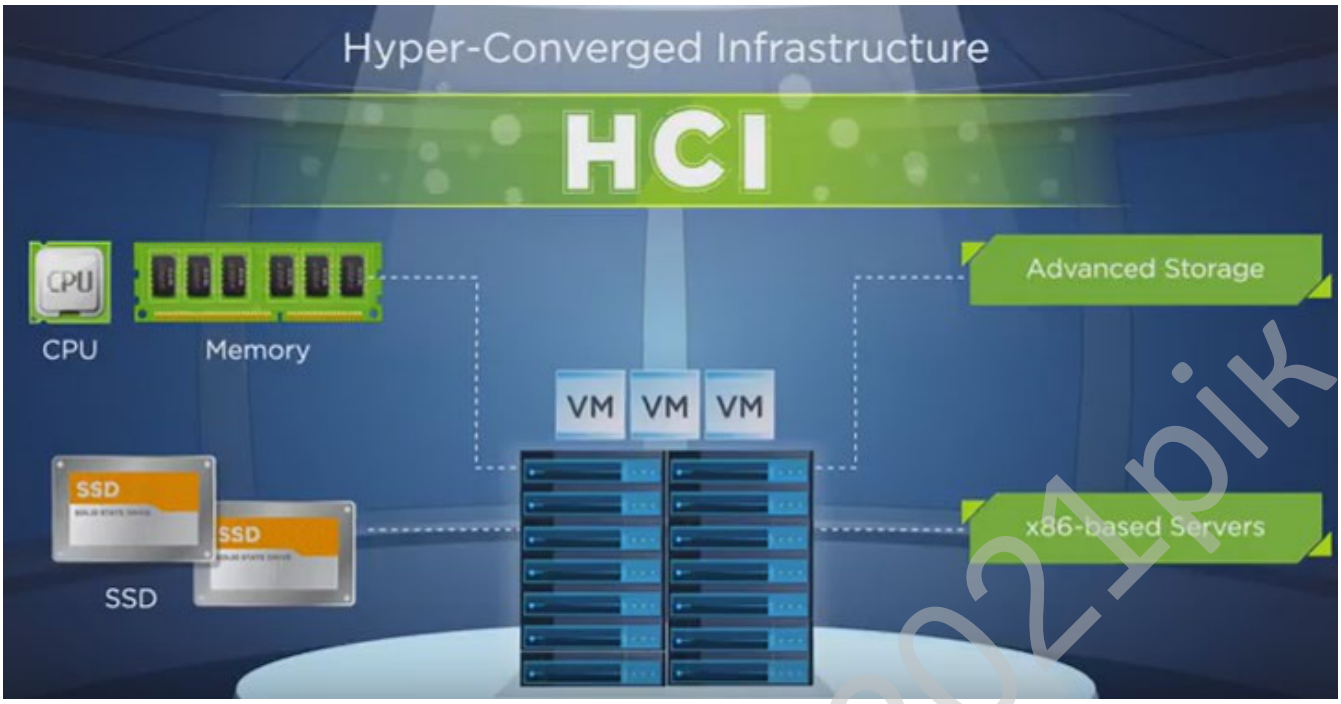

Рисунок 3.1 – Структурна схема системи

Традиційна трирівнева архітектура вимагає більших витрат на створення, складна в експлуатації й масштабуванні. Не потрібно чекати, поки IT-інфраструктура зможе підтримувати потреби застосунків. Розгорніть гіперконвергентну інфраструктуру без втрати контролю, підвищення витрат і зниження рівня безпеки.

Принцип роботи гіперконвергентної інфраструктури:

– Всі важливі процеси ЦОД виконуються на тісно інтегрованому програмному рівні, а не на спеціалізованому устаткуванні.

– Гіперконвергентна платформа складається із трьох програмних компонентів: засобів віртуалізації сховища, засобів віртуалізації обчислювальних ресурсів і системи керування.

– ПЗ для віртуалізації абстрагує й поєднує в пули базові ресурси, а потім динамічно виділяє їхнім застосункам, які виконуються у ВМ або контейнерах.

– Конфігурація визначається політиками, які враховують потреби застосунків, що усуває необхідність у використанні таких складних об'єктів, як дискові й логічні томи.

– Розширені можливості керування скорочують число завдань, виконуваних вручну, і допомагають повністю автоматизувати експлуатацію.

Розходження між гіперконвергентною й конвергентною інфраструктурою

Гіперконвергентні й конвергентні IT-інфраструктури поєднують чотири компоненти ЦОД. При цьому гіперконвергентні системи роблять це за допомогою програмного забезпечення (тому вони не мають прив'язки до певного устаткування), а конвергентні рішення залежать від устаткування. Для створення конвергентної інфраструктури ЦОД використовуються в основному ті ж продукти, що й у традиційної IT-середовищу, тільки зі спрощеною архітектурою й оптимізованим керуванням.

Переваги гіперконвергентної інфраструктури перед традиційної трирівневою архітектурою

Спрощення експлуатації

Усунете процеси, виконувані вручну, і необхідність у розрізнених групах фахівців для керування різними процесами експлуатації. Один об'єднаний IT-відділ може виконувати моніторинг обчислювальних ресурсів і сховища й управляти ними, завдяки чому забезпечується економія часу співробітників.

Скорочення витрат

Скоротите капітальні витрати завдяки використанню горизонтально й вертикально масштабованої архітектури на базі стандартних серверів x86 і економічних спеціалізованих мереж. Надалі можна додавати ресурси в міру необхідності без переривання процесів.

Підвищена адаптивність

Прискорте реагування на швидко мінливі потреби бізнесу. Устаткування можна запустити за кілька годин, а робітники навантаження ініціалізуються за

кілька хвилин. Підвищте продуктивність важливих застосунків, наприклад реляційних баз даних.

Впровадження гіперконвергентної інфраструктури: три фактори, які варто врахувати

Переваги гіперконвергентної інфраструктури

Гіперконвергентна інфраструктура допомагає усунути розрізненість, характерну для традиційної IT-середовища, і реалізувати комплексне керування за допомогою єдиного засобу, завдяки чому можна знизити операційні й капітальні витрати, пов'язані з IT-інфраструктурою, без збитку для безпеки, гнучкості або масштабованості.

Яким образом рішення інтегрується з існуючим середовищем?

Для реалізації максимальних переваг і наступного переходу до повністю програмно-визначаємого ЦОД гіперконвергентна інфраструктура не повинна бути прив'язана до конкретної апаратної платформи. Гіперконвергентна інфраструктура VMware надає максимальну волю вибору й гнучкість, завдяки чому можна вибрати кращу платформу, забезпечити повну інтеграцію з наявною інфраструктурою, а також зберегти й оптимізувати існуючі технології.

Чи надає рішення можливість ефективно й економично збільшувати обсяг ресурсів у міру необхідності в зручний час, у будь-якому місці й зручному способі?

Масштабування ЦОД може бути дорогим і складним процесом. Гіперконвергентна інфраструктура VMware забезпечує масштабованість і зручність, допомагаючи прискорити реагування на швидко мінливі потреби бізнесу. На даний момент VMware – це єдиний постачальник, що пропонує повний набір продуктів для створення повністю програмно-визначаємого ЦОД у виробничому середовищі. Завдяки цьому забезпечується адаптивність для розгортання застосунків, стандартизації IT-рішень у різних середовищах і підготовки IT-середовища до використання загальнодоступних хмар.

3.3 Розробка функціональної схеми

Гіперконвергенція – об'єднання IT-ресурсів у цілісне рішення за допомогою типових апаратних блоків і програмно обумовлених пристроїв – стала одним із самих модних трендів в IT. Спробуємо розібратися в перевагах гіперконвергентних систем, напрямках подальшого розвитку рішень і сценаріях їхнього використання в бізнесі.

Етапи розвитку обчислювальних інфраструктур

Віртуалізація машин, що застосовувалася ще в радянський час, дозволила оптимізувати використання обчислювальних ресурсів комп'ютера, відокремити виконання коду від конкретного апаратного забезпечення й створила передумови до виникнення масштабованих віртуальних обчислювальних потужностей на blade-серверах, що містять, що додаються по необхідності «леза» – материнські плати із процесорами, установлювані в blade-кошики.

На початку 90-х з'явилася можливість нарощувати обсяг дискового простору шляхом додавання дисків у виділене сховище даних – СЗД. Для роботи з ними використовувалася розроблена в 1987 році технологія віртуалізації даних RAID, що поєднує диски в логічні елементи, з якими й працювали операційні системи.

Архітектура, що включає кластер комп'ютерів або блейд-серверів і пов'язаних з ними швидкісним каналом (fibre channel) СЗД, де-факто стала стандартом в 2005 році й донині використовується багатьма компаніями.

Цифрова трансформація ставить перед розроблювачами комп'ютерних систем всі нові завдання. Складність рішень постійно росте, обчислювальні системи потрібно запускати швидко, забезпечуючи необхідну гнучкість при подальших змінах конфігурації. У відповідь на запити ринку з'явилися хмарні технології, що надають обчислювальні потужності як сервіс на вимогу, і конвергентні рішення – заздалегідь інтегровані обчислювальні модулі, що

включають обчислювальні потужності, системи зберігання, мережне устаткування, засоби віртуалізації й оркестрації.

Раніше компанія купувала комп'ютери й СЗД, з'єднувала їх і розвертала платформу віртуалізації. Це вимагало багато часу й участі кваліфікованих фахівців. З появою конвергентних систем (у термінології Gartner – інтегрованих інфраструктур) стало досить купити заздалегідь інтегрований модуль (стійку), готовий для установки прикладних програм.

Наступний крок – розгорнути платформу віртуалізації на інтегрованих блоках або серверах, об'єднавши в єдине ціле їхні внутрішні диски/дискові полки на програмному рівні, у такий спосіб формуючи віртуальне відказостійке сховище. З погляду нарощування обчислювальних потужностей гіперконвергентні інфраструктури, що виходять (HCI – Hyper-Converged Infrastructure) уже не обмежені можливостями одного інтегрованого модуля пристрою або сервера – платформа поєднує їхні ресурси в єдине ціле, з якого можна нарізати потрібні для конкретного завдання обчислювальні потужності.

Весь «інтелект» HCI для забезпечення відказостійкісти й продуктивності опирається на програмно обумовлені технології (віртуальні машини, програмно обумовлені сховища).

Такий підхід сильно спрощує керування складною системою, що тепер здійснюється через загальну консоль адміністрування, забезпечує більшу гнучкість і масштабованість, а також виконання заздалегідь заданих вимог до відказостійкісти й продуктивності. При цьому можна обійтися без дорогих СЗД, відмовитися від мереж fiber channel і зовнішніх мереж зберігання даних – гіперконвергентні системи, як правило, складаються з комодіті-рішень, побудованих на стандартній для галузі архітектурі x86, що значно підвищує економічну ефективність їхнього використання.

Рисунок 3.2 – Функціональна схема системи

Сценарії застосування

HCI не є рішенням на всі випадки життя. Масштабування може вийти не збалансованим: наприклад, клієнтові потрібно тільки наростити обчислювальну потужність, а доводиться докуповувати весь блок. Крім того, є проблеми з розгортанням деяких застосунків у віртуальних середовищах.

Два основних сценарії використання гіперконвергентних систем – віддалені офіси й віртуалізовані робітники місця. При гомогенному навантаженні, як у випадку VDI, проявляється основна перевага

гіперконвергентной системи – легка масштабованість. Допустимо, сьогодні потрібно 1 тис. робочих місць – купуємо десять вузлів HCI. Завтра потрібно 1,5 тис. – купуємо ще п'ять. У такий спосіб сильно спрощується розгортання обчислювальної інфраструктури віддаленого офісу або філії. При цьому не треба шукати місцевих фахівців для запуску й підтримки системи, тим більше що їх може й не бути.

Втім, можливі й інші сценарії застосування. Наприклад, об'єднання різнорідних обчислювальних систем при злитті компаній легше здійснити при інтеграції на HCI-системах. Та й не тільки при об'єднанні – міграція с дорогих високонадійних обчислювальних систем, на яких працюють бізнес-критичні застосунки, на комодіті-рішення, використовувані в HCI, може заощадити багато грошей, не знижуючи при цьому показники надійності.

І, звичайно, гнучкі гіперконвергентні системи зручні для розробки й тестування нових бізнес-застосунків.

Перехід на HCI

Найпростіший спосіб перейти на гіперконвергентну систему – купити заздалегідь інтегрований програмно-апаратний комплекс. Такий підхід ефективний при розгортанні нових філій підприємств і банків, створенні нових компаній.

Але вже працюючі на ринку організації мають свою IT-інфраструктуру й не хочуть втрачати зроблені в неї інвестиції. У компаній існують програми закупівель, орієнтовані на конкретних вендорів, є експертиза по роботі з устаткуванням і програмним забезпеченням конкретних виробників.

Постачальники у відповідь на запит ринку почали пропонувати різні варіанти гіперконвергентних систем. Постачальники апаратних рішень забезпечують можливість роботи з різними системами віртуалізації. Так, Dell EMC пропонує гіперконвергентні системи на базі VMware, Nutanix і Azure Stack, але може поставити сервера й з усім інтегрованим стеком Red Hat.

Постачальники програмних рішень також провели диверсифікованість апаратних платформ.

Написати програму, що працює з конкретним апаратним комплексом, значно простіше, ніж забезпечити її роботу з усім спектром «заліза», представленим на ринку. Це завдання різного масштабу. Рішення подорослішали й тепер з певними обмеженнями можуть бути запущені практично на будь-якому стандартному устаткуванні. Але, як відзначив експерт по хмарних платформах компанії Dell EMC, отут є підводні камені. Робота 3D залежить від особливостей апаратної частини й версій її прошивання. Наприклад, продукт Dell EMC ScaleIO працює через гіпервізор і мало пов'язаний з «залізом». Але з певного часу його перестали продавати як окреме ПЗ, оскільки клієнти іноді встановлювали його на екзотичне самозбірне «залізо», а потім висували претензії компанії Dell EMC. Тому було ухвалене рішення продавати ScaleIO тільки на сертифікованому устаткуванні, по суті по моделі гіперконвергентної інфраструктури VxRail. Проводять сертифікацію конкретних серверів для своїх гіперконвергентних рішень компанії VMware і Red Hat.

Що далі?

У єдине ціле можна об'єднати не тільки обчислювальні потужності й системи зберігання даних, але й віртуалізовані мережі, що вже стає стандартом для HCI. Пропонуються рішення, інтегровані із системами резервування й аварійне відновлення даних.

Так, в Dell EMC крім класичної гіперконвергентної інфраструктури VxRail з'явилося більше складне гіперконвергентне рішення Dell EMC VxRack SDDC на базі VMware Cloud Foundation, куди входять віртуальне сховище VSAN і програмно обумовлені мережі VMware NSX. Рішення розширюється практично будь-якими продуктами VMWare, включаючи систему автоматизації керування IT-процесами VMware vRealize Operations, систему створення персоналізованої інфраструктури, застосунків і спеціалізованих IT-послуг VMware vRealize Automation.

Причому все це може поставлятися як попередньо інтегрований комплекс «з коробки».

Таким чином, гіперконвергентні рішення виходять на новий рівень. Якщо раніше пропонувалася тільки інфраструктура, на якій фахівці замовника «піднімали» свої системи, то тепер гіперконвергентні рішення пропонуються як платформа, на зразок хмарного PaaS. Це сильно полегшує праця розроблювачів, дозволяючи автоматизувати створення середовищ виконання програм, розгортання віртуальних машин, контейнерів і мікросервісів.

Ще один напрямок розвитку гіперконвергентних систем – надання їхнім хмарним провайдером в оренду по моделі dedicated. Компанія може не купувати HCI-систему, а користуватися нею із ЦОДу провайдеру по сервісній моделі. Або розгорнути в себе й масштабувати в ЦОД провайдеру, створюючи розтягнутий кластер. Аналогічний кластер можна створити, об'єднавши зі своєї локальної HCI-системою гіперконвергентну, розташовану по моделі colocation у ЦОДі провайдеру.

На Заході вже з'явилися пропозиції as-a-Service, наприклад на базі NetApp HCI. Це дозволить клієнтам створити гіперконвергентні рішення із хмарою VMware, об'єднавши локальні й хмарні віртуалізованні системи зберігання за допомогою vSAN у розтягнутий кластер

Сьогодні обчислювальна інфраструктура підприємств містить у собі усе більше різноманітних систем і її розвиток спрямований на перехід від забезпечення працездатності обчислювальних систем до забезпечення доступності й працездатності пропонованих клієнтові сервісів. У цих умовах одне з головних переваг гіперконвергентних систем – спрощення керування обчислювальною інфраструктурою, причому як на площадці замовника, так і в провайдеру.

3.4 Розробка діаграми процесів

Розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3. Основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей.

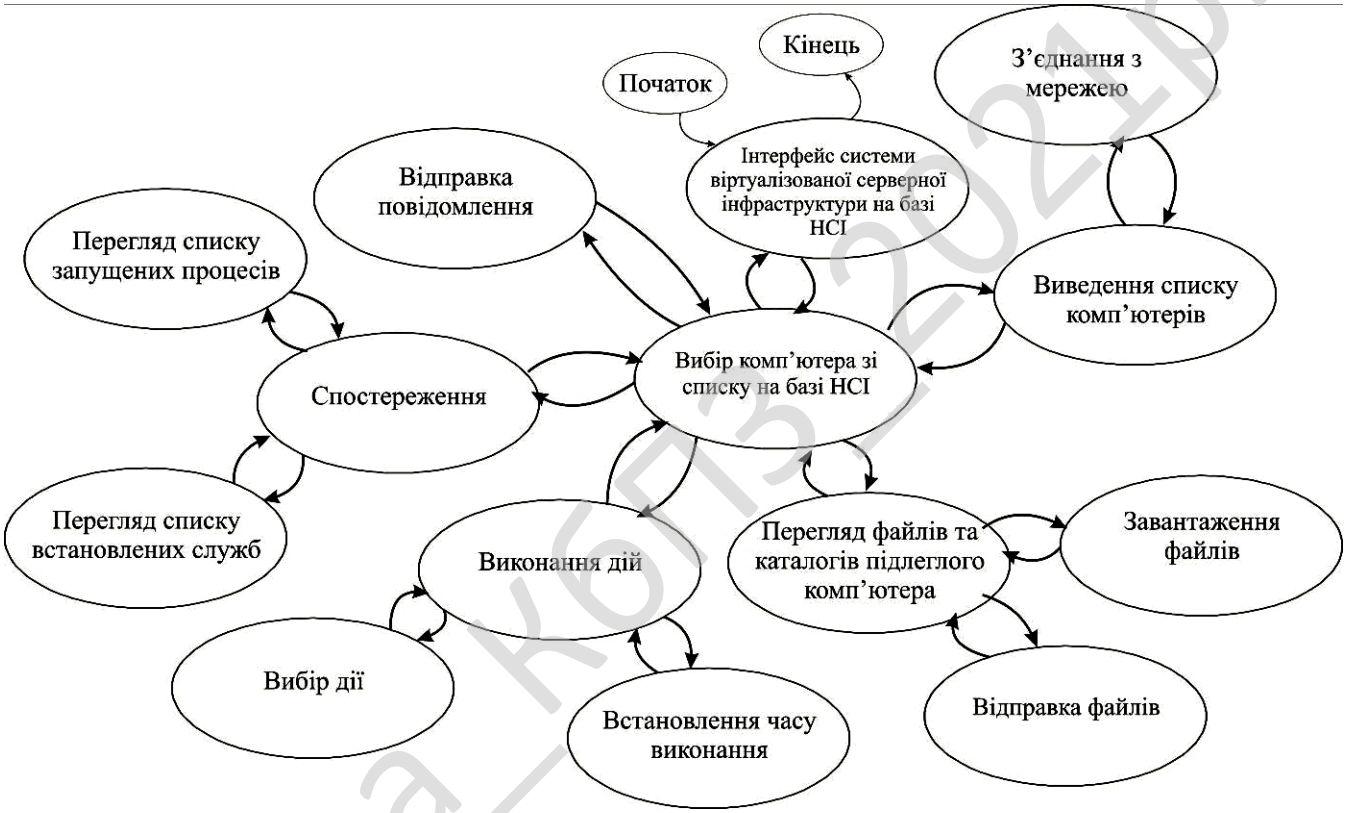

Рисунок 3.3 – Діаграма взаємодії процесів

Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників. Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

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

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

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

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

Діаграми потоків даних містять чотири типи елементів:

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

– Сховища даних (репозиторії).

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

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

Було створено блок-схеми роботи системи. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується. Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо.

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

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

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

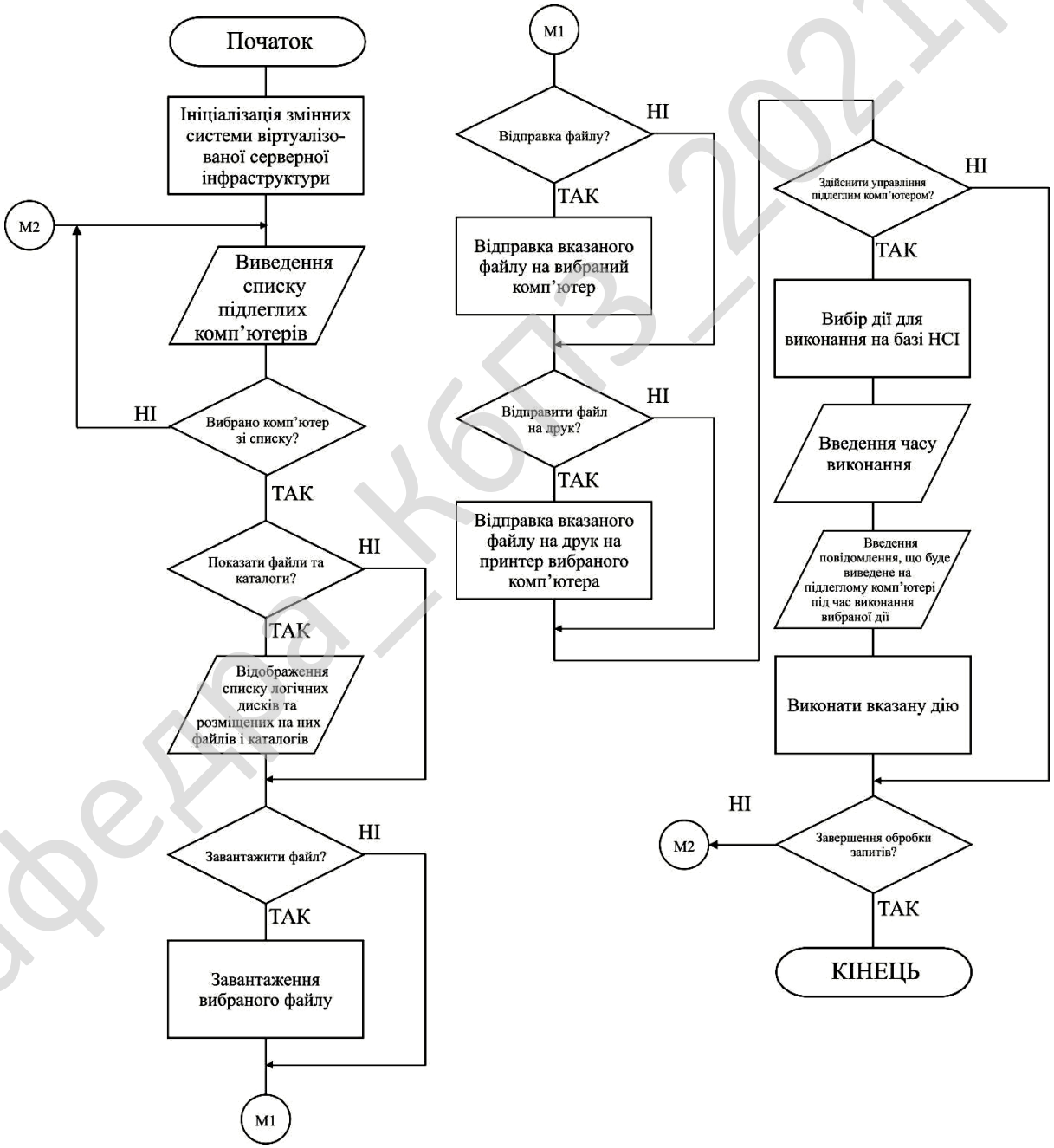

Рисунок 4.1 – Блок-схема основної програми

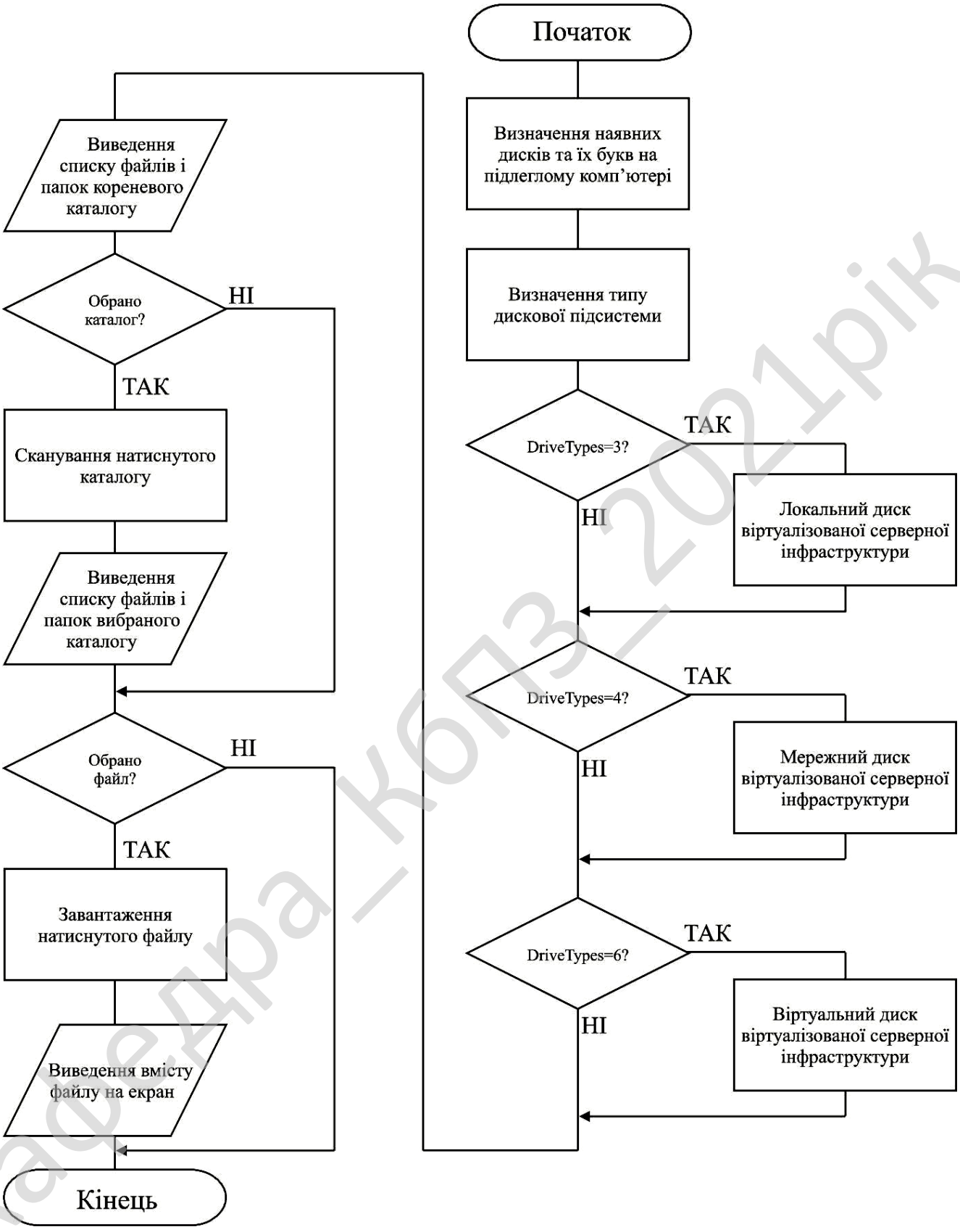

Рисунок 4.2 – Блок-схема роботи підпрограми

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

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

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

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

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

У 1991 був утворений консорціум ODMG, основною метою якого стало вироблення промислового стандарту об'єктно-орієнтованих баз даних. Між 1993

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

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

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

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

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

СКБД третього покоління повинні включити в себе СКБД другого покоління. Системи другого покоління внесли вирішальний вклад у двох областях – непроцедурний доступ за допомогою мови запитів SQL і незалежність

даних. Ці досягнення обов'язково повинні враховуватися в системах третього покоління.

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

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

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

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

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

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

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

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

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

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

Основні характеристики СКБД

1. Контроль за надлишковістю даних.
2. Несуперечливість даних.
3. Підтримка цілісності бази даних (коректність та несуперечливість).
4. Цілісність описується за допомогою обмежень.
5. Незалежність прикладних програм від даних.
6. Спільне використання даних.
7. Підвищений рівень безпеки.

Можливості СКБД

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

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

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

Основні компоненти середовища СКБД

1. Апаратне забезпечення.

2. Програмне забезпечення.

3. Дані.

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

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

Архітектура СКБД

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

Рівні:

1. Зовнішній – представлення БД з точки зору користувача.

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

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

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

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

Було використано **MongoDB** – документо-орієнтована система керування базами даних (СКБД) з відкритим вихідним кодом, яка не потребує опису схеми таблиць.

MongoDB займає нішу між швидкими і м асштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними і зручними у формуванні запитів.

Код MongoDB написаний на мові C++ і поширюється в рамках ліцензії AGPLv3.

MongoDB підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів, підтримує журналювання операцій зі зміни і додавання даних в БД, може працювати відповідно до парадигми Map/Reduce, підтримує реплікацію і побудову відмовостійких конфігурацій.

У MongoDB є вбудовані засоби із забезпечення шардінгу (розподіл набору даних по серверах на основі певного ключа), комбінуючи який з реплікацією даних можна побудувати горизонтально масштабований кластер зберігання, в якому відсутня єдина точка відмови (збій будь-якого вузла не позначається на роботі БД), підтримується автоматичне відновлення після збою і перенесення навантаження з вузла, який вийшов з ладу.

Розширення кластера або перетворення одного сервера на кластер проводиться без зупинки роботи БД простим додаванням нових машин.

При розробці автори виходили з необхідності спеціалізації баз даних, завдяки чому їм вдалося відійти від принципу «один розмір під усе». За рахунок мінімізації семантики для роботи з транзакціями з'являється можливість вирішення цілого ряду проблем, пов'язаних з нестачею продуктивності, причому горизонтальне масштабування стає простішим. Використовувана модель документів зберігання даних (JSON/BSON) простіше кодується, простіше управляється (у тому числі за рахунок застосування так званого безсхемного стилю (schemaless style), а внутрішнє угруповання релевантних даних забезпечує додатковий виграш в швидкодії.

Нереляційний підхід досить зручний для створення баз даних, у яких горизонтальне масштабування означає розгортання на множині машин. Можливість забезпечувати найкращу продуктивність повинна існувати паралельно з підтримкою більшої функціональності, ніж це дозволяє використання пар «ключ-значення» (у чистому вигляді).

Технологія баз даних має працювати скрізь, починаючи з серверів користувача та віртуальних машин і закінчуючи хмарними технологіями.

MongoDB, на думку розробників, має заповнити розрив між простими сховищами даних типу «ключ-значення» (швидкими і легко масштабованими) і великими СКБД (зі структурними схемами і потужними запитами).

Основні можливості MongoDB:

1. Документо-орієнтоване сховище (проста та потужна JSON-подібна схема даних).

2. Досить гнучка мова для формування запитів.

3. Динамічні запити.

4. Повна підтримка індексів.

5. Профілювання запитів.

7. Швидкі оновлення «на місці».

8. Ефективне зберігання бінарних даних великих обсягів, наприклад, фото та відео.

9. Журналювання операцій, що модифікують дані в БД.

10. Підтримка відмовостійкості і масштабованості: асинхронна реплікація, набір реплік і шардінг.

11. Може працювати відповідно до парадигми MapReduce.

СКБД управляє наборами JSON-подібних документів, що зберігаються в бінарному форматі BSON. Зберігання і пошук файлів в MongoDB відбувається завдяки викликам протоколу GridFS.

Подібно до інших документо-орієнтованих СКБД (CouchDB тощо), MongoDB не є реляційною СКБД.

Є докладна і якісна документація, велике число прикладів і драйверів для популярних мов Java, C++, C#, PHP, Python, Perl, Ruby.

При випуску одразу було заявлено, що реліз MongoDB 1.0 готовий до використання у виробництві як одиничний хост, так і у зв'язках master/slave.

Код цього релізу досить стабільний і перевірений в промисловій експлуатації протягом 1,5 років. MongoDB рекомендується розгортати мінімум на двох серверах використовуючи реплікацію Master/Slave.

Це забезпечує наявність актуальних даних при виході з ладу однієї з СКБД. MongoDB – продукт досить молодий, і відтак у ньому зустрічаються помилки, з'являються нові можливості тощо. Характерний високий темп розробки (проект пишуть не тільки волонтери, а й компанія людей на повній зайнятості). Компанія-розробник надає платні підтримку, хостинг, консультації.

Приклади запитів

Запити можуть витягати дані з вбудованих об'єктів та масивів. Якщо в колекцію users вставлений такий об'єкт:

```
{
   "username" : "bob",
   "address" : {
     "street" : "123 Main Street",
     "city" : "Springfield",
```

```
        "state" : "NY"
      }
    }
```

Ми можемо запитати цей документ (і всі документи з адресою в Нью - Йорку) за допомогою:

```
> db.users.find({"address.state" : "NY"})
```

Можна також запитати елементи масиву:

```
> db.food.insert({"fruit" : ["peach", "plum", "pear"]})
> db.food.find({"fruit" : "pear"})
```

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

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

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

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

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

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

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

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

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

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

ППІ, як правило, пов'язаний із бібліотеками програмного забезпечення: ППІ описує і вказує очікувану поведінку в той час, як біблі отека є фактичною реалізацією даного набору правил. Один ППІ може мати декілька реалізацій (або жодної, будучи абстрактним) у вигляді різних бібліотек, які мають такий же інтерфейс.

Прикладний програмний інтерфейс також може бути пов'язаним з платформами програмування: платформа може бути заснована на кількох бібліотеках реалізує декілька інтерфейсів ППІ, але на відміну від звичайного використання ППІ, доступ до поведінки вбудований в платформу

опосередкований шляхом розширення його змісту новими класами і вставлений в саму платформу. Крім того, загальний потік управління програми може бути під контролем абонента.

Прикладний програмний інтерфейс може бути також реалізацією протоколу.

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

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

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

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

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

орієнтованої архітектури (SOA) на більш прямі передачі репрезентативного стану (REST) стилів веб-ресурсів та ресурсів-орієнтованої архітектури (ROA).

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

Було використано метод розробки динамічних систем (Dynamic Systems Development Method, DSDM) – це головним чином методика розробки програмного забезпечення, що базується на концепції швидкої розробки додатків (Rapid Application Development, RAD). У 2007 році **DSDM** став основним підходом до управління проектом і розробки додатків. DSDM – це ітеративний і інкрементний підхід, який надає особливого значення тривалого участі в процесі користувача/споживача.

Мета методу – здати готовий проект вчасно і вкластися в бюджет, але в той же час регулюючи зміни вимог до проекту під час його розробки. DSDM входить в сімейство гнучкої методології розробки програмного забезпечення, а також розробок, що не входять у сферу інформаційних технологій.

Остання версія DSDM називається DSDM Atern. Назва Atern – це скорочення від Arctic Tern (пер. Полярна крачка). Полярна крачка – птиця, яка може подорожувати на великі відстані. Вона символізує безліч аспектів методу, наприклад визначення пріоритету або кооперування, які є природним способом ведення робочого процесу.

Попередня версія DSDM (випущена в травні 2003 року), яка все ще діє і широко використовується, – це DSDM 4.2, є дещо розширеною версією DSDM 4. Розширена версія містить настанови щодо того, як використовувати DSDM спільно з Екстремальним програмуванням (Extreme Programming).

Як розширення концепції швидкої розробки додатків, DSDM фокусується на проектах інформаційних систем, що характеризуються стислими термінами і

бюджетами. У DSDM присутні вказівки на типові помилки проектів інформаційних систем, таких як перевищення бюджету, запізнення з термінами здачі (виконання), недостатнє залучення користувачів і топ-менеджерів в роботу над проектом. DSDM складається з:

– трьох стадій: передпроектна стадія життєвого циклу проекту і постпроектна стадія;

– стадія життя проекту складається з 5 етапів: дослідження реалізованості, дослідження економічної доцільності, створення функціональної моделі, проектування і розробка, етап реалізації.

При деяких умовах існує можливість включення в DSDM частин інших методик, таких як Rational Unified Process (RUP), Екстремальне програмування, PRINCE2. Інший гнучкий метод, схожий на DSDM по процесу і концепції – Scrum.

Метод DSDM був розроблений у Великобританії в 1990-х Консорціумом DSDM. Консорціум DSDM – це асоціація розробників та експертів в області програмного забезпечення, створена з метою «спільної розробки і просування незалежного фреймворку RAD» комбінуванням кращого практичного досвіду учасників асоціації. Консорціум DSDM – це некомерційна організація незалежних розробників, які володіють та управляють фреймворком DSDM. Перша версія фреймворку була завершена в січні 1995 року і опублікована в лютому 1995 року. У липні 2006 року була представлена відкрита версія DSDM 4.2, яка стала доступна приватним особам для перегляду і використання. Тим не менш, всі, хто поширює DSDM, повинні бути членами цього некомерційного консорціуму.

На початку 1990-х в індустрії інформаційних технологій став поширюватись новий термін – швидка розробка додатків (Rapid Application Development, RAD). Інтерфейси прикладних програм еволюціонували від старих зелених екранів до графічних інтерфейсів користувача, які використовуються і зараз. На ринок почали виходити нові інструменти для створення додатків, наприклад PowerBuilder. Вони дозволили розробникам простіше ділитися

планованими розробками з покупцями – з'явилося прототипування і почалося руйнування класичних, послідовних (каскадних) методів розробки.

Тим не менш, новий рух RAD було дуже неструктурованим: не існувало узгодженого опису цього методу і у багатьох організацій були створені власні опису і підходи до нього. Безліч великих корпорацій були зацікавлені в перспективах, що надаються методом, але вони також були стурбовані тим, щоб не знизився рівень якості їх продукції в кінцевому результаті.

Консорціум DSDM був утворений в 1994 році, коли група людей зустрілася на заході, організованому Butler Group в Лондоні. Всі, хто прийшов на цей захід, працювали у великих організаціях, таких як British Airways, American Express, Oracle and Logica (такі компанії як Data Sciences і Allied Domecq з тих пір були поглинені іншими організаціями).

На цьому зібранні було вирішено, що Дженніфер Степлтон, тоді представляла компанію Logica, розробить архітектуру комлексного, орієнтованого на користувача методу з хорошим контролем якості для ітеративної і інкрементній розробки. Підсумкова архітектура була спроектована так, щоб бути повністю сумісна зі стандартом ISO 9000 і PRINCE2. Коли архітектура була готова (через місяць після зборів), Консорціум сформував групи для її поширення у всіх областях розробки програмного забезпечення, які включали в себе: методи та засоби управління проектом, контроль якості та тестування, методи і засоби розробки. Контрольна група, очолювана творцем архітектури і складається з глав цих груп, повинна була забезпечити розуміння методу так, як він спочатку замислювався.

Незважаючи на те, що багато членів Консорціуму були прямими конкурентами, вони вільно ділилися тим, як вони вирішують проблеми, що виникають. Практика показала, що найкращий результат може бути досягнутий тільки працюючи як одне ціле. Консорціум збільшився за перший рік від декількох організацій до шістдесяти; опис методу ставало все більш і більш повним. Версія 1 була сформована в грудні 1994 року і опублікована в лютому

1995 року. Результатом став універсальний метод, що охоплює людей, процеси та інструменти. Він сформувався на основі досвіду організацій, різних за родом своєї діяльності і розмірами.

Метод DSDM – принципи. Існує 9 принципів, що складаються з 4 основних та 5 початкових точок:

1. Залучення користувача – це основа ведення ефективного проекту, де розробники ділять з користувачами робочий простір і тому прийняті рішення будуть більш точними.

2. Команда повинна бути уповноважена приймати важливі для проекту рішення без узгодження з начальством.

3. Часта поставка версій результату, з урахуванням такого правила, що «поставити щось хороше раніше – це завжди краще, ніж поставити все ідеально зроблена в кінці». Аналіз поставок версій з попередньої ітерації враховується на наступній.

4. Головний критерій – як можна більш швидка поставка програмного забезпечення, яке відповідає поточним потребам ринку. Але в той же час постачання продукту, який задовольняє потребам ринку, не менш важлива, ніж вирішення критичних проблем у функціоналі продукту.

5. Розробка – ітеративна та інкрементна. Вона ґрунтується на зворотного зв'язку з користувачем, щоб досягти оптимальної з економічної точки зору рішення.

6. Будь-які зміни під час розробки – оборотні.

7. Вимоги встановлюються на високому рівні перш, ніж почнеться проект.

8. Тестування інтегровано в життєвий цикл розробки.

9. Взаємодія і співпраця між усіма учасниками необхідно для його ефективності.

Передумови для використання DSDM.

Щоб успішно використовувати DSDM, необхідно щоб був виконаний ряд передумов. По-перше, необхідно організувати взаємодію між проектною

командою, майбутніми користувачами і вищим керівництвом. По-друге, повинна бути можливість поділу проекту на менші частини, що дозволить використовувати ітеративний підхід.

Два приклади проектів, для яких DSDM не дуже підходить:

1. Проекти, критичні безпеки розширене тестування та затвердження в таких проектах конфліктують з метою методу DSDM укластися в терміни і бюджет.

2. Проекти, чия мета зробити компоненти багаторазового використання – вимоги в таких проектах занадто високі і не вкладаються в принцип 80 %/20%.

Життєвий цикл проекту. Фреймворк DSDM складається з трьох послідовних стадій, які називаються передпроектна стадія, стадія життєвого циклу проекту і постпроектна стадія. Стадія життєвого циклу проекту – сама продумана і детально розроблена з усіх інших. Вона складається з п'яти етапів, які формують ітеративний, інкрементний підхід до розробки інформаційних систем.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм Madryga. Алгоритм Madryga складається із двох вкладених циклів. Зовнішній цикл повторюється вісім разів (для гарантії надійності число циклів можна збільшити) і полягає в застосуванні внутрішнього циклу до відкритого тексту. Внутрішній цикл перетворює відкритий текст у шифртекст і виконується однократно над кожним 8-бітовим блоком (байтом) відкритого тексту. Таким чином, весь відкритий текст послідовно вісім разів обробляється алгоритмом.

Ітерація внутрішнього циклу оперує з 3-байтовим вікном даних, називаним робочим кадром (рисунок 4.3). Це вікно зрушується на 1 байт за ітерацію. (При роботі з останніми 2 байтами дані покладаються циклічно замкнутими). Перші два байти робочого кадру циклічно зрушуються на змінне число позицій, а для

останнього байта виконується операція XOR з декількома бітами ключа. У міру переміщення робочого кадру всі байти послідовно циклічно зрушуються й піддаються операції XOR із частинами ключа. Послідовні циклічні зрушення перемішують результати попередніх операцій XOR і циклічного зрушення, причому на циклічне зрушення впливають результати XOR. Завдяки цьому процес у цілому оборотний.

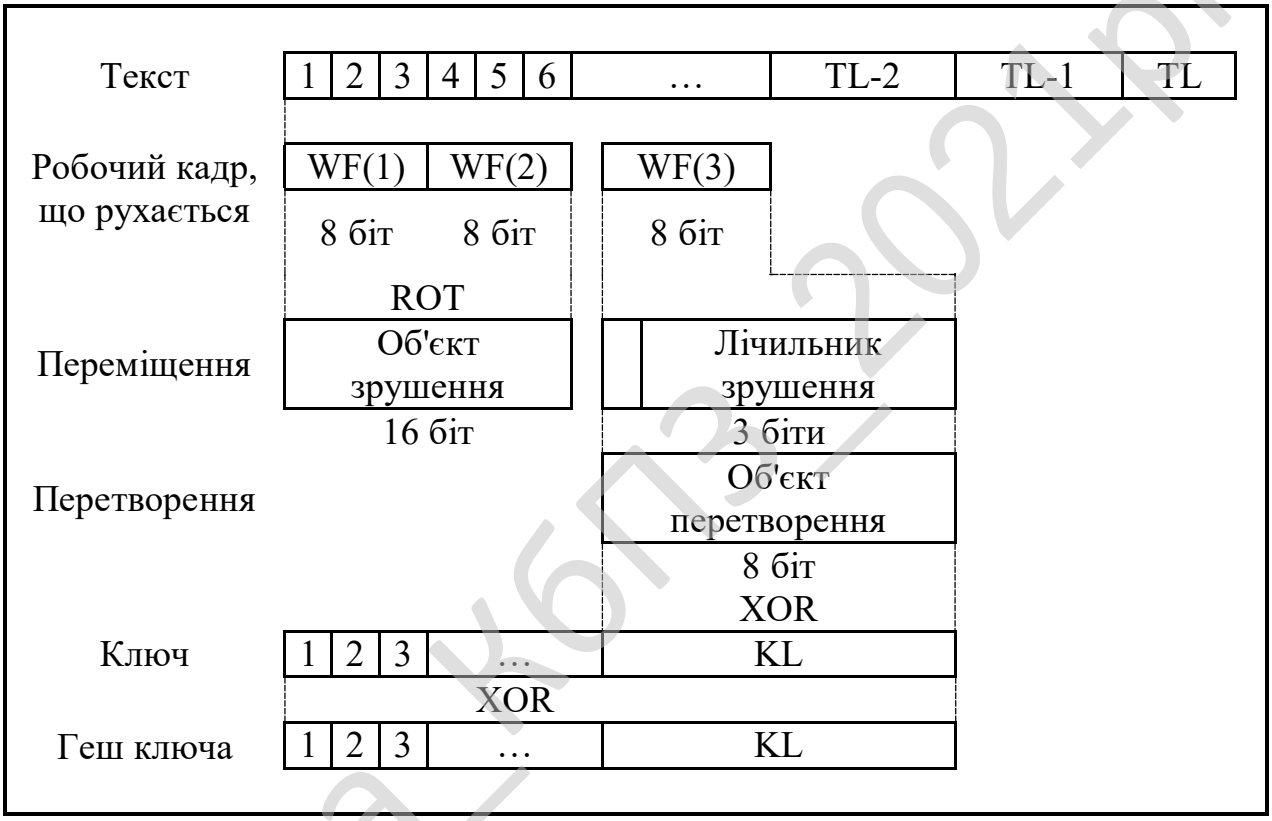

Рисунок 4.3 – Одна ітерація алгоритму Madryga

Оскільки кожний байт даних впливає на два байти ліворуч і на один байт праворуч від себе, після восьми проходів кожний байт шифртексту залежить від 16 байтів ліворуч і 8 байтів праворуч.

При шифруванні кожна ітерація внутрішнього циклу встановлює робочий кадр на передостанній байт відкритого тексту й циклічно переміщає його до третього з кінця байту відкритого тексту.

Спочатку весь ключ піддається операції XOR з випадковою константою й потім циклічно зрушується вправо на 3 біти (ключ і дані рухаю ться в різних напрямках, щоб мінімізувати надлишкові операції з бітами ключа).

Молодші три біти молодшого байта робочого кадру зберігаються, вони визначають циклічне зрушення інших двох байтів. Далі конкатенація двох старших байтом циклічно зрушується вліво на змінне число біт (від 0 до 7). Потім над молодшим байтом робочого кадру виконується операція XOR з молодшим байтом ключа. Нарешті робочий кадр зміщається вправо на один байт і весь процес повторюється.

Випадкова константа призначена для перетворення ключа в псевдовипадкову послідовність. Довжина константи повинна бути рівній довжині ключа. При обміні даними абоненти повинні користуватися однієї й тією же константою. Для 64-бітового ключа Madryga рекомендує константу 0x0fle2d3c4b5a6978.

При розшифруванні процес повторюється у зворотному порядку. У кожній ітерації внутрішнього циклу робочий кадр установлюється на байт, третій ліворуч від останнього байта шифртексту, і циклічно зрушується у зворотному напрямку до байта, розташованого на 2 байти уліво відносно останнього байта шифртексту. 2 байти шифртексту в процесі циклічно зрушуються вправо, а ключ – уліво. Після циклічних зрушень виконується операція XOR.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено розроблена у магістерській дипломній роботі система віртуалізованої серверної інфраструктури на базі HCI. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи: Функціональних кнопок ПЗ; Розділу обрання групи; Розділу виведення результату роботи системи; Функції представлені у графічному вигляді.

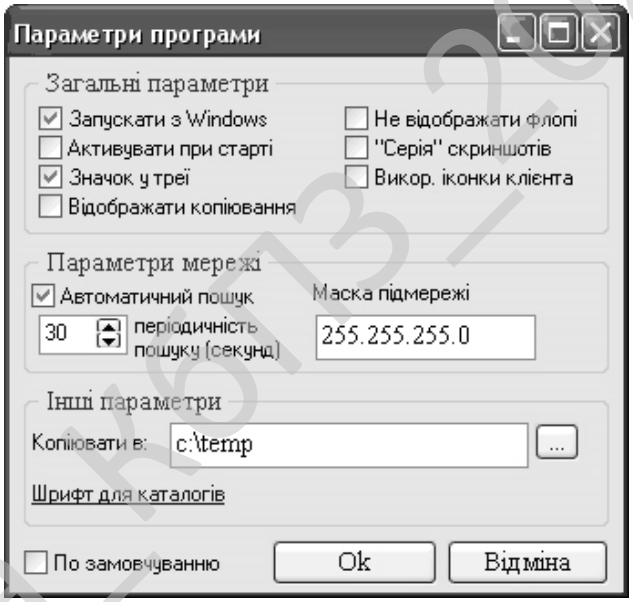

Рисунок 5.1 – Головне вікно ПЗ

Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

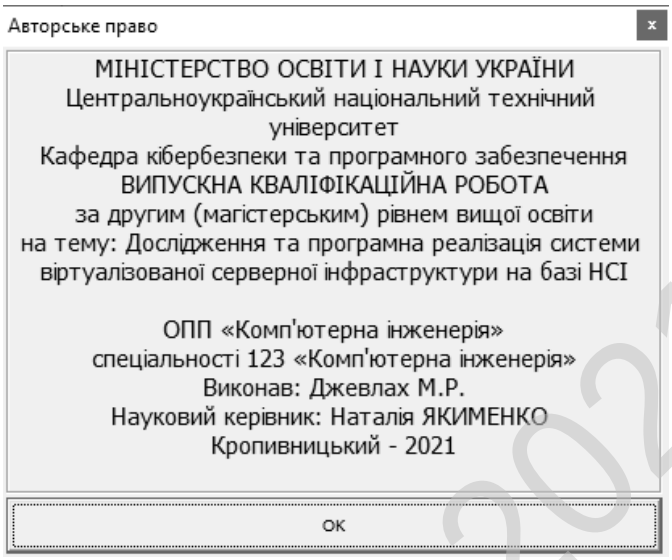

Рисунок 5.2 – Авторське право

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

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

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

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

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

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в IT рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

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

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

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

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

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

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

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

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

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

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

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

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

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

Будь-який спосіб тестування «чорної скриньки» повинен:

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

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

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

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій.

– Помилок інтерфейсу.

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

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

– Помилок ініціалізації та завершення.

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

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

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

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

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

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

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи віртуалізованої серверної інфраструктури на базі HCI.

Метою розробки є дослідження та програмна реалізація системи віртуалізованої серверної інфраструктури на базі HCI.

Об'єктом дослідження є процес віртуалізованої серверної інфраструктури на базі HCI.

Предметом дослідження є методи віртуалізованої серверної інфраструктури на базі HCI.

Методи дослідження базуються на методах теорії комп/ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод віртуалізованої серверної інфраструктури на базі HCI.

– Розроблено вітчизняний продукт віртуалізованої серверної інфраструктури на базі HCI, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 24 днів (один місяць). В магістерській роботі було проведено дослідження та виконана програмна реалізація системи віртуалізованої серверної інфраструктури на базі HCI. Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп'ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	30 (2 ост. цифри № зал $*10^1$)
3. Запланований термін розробки, днів	Fpq	24 (1 місяць)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Г
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	8
8. Кількість форм вихідної інформації.	–	6
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	1
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	3
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	4
17. Складність кінцевого програмного продукту (1-6)	–	5
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	3
20. Вимоги до швидкодії ПП (1-6)	–	3
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	4
23. Професійний рівень аналітиків (1-6)	–	3
24. Професійний рівень програмістів (1-6)	–	4
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	1
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	3
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	30000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	40
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \qquad (7.2)$$

де: *Wi* – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+4{,}22+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45 \cdot 2{,}2^{1{,}027} = 5{,}5 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПВ_j, \qquad (7.3)$$

де: $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 5{,}5 \cdot (1 \cdot 1{,}09 \cdot 1{,}30 \cdot 0{,}91 \cdot 1 \cdot 1 \cdot 1 \cdot 1{,}15 \cdot 1 \cdot 0{,}87 \cdot 1{,}10 \cdot$$

$$\cdot 1{,}22 \cdot 1{,}12 \cdot 1{,}10 \cdot 1 \cdot 1 \cdot 1{,}10) = 12{,}9 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \qquad (7.4)$$

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 12{,}9^{0{,}33+0{,}2(1{,}027-1{,}01)} \cdot 100 = 131 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	15	Д7
Робочий проект	131	Ф 7.1-7.4
Впровадження	15	Д13
Всього	180	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \qquad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{180 \cdot 1}{24 - 3} = 8{,}6 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	385	12	4620	77
Монітор	160	12	1920	32
Клавіатура	140	12	1680	28
Маніпулятор «мишка»	30	12	360	6
Принтер матричний	185	1	185	3
Принтер лазерний	355	2	710	12
Принтер струминний	300	1	300	5
Сканер	155	2	310	5
Концентратор– маршрутизатор	155	2	310	5
Кабельні господарства ЛОМ на 1 м. п.	2,5	70	175	3
Кабельне господарство електромережі	48	50	2400	40
Копіювальний апарат	285	1	285	5
Усього за рік:			$З_ч$	221

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{міс}}{1,2}, \quad (7.6)$$

$$Ф_{др}^{с} = \frac{221 \cdot 1}{1,2} = 184,1 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$Ч_{ел}$ = 184,1/(24·8) = 1 ставки.

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2016, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	1	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	1	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	1	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,5
	Підтримка постійних клієнтів	1	
	Оформлення договорів, ведення тендерів	1	
	Контроль взаєморозрахунків з постачальниками	1	
Всього		4	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,5
	Створення графічних і стилістичних елементів сайту	1	
	Оформлення банерів і промо-сторінок	1	
	Розміщення графіки і контенту на Інтернет сторінках	1	
Всього		4	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,5
	Верстка друкованих видань	1	
	Додрукова підготовка макетів	1	
	Розміщення графіки і контенту на Інтернет сторінках	1	
Всього		4	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (IT-менеджер)	1	10500	10500
Продакт-менеджер	0,5	8500	4250
Інженер-програміст	8,6	7500	64500
Інженер-електронщик	1	6000	6000
Інженер-системотехнік	0,5	6000	3000
Адміністратор мережі	0,5	8000	4000
Дизайнер WEB	0,5	10000	5000
Всього за період розробки	$R_{сп}$ = 12,6	-	$Ф_{роб}$ = 97250

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{97250}{12,6 \cdot 24} = 322 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 13 робочих місць;

$S_у$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» (м. Кіровоград) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по міст у складає 800…1600 $у.о./м^2$. Приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 13 \cdot 8 \cdot 20000 = 2080000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 208000 грн. Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$I_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$I_{нв} = 13 \cdot 3500 = 45500 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Brain за 14.10.21 – джерело http: //brain.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп'ютер		11457
Системний блок		7509
Процесор	Intel Core i7-4790 (4(8) ядра по 3.6 - 4. GHz); Cache Memory 8 MB	4200

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Системна плата	1st Player ATX NEW	1525
Відеокарта	PCIeX: ATI HD5670 SAPPHIR 1024MB/128bit/DDR3/TV/DualDVI	430
Жорсткий диск	HDD: 500 Gb 7200 Serial ATA WD 16MB	490
Оперативна пам’ять	Kingston DDR3 2GB (KVR1333D3N9/2G Intel/AMD – 2 шт	333
DVD-привод	-	-
Корпус	ATX Middle Tower GIGABYTE GZ-X Silver 500W (GZ-X4 Silver)	411
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: Al Type Cards, black	120
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	2600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Сканер	Epson Perfection V37 Photo	2970
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійног живлення	UPS APC BACK-UPS ES 525VA 230V RUSSIA (BE525-RS)	1348

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп'ютери	13	11457	14894,1	163835,1
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2970	297	3267
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	185653,6

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	2080000	-	-
2. Передавальні пристрої	208000	-	-
Всього по групі	2288000	5	114400

Продовження таблиці 7.8

1	2	3	4
Група 4			
3. Обчислювальна техніка	185654	-	-
Всього по групі	185654	50	92827
Група 5, 6			
4. Вимірювальні пристрої	3999	25	-
5. Транспортні засоби	0	20	-
6. Господарський інвентар	45500	25	-
Всього по групі	49499	-	12374,75
7. Нематеріальні активи	30000	10	3000
Разом	$К_р$ = 2553153		$А_р$ = 222602

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_e}, \quad (7.11)$$

де: N_e – кількість екземплярів програм, шт.

$$З_о = 322 \cdot 180/30 = 1932 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 1932 \cdot 10 \cdot 0{,}01 = 193 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $H_c = 22\%$ від суми основної та додаткової зарплати:

$$C_{оц} = 0{,}01 \cdot H_c (З_о + З_д), \quad (7.13)$$

де: H_c – відрахування на соціальні потреби, %.

$$C_{оц} = 0{,}01 \cdot 22(1932+193) = 468 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $H_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot H_г \cdot 0{,}01, \quad (7.14)$$

де: $H_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 1932 \cdot 15 \cdot 0{,}01 = 290 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.;

$З_{М2}$ – вартість запам’ятовуючих пристроїв, грн.;

$З_{М3}$ – вартість фарби, картріджей, тонеру, грн.;

$N_е$ – кількість екземплярів програм, шт.

Згідно виданих викладачем норм $n = 0{,}33$ приймаємо одну пачку паперу на три місяці розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п = 121$ грн., визначаємо вартість паперу за період розробки $N_м = 1$ міс:

$$З_{М1} = Ц_п \cdot N_м \cdot n. \quad (7.16)$$

$$З_{М1} = 121 \cdot 1 \cdot 0{,}33 = 40 \text{ грн.}$$

Згідно виданих викладачем норм до вартості запам’ятовуючих пристроїв входить вартість CD дисків в кількості, що дорівнює кількості екземплярів програм та одного DVD диска для збереження резервної копії програми:

$$З_{М2} = \sum Ц_д, \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 3 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 3 грн./шт.

$$З_{М2} = 30 \cdot 3 + 3 = 93 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{МЗ} = \sum Ц_{з}., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{МЗ} = 574+558+570 = 1702 \text{ грн.}$$

$$З_{М} = (40 + 93 + 1702)/30 = 61 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 1932 \cdot 15 \cdot 0{,}01 = 290 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 30 прим.):

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $А_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$А_м = 222602 \cdot 1/(30 \cdot 12) = 618 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$С_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$С_п = 1932+193+468+290+61+290+618 = 3852 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	1932
2. Додаткова зарплата виконавців	$З_д$	193
3. Відрахування на соціальні потреби	$С_{оц}$	468
4. Загальногосподарські витрати	$Г_{осп}$	290
5. Витрати на матеріали	$З_М$	61
6. Освоєння нових операційних систем, мов програмування	$О_п$	290
7. Амортизація основних фондів	$А_м$	618
8. Повна собівартість програмного забезпечення	$С_п$	3852
9. Плановий прибуток	$П_р$	1541
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	5393
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	1077
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	6470

Визначимо плановий прибуток за рівнем рентабельності ($Р_п$) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 40%.

$$П_р = 0{,}01 \cdot Р_п \cdot С_п, \qquad (7.22)$$

де: $Р_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 40 \cdot 3852 = 1541 \text{ грн.}$$

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об’єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	6470
Всього капітальних витрат	–	6470

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Витрати на профілактичні роботи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_c), \qquad (7.23)$$

де: $Т_р$ – кількість годин обслуговування кожного комп’ютера за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	32208	20130
2. Витрати на електроенергію	$З_{ел}$	22680	22050
3. Витрати на амортизацію	$З_{ам}$	0	3235
Всього витрат за рік	$І$	54888	45415

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 240 годин на рік до 150 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 240 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 32208\ грн,$$

до:

$$З_{р\,нов} = 150 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 20130\ грн.$$

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \qquad (7.24)$$

$$З_{ел\,баз} = 10 \cdot 0{,}15 \cdot 7200 \cdot 2{,}1 = 22680 \text{ грн.}$$

$$З_{ел\,нов} = 10 \cdot 0{,}15 \cdot 7000 \cdot 2{,}1 = 22050 \text{ грн.}$$

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	50	–	6470	–	3235
Всього відрахувань	-	–	6470	–	3235

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (5393 - 3852) \cdot 30 - (0,05 \cdot 2288000 + 0,5 \cdot 185654 + +0,25 \cdot 49499 + 0,1 \cdot 30000) \cdot 1/12 = 27680 \ грн.$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_p^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: K_p^* – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{265153}{(5393 - 3852) \cdot 30 \cdot 12 / 1} = 0,5 \ року.$$

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	30
2. Повна собівартість розробленої програми	Грн.	3852
3. Ціна розробленої програми	Грн.	5393
4. Плановий прибуток від реалізації розробленої програми	Грн.	1541
5. Рентабельність програмної продукції	%	40
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	2553153
7. Загальний прибуток від реалізації програмної продукції	Грн.	46230
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	27680
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	6470
11.Величина економічного ефекту у користувача програмної продукції	Грн.	6238
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,7

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \qquad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (54888\text{-}45415)\text{-}0,\ 5 \cdot 6470 = 6238 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \tag{7.28}$$

$$T_{сп} = \frac{6470}{54888 - 45415} = 0,7 \ року .$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Важливість небезпеки взагалі і охорони праці, як її складової частини, зокрема, не викликає сумнівів з давніх часів. Умови праці розглядав ще Арістпотель (384-322 до н.е.).

Розвиток галузі інформаційних технологій (ІТ) привернув увагу спільноти на умови праці програмістів та інших спеціалістів ІТ сектора економіки.

Загальні положення державної політики в галузі охорони праці регламентуються Законом України “Про охорону праці” [2], а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

Відомо, що програмісти у поцессі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби польців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення газу.

Вплив негативних чинників може привести до професійних захворювань.

У даному розділі розглянемо шкідливі чинники роботи програмістів та інших спеціалістів ІТ, та проведемо аналіз умов праці. При цьорму будемо

керуватись наступними нормативно-правовими актами: «Правила охорони праці під час експлуатації електронно-обчислювальних машин» НПАОП 0.00-1.28-10, «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

На умови праці програміста впливають наступні фактори:

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення впливу комп'ютера на організм програміста визначимо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста,

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Програміст працює з електронно-обчислювальною машиною (ЕОМ) та іншим обладнанням, яке є джерелом небезпеки ураження електричним струмом [1]. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. Так як програміст постійно перебуває в приміщенні, тому для комфортних умов праці в цьому приміщенні необхідно створити належний мікроклімат.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– монотонність праці;

– електромагнітні (у т.ч. високочастотні) електромагнітні випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шуми;

– статичні навантаження на кістково-м’язовий апарат;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Довжина	8,56
Ширина	5,43
Висота	2.9

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6,64
Обсяг, V	$м^3$	не менше 20.0	19,25

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

У зазначеному приміщенні працюють 7 людей. За даними, які наведено у табл. 8.1, та табл. 8.2, можна зробити висновок, що площа приміщення у розрахунку на одного робочого місця програміста відповідає нормативним вимогам як Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», так і СанПіН 3.3.2 – 007 – 98.

А об'єм приміщення у розрахунку на одного робочого місця програміста відповідає нормативним вимогам Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», але дещо нижчий вимог СанПіН 3.3.2 – 007 – 98 (19,25 $м^3$ замість 20 $м^3$). Це цілком прийнятно, враховуючи хронологію та пріоритет опублікування вищезгаданих нормативних актів.

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного вдача Украйни, робота, виконувана в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря в приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виделення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату

складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщенн я. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, ºС	Воло-гість,%	Швидкість повітря, м/с	Температура, ºС	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-23	40-55	0,1
Тепла	23-25	50-70	0,1	24-25	55-65	0,1

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застовується у холодний період року.

В літню пору застосується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер HP 1100, електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

Працю працівника, який постійно працює за комп’ютером, згідно ДБН В.2.5 – 28 – 2006 р можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи

приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 лк. Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значеннь;

– бажано, щоб робочий стіл при необхідності можна було регулювати по висоті в межах 680-780 мм, а висота над рівнем підлоги робочої поверхні, на якій працює програміст, повинна складати 720 мм.

Таким чином можна припустити, що основною причиною можливого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Додаткові рекомендовані заходи: забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення та занулення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

8.5 Розрахунок штучного освітлення

Розрахунок штучного освітлення проводемо за методом коефіцієнта використання світлового потоку для приміщення ширина якого складає 5,42 м, довжина – 8,52 м, висота – 2,9 м.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою:

$$F=ESKZ/n,$$

де:

F – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк; E = 300 Лк;

S – площа освітлюваного приміщення (у нашому випадку S=5,42 х 8,52 = 46,2 м^2);

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, в нашому випадку Z = 1,1);

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку K = 1,5);

n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін ($\rho_{стін.}$) і стелі ($\rho_{стелі}$), значення коефіцієнтів дорівнюють $\rho_{стін}$ =

50% і $\rho_{стелі}$ = 50%.

Обчислимо індекс приміщення за формулою:

$$i=S/(h(A+B)),$$

де:

S – площа приміщення, S = 46,2 м^2;

h – розрахункова висота підвісу, h = 2,9 м;

A – ширина приміщення, A = 5,42 м;

B – довжина приміщення, B = 8,52 м.

Підставимо всі значення у формулу та визначемо індекса приміщення:

$$i=1,14.$$

Знаючи індекс приміщення, за знаходимо n = 0.46 (з таблиці коефіцієнтів використання світлового потоку (η) світильників з відповідним типом лампам [3]). Підставимо всі значення у формулу, визначемо світловий потік: *F=38743 Лм.*

Будемо використовувати лампи TL-F 12 36W LED 6000K IP65, світловий потік яких $F_л$ = 3000 Лм.

Число ламп визначається по формулі:

$$N=F/F_л$$

де:

F – світловий потік,

$F_л$ – світловий потік однієї лампи.

Підставимо всі значення у формулу та визначимо індексу приміщення:

$$N= 38743/ 3000=12,9 \text{ шт.}$$

Кожен світильник комплектується двома лампами. Тому необхідно використовувати 7 світильників з 14 лампами в них.

8.6 Висновки до розділу

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

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я програміста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи віртуалізованої серверної інфраструктури на базі HCI.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів віртуалізованої серверної інфраструктури на базі HCI.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем віртуалізованої серверної інфраструктури на базі HCI.

– Досліджена система віртуалізованої серверної інфраструктури на базі HCI.

– На основі отриманих результатів досліджень створена програмна реалізація системи віртуалізованої серверної інфраструктури на базі HCI.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання віртуалізованої серверної інфраструктури на базі HCI.

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

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

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

Програма реалізована на мові високого рівня Embarcadero RAD Studio Delphi 10.3.2 Rio Architect. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows XP/Vista/7/8/10.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Madryga.

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.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 6238 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,7 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Джевлах М.Р. Дослідження та програмна реалізація системи віртуалізованої серверної інфраструктури на базі HCI // Збірник праць молодих науковців ЦНТУ. – Вип. 12. – Кропивницький: ЦНТУ, 2022.
2. А. Я. Архангельский, Программирование в Delphi 7, М., БИНОМ, 2004.
3. Г. В. Галисеев, Компоненты в Delphi 7, М., С -Пб, Киев, Диалектика, 2004.
4. Е. А. Еремин, Популярные лекции об устройстве компьютера, С-Пб, БХВ-Санкт-Петербург, 2003.
5. Таненбаум Э. Компьютерные сети (3-е изд.). Пер. с англ. – С.-Пб.:"Питер", 2002 – 848 с
6. Олифер В. Г., Олифер Н. А. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 3-е издание. С.-Пб.: "Питер", 2005 – 960 с
7. Руссинович М., Соломон Д. Внутреннее устройство Microsoft Windows: Windows Server 2003, Windows XP и Windows 2000. Мастер-класс. 4-е издание. М.: Издательство "Русская редакция"; СПб.: Питер, 2006. 992 стр.: ил
8. Рассел, Ш. Кроуфорд, Дж. Джеренд. Microsoft Windows Server 2003. Справочник администратора. М.:Эком, 2005 – 1391 с
9. Вишневский А., Кокорева О., Чекмарев А. Microsoft Windows Server 2003. Русская версия. С.-Пб.: БХВ-Петербург, 2003 – 1120 с
10. Уильям Р. Станек. Microsoft Windows Server 2003. Справочник администратора. М.: Издательско-торговый дом "Русская редакция", 2004 – 648 с.
11. Мохамад Гани Абу Таам Разработка математической GERT-модели технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / А.А.Смирнов, Мохамад Гани Абу

Таам // Информационные системы в управлении, образовании, промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Вид-во ТОВ «Щедра садиба плюс», 2014. – 498 с.

12. Мохамад Гани Абу Таам Метод управления доступом в интеллектуальных узлах коммутации / Мохамад Гани Абу Таам, А.А.Смирнов // Информационные технологии и защита информации в информационно-коммуникационных системах: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Вид-во ТОВ «Щедра садиба плюс», 2015. – 486 с.

13. Мохамад Гани Абу Таам Математическая GERT-модель технологии передачи метаданных в облачные антивирусные системы / В.В.Босько, А.А.Смирнов, И.А.Березюк, Мохамад Гани Абу Таам // Збірник наукових праць "Системи обробки інформації". – Випуск 1(117). – Х.: ХУПС – 2014. – С. 137-141.

14. Мохамад Гани Абу Таам Структурно-логическая GERT-модель технологии распространения компьютерных вирусов / А.А.Смирнов, И.А.Березюк, Мохамад Гани Абу Таам // Системи управління, навігації та зв'язку. – Випуск 1(29). – П.: ПНТУ. – 2014. – С. 120-125.

15. Мохамад Гани Абу Таам Сравнительные исследования математических моделей технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Гани Абу Таам, А.А. Смирнов, А.В. Коваленко, С.А. Смирнов // Збірник наукових праць "Системи обробки інформації". – Випуск 9(125). – Х.: ХУПС – 2014. – С. 105-110.

16. Мохамад Гани Абу Таам Математическая модель интеллектуального узла коммутации с обслуживанием информационных пакетов различного приоритета / Мохамад Гани Абу Таам, А.А. Смирнов, Н.С. Якименко, С.А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. Випуск 4 (41). – Харків: ХУПС. – 2014. – С. 48-52.

17. Мохамад Гани Абу Таам Исследование показателей качества функционирования интеллектуальных узлов коммутации в

телекоммуникационных системах и сетях / Мохамад Гани Абу Таам, А.А. Смирнов, Н.С. Якименко, С.А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України. – Випуск 4(17). – Харків: ХУПС. – 2014. – С.90-95.

18. Мохамад Гани Абу Таам Усовершенствованный алгоритм управления доступом к «облачным» телекоммуникационным ресурсам / Мохамад Гани Абу Таам, А.А. Смирнов, Н.С. Якименко, С.А. Смирнов // Збірник наукових праць "Системи обробки інформації". – Випуск 1(126). – Х.: ХУПС – 2015. – С. 150-153.

19. Мохамад Гани Абу Таам Анализ и исследование методов управления сетевыми ресурсами для обеспечения антивирусной защиты данных / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Системи озброєння і військова техніка. – Випуск 3(43) – Х.: ХУПС – 2015. – С. 100-107.

20. Мохамад Гани Абу Таам Исследование эффективности метода управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України. – Випуск 3(19). – Х.: ХУПС. – 2015. – С. 134-141.

21. Mohamad Abou Taam Method of controlling access to intellectual switching nodes of telecommunication networks and systems / A.A. Smirnov, Mohamad Abou Taam, S.A. Smirnov // International Journal of Computational Engineering Research (IJCER). – Volume 5, Issue 5. – India. Delhi. – 2015. – P. 1-7.

22. Мохамад Гани Абу Таам GERT-модель технологии передачи данных в облачные антивирусные системы / А.А. Смирнов, В.В. Босько, Мохамад Гани Абу Таам // Збірник тез доповідей науково-практичної конференції «Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку». м. Харків. 12-13 березня 2014 р. – Харків. АВВ МВС. – 2014. – С. 18-19.

23. Мохамад Гани Абу Таам Математическое моделирование технологии передачи сигнатур в облачные антивирусные системы / Мохамад Гани Абу Таам, А.А. Смирнов // Збірник тез VI міжнародної науково-практичної конференції

“Проблеми і перспективи розвитку ІТ-індустрії”. м. Харків. 17-18 квітня 2014 р. – Харків: ХНЕУ. – 2014. – С. 260.

24. Мохамад Гани Абу Таам Анализ требований к качеству обслуживания в информационно-телекоммуникационных системах / А.А. Смирнов, Мохамад Гани Абу Таам // Збірник тез XVI міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 11-12 квітня 2014 р. – Кіровоград: КНТУ. – 2014. – С. 124-126.

25. Мохамад Гані Абу Таам Дослідження та реалізація GERT-моделі технології розповсюдження комп’ютерних вірусів для захисту телекомунікаційних систем / Мохамад Гані Абу Таам, С.А. Смирнов // Збірник тез науково-практичної конференції «Інформаційні технології та комп’ютерна інженерія». м. Кіровоград. 4 грудня 2014 р. – Кіровоград: КНТУ. – 2014. – С. 168.

26. Мохамад Гани Абу Таам Исследование математических моделей технологии распространения компьютерных вирусов / А.А. Смирнов, Мохамад Гани Абу Таам, С.А. Смирнов // Збірник наукових праць міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 25-28 лютого 2015 р. – Київ: Європейський університет. – 2015. – С. 90-91.

27. Мохамад Гани Абу Таам Метод управления доступом к «облачным» ресурсам для защиты телекоммуникационных систем / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Збірник тез всеукраїнської науково-практичної конференції «Інформаційна безпека держави, суспільства та особистості». м. Кіровоград. 16 квітня 2015. – Кіровоград: КНТУ. – 2015. – С. 50-52.

28. Мохамад Гани Абу Таам Разработка метода управления доступом в интеллектуальных узлах коммутации / А.А. Смирнов, Мохамад Гани Абу Таам, С.А. Смирнов // Збірник тез VII міжнародної науково-практичної конференції “Проблеми і перспективи розвитку ІТ-індустрії”. м. Харків. 17-18 квітня 2015 р. – Харків: ХНЕУ. – 2015. – С. 14.

29. Мохамад Гани Абу Таам Реализация метода управления доступом в интеллектуальных узлах коммутации / А.А. Смирнов, Мохамад Гани Абу Таам // Збірник тез XVII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 17-18 квітня 2015 р. – Кіровоград: КНТУ. – 2015. – С. 91-92.

30. Мохамад Гани Абу Таам Реализация математической модели интеллектуального узла коммутации для обеспечения защищенности телекоммуникационной сети / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Збірник тез II Міжнародної науково-практичної Інтернет-конференції «Інформаційна та економічна безпека» (INFECO-2015)». м. Харків. 21-22 травня 2015 р. – Харків: ХІБС УБС НБУ. – 2015. – С. 20-24.

31. Мохамад Гани Абу Таам Разработка математической модели технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Сборник тезисов XI международной конференции "Стратегия качества в промышленности и образовании". г. Варна. Болгария. 01 – 06 июня 2015 г – Варна. ТУВ. – 2015. – С. 488-491

32. Мохамад Гани Абу Таам Метод управления доступом к облачным телекоммуникационным ресурсам для обеспечения защиты данных / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Збірник тез Міжнародної науково-практичної конференції «Комп'ютерні технології та інформаційна безпека». м. Кіровоград. 2-3 липня 2015 р. – Кіровоград: КНТУ. – 2015. – С. 4-5.

33. Мохамад Гани Абу Таам Имитационная модель системы управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Збірник тез першої всеукраїнської науково-практичної конференції «Перспективні напрями захисту інформації». м. Затока. 7-9 вересня 2015 р. – Одеса: ОНАЗ. – 2015. – С. 90-94.

34. МСЭ-Т Рекомендация G.101. Международные телефонные соединения и цепи – Общие определения //11/2003. [Електронний ресурс]. – Режим доступу до ресурсу: http://www. telecom61.ru›SharedFiles/Download.aspx? …pageid=106

35. Одом Ш. Коммутаторы CISCO / Ш. Одом, Х. Ноттингем – М.:"Кудиц-Образ", 2003. – 528 с.

36. Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы: учебник для вузов / В.Г. Олифер, Н.А. Олифер. –2-е изд. – СПб.: Питер, 2007. – 958 с.

37. Руководство по технологиям объединенных сетей. 4-е изд. / пер.с англ. и ред. А.Н. Крикуна – М.: Изд. дом «Вильямс», 2005. – 1040 с.

38. Свами М.Н., Тхуласираман К. Графы, сети и алгоритмы: пер. с англ. / М.Н. Свами, К. Тхуласираман; под ред. В.А. Горбатова. – М.: Мир, 1984. – 454 с.

39. Семенов С.Г. Анализ методов прогнозирования в телекоммуникационных сетях автоматизированных систем управления / С.Г.Семенов // Збірник наукових праць «Системи управління, навігації та зв'язку», – К.:ЦНДІ навігації і управління, – 2008.-Вип. 2(6) .- С.134-137

40. Семенов С.Г. Математическая модель процесса доставки информационных пакетов в компьютерной сети системы критического применения / С.Г.Семенов, И.В.Ильина // Науково-технічний журнал «Радіоелектронні і комп'ютерні системи» Х.:ХАІ, – 2008.-Вип. 1(28) – С.162-165

41. Семенов С.Г. Оптимизация трафика на основе сбалансированной загрузки информационно-телекоммуникационной сети // Системи обробки інформації. – Х.: ХВУ, 2004. – № 8(36). – С.206-210

42. Семенов С.Г. Математическая модель мультисервисного канала связи на основе экспоненциальной GERT-сети / С.Г. Семенов, Є.В. Мелешко, Я.В.

Ілюшко // Системи озброєння і військова техніка. – Х.:ХУ ПС. – 2011. –Вип. 3(27). – С. 64-67.

43. Семенов С.Г. Математична модель системи криптографічного захисту електронних повідомлень на основі GERT-мережі / С.Г. Семенов, О.О. Сур // Системи управління, навігації та зв'язку. – К.:ЦНДІ навігації і управління. – 2012. – Том 1. Вип. 1(21). – С. 131-137

44. Семенов С.Г. Исследования вероятностно-временных характеристик мультисервисного канала связи с использованием математического аппарата GERT-сети / С.Г. Семенов, В.В. Босько, І.А. Березюк // Системи обробки інформації. – Х.: ХУ ПС. – 2012. – Том 1. Вип. 3(101). – С. 139-142.

45. Семенов С.Г. Моделирование защищенного канала связи с использованием экспоненциальной GERT-сети / С.Г. Семенов, А.А. Можаев // Информатика, математическое моделирование, экономика. – Смоленськ.: Смоленский филиал АНО ВПО ЦС РФ "Российский университет кооперации". – 2012. – Том.1. – С. 152-160.

46. Семенов С.Г. Методика математического моделирования защищенной ИТС на основе многослойной GERT-сети / С.Г. Семенов // Вісник Національного технічного університету «Харківський політехнічний інститут». – Х.:НТУ «ХПІ». – 2012. –№62 (968). – С 173-181.

47. Семенов С.Г. Защита данных в компьютеризированных управляющих системах / С.Г. Семенов, В.В. Давыдов, С.Ю. Гавриленко. – LAP Lambert Academic Publishing GmbH & Co. KG (Саарбрюккен, Германия), 2014. – 236 с.

48. Смирнов А.А. Анализ и сравнительное исследование перспективных направлений развития цифровых телекоммуникационных систем и сетей / А.А.Смирнов, В.В.Босько, Е.В.Мелешко // Системи обробки інформації. – Х.: ХУ ПС, 2008. – Вип.7(74). – С.120-123.

49. Смирнов А.А. Усовершенствование метода управления очередями в многопротокольных узлах телекоммуникационной сети / А.А.Смирнов, Е.В.Мелешко // Збірник тез та доповідей другої всеукраїнської науково-

практичної конференції «Системний аналіз. Інформатика. Управління». Запоріжжя. Тези доповідей. Запоріжжя: КПУ, 2011.

50. Современные телекоммуникации. Технологии и экономика / [В.Л. Банкет, О.В. Бондаренко, П.П. Воробьенко и др.]; под ред. С.А. Довгого. – М.: Эко-Трендз, 2003. – 320 с.

51. Столлингс В. Современные компьютерные сети / Вильям Столлингс.– СПб.: Питер, 2003. – 778 с.

52. Таненбаум Э. Компьютерные сети / Эндрю Таненбаум; пер. с англ. А. Леонтьев. – СПб.: Питер, 2002. – 848 с.

53. Телекоммуникационные системы и сети: учебное пособие. В 3 томах / [В.В. Величко, Е.А. Субботин, В.П. Шувалов, А.Ф. Ярославцев]; под ред. В.П. Шувалова. – М.: Горячая линия-Телеком, 2005, т. 3 – 592 с.

54. Уолрэнд Дж. Телекоммуникационные и компьютерные сети / Дж. Уолрэнд. – М.: Постмаркет, 2001. – 480 с.

55. Хайкин С. Нейронные сети: полный курс / С. Хайкин. – М.:Вильямс, 2006. – 1103 с.

56. Шелухин О.И. Фрактальные процессы в телекоммуникациях: моногр. / О.И. Шелухин, А.М. Тенякшев, А.В. Осин – М.: Радиотехника, 2003. – 480 с.

57. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

58. Про охорону праці: Закон України від 14.10.1992 р. № 2694-XII. Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12#Text

59. Центр післядипломної освіти та підвищення кваліфікації Режим доступу до ресурсу: https://cpo.stu.cn.ua/

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРМ-123.21.0003.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Джевлах М.Р.			Дослідження та програмна реалізація системи віртуалізованої серверної інфраструктури на базі НСІ	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Якименко Н.М.				М	
Н. Контр.		Гермак В.С.				ЦНТУ КІ-20М-1,4	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи віртуалізованої серверної інфраструктури на базі HCI.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 42-13 від 02.08.2021 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи віртуалізованої серверної інфраструктури на базі HCI.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) 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;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи віртуалізованої серверної інфраструктури на базі HCI;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Embarcadero RAD Studio Delphi 10.3.2 Rio Architect.

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 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2021 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 109 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2021 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 20.12.2021 р.

Додаток Б

(обов'язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Якименко Н.М.

Дослідження та програмна реалізація системи віртуалізованої серверної інфраструктури на базі HCI

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 81

Літера: РП

Кропивницький – 2021 року

Серверна частина

Файл ProjectServer.dpr - файл проекту

```
program ProjectServer;

uses
  Forms,
  UnitServer in 'UnitServer.pas' {fmMain},
  uServerTCPParser in 'uServerTCPParser.pas',
  uOptions in 'uOptions.pas' {fmOptions},
  uAbout in 'uAbout.pas' {fmAbout},
  uInfo in 'uInfo.pas' {fmInfo},
  uAllowWork in 'uAllowWork.pas' {fmAllowWork},
  uRemoteConnect in 'uRemoteConnect.pas' {fmRemoteConnect};

{$R *.RES}

begin
  Application.Initialize;
  Application.HintHidePause:=65500;
  Application.CreateForm(TfmMain, fmMain);
  Application.CreateForm(TfmAbout, fmAbout);
  Application.CreateForm(TfmInfo, fmInfo);
  Application.CreateForm(TfmAllowWork, fmAllowWork);
  Application.CreateForm(TfmRemoteConnect, fmRemoteConnect);
  Application.CreateForm(TfmOptions, fmOptions);
  Application.Run;
end.
```

Файл ProjectServer.dpr - програма-сервер

```
unit UnitServer;

interface

uses
  Windows, ShellAPI, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, Menus, ComCtrls, CommCtrl, ExtCtrls, Buttons,
uTCPParser, uBaseLanServer, uServerTCPParser, uServiceController, scktcomp,
  DropTarget, DropSource, ImgList, CoolTrayIcon, ToolWin;

type
  TDisplayStatus=(dsUnknown,dsFolders,dsProcesses,dsServices,dsSearchResults);

  TfmMain = class(TForm)
    BaseLanServer: TBaseLanServer;
    TCPParserCollector: TTCPParserCollector;
    MainMenu: TMainMenu;
    mmAbout: TMenuItem;
    mmOptions: TMenuItem;
    mmConnect: TMenuItem;
    mmDisconnect: TMenuItem;
    mmExit: TMenuItem;
    mmAction: TMenuItem;
    mmLogOut: TMenuItem;
    mmShutDown: TMenuItem;
    mmReboot: TMenuItem;
    mmForceShutDown: TMenuItem;
    mmSendMessage: TMenuItem;
    mmFindInFiles: TMenuItem;
    mmAllowWork: TMenuItem;
    mmCloseCD: TMenuItem;
    mmOpenCD: TMenuItem;
    mmBroadCastClients: TMenuItem;
    mmRemoteSearch: TMenuItem;
    mmShutDownClient: TMenuItem;

    pmTray: TPopupMenu;
    pmConnect: TMenuItem;
    pmDisconnect: TMenuItem;
    pmRestore: TMenuItem;
    pmCollapse: TMenuItem;
    pmOptions: TMenuItem;
    pmExit: TMenuItem;

    pmlvDirs: TPopupMenu;
    pmCopy: TMenuItem;
    pmPrint: TMenuItem;
    pmDelete: TMenuItem;
    pmRefreshDir: TMenuItem;
    pmProperty: TMenuItem;
    pmOpen: TMenuItem;
    pmSendFiles: TMenuItem;
    pmCreateShortCut: TMenuItem;
    pmRename: TMenuItem;

    pmClients: TPopupMenu;
    pmPCInfo: TMenuItem;
    pmSendMessage: TMenuItem;
    pmOpenAsSystem: TMenuItem;
    pmClientVersion: TMenuItem;
    pmUpdateClient: TMenuItem;
    mmConnection: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
```

```
N5: TMenuItem;
N6: TMenuItem;
N7: TMenuItem;
N8: TMenuItem;
N9: TMenuItem;
N10: TMenuItem;
N11: TMenuItem;
N12: TMenuItem;
N13: TMenuItem;
N14: TMenuItem;
N15: TMenuItem;
N16: TMenuItem;

Panel1: TPanel;
Panel3: TPanel;
Panel4: TPanel;

lbNetClients: TListBox;
cbDrive: TComboBox;
lvDirs: TListView;
edDir: TEdit;

Splitter1: TSplitter;
sbVolume: TScrollBar;
SystemLargeImageList: TImageList;
ButtonImageList: TImageList;

DropFileSource: TDropFileSource;
DropFileTarget: TDropFileTarget;

OpenDialog: TOpenDialog;

TrayIcon: TCoolTrayIcon;

CoolBar1: TCoolBar;
CoolBar2: TCoolBar;
XPToolBarClients: TToolBar;
XPToolBarFileOperations: TToolBar;
XPToolBarFolderOperations: TToolBar;
XPToolBarProcessOperations: TToolBar;
XPToolBarActions: TToolBar;
XPToolBarShut: TToolBar;
tbUp: TToolButton;
tbRefresh: TToolButton;
tbCopy: TToolButton;
tbSendFiles: TToolButton;
tbDeleteFiles: TToolButton;
tbPrintFile: TToolButton;
tbScreenShot: TToolButton;
tbProcesses: TToolButton;
tbServices: TToolButton;

tbBroadcastClients: TToolButton;
tbShutdownActions: TToolButton;
tbOpenCD: TToolButton;
tbCloseCD: TToolButton;
mmUpdateClient: TMenuItem;
ClientImageList: TImageList;
N1: TMenuItem;
pmScreenShot: TMenuItem;
pmShowProcess: TMenuItem;
pmServices: TMenuItem;
N17: TMenuItem;
mmShow: TMenuItem;
mmShowFolders: TMenuItem;
mmScreenShot: TMenuItem;
mmProcesses: TMenuItem;
mmServices: TMenuItem;
N18: TMenuItem;
```

```
    mmPCInfo: TMenuItem;
    mmClientVersion: TMenuItem;
    N19: TMenuItem;
    pmActivateClient: TMenuItem;
    tbViewStyle: TToolButton;
    SystemSmallImageList: TImageList;
    ToolButton1: TToolButton;
    N20: TMenuItem;
    StatusBar1: TStatusBar;

    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure FormResize(Sender: TObject);

    procedure BaseLanServerChangeClients(Sender: TObject);

    procedure TCPParserCollectorAdddingParser(Sender: TObject;var TCPParser:
TCustomTCPParser);
    procedure TCPParserCollectorAddParser(Sender: TObject;
      var TCPParser: TCustomTCPParser);
    procedure TCPParserCollectorStartDataReceive(
      const Socket: TCustomWinSocket; const DataHeader: TDataHeader;
      var DataStream: TStream);



    procedure mmDisconnectClick(Sender: TObject);
    procedure mmConnectClick(Sender: TObject);
    procedure mmBroadcastClientsClick(Sender: TObject);
    procedure mmExitClick(Sender: TObject);
    procedure mmOptionsClick(Sender: TObject);
    procedure mmAboutClick(Sender: TObject);
    procedure mmLogOutClick(Sender: TObject);
    procedure mmShutDownClick(Sender: TObject);
    procedure mmRebootClick(Sender: TObject);
    procedure mmForceShutDownClick(Sender: TObject);
    procedure mmAllowWorkClick(Sender: TObject);
    procedure mmSendMessageClick(Sender: TObject);
    procedure mmHelpClick(Sender: TObject);
    procedure mmRemoteSearchClick(Sender: TObject);
    procedure mmUpdateClientClick(Sender: TObject);

    procedure pmRestoreClick(Sender: TObject);
    procedure pmCollapseClick(Sender: TObject);
    procedure pmRenameClick(Sender: TObject);
    procedure pmPropertyClick(Sender: TObject);
    procedure pmlvDirsPopup(Sender: TObject);
    procedure pmOpenAsSystemClick(Sender: TObject);
    procedure mmPCInfoClick(Sender: TObject);
    procedure mmClientVersionClick(Sender: TObject);
    procedure mmRefreshDirClick(Sender: TObject);

    procedure pmActivateClientClick(Sender: TObject);

    procedure lvDirsMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure lvDirsMouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure lvDirsMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure pmOpenClick(Sender: TObject);
    procedure lvDirsKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure lvDirsEdited(Sender: TObject; Item: TListItem;
      var S: String);
    procedure lvDirsColumnClick(Sender: TObject; Column: TListColumn);
    procedure lvDirsCompare(Sender: TObject; Item1, Item2: TListItem;
      Data: Integer; var Compare: Integer);
```

```
    procedure tbUpClick(Sender: TObject);
    procedure tbRefreshClick(Sender: TObject);
    procedure pmCopyClick(Sender: TObject);
    procedure pmSendFilesClick(Sender: TObject);
    procedure pmPrintClick(Sender: TObject);
    procedure pmDeleteClick(Sender: TObject);
    procedure mmScreenShotClick(Sender: TObject);
    procedure sbOpenCDClick(Sender: TObject);
    procedure sbCloseCDClick(Sender: TObject);
    procedure mmProcessesClick(Sender: TObject);
    procedure mmServicesClick(Sender: TObject);
    procedure sbVolumeScroll(Sender: TObject; ScrollCode: TScrollCode;
      var ScrollPos: Integer);
    procedure tbShutdownActionsClick(Sender: TObject);

    procedure cbDriveChange(Sender: TObject);

    procedure Splitter1CanResize(Sender: TObject; var NewSize: Integer;
      var Accept: Boolean);
    procedure DropFileTargetDrop(Sender: TObject; ShiftState: TShiftState;
      Point: TPoint; var Effect: Integer);
    procedure DropFileSourceDrop(Sender: TObject; DragType: TDragType;
      var ContinueDrop: Boolean);
    procedure DropFileTargetDragOver(Sender: TObject;
      ShiftState: TShiftState; Point: TPoint; var Effect: Integer);
    procedure CoolBar2Resize(Sender: TObject);
    procedure tbViewStyleClick(Sender: TObject);
    procedure N20Click(Sender: TObject);
    procedure lvDirsChange(Sender: TObject; Item: TListItem;
      Change: TItemChange);
  private
    { Private declarations }
    DirectoryIndex:integer;
    FDontShowFloppy: boolean;
    FAllCopyDirectory:string;
    FDisplayStatus: TDisplayStatus;
    FSortColumn: integer;
    FSortDirection:integer;
    FAllowFilePropertys: Boolean;
    FAllowLargeFileIcons: Boolean;

    procedure SetDontShowFloppy(const Value: boolean);
    procedure SetFAllCopyDirectory(const Value: string);
    procedure SetDisplayStatus(const Value: TDisplayStatus);
    procedure SetSortColumn(const Value: integer);
    procedure SetAllowFilePropertys(const Value: Boolean);
    procedure SetAllowLargeFileIcons(const Value: Boolean);

  public
    { Public declarations }
    Header:TDataHeader;
    Stream:TStream;
    ActiveParser:TTCPParser;

    ShowScreenShot:Boolean;
    ShowCopyProgress:boolean; // установлювати ця властивість на самому
      // парсері безглуздо - однаково, відображатися повинні тільки
      // файлові копіювання, а Custom Parser'у однаково, що він приймає

    MultiScreenShot:Boolean;

    DragFlag:boolean;

    procedure ActivateButtons(const Active:Boolean);
    procedure SetOptions(TCPParser:TTCPParser);


    procedure ParserOnGetDrives(Sender:
TObject;ErrCode:integer;ErrString:string);
```

```
    procedure ParserOnGetDirs(Sender:TObject);
    procedure ParserOnProgress(Sender:TObject);
    procedure ParserOnGetProcesses(Sender:TObject);
    procedure ParserGetProcessInfo(Sender:TObject);
    procedure ParserOnGetFile(Sender: TObject;ErrCode:integer;ErrString:string);
    procedure ParserOnExecute(Sender: TObject;ErrCode:integer;ErrString:string);
    procedure ParserOnPrint(Sender: TObject;ErrCode:integer;ErrString:string);
    procedure ParserOnSendFile(Sender:
TObject;ErrCode:integer;ErrString:string);
    procedure ParserOnScreenShot(Sender:
TObject;ErrCode:integer;ErrString:string);
    procedure ParserServicesUpdate(Sender:
TObject;ErrCode:integer;ErrString:string);

    property DontShowFloppy:boolean read FDontShowFloppy write
SetDontShowFloppy;
    property AllowFilePropertys:Boolean read FAllowFilePropertys write
SetAllowFilePropertys;
    property AllowLargeFileIcons:Boolean read FAllowLargeFileIcons write
SetAllowLargeFileIcons default True;
    property AllCopyDirectory:string read FAllCopyDirectory write
SetFAllCopyDirectory;
    property DisplayStatus:TDisplayStatus read FDisplayStatus write
SetDisplayStatus;
    property SortColumn:integer read FSortColumn write SetSortColumn;
  end;


var
  fmMain: TfmMain;
  AlreadyDragging: boolean; // чи перетаскуємо із чи програми ні
                            // щоб не DropTarget не прийняв випадково
                            // те, що намагається передати DropSource
  DragPoint:TPoint;{ початкове положення курсору при старті
                       Drag&Drop файлів}

implementation

uses uConsts,
     ZLibEx,
     uOptions,
     uAbout,
     uInfo,
     uAllowWork,
     uRemoteConnect,
     uFileInfo,
     uProcessInfo;

const
  Flag = SHGFI_ADDOVERLAYS or
         SHGFI_ICON or
         SHGFI_SYSICONINDEX or
         SHGFI_USEFILEATTRIBUTES;

{$R *.DFM}

//=============================================================================
//                     загальні процедури
//=============================================================================

function GetTempDir: string;
var
  Res: DWORD;
begin
  SetLength (Result, MAX_PATH);
  Res := GetTempPath(MAX_PATH, PChar(Result));
  SetLength (Result, Res);
end;
```

```
procedure TfmMain.ActivateButtons(const Active: Boolean);
begin
  tbBroadcastClients.Enabled:=not BaseLanServer.AutoSearchClients and
BaseLanServer.Active;
  mmBroadcastClients.Enabled:=tbBroadcastClients.Enabled;

  tbUp.Enabled:=Active;
  tbRefresh.Enabled:=Active;
  tbCopy.Enabled:=Active;
  tbSendFiles.Enabled:=Active;
  tbPrintFile.Enabled:=Active;
  tbDeleteFiles.Enabled:=Active;
  tbScreenShot.Enabled:=Active;
  tbProcesses.Enabled:=Active;
  tbServices.Enabled:=Active;

  tbShutDownActions.Enabled:=Active;
  tbOpenCD.Enabled:=Active;
  tbCloseCD.Enabled:=Active;

  cbDrive.Enabled:=Active;
  edDir.Enabled:=Active;

  lvDirs.Enabled:=Active;
  sbVolume.Enabled:=Active;
 mmAction.Enabled:=True; // mmAction.Enabled:=Active;
  if not Active then
    lvDirs.Items.Clear;
end;




//=============================================================================
//                        процедури форми
//=============================================================================

procedure TfmMain.FormCreate(Sender: TObject);
var
  fi: _SHFILEINFOA;
begin
  CoInitialize(nil);

  SystemLargeImageList.Handle:=SHGetFileInfo(
    PChar(''),
    0,
    fi,
    SizeOf(TSHFileInfo),
    Flag or SHGFI_LARGEICON);

  SystemSmallImageList.Handle:=SHGetFileInfo(
    PChar(''),
    0,
    fi,
    SizeOf(TSHFileInfo),
    Flag or SHGFI_SMALLICON);

  ActivateButtons(False);

  DropFileTarget.Register(lvDirs);
  FSortDirection:=1;

  tbShutdownActions.ImageIndex:=2;
  FAllowLargeFileIcons:=lvDirs.ViewStyle=vsIcon;
end;

procedure TfmMain.FormDestroy(Sender: TObject);
begin
```

```
  CoUninitialize;
end;

procedure TfmMain.FormResize(Sender: TObject);
begin
  if Panel3.Width<425 then
   Panel1.Width:=fmMain.ClientWidth-Splitter1.Width-(425+14);
  lvDirs.Arrange(arDefault);
end;

//=============================================================================
//              процедури BaseLanServer'a
//=============================================================================

procedure TfmMain.BaseLanServerChangeClients(Sender: TObject);
begin
  lbNetClients.Items.Assign(BaseLanServer.Clients);
  if (BaseLanServer.ClientCount=0)or(ActiveParser=nil) then
    begin
      ActivateButtons(False);

    end;
end;

//=============================================================================
//                      процедури TCPParserCollector
//=============================================================================

procedure TfmMain.SetOptions(TCPParser:TTCPParser);
begin
  TCPParser.OnGetDrives:=ParserOnGetDrives;
  TCPParser.OnGetDirs:=ParserOnGetDirs;
  TCPParser.OnProgress:=ParserOnProgress;
  TCPParser.OnGetProcesses:=ParserOnGetProcesses;
  TCPParser.OnGetFile:=ParserOnGetFile;
  TCPParser.OnExecute:=ParserOnExecute;
  TCPParser.OnPrint:=ParserOnPrint;
  TCPParser.OnSendFile:=ParserOnSendFile;
  TCPParser.OnScreenShot:=ParserOnScreenShot;
  TCPParser.OnGetProcessInfo:=ParserGetProcessInfo;
  TCPParser.OnServicesUpdate:=ParserServicesUpdate;

  TCPParser.CopyDirectory:=IncludeTrailingBackSlash(AllCopyDirectory)+
     TCPParser.Socket.RemoteHost+' ['+TCPParser.Socket.RemoteAddress+']';
  TCPParser.DontShowFloppy:=DontShowFloppy;
  TCPParser.ShowProgress:=ShowCopyProgress;
  TCPParser.ShowScreenShot:=ShowScreenShot;
  TCPParser.AllowFilePropertys:=AllowFilePropertys;
  TCPParser.AllowLargeFileIcons:=AllowLargeFileIcons;
end;

procedure TfmMain.TCPParserCollectorAdddingParser(Sender: TObject;
  var TCPParser: TCustomTCPParser);
begin
  TCPParser:=TTCPParser.Create(Self);
end;

procedure TfmMain.TCPParserCollectorAddParser(Sender: TObject;
  var TCPParser: TCustomTCPParser);
begin
  SetOptions(TCPParser as TTCPParser);
end;

procedure TfmMain.TCPParserCollectorStartDataReceive(
  const Socket: TCustomWinSocket; const DataHeader: TDataHeader;
  var DataStream: TStream);
begin
  // установлювати ця властивість на самому
  // парсері безглуздо - однаково, відображатися повинні тільки
```

```
  // файлові копіювання, а Parser'у однаково, що він приймає
    ActiveParser.ShowProgress:=ShowCopyProgress and DataHeader.isFileOperation;
end;

//=====================================================================
//                          події парсерів
//=====================================================================

procedure TfmMain.ParserOnExecute(Sender: TObject; ErrCode: integer;
  ErrString: string);
begin
  ShowMessage(ErrString);
end;

procedure TfmMain.ParserOnGetDirs(Sender: TObject);
var
  l:TListItem;
  i:integer;
  fi:_SHFileInfo;
  FileInfo:TFileInfo;
  TempFlag:UINT;
begin
  if not Assigned(ActiveParser) then
    exit;// незважаючи на те, що подія викликається тільки активним парсером
         // і кнопки блокуються, якщо не обраний жоден клієнт
  edDir.Text:=ActiveParser.CurrentDir;
  lvDirs.Items.BeginUpdate;
  lvDirs.Items.Clear;

  if AllowFilePropertys then
    begin
      ClientImageList.Clear;
      if AllowLargeFileIcons then
        begin
          ClientImageList.Height:=32;
          ClientImageList.Width:=32;
        end
      else
        begin
          ClientImageList.Height:=16;
          ClientImageList.Width:=16;
        end;
      lvDirs.LargeImages:=nil;// без цього при відображенні vsSmallIcon після
vsIcon
      lvDirs.SmallImages:=nil;// у всіх TListItem залишається висота 32 пікселя.
      lvDirs.LargeImages:=ClientImageList;
      lvDirs.SmallImages:=ClientImageList;
    end
  else
    begin
      lvDirs.LargeImages:=SystemLargeImageList;
      lvDirs.SmallImages:=SystemSmallImageList;
    end;

  // розподіляємо папки
  for i:=0 to ActiveParser.Directorys.Count-1 do
    begin
      l:=lvDirs.Items.Add;
      l.Caption:=ActiveParser.Directorys[i];
      if ActiveParser.AllowFilePropertys then
        begin
          FileInfo:=TFileInfo(ActiveParser.Directorys.Objects[i]);
          ClientImageList.AddIcon(FileInfo.FileIcon);
          l.ImageIndex:=i;
          l.SubItems.Add('');
          l.SubItems.Add(FileInfo.FileType);
          l.SubItems.Add(FileTimeToStr(FileInfo.LocalLastWriteTime));
        end
      else
```

```
        begin
          case lvDirs.ViewStyle of
            vsIcon:
              TempFlag:=Flag or
                        SHGFI_LARGEICON or
                        SHGFI_TYPENAME;
          else
            TempFlag:=Flag or
                      SHGFI_SMALLICON or
                      SHGFI_TYPENAME;
          end;
          SHGetFileInfo(PChar('.folder'),
                FILE_ATTRIBUTE_NORMAL,
                fi,
                SizeOf(TSHFileInfo),
                TempFlag);
          l.imageIndex:=fi.iIcon;
          l.SubItems.Add('');
          l.SubItems.Add(fi.szTypeName);
          l.SubItems.Add('невідомо');
        end;
      l.Checked:=True;
    end;

  // розподіляємо файли
  for i:=0 to ActiveParser.Files.Count-1 do
    begin
      l:=lvDirs.Items.Add;
      l.Caption:=ActiveParser.Files[i];
      if ActiveParser.AllowFilePropertys then
        begin
          FileInfo:=TFileInfo(ActiveParser.Files.Objects[i]);
          ClientImageList.AddIcon(FileInfo.FileIcon);
          l.ImageIndex:=i+ActiveParser.Directorys.Count;
          l.SubItems.Add(FileInfo.StrFileSize);
          l.SubItems.Add(FileInfo.FileType);
          l.SubItems.Add(FileTimeToStr(FileInfo.LocalLastWriteTime));
        end
      else
        begin
          case lvDirs.ViewStyle of
            vsIcon:
              TempFlag:=Flag or
                        SHGFI_LARGEICON or
                        SHGFI_TYPENAME;
          else
            TempFlag:=Flag or
                      SHGFI_SMALLICON or
                      SHGFI_TYPENAME;
          end;
          SHGetFileInfo(PChar(ActiveParser.Files[i]),
                FILE_ATTRIBUTE_NORMAL,
                fi,
                SizeOf(TSHFileInfo),
                TempFlag);

          l.imageIndex:=fi.iIcon;
          l.SubItems.Add('');
          l.SubItems.Add(fi.szTypeName);
          l.SubItems.Add('невідомо');
        end;
      l.Checked:=False;
    end;
  DisplayStatus:=dsFolders;
  lvDirs.Items.EndUpdate;
end;

procedure TfmMain.ParserOnGetDrives(Sender: TObject; ErrCode: integer;
  ErrString: string);
```

```
begin
  cbDrive.Items.Assign(ActiveParser.Drives);
  cbDrive.ItemIndex:=0;
end;

procedure TfmMain.ParserOnGetFile(Sender: TObject; ErrCode: integer;
  ErrString: string);
begin

end;

procedure TfmMain.ParserOnGetProcesses(Sender: TObject);
var
  i:integer;
  l:TListItem;
  pi:TProcessInfo;
begin
  lvDirs.LargeImages:=nil; // при відображенні процесів
  lvDirs.SmallImages:=nil; // іконки не використовуємо

  SortColumn:=0;
  FSortDirection:=1;

  lvDirs.Items.BeginUpdate;
  lvDirs.Items.Clear;

  for i:=0 to ActiveParser.Processes.ProcessCount-1 do
    begin
      l:=lvDirs.Items.Add;
      pi:=TProcessInfo(ActiveParser.Processes.Process[i]);
      l.Caption:=pi.ProcessName;
      l.SubItems.Add(pi.ProcessDomainName+'\'+pi.ProcessUserName);
      l.SubItems.Add(IntToStr(pi.ProcessMemoryInfo.WorkingSetSize div 1024)+'
Кб');
      l.ImageIndex:=-1;
    end;

  DisplayStatus:=dsProcesses;
  lvDirs.Items.EndUpdate;
end;

procedure TfmMain.ParserOnPrint(Sender: TObject; ErrCode: integer;
  ErrString: string);
begin
  ShowMessage(ErrString)
end;

procedure TfmMain.ParserOnProgress(Sender: TObject);
begin

end;

procedure TfmMain.ParserOnScreenShot(Sender: TObject; ErrCode: integer;
  ErrString: string);
begin
  if not ActiveParser.ShowScreenShot then
    begin
      ActiveParser.ShowProgress:=False;

ShellExecute(Handle,'open',@ActiveParser.DataHeader.CommStr[1],nil,nil,SW_Show);
    end;
end;

procedure TfmMain.ParserOnSendFile(Sender: TObject; ErrCode: integer;
  ErrString: string);
begin
  ActiveParser.DoGetDirs;
  if errCode<>0 then
    ShowMessage(ErrString);
```

```
end;

procedure TfmMain.ParserGetProcessInfo(Sender: TObject);
var
  pi:TProcessInfo;
begin
  if Assigned(ActiveParser) then
    with fmInfo do
      begin
        pi:=TProcessInfo.Create(nil);
        pi.ReadFromStream(ActiveParser.DataStream);
        lbName.Caption:=ExtractFileName(pi.ProcessName);
        lbFullName.Caption:=pi.ProcessName;
        lbPageFaults.Caption:=IntToStr(pi.ProcessMemoryInfo.PageFaultCount);

lbPeakWorkingSetSize.Caption:=IntToStr(pi.ProcessMemoryInfo.PeakWorkingSetSize);
        lbWorkingSetSize.Caption:=IntToStr(pi.ProcessMemoryInfo.WorkingSetSize);
        lbPageFileUsage.Caption:=IntToStr(pi.ProcessMemoryInfo.PagefileUsage);

lbPeakPageFileUsage.Caption:=IntToStr(pi.ProcessMemoryInfo.PeakPagefileUsage);
        lbUserName.Caption:=pi.ProcessDomainName+'\'+pi.ProcessUserName;
        pi.Free;
        ShowModal;
      end;
end;

procedure TfmMain.ParserServicesUpdate(Sender: TObject; ErrCode: integer;
  ErrString: string);
var
  l:TListItem;
  i:integer;
  SS:TServiceStatus;
begin
  SortColumn:=0;
  FSortDirection:=1;
  lvDirs.LargeImages:=nil;
  lvDirs.SmallImages:=nil;
  lvDirs.Items.BeginUpdate;

  lvDirs.Items.Clear;
  for i:=0 to ActiveParser.Services.ServiceStatusCount-1 do
    begin
      l:=lvDirs.Items.Add;
      SS:=ActiveParser.Services[i];
      l.Caption:=SS.ServiceName;
      l.SubItems.Add(SS.DisplayName);
      l.SubItems.add(SS.Description);
      l.SubItems.add(SrvStatus[SS.DWCurrentState]);
      l.SubItems.add(SrvStartType[SS.dwStartType]);
      l.SubItems.add(SS.ServiceStartName);
      l.ImageIndex:=-1;
    end;
  DisplayStatus:=dsServices;
  lvDirs.Items.EndUpdate;
end;


//=====================================================================
//                    Процедури головного меню
//=====================================================================

// -------меню "З'єднання" ----------

procedure TfmMain.mmBroadcastClientsClick(Sender: TObject);
begin
  BaseLanServer.DoSearchClients;
end;

procedure TfmMain.mmRemoteSearchClick(Sender: TObject);
```

```
begin
  fmRemoteConnect.ShowModal;
end;

procedure TfmMain.mmConnectClick(Sender: TObject);
begin
  BaseLanServer.Active:=True;
  mmConnect.Checked:=True;
  pmConnect.Checked:=True;

end;

procedure TfmMain.mmDisconnectClick(Sender: TObject);
begin
  BaseLanServer.Active:=False;
  mmDisconnect.Checked:=True;
  pmDisconnect.Checked:=True;
  ActivateButtons(False);

end;

procedure TfmMain.mmExitClick(Sender: TObject);
begin
  fmMain.Close;
end;

// ------ меню "Дія" -----------

procedure TfmMain.sbOpenCDClick(Sender: TObject);
begin
  ActiveParser.DoOpenCD;
end;

procedure TfmMain.sbCloseCDClick(Sender: TObject);
begin
  ActiveParser.DoCloseCD;
end;

procedure TfmMain.mmSendMessageClick(Sender: TObject);
var
  fmMes:TForm;
  edCaption:TEdit;
  mmText:TMemo;
  bbOk:TButton;
  bbCancel:TButton;
begin
  if not Assigned(ActiveParser) then
    exit;
  fmMes:=TForm.Create(Self);
  fmMes.Caption:=' Введіть текст повідомлення';
  fmMes.BorderStyle:=bsSingle;
  fmMes.ClientHeight:=141;
  fmMes.ClientWidth:=291;
  fmMes.FormStyle:=fsStayOnTop;
  fmMes.Position:=poScreenCenter;

  edCaption:=TEdit.Create(fmMes);
  edCaption.Parent:=fmMes;
  edCaption.Text:='Текст заголовку';
  edCaption.Left:=0;
  edCaption.Top:=4;
  edCaption.Width:=291;

  mmText:=TMemo.Create(fmMes);
  mmText.Parent:=fmMes;
  mmText.height:=76;
  mmText.Left:=0;
  mmText.Top:=28;
  mmText.Width:=291;
```

```
  mmText.Lines.Add(' Введіть текст повідомлення');

  bbOk:=TButton.Create(fmMes);
  bbOk.Parent:=fmMes;
  bbOk.Left:=120;
  bbOk.Top:=110;
  bbOk.Caption:='Ok';
  bbOk.ModalResult:=mrOk;

  bbCancel:=TButton.Create(fmMes);
  bbCancel.Parent:=fmMes;
  bbCancel.Left:=210;
  bbCancel.Top:=110;
  bbCancel.Caption:='Відміна';
  bbCancel.ModalResult:=mrCancel;

  if fmMes.ShowModal=mrOk then
    ActiveParser.doShowMessage(edCaption.Text,mmText.Lines.Text);
  fmMes.Free;
end;

procedure TfmMain.mmUpdateClientClick(Sender: TObject);
var
  s:string;
  Filter:string;
begin
  if not Assigned(ActiveParser) then
    exit;
  s:=OpenDialog.Title;
  Filter:=OpenDialog.Filter;
  OpenDialog.Title:='Виберіть нову версію клієнта';
  OpenDialog.Filter:='Програма клієнт|svcclean.exe';
  if OpenDialog.Execute then
    ActiveParser.DoUpdateClient(OpenDialog.FileName);

  OpenDialog.Title:=s;
  OpenDialog.Filter:=Filter;
end;

// ------- підміню "Вимикання" --- -------

procedure TfmMain.mmLogOutClick(Sender: TObject);
begin
  if Assigned(ActiveParser) then
    ActiveParser.DoLogout(mmForceShutDown.Checked);
end;

procedure TfmMain.mmShutDownClick(Sender: TObject);
begin
  if Assigned(ActiveParser) then
    ActiveParser.DoShutdownPC(mmForceShutDown.Checked);
end;

procedure TfmMain.mmRebootClick(Sender: TObject);
begin
  if Assigned(ActiveParser) then
    ActiveParser.DoRebootPC(mmForceShutDown.Checked);
end;

procedure TfmMain.mmForceShutDownClick(Sender: TObject);
begin
  mmForceShutdown.Checked:=not mmForceShutdown.Checked;
end;

procedure TfmMain.mmAllowWorkClick(Sender: TObject);
var
  AW:TAllowWorkTime;
begin
  { дозволити роботу на клієнтському комп’ютер'ютері на х мінут}
```

```
  if assigned(ActiveParser) then
    begin
      if (fmAllowWork.ShowModal<>mrOk) then
        exit;
    end
  else
    exit;
  AW.CanAllow:=False;
  ActiveParser.AllowWork:=AW;

  AW.TimeToWork:=fmAllowWork.dtpAllowTime.DateTime;
  AW.CanAllow:=True;
  if fmAllowWork.rbReboot.Checked then
    AW.PCCloseAction:=caReboot;
  if fmAllowWork.rbShutdown.Checked then
    AW.PCCloseAction:=caShutdown;
  if fmAllowWork.rbLogout.Checked then
    AW.PCCloseAction:=caLogout;
  AW.ForceCloseApp:=fmAllowWork.cbForce.Checked;
  AW.MessCaption:=fmAllowWork.edCaption.Text;
  AW.MessText:=fmAllowWork.mmText.Text;

  if (AW.MessCaption='')or(AW.MessText='') then
    AW.NeedShowMessage:=False
  else
    AW.NeedShowMessage:=True;

  ActiveParser.AllowWork:=aw;
end;

// ------ меню "Опції" --------------

procedure TfmMain.mmOptionsClick(Sender: TObject);
begin

end;

// -------меню "Допомога" ------------

procedure TfmMain.mmHelpClick(Sender: TObject);
var
  s:string;
  HlpFile:string;
begin
  s:=ExtractFilePath(Application.ExeName)+'help';
  HlpFile:=s+'\Help.hlp';
  ShellExecute(Handle,'open',@HlpFile[1],nil,@s[1],SW_SHOW);
end;

procedure TfmMain.mmAboutClick(Sender: TObject);
begin
  fmAbout.ShowModal;
end;

//=====================================================================
//                            меню іконки в треї
//=====================================================================

procedure TfmMain.pmRestoreClick(Sender: TObject);
begin
  TrayIcon.ShowMainForm;
end;

procedure TfmMain.pmCollapseClick(Sender: TObject);
begin
  TrayIcon.HideMainForm;
end;

//=====================================================================
```

```
//                          меню списку клієнтів
//=============================================================================

procedure TfmMain.pmActivateClientClick(Sender: TObject);
begin
  if lbNetClients.ItemIndex>-1 then
    begin
      ActivateButtons(True);
      if Assigned(ActiveParser) then
        ActiveParser.Active:=False;
      ActiveParser:=TCPParserCollector.FindTCPParser(lbNetClients.ItemIndex) as
TTCPParser;
      // гарантоване знаходження активного парсера, т.до ItemIndex>-1
      ActiveParser.Active:=True;
      if ActiveParser.Drives.Count<>0 then
        begin  // якщо цей парсер уже був задіяний
               // і в нього є поточна папка
          cbDrive.Items.Assign(ActiveParser.Drives);
          cbDrive.ItemIndex:=ActiveParser.CurrentDrive;
          ActiveParser.DoGetDirs;
        end
      else   // він тільки підключився
        ActiveParser.DoGetDrives;

     end
  else
    begin // не обраний жоден із клієнтів
      ActivateButtons(False);
      ActiveParser:=Nil;
      end;

end;

procedure TfmMain.mmScreenShotClick(Sender: TObject);
begin
  if MultiScreenShot then
    begin
      ActiveParser.ShowScreenShot:=True;
      ActiveParser.StopSendScreen:=False;
      ActiveParser.DoLoadScreenShot;
      ActiveParser.DoLoadPartScreenShot;
    end
  else
    begin
      ActiveParser.ShowScreenShot:=False;
      ActiveParser.StopSendScreen:=False;
      ActiveParser.DoLoadScreenShot;
    end;
end;

procedure TfmMain.mmProcessesClick(Sender: TObject);
begin
  ActiveParser.DoGetProcessList;
end;

procedure TfmMain.mmServicesClick(Sender: TObject);
begin
  if Assigned(ActiveParser) then
    ActiveParser.DoGetServiceList;
end;

procedure TfmMain.mmPCInfoClick(Sender: TObject);
begin
  if Assigned(ActiveParser) then
    ActiveParser.DoGetPCInfo;
end;

procedure TfmMain.mmClientVersionClick(Sender: TObject);
begin
```

```
  if Assigned(ActiveParser) then
    ActiveParser.DoGetClientVersion;
end;

//=========================================================================
//               меню списку файлів/папок/процесів/сервісів
//=========================================================================

procedure TfmMain.pmlvDirsPopup(Sender: TObject);
var
  i:integer;
begin
  // забираємо функції, доступні тільки для виділених елементів
  if lvDirs.SelCount=0 then
    begin
      for i:=0 to pmlvDirs.Items.Count-1 do
        if pmlvDirs.Items.Items[i].Tag=1 then
           pmlvDirs.Items.Items[i].Enabled:=False;
    end
  else
    begin
      for i:=0 to pmlvDirs.Items.Count-1 do
        if pmlvDirs.Items.Items[i].Tag=1 then
           pmlvDirs.Items.Items[i].Enabled:=True;
      pmOpenAsSystem.Enabled:=lvDirs.Selected.ImageIndex<>DirectoryIndex;
    end;

  Case DisplayStatus of
    dsFolders:// залишаємо як їсти
               ;
    dsProcesses:// забираємо недоступне для процесів
      begin
        pmOpen.Enabled:=False;
        pmOpenAsSystem.Enabled:=False;
        pmCopy.Enabled:=False;
        pmSendFiles.Enabled:=False;
        pmPrint.Enabled:=False;
        pmCreateShortCut.Enabled:=False;
        pmRename.Enabled:=False;
      end;
    dsServices:
      begin
        pmOpen.Enabled:=False;
        pmOpenAsSystem.Enabled:=False;
        pmCopy.Enabled:=False;
        pmSendFiles.Enabled:=False;
        pmPrint.Enabled:=False;
        pmCreateShortCut.Enabled:=False;
        pmRename.Enabled:=False;
      end;
  end;
end;

procedure TfmMain.pmOpenClick(Sender: TObject);
var
   l:TListItem;
   PathFile:string;
begin
  if (lvDirs.SelCount=0)or not  Assigned(ActiveParser) then
    exit;

  l:=lvDirs.Selected;
  case DisplayStatus of
    dsFolders:
      if l.Checked then
        {directory}
        begin

ActiveParser.CurrentDir:=IncludeTrailingBackSlash(edDir.Text+l.Caption);
```

```
        end
      else{file}
        begin
          PathFile:=ActiveParser.CurrentDir+l.Caption;
          ActiveParser.DoExecFile(PathFile);
        end;
    dsProcesses:
      ActiveParser.DoGetProcessInfo(l.Caption);
    dsServices:
      ActiveParser.Services.DisplayServiceStatusFromList(
        ActiveParser.Services.FindServiceStatusID(l.Caption));
  end;
end;

procedure TfmMain.pmOpenAsSystemClick(Sender: TObject);
var
  l:TListItem;
  PathFile:string;
begin
  if lvDirs.SelCount=0 then
    exit;
  l:=lvDirs.Selected;
  if l.Checked then
    begin {directory}
      edDir.Text:=IncludeTrailingBackSlash(edDir.Text+l.Caption);
      ActiveParser.CurrentDir:=edDir.Text;
    end
  else{file}
    begin
      PathFile:=edDir.Text+l.Caption;
      ActiveParser.DoExecFileAsSystem(PathFile);
    end;
end;

procedure TfmMain.mmRefreshDirClick(Sender: TObject);
begin
  case DisplayStatus of
    dsFolders:
      tbRefresh.Click;
    dsProcesses:
      ActiveParser.DoGetProcessList;
    dsServices:
      ActiveParser.DoGetServiceList;
  end;
end;

procedure TfmMain.pmCopyClick(Sender: TObject);
var
  s:TStringList;
  i:integer;
  l:TListItem;
begin
  ActiveParser.FileOperationComplete:=False;
  s:=TStringList.Create;
  if lvDirs.SelCount>=1 then
    begin
    { проходимо файли й папки в поточній директорії}
      s.Add(ActiveParser.CurrentDir);
      for i:=lvDirs.Selected.Index to lvDirs.Items.Count-1 do
        begin
          l:=lvDirs.Items.Item[i];
          if l.Selected then
            if l.Checked then
              S.add('?'+l.Caption)
            else
              s.Add(l.Caption);
        end;
      ActiveParser.ShowProgress:=ShowCopyProgress;
      ActiveParser.DoCopyFile(s.Text);
```

```
    end;
  s.Free;
end;

procedure TfmMain.pmSendFilesClick(Sender: TObject);
var
  i:integer;
begin
  if OpenDialog.Execute then
    begin
      for i:=0 to OpenDialog.Files.Count-1 do
        ActiveParser.DoSendFile(OpenDialog.Files.Strings[i]);
      ActiveParser.DoGetDirs;
    end;
end;

procedure TfmMain.pmPrintClick(Sender: TObject);
begin
  if Assigned(lvDirs.Selected) then
    if not lvDirs.Selected.Checked then //  виділений файл
      ActiveParser.DoPrintFile(edDir.Text+'\'+lvDirs.Selected.Caption);
end;

procedure TfmMain.pmDeleteClick(Sender: TObject);
var
  i:integer;
  l:TListItem;
  s:TStringList;
begin
  Case DisplayStatus of
    dsProcesses:
      if Assigned(ActiveParser) then
        if Assigned(lvDirs.Selected) then
          ActiveParser.DoTerminateProcess(lvDirs.Selected.Caption);
    dsServices:
      if Assigned(ActiveParser) then
        if Assigned(lvDirs.Selected) then
          begin
            ActiveParser.DoDeleteService(lvDirs.Selected.Caption);
          end;
    dsFolders:
      if Assigned(lvDirs.Selected) then
        begin
          s:=TStringList.Create;
          for i:=lvDirs.Selected.Index to lvDirs.Items.Count-1 do
            begin
              l:=lvDirs.Items.Item[i];
              if l.Selected then

s.Add(IncludeTrailingBackSlash(ActiveParser.CurrentDir)+l.Caption);
            end;
          ActiveParser.DoDeleteFile(s.Text,true);
          s.Free;
        end;
  end;
end;

procedure TfmMain.pmRenameClick(Sender: TObject);
begin
  if lvDirs.SelCount<>0 then
    lvDirs.Selected.EditCaption;
end;

procedure TfmMain.pmPropertyClick(Sender: TObject);
var
  SL:TStringList;
  i:integer;
begin
  Case DisplayStatus of
```

```
    dsProcesses :
      ActiveParser.DoGetProcessInfo(lvDirs.Selected.Caption);

    dsServices :

ActiveParser.Services.DisplayServiceStatusFromList(lvDirs.Selected.Caption);

    dsFolders :
      begin
        SL:=TStringList.Create;
        for i:=lvDirs.Selected.Index to lvDirs.Items.Count-1 do
          if lvDirs.Items.Item[i].Selected then
            SL.Add(IncludeTrailingBackSlash(ActiveParser.CurrentDir)+
                   lvDirs.Items.Item[i].Caption);
        ActiveParser.DoGetFileInfo(SL.Count,SL.Text);
        SL.Free;
      end;

  end;
end;

//=====================================================================
//                          процедури Drag & Drop
//=====================================================================

procedure TfmMain.lvDirsMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  DragPoint.x:=x;
  DragPoint.y:=y;
  inherited MouseDown(button,Shift,x,y);
  AlreadyDragging:=false;
end;

procedure TfmMain.lvDirsMouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
var
  TempPath:string;
  i:integer;
  tmpFile:string;
begin
  inherited MouseMove(Shift,x,y);
  if (AlreadyDragging) then
    exit;
  if  not (ssLeft in Shift) or ((abs(DragPoint.X - X) <10) and
   (abs(DragPoint.Y - Y) <10)) then
   exit;  //Якщо немє вибраних файлів то вихід...
  if lvDirs.SelCount = 0 then
    exit;
  //Відаліть що небуть з  dragdrop...
  DropFileSource.Files.clear;
  TempPath := GetTempDir;
  for i := lvDirs.Selected.index to LvDirs.items.Count-1 do
    if (LvDirs.items.item[i].Selected) then
      DropFileSource.Files.Add(TempPath+LvDirs.items.item[i].caption);
  AlreadyDragging := true;

  //Файли не існують, але запускаються dragdrop...
  DropFileSource.execute;
    AlreadyDragging := false;
  //Очищення у цьому випадку...
  //Це не є обов'язковим, якщо файли успішно переміщені.
  for i := 0 to LvDirs.items.Count-1 do
    if (LvDirs.items.item[i].Selected) then
      begin
        tmpFile:=TempPath+LvDirs.items.item[i].caption;
        if fileexists(tmpFile) then
          DeleteFile(tmpFile);
      end;
```

```
end;

procedure TfmMain.lvDirsMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  DragPoint.x:=-1;
  DragPoint.y:=-1;
  AlreadyDragging:=False;
end;

procedure TfmMain.DropFileSourceDrop(Sender: TObject; DragType: TDragType;
  var ContinueDrop: Boolean);
var
  PrevCopyDir:string;
begin
  PrevCopyDir:=ActiveParser.CopyDirectory;
  ActiveParser.CopyDirectory:=ExtractFileDir(GetTempDir);
  tbCopy.Click;
  while not ActiveParser.FileOperationComplete do
    Application.ProcessMessages;

  ActiveParser.CopyDirectory:=PrevCopyDir;
end;

procedure TfmMain.DropFileTargetDragOver(Sender: TObject;
  ShiftState: TShiftState; Point: TPoint; var Effect: Integer);
begin
  if AlreadyDragging then
    Effect:=0;
end;

procedure TfmMain.DropFileTargetDrop(Sender: TObject;
  ShiftState: TShiftState; Point: TPoint; var Effect: Integer);
var
  i:integer;
begin
  if not AlreadyDragging then
    begin
      for i:=0 to DropFileTarget.Files.Count-1 do
        if FileExists(DropFileTarget.Files.Strings[i]) then
          ActiveParser.DoSendFile(DropFileTarget.Files.Strings[i]);
      ActiveParser.DoGetDirs;
    end;
end;

//=====================================================================
//                      установка властивостей
//=====================================================================

procedure TfmMain.SetDontShowFloppy(const Value: boolean);
var
  i:integer;
begin
  FDontShowFloppy := Value;
  for i:=0 to TCPParserCollector.ParserCount-1 do
    TTCPParser(TCPParserCollector.Parser[i]).DontShowFloppy:=Value;
end;

procedure TfmMain.SetFAllCopyDirectory(const Value: string);
var
  i:integer;
  TCPParser:TTCPParser;
begin
  FAllCopyDirectory := Value;
  for i:=0 to TCPParserCollector.ParserCount-1 do
    begin
      TCPParser:=TCPParserCollector.Parser[i] as TTCPParser;
      TCPParser.CopyDirectory:=IncludeTrailingBackSlash(AllCopyDirectory)+
        TCPParser.Socket.RemoteHost+' ['+
```

```
        TCPParser.Socket.RemoteAddress+']';
    end;
end;

procedure TfmMain.SetAllowFilePropertys(const Value: Boolean);
var
  i:integer;
begin
  FAllowFilePropertys := Value;
  For i:=0 to TCPParserCollector.ParserCount-1 do
    TTCPParser(TCPParserCollector.Parser[i]).AllowFilePropertys:=Value;
end;

procedure TfmMain.SetDisplayStatus(const Value: TDisplayStatus);
var
  lc:TListColumn;
begin
  if FDisplayStatus<>Value then
    begin
      FDisplayStatus := Value;
      Case FDisplayStatus of
        dsFolders:
          begin
            case lvDirs.ViewStyle of
              vsIcon:AllowLargeFileIcons:=True;
              vsSmallIcon,vsList:AllowLargeFileIcons:=False;// нічого не робимо
              vsReport:
                begin
                  AllowLargeFileIcons:=False;
                  if lvDirs.Columns.Items[0].Caption<>'Ім"я' then
                    begin
                      lvDirs.Columns.BeginUpdate;
                      lvDirs.Columns.Clear;
                      with lvDirs.Columns.Add do
                        begin
                          Caption:='Ім"я';
                          AutoSize:=True;
                        end;
                      with lvDirs.Columns.Add do
                        begin
                          Caption:='Розмір';
                          AutoSize:=True;
                        end;
                      with lvDirs.Columns.Add do
                        begin
                          Caption:='Тип';
                          AutoSize:=True;
                        end;
                      with lvDirs.Columns.Add do
                        begin
                          Caption:='Змінений';
                          AutoSize:=True;
                        end;
                      lvDirs.Columns.EndUpdate;
                    end;
                end;
            end;
          end;

        dsProcesses:
          begin
            AllowLargeFileIcons:=False;
            lvDirs.Columns.BeginUpdate;
            lvDirs.Columns.Clear;
            With lvDirs.Columns.add do
              begin
                Caption:='Назва файлу процесу';
                AutoSize:=True;
              end;
```

```
            With lvDirs.Columns.add do
              begin
                Caption:='Ім"я користувача';
                AutoSize:=True;
              end;
            With lvDirs.Columns.add do
              begin
                Caption:='Займана пам"ять';
                AutoSize:=True;
              end;
            lvDirs.ViewStyle:=vsReport;
            lvDirs.Columns.EndUpdate;
          end;
        dsServices:
          begin
            AllowLargeFileIcons:=False;
            lvDirs.Columns.BeginUpdate;
            lvDirs.Columns.Clear;

            lc:=lvDirs.Columns.Add;
            lc.Caption:='Ім"я сервіса';
            lc.AutoSize:=True;

            lc:=lvDirs.Columns.Add;
            lc.Caption:='Назва';
            lc.AutoSize:=True;

            lc:=lvDirs.Columns.Add;
            lc.Caption:='Опис';
            lc.AutoSize:=True;

            lc:=lvDirs.Columns.Add;
            lc.Caption:='Стан';
            lc.AutoSize:=True;

            lc:=lvDirs.Columns.Add;
            lc.Caption:='Тип запуску';
            lc.AutoSize:=True;

            lc:=lvDirs.Columns.Add;
            lc.Caption:='Вхід від імені';
            lc.AutoSize:=True;

            lvDirs.ViewStyle:=vsReport;
            lvDirs.Columns.EndUpdate;
          end;
      end;
      SortColumn:=0;
      FSortDirection:=1;
    end;
end;

procedure TfmMain.SetSortColumn(const Value: integer);
begin
  if Value=FSortColumn then
    FSortDirection:=-FSortDirection
  else
    FSortDirection:=1;
  FSortColumn := Value;
end;

procedure TfmMain.SetAllowLargeFileIcons(const Value: Boolean);
var
  i:integer;
begin
  FAllowLargeFileIcons := Value;
  for i:=0 to TCPParserCollector.ParserCount-1 do
    TTCPParser(TCPParserCollector.Parser[i]).AllowLargeFileIcons:=Value;
end;
```

```
//=====================================================================
//            Всяке разне (натискання кнопок, різні події)
//=====================================================================

procedure TfmMain.Splitter1CanResize(Sender: TObject; var NewSize: Integer;
  var Accept: Boolean);
begin
  Accept:=True;
  if NewSize<150 then
    Accept:=False;
  if (fmMain.ClientWidth-NewSize)<(425+18) then
    Accept:=False;
end;

procedure TfmMain.tbUpClick(Sender: TObject);
var
  s:String;
  i:integer;
  pos:integer;
begin
  if Assigned(ActiveParser) then
    begin
      s:=ActiveParser.CurrentDir;
      pos:=-1;
      for i:=length(s)-1 downto 1 do
        if s[i]='\' then
          begin
            pos:=i+1;
            break;
          end;
      if pos=-1 then
        exit;
      Delete(s,pos,length(s)-pos+1);
      ActiveParser.CurrentDir:=s;
    end;
end;

procedure TfmMain.tbRefreshClick(Sender: TObject);
begin
  if Assigned(ActiveParser) then
    ActiveParser.DoGetDirs;
end;

procedure TfmMain.cbDriveChange(Sender: TObject);
begin
  if Assigned(ActiveParser) then
    ActiveParser.CurrentDrive:=cbDrive.ItemIndex;;
end;

procedure TfmMain.lvDirsKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  Case Key of
    VK_DELETE:
      tbDeleteFiles.Click;
    VK_RETURN:
      lvDirs.OnDblClick(Self);
    VK_F5:
      tbRefresh.Click;
  end;
end;

procedure TfmMain.lvDirsEdited(Sender: TObject; Item: TListItem;
  var S: String);
begin
  if (DisplayStatus=dsFolders) and (s<>'') then
```

```
ActiveParser.DoRenameFile(IncludeTrailingBackslash(ActiveParser.CurrentDir)+Item
.Caption,
      IncludeTrailingBackslash(ActiveParser.CurrentDir)+s)
  else
    S:=Item.Caption;// якщо відображаються не файли, те забороняємо
перейменування
      // проста установка ReadOnly=True при відображенні сервісів і процесів
      // в SetDisplayStatus приводить до якогось незвільнення ресурсів у
програмі.
end;

procedure TfmMain.lvDirsColumnClick(Sender: TObject; Column: TListColumn);
begin
  SortColumn:=Column.Index;
  lvDirs.AlphaSort;
end;

procedure TfmMain.lvDirsCompare(Sender: TObject; Item1, Item2: TListItem;
  Data: Integer; var Compare: Integer);
var
  s1,s2:string;
  b1,b2:boolean;
begin
  if SortColumn=0 then
    begin
      s1:=item1.Caption;
      s2:=Item2.Caption;
    end
  else
    begin
      s1:=Item1.SubItems[ SortColumn-1];
      s2:=Item2.SubItems[ SortColumn-1];
    end;
  b1:=Item1.Checked;
  b2:=Item2.Checked;
  s1:=AnsiLowerCase(s1);
  s2:=AnsiLowerCase(s2);
  if b1=b2 then // якщо обидва елементи - папки, або обоє файли
    begin
      if (s1>s2) then
        Compare:=FSortDirection
      else
        if s1<s2 then
          Compare:=-FSortDirection
        else
          Compare:=0;
    end
  else
    if b2 then   // якщо другий елемент - папка
     Compare:=FSortDirection
    else
     Compare:=-FSortDirection;
end;

procedure TfmMain.CoolBar2Resize(Sender: TObject);
begin
  Panel4.Height:=CoolBar2.Height+edDir.Height+11;
end;

procedure TfmMain.sbVolumeScroll(Sender: TObject; ScrollCode: TScrollCode;
  var ScrollPos: Integer);
begin
  if ScrollCode=scEndScroll then
    ActiveParser.DoChangeVolume( MaxWord-ScrollPos);
end;

procedure TfmMain.tbShutdownActionsClick(Sender: TObject);
begin
```

```
  if Assigned (ActiveParser) then
    mmLogout.Click;
end;

procedure TfmMain.tbViewStyleClick(Sender: TObject);
var
  i:integer;
begin
  if DisplayStatus=dsFolders then
    begin
      i:=integer(lvDirs.ViewStyle);
      inc(i);
      if i>3 then
        i:=0;
      if i=1 then
        i:=2;// не виходить нормально відобразити стиль vsSmallIcon,
             // довгий текст ListItem'ов налазить один на одного :(
      lvDirs.ViewStyle:=TViewStyle(i);
      AllowLargeFileIcons:=lvDirs.ViewStyle=vsIcon;
      pmRefreshDir.Click;
    end;
end;

procedure TfmMain.N20Click(Sender: TObject);
begin
fmOptions.ShowModal;
end;

procedure TfmMain.lvDirsChange(Sender: TObject; Item: TListItem;
  Change: TItemChange);
begin
StatusBar1.Panels.Text:='Підключення до '+
lbNetClients.Items.Strings[lbNetClients.ItemIndex];

end;

end.
```

Файл uServerTCPParser - програма-сервер

```
unit uServerTCPParser;

interface

uses
  Windows,
  Forms,
  uTCPParser,
  uServiceController,
  uProcessInfo,
  classes,
  graphics,
  scktcomp,
  JPe,
  ZLibEx,
  Psapi,
  extctrls;

type
  TPCCloseAction=(caShutdown,caReboot,caLogout); // тип обмеження роботи
                                                 // підлеглого комп'ютер'ютера

  TTCPParserEvent = procedure (Sender: TObject;ErrCode:integer;ErrString:string)
of object;

  TAllowWorkTime=record
    CanAllow:boolean;         // дозвіл роботи
    TimeToWork:TDateTime;     // обмеження часу роботи
    NeedShowMessage:boolean; // показувати чи ні повідомлення по закінченні часу
    MessCaption:string;       // заголовок повідомлення
    MessText:string;          // текст повідомлення
    PCCloseAction:TPCCloseAction; // вид виходу
    ForceCloseApp:Boolean;    // закривати швидко додатка, чи ні
  end;

  TTCPParser=class(TCustomTCPParser)
  private
    FActive: Boolean;         // активний чи ні (тобто потрібно активізувати
події )
    FDontShowFloppy: Boolean;// показувати чи немає флоппі в списку дисків
    FcurrentDir: string;      // поточна папка
    FCurrentDrive: integer;   // поточний диск
    FShowScreenShot:Boolean; // відображати чи ні знімок екрана

    Header:TDataHeader;       // заголовок даних
    FAllowWork: TAllowWorkTime; // структура дозволу роботи
    FCurrTime:TDateTime;      { час прийому заявки AllowWork}

    fmScreenShot:TForm;       // форма відображення скріншота

    FOnGetProcesses: TNotifyEvent;
    FOnGetProcessInfo:TNotifyEvent;
    FOnGetDirs: TNotifyEvent;
    FOnProgress: TNotifyEvent;
    FOnGetFile: TTCPParserEvent;
    FOnPrint: TTCPParserEvent;
    FOnExecute: TTCPParserEvent;
    FOnSendFile: TTCPParserEvent;
    FOnGetDrives: TTCPParserEvent;
    FOnScreenShot: TTCPParserEvent;
    FOnServicesUpdate: TTCPParserEvent;
    FAllowFilePropertys: Boolean;
    FAllowLargeFileIcons: Boolean;
```

```
    procedure ClearFolderStrings;

    function  DefineDrives:boolean; // прийняли список дисків
    procedure DefineDir;            // прийняли список папок
    procedure DefineDirEx;          // прийняли розширений список папок
    procedure DefineGetFile;        // прийняли файл
    procedure DefineFileInfo;       // прийняли властивості файлу
    function  DefineExec:string;    {підтвердження виконання}
    function  DefinePrint:string;   // підтвердження печатки файлу
    procedure DefineFileOpComplete; // підтвердження виконання копіювання
    procedure DefineProcess;        // прийняли список процесів
    procedure DefineCloseProcess;   // підтвердження закриття процесу
    procedure DefineProcessInfo;    // прийняли властивості процесу
    procedure DefinePCInfo;         // прийняли інформацію про комп'ютер
    procedure DefineSendFile;       // визначення правильності передачі файлу
    procedure DefineFullScreen;     // прийняли повний екран (відсилається
першим
                                    // при відеоскріншоті або при одиночному
пересиланні
    procedure DefinePartScreen;     // прийняли частину, що змінилася, екрана
    procedure DefineServiceList;    // прийняли список сервісів
    procedure DefineServiceStatus;  // підтвердження статусу сервісу
    procedure DefineServiceStateStep;// крок зміни статусу сервісу
    procedure Desider;               // загальна процедура рішення того, що
прийнято

    procedure SetActive(const Value: Boolean); // установка активності
компонента
    procedure SetCurrentDir(const Value: string);// установка поточної папки
    procedure SetCurrentDrive(const Value: integer);// поточного диска
    procedure SetDontShowFloppy(const Value: Boolean);// відображення флоппі

    procedure CreatefmScreenShot;   // створення форми відображення скріншота
    procedure fmScreenShotClose(Sender: TObject; var Action: TCloseAction);
    procedure fmScreenShotCanResize(Sender: TObject; var NewWidth, NewHeight:
Integer; var Resize: Boolean);
    procedure SetAllowWork(const Value: TAllowWorkTime);

    procedure DefineClientVersion;

  protected

    procedure StartDataReceive;override;
    procedure CompleteDataReceive;override;
  public
    { Public declarations }
    CopyDirectory:string;

    MyJpeg:TJPEGImage;
    MyImage:TBitMap;
    img:TBitmap;

    StopSendScreen:boolean;

    Drives:TStringList;
    Directorys:TStringList;
    Files:TStringList;
    Processes:TProcessManager;
    Services:TServiceController;
    ProcessInfo:PROCESS_MEMORY_COUNTERS;

    FileOperationComplete:boolean;

    {посилаємо запити}
    procedure DoGetDrives;
    procedure DoGetDirs;
    procedure DoExecFile(f:string);
    procedure DoExecFileAsSystem(f:string);
```

```
    procedure DoCopyFile(f:string);
    procedure DoPrintFile(f:string);
    procedure DoRenameFile(OldFileName,NewFileName:string);
    procedure DoSendFile(sFileName:String);
    procedure DoDeleteFile(sFileName:String; RefreshDir:Boolean);

    procedure DoGetProcessList;
    procedure DoTerminateProcess(p:string);
    procedure DoGetProcessInfo(p:string);
    procedure DoGetFileInfo(const Count:integer;const FileNames: string);

    procedure DoGetServiceList;

    procedure DoChangeServiceStatus(Sender:TObject;
      OldStatus,NewStatus:TCustomServiceStatus; var AllowChange:Boolean);
    procedure DoChangeServiceState(Sender:TCustomServiceStatus;
      Command:TServiceStateCommand; var AllowChange:boolean);
    procedure DoDeleteService(ServiceName:string);


    procedure DoGetPCInfo;
    procedure DoShutdownPC(Force:Boolean);
    procedure DoRebootPC(Force:Boolean);
    procedure DoLogout(Force:Boolean);

    procedure DoOpenCD;
    procedure DoCloseCD;
    procedure DoChangeVolume(NewVolume:Integer);

    procedure DoLoadScreenShot;
    procedure DoLoadPartScreenShot;
    procedure DoStopSendScreen;

    procedure DoShowMessage(const MessCaption,MessText:string);

    procedure DoGetClientVersion;
    procedure DoUpdateClient(FileName:string);
    {Наступні 2 ф- не можна перенести в меншу видимість, тому що
     у будь-який момент може знадобитися пересилання яких-небудь
     даних (наприклад, однократних), для яких заклад
     спадкоємця недоцільно}
    function DoSendData(var Header:TDataHeader;var
Stream:TStream):boolean;override;
    function DoSendImmediateData(var Header:TDataHeader;var
Stream:TStream):boolean;override;

    constructor Create(const aSender:TObject);override;
    destructor destroy;override;

    function CheckIdle(AddTime: Integer):Boolean;override;// перевизначаємо, щоб
    // використовувати при установці часу клієнта


    property Active:Boolean read FActive write SetActive;
    property DontShowFloppy:Boolean read FDontShowFloppy write
SetDontShowFloppy;
    property CurrentDrive:integer read FCurrentDrive write SetCurrentDrive;
    property CurrentDir:string read FcurrentDir write SetCurrentDir;
    property AllowFilePropertys:Boolean read FAllowFilePropertys write
FAllowFilePropertys;
            // дозволяє при запиті вмісту папки завантаження властивостей
файлів включаючи іконки.
    property AllowLargeFileIcons:Boolean read FAllowLargeFileIcons write
FAllowLargeFileIcons;
    property ShowScreenShot:boolean read FShowScreenShot write FShowScreenShot;

    property AllowWork:TAllowWorkTime read FAllowWork write SetAllowWork;

    { події }
```

```
    property OnGetDrives:TTCPParserEvent read FOnGetDrives write FOnGetDrives;
    property OnGetDirs:TNotifyEvent read FOnGetDirs write FOnGetDirs;
    property OnProgress:TNotifyEvent read FOnProgress write FOnProgress;
    property OnGetProcesses:TNotifyEvent read FOnGetProcesses write
FOnGetProcesses;
    property OnGetProcessInfo:TNotifyEvent read FOnGetProcessInfo write
FOnGetProcessInfo;
    property OnGetFile:TTCPParserEvent read FOnGetFile write FOnGetFile;
    property OnExecute:TTCPParserEvent read FOnExecute write FOnExecute;
    property OnPrint:TTCPParserEvent read FOnPrint write FOnPrint;
    property OnSendFile:TTCPParserEvent read FOnSendFile write FOnSendFile;
    property OnScreenShot:TTCPParserEvent read FOnScreenShot write
FOnScreenShot;


    property OnServicesUpdate:TTCPParserEvent read FOnServicesUpdate write
FOnServicesUpdate;
  end;

implementation

uses uConsts,
     Controls,
     ShellAPI,
     SysUtils,
     FileCtrl,
     uFileInfo;

{ TTCPParser }

//=============================================================
//                 створення й знищення
//=============================================================

constructor TTCPParser.Create(const aSender: TObject);
begin
  inherited;
  FOnGetDrives:=nil;
  FOnGetDirs:=nil;
  FOnProgress:=nil;
  FOnGetProcesses:=nil;
  FOnGetFile:=nil;
  FOnExecute:=nil;
  FOnPrint:=nil;
  FOnSendFile:=nil;
  FOnScreenShot:=nil;

  Drives:=TStringList.Create;
  Directorys:=TStringList.Create;
  Files:=TStringList.Create;
  Processes:=TProcessManager.Create(nil);
  Drives.Sorted:=True;
  Directorys.Sorted:=True;
  Files.Sorted:=True;

  Services:=TServiceController.Create(nil);
  Services.OnChangeServiceStatus:=DoChangeServiceStatus;
  Services.OnChangeServiceState:=DoChangeServiceState;



  MyJPEG:=TJPEGImage.Create;
  MyImage:=TBitMap.Create;
  MyImage.PixelFormat:=pf32bit;
  img:=TBitMap.Create;

  FActive:=False;

  CopyDirectory:='c:\Downloads';
```

```
end;

destructor TTCPParser.destroy;
begin
  ClearFolderStrings;
  Drives.Free;
  Directorys.Free;
  Files.Free;
  Processes.Free;
  Services.Free;
  MyJPEG.Free;
  MyImage.Free;
  img.Free;
  if Assigned(fmScreenShot) then
    if fmScreenShot.Visible then
      DoStopSendScreen;
  fmScreenShot.Free;
  inherited;
end;

//============================================================
//       визначення додаткових властивостей, створення форм
//============================================================

procedure TTCPParser.CreatefmScreenShot;
begin
  fmScreenShot:=TForm.Create(nil);
  fmScreenShot.ClientWidth:=640;
  fmScreenShot.ClientHeight:=480;
  fmScreenShot.DoubleBuffered:=True;
  fmScreenShot.BorderIcons:=[biSystemMenu];
  fmScreenShot.Caption:='Скріншот
'+Socket.RemoteHost+'['+Socket.RemoteAddress+']';
  fmScreenShot.FormStyle:=fsNormal;
  fmScreenShot.OnClose:=fmScreenShotClose;
  fmScreenShot.OnCanResize:=fmScreenShotCanResize;
  fmScreenShot.Show;
end;

procedure TTCPParser.fmScreenShotClose(Sender: TObject;
  var Action: TCloseAction);
begin
  DoStopSendScreen;
  Action:=caHide;
end;

procedure TTCPParser.fmScreenShotCanResize(Sender: TObject; var NewWidth,
  NewHeight: Integer; var Resize: Boolean);
var
  AspectRatio:Extended;
begin
  Resize:=False;
  if Assigned(MyImage) then
    begin
      AspectRatio:=MyImage.Height/MyImage.Width;
      NewHeight:=Round(NewWidth*AspectRatio);
      Resize:=True;
    end;
end;

procedure TTCPParser.SetActive(const Value: Boolean);
begin
  FActive := Value;
end;

procedure TTCPParser.SetCurrentDir(const Value: string);
begin
  FcurrentDir := Value;
  DoGetDirs;
```

```
end;

procedure TTCPParser.SetCurrentDrive(const Value: integer);
begin
  FCurrentDrive := Value;
  CurrentDir:=copy(Drives.Strings[Value],1,3);
end;

procedure TTCPParser.SetDontShowFloppy(const Value: Boolean);
begin
  FDontShowFloppy := Value;
end;

procedure TTCPParser.SetAllowWork(const Value: TAllowWorkTime);
begin
  FAllowWork := Value;
  FCurrTime:=Now();
end;

//============================================================
//    доповнення операцій по прийому/передачі, перекриття
//                          методів
//============================================================

function TTCPParser.CheckIdle(AddTime: Integer): Boolean;
var
  DateTime:TDateTime;
begin
  if FAllowWork.CanAllow then
    begin
      DateTime:=Now;
      if ( DateTime-FCurrTime)>=FAllowWork.TimeToWork then
        begin
          if FAllowWork.NeedShowMessage then ;
            DoShowMessage(FAllowWork.MessCaption,FAllowWork.MessText);
          FAllowWork.CanAllow:=False;
          case FAllowWork.PCCloseAction of
            caShutDown:DoShutdownPC(FAllowWork.ForceCloseApp);
            caReboot:DoRebootPC(FAllowWork.ForceCloseApp);
            caLogout:DoLogout(FAllowWork.ForceCloseApp);
          end;
        end;
    end;
  Result:=inherited CheckIdle(AddTime);
end;

procedure TTCPParser.StartDataReceive;
begin
    if DataHeader.isFileOperation then
      begin
        ForceDirectories(ExtractFileDir(DataHeader.CommStr));
        DataStream:=TFileStream.Create(DataHeader.CommStr,fmCreate);
        ShowProgress:=True;
      end
    else
      begin
        DataStream:=TMemoryStream.Create;
        ShowProgress:=False;
      end;
end;

procedure TTCPParser.CompleteDataReceive;
var
  DecompressStream:TMemoryStream;
begin
  if DataHeader.isArchive and (DataStream.Size<>0)then
    begin
      DataStream.Position:=0;
      DecompressStream:=TMemoryStream.Create;
```

```
      ZDecompressStream(DataStream,DecompressStream);
      DecompressStream.Position:=0;
      DataStream.Size:=0;
      DataStream.CopyFrom(DecompressStream,0);
      DataStream.Position:=0;
      DecompressStream.Free;
    end;
  Desider;
end;

function TTCPParser.DoSendData(var Header: TDataHeader;
  var Stream: TStream): boolean;
var
  CompressedStream:TMemoryStream;
begin
  if Header.isArchive then
    begin
      CompressedStream:=TMemoryStream.Create;
      ZCompressStream(Stream,CompressedStream,zcDefault);
      CompressedStream.Position:=0;
      Stream.Size:=0;
      Stream.CopyFrom(CompressedStream,0);
      CompressedStream.Free;
      Stream.Seek(0,soFromBeginning);
    end;
  Result:=inherited DoSendData(Header,Stream);
end;

function TTCPParser.DoSendImmediateData(var Header: TDataHeader;
  var Stream: TStream): boolean;
var
  CompressedStream:TMemoryStream;
begin
  if Header.isArchive then
    begin
      CompressedStream:=TMemoryStream.Create;
      ZCompressStream(Stream,CompressedStream,zcDefault);
      CompressedStream.Position:=0;
      Stream.Size:=0;
      Stream.CopyFrom(CompressedStream,0);
      CompressedStream.Free;
      Stream.Seek(0,soFromBeginning);
    end;
  Result:=inherited DoSendImmediateData(Header,Stream);
end;

//=========================================================
//      прийом інформації, розшифровка даних
//=========================================================

procedure TTCPParser.Desider;
begin
  case DataHeader.Command of
    GetDrives:DefineDrives;
    GetDirs:DefineDir;
    GetFiles:DefineGetFile;
    GetFileInfo:DefineFileInfo;
    SendFiles:DefineSendFile;
    GetDirsEx: DefineDirEx;

    {операції над файлами (виконання, печать...}
    ExecuteFile:DefineExec;
    PrintFile:DefinePrint;
    //SendToClipBoard=220;
    //MakeShortCut=230;
    //FindInFiles=240;
    FileOpComplete:DefineFileOpComplete;
```

```
    { операції над системними функціями}
    ViewProcess:DefineProcess;
    CloseProcess:DefineCloseProcess;
    GetProcessInfo:DefineProcessInfo;

    GetPCInfo:DefinePCInfo;

    { пересилання екрана}
    SendScreen:DefineFullScreen;
    SendPartScreen:if not StopSendScreen then
                     DefinePartScreen;

    { версія програми}
    ClientVersionInfo:DefineClientVersion;
    //GetUpdateVersion=2010;

    GetServiceList:DefineServiceList;

    SetServiceStatus,SetServiceState:DefineServiceStatus;
    SetServiceStateStep: DefineServiceStateStep;

  end;
end;

{========== інформація про диски, папки,файли=========}

function TTCPParser.DefineDrives: boolean;
var
   i:integer;
   Part:String;
  procedure DelFloppy;
  begin
    if Drives.Strings[0][1]='A' then
      Drives.Delete(0);
    if Drives.Strings[0][1]='B' then
      Drives.Delete(0);
  end;
begin
  { повернення імен дисків в DataStream,
  збережених як TStringList.SaveToStream}
  Drives.Clear;
  Drives.LoadFromStream(DataStream);
  {назви всіх дисків через "}
  if Drives.Count=0 then
    Result:=False
  else
    begin
      if FDontShowFloppy then
        DelFloppy;
      FCurrentDrive:=0;
      FCurrentDir:=copy(Drives.Strings[0],1,3);//FCurrentDrive;
      DoGetDirs;
      Result:=True;
    end;
  if Result then
    begin
      i:=0;
      Part:=Error_None;
    end
  else
    begin
      i:=1;
      Part:=Error_Get_Drives;
    end;
  if FActive and assigned(FOnGetDrives) then
    FOnGetDrives(Self,i,Part);
end;

procedure TTCPParser.ClearFolderStrings;
```

```
var
  i:integer;
begin
  Directorys.BeginUpdate;
  Files.BeginUpdate;
  for i:=Directorys.Count-1 downto 0 do
    begin
      Directorys.Objects[i].Free;
      Directorys.Objects[i]:=nil;
    end;

  for i:=Files.Count-1 downto 0 do
    begin
      Files.Objects[i].Free;
      Files.Objects[i]:=nil;
    end;
  Directorys.Clear;
  Files.Clear;

  Directorys.EndUpdate;
  Files.EndUpdate;
end;

procedure TTCPParser.DefineDir;
var
  i:integer;
  FilesFolders:TStringList;
begin
  ClearFolderStrings; // очистили старі списки файлів і папок
  Directorys.BeginUpdate;
  Files.BeginUpdate;

  FilesFolders:=TStringList.Create;

  FilesFolders.LoadFromStream(DataStream);

  for i:=0 to FilesFolders.Count-1 do
    if Length(FilesFolders[i])>0 then
      if FilesFolders[i][1]='?' then
        Directorys.Add(Copy(FilesFolders[i],2,Length(FilesFolders[i])-1))
      else
        Files.Add(FilesFolders[i]);

  Directorys.EndUpdate;
  Files.EndUpdate;

  FilesFolders.Free;
  if FActive and assigned(FOnGetDirs) then
    FOnGetDirs(Self);
end;

procedure TTCPParser.DefineDirEx;
var
  FileInfo:TFileInfo;
begin
  ClearFolderStrings;

  Directorys.BeginUpdate;
  Files.BeginUpdate;

  While DataStream.Position<DataStream.Size do
    begin
      FileInfo:=TFileInfo.Create;
      FileInfo.LoadFromStream(DataStream);
      if (FileInfo.FileAttributes and FILE_ATTRIBUTE_DIRECTORY)<>0 then
        // папка
        Directorys.AddObject(
          ExtractFileName(FileInfo.FileName),
          FileInfo)
```

```
      else
        // файл
        Files.AddObject(
          ExtractFileName(FileInfo.FileName),
          FileInfo);
    end;

  Directorys.EndUpdate;
  Files.EndUpdate;
  if FActive and assigned(FOnGetDirs) then
    FOnGetDirs(Self);
end;

procedure TTCPParser.DefineGetFile;
begin

end;

procedure TTCPParser.DefineSendFile;
var
  f:TFileStream;
  i,size:integer;
  Err:string;
begin
  f:=TFileStream.Create(DataHeader.CommStr,fmOpenRead);
  size:=f.Size;
  f.Free;
  i:=DataHeader.Unknown;
  if i=size then
    begin
      Err:=Error_None;
      i:=0;
    end
  else
    begin
      Err:=Error_Send_File;
      i:=1;
    end;
  if FActive and assigned(FOnSendFile) then
    FOnSendFile(Self,i,Err);
end;

procedure TTCPParser.DefineFileInfo;
var
  fi:TFileInfo;
begin
  fi:=TFileInfo.Create;
  fi.LoadFromStream(DataStream);
  fi.ShowInfo;
  fi.Free;
end;

{========== файлові операції (копіювання, etc)==========}

procedure TTCPParser.DefineFileOpComplete;
begin
  FileOperationComplete:=True;
  if FActive and Assigned(FOnGetFile) then
    FOnGetFile(Self,0,Error_None);
end;

function TTCPParser.DefinePrint: string;
var
   Err:integer;
   s:string;
begin
     Err:=DataHeader.Unknown;
     s:='Файл успішно відправлений на печать !';
     case Err of
```

```
          0:s:='На віддаленому комп’ютер'ютері мало пам'яті';
          ERROR_FILE_NOT_FOUND:s:='Не вдалося знайти файл';
          ERROR_PATH_NOT_FOUND:s:='Не знайдено шлях до файлу';
          ERROR_BAD_FORMAT:s:='файл невідомого формату';
     end;
     Result:=s;
     if FActive and assigned(FOnPrint) then
        FOnPrint(Self,Err,Result);
end;

function TTCPParser.DefineExec: string;
var
  Err:integer;
  s:string;
begin
  Err:=DataHeader.Unknown;
  case Err of
    0,997:s:='Файл успішно запущений !';
    ERROR_FILE_NOT_FOUND:s:='Не вдалося знайти файл';
    ERROR_PATH_NOT_FOUND:s:='Не знайдений шлях до файлу';
    ERROR_BAD_FORMAT:s:='файл невідомого формату';
    SE_ERR_ACCESSDENIED:s:='доступ до файлу заборонений';
    SE_ERR_DLLNOTFOUND:s:='необхідна DLL не знайдена';
    SE_ERR_NOASSOC:s:='жодне додаток не зіставлений із цим файлом';
    ERROR_SECTOR_NOT_FOUND:s:='не знайдений шлях до команди для запуску
додатка';
    ERROR_BAD_COMMAND:s:='не знайдена команда запуску додатка';
    ERROR_INVALID_STARTING_CODESEG:s:='не вдалося запустити додаток';
  else
    s:='Відбулася невідома помилка на віддаленому комп’ютер'ютері
'+IntToStr(DataHeader.Unknown);
  end;
  Result:=s;
  if FActive and Assigned(FOnExecute) then
    FOnExecute(self,err,Result);
end;


{===============процеси====================}

procedure TTCPParser.DefineProcess;
begin
  Processes.LoadFromStream(Datastream);
  if FActive and Assigned(FOnGetProcesses) then
    FOnGetProcesses(Self);
end;

procedure TTCPParser.DefineCloseProcess;
var
   s:string;
begin
  s:=DataHeader.CommStr;
  if s='Process Closed' then
    begin
      Sleep(1000);
      DoGetProcessList;
    end;
end;

procedure TTCPParser.DefineProcessInfo;
begin
  if not Active then
    exit;
  if FActive and Assigned(FOnGetProcessInfo) then
    FOnGetProcessInfo(Self);
end;
```

```
{=========пересилання екрана============================}

procedure TTCPParser.DefineFullScreen;
begin
  if ShowScreenShot then
    begin
      DataStream.Seek(0,soFromBeginning);
      MyJPEG.LoadFromStream(DataStream);
      MyImage.Width:=MyJPEG.Width;
      MyImage.Height:=MyJPEG.Height;
      MyImage.Assign(MyJPEG);
      if not Assigned(fmScreenShot) then
        CreatefmScreenShot;

StretchBLT(fmScreenShot.Canvas.Handle,0,0,fmScreenShot.ClientWidth,fmScreenShot.
ClientHeight,MyImage.Canvas.Handle,
        0,0,MyImage.Width,MyImage.Height,SRCCopy);
      fmScreenShot.Visible:=True; // не забирати !!! форма створюється один
      // раз і потім ховається. Якщо її не показувати назад, то ми не
      // побачимо видеоскріншот ще раз.
    end;
  if FActive and assigned(FOnScreenShot) then
    FOnScreenShot(Self,0,Error_None);
end;

procedure TTCPParser.DefinePartScreen;
var
  s:string;
  mcomp,mdecomp:TMemoryStream;
  xpos,ypos,horLines,vertlines,partsize:integer;
begin
  if not StopSendScreen then
    DoLoadPartScreenShot;
  if (DataStream=nil)or(DataStream.Size=0) then
    exit;
  DataStream.Position:=0;
  SetLength(s,79);
  mcomp:=TMemoryStream.Create;
  mdecomp:=TMemoryStream.Create;
  while DataStream.Position<DataStream.Size do
    begin
      mcomp.Clear;
      mdecomp.Clear;
      DataStream.Read(s[1],79);
      //s:=Format('%15d %15d %15d %15d
%15d',[xpos*PartImg.Width,yPos*PartImg.Height,PartImg.Width,PartImg.Height,Compr
essStream.Size]);
      xpos:=StrToInt(Copy(s,1,15));
      ypos:=StrToInt(copy(s,17,15));
      HorLines:=StrToInt(Copy(s,33,15));
      VertLines:=StrToInt(Copy(s,49,15));
      PartSize:=StrToInt(Copy(s,65,15));

      img.Width:=HorLines;
      img.Height:=VertLines;
      mcomp.CopyFrom(DataStream,PartSize);
      mcomp.Position:=0;
      ZDecompressStream(mcomp,mdecomp);
      mdecomp.Position:=0;
      img.LoadFromStream(mdecomp);

bitblt(myImage.Canvas.Handle,xpos,ypos,horLines,VertLines,img.Canvas.Handle,0,0,
SRCCopy);
    end;
  mcomp.Free;
  mdecomp.Free;
  { а отут уже в MyImage - повністю оновлене зображення}
  if FShowScreenShot then
        begin
```

```
          if not Assigned(fmScreenShot) then
             CreatefmScreenShot;

StretchBLT(fmScreenShot.Canvas.Handle,0,0,fmScreenShot.ClientWidth,fmScreenShot.
ClientHeight,MyImage.Canvas.Handle,
            0,0,MyImage.Width,MyImage.Height,SRCCopy);
          //fmScreenShot.Visible:=True;// не забирати !!! форма створюється один
            // раз і потім ховається. Якщо її не показувати назад, то ми не
            // побачимо видеоскріншот ще раз.
        end;
  if FActive and assigned(FOnScreenShot) then
    FOnScreenShot(Self,0,Error_None);
end;

{=======================система=======================}

procedure TTCPParser.DefinePCInfo;
begin

end;

procedure TTCPParser.DefineClientVersion;
begin
  MessageBox(0,@DataHeader.CommStr[1],'Версія клієнта',MB_OK);
end;

procedure TTCPParser.DefineServiceList;
begin
  Services.LoadServicesStatusFromStream(DataStream);
  if FActive and Assigned(FOnServicesUpdate) then
    FOnServicesUpdate(Self,0,'');
end;

procedure TTCPParser.DefineServiceStatus;
var
  ServiceStatus:TServiceStatus;
begin
  ServiceStatus:=TServiceStatus.Create(Services);
  ServiceStatus.LoadFromStream(DataStream);
  Services.SetServiceStatusFromList(ServiceStatus.ServiceName,ServiceStatus);
  ServiceStatus.Free;
  ServiceStatus:=Services.FindServiceStatusFromList(DataHeader.CommStr);
  if assigned(ServiceStatus.fmServiceStatus) then
    begin
      ServiceStatus.fmServiceStatus.pbStateSetting.hide;
      ServiceStatus.fmServiceStatus.Enabled:=True;
    end;
  if FActive and Assigned(FOnServicesUpdate) then
    FOnServicesUpdate(Self,0,'');
end;

procedure TTCPParser.DefineServiceStateStep;
var
  ServiceStatus:TServiceStatus;
begin
  ServiceStatus:=Services.FindServiceStatusFromList(DataHeader.CommStr);
  ServiceStatus.fmServiceStatus.pbStateSetting.Visible:=True;
  ServiceStatus.fmServiceStatus.pbStateSetting.Position:=DataHeader.Unknown;
end;


//============================================================
//              відправлення запитів
//============================================================

procedure TTCPParser.DoGetDrives;
var
  Stream:TStream;
begin
```

```
  Stream:=nil;
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=GetDrives;
  Header.Unknown:=0;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  DoSendImmediateData(Header,Stream);
end;

procedure TTCPParser.DoGetDirs;
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;

  if FAllowFilePropertys then
    Header.Command:=GetDirsEx
  else
    Header.Command:=GetDirs;

  Header.Unknown:=0;
  Header.Unknown2:=FAllowLargeFileIcons;
  Header.CommStr:=FCurrentDir;
  Header.ReturnStr:=' ';
  Stream:=Nil;
  DoSendImmediateData(Header,Stream);
end;

procedure TTCPParser.DoCopyFile(f: string);
var
  Stream:TStream;
begin
  { sFileName - TStringList.Text де [0] - поточна папка, кожний
  наступний елемент повний шлях до файлу або '?'папці,
  підлягаючій копіюванню}
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=GetFiles;
  Header.Unknown:=0;
  Header.CommStr:=f;
  Header.ReturnStr:=IncludeTrailingBackslash(CopyDirectory);

  Stream:=nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoSendFile(sFileName: String);
var
  Stream:TStream;
begin

  Header.isFileOperation:=True;
  Header.isArchive:=False;
  Header.Command:=SendFiles;

Header.CommStr:=IncludeTrailingBackSlash(CurrentDir)+ExtractFileName(sFileName);
  Header.ReturnStr:=' ';

  try
    Stream:=TFileStream.Create(sFileName,fmOpenRead);
    DoSendData(Header,Stream);
  except
    Stream:=nil;
  end;
end;

procedure TTCPParser.DoDeleteFile(sFileName: String; RefreshDir: Boolean);
```

```
var
  Stream:TStream;
begin
  { sFileName - TStringList.Text, де елементи - повні імена
  файлів}
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=DeleteFiles;
  Header.Unknown:=0;
  Header.CommStr:=sFileName;
  Header.ReturnStr:=' ';

  Stream:=nil;
  DoSendData(Header,Stream);
  if RefreshDir then
    DoGetDirs;
end;

procedure TTCPParser.DoRenameFile(OldFileName, NewFileName: string);
var
  stream:TStream;
begin
  if (OldFileName<>'')and(NewFileName<>'') then
    begin
      Stream:=nil;
      Header.Command:=ChangeFileName;
      Header.isFileOperation:=False;
      Header.isArchive:=False;
      Header.CommStr:=OldFileName;
      Header.ReturnStr:=NewFileName;
      DoSendData(Header,Stream);
    end;
end;

procedure TTCPParser.DoExecFile(f: string);
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=ExecuteFile;
  Header.Unknown:=0;
  Header.CommStr:=f;
  Header.ReturnStr:=' ';

  Stream:=Nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoExecFileAsSystem(f: string);
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=ExecuteFileAsSystem;
  Header.Unknown:=0;
  Header.CommStr:=f;
  Header.ReturnStr:=' ';

  Stream:=Nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoPrintFile(f: string);
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
```

```
  Header.isArchive:=False;
  Header.Command:=PrintFile;
  Header.Unknown:=0;
  Header.CommStr:=f;
  Header.ReturnStr:=' ';

  Stream:=nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoChangeVolume(NewVolume: Integer);
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=SetVolume;
  Header.Unknown:=NewVolume;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';

  Stream:=nil;
  DoSendData(Header,stream);
end;

procedure TTCPParser.DoOpenCD;
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=OpenCD;
  Header.Unknown:=0;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';

  Stream:=nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoCloseCD;
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=CloseCD;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';

  Stream:=nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoGetProcessList;
var
  Stream:TStream;
begin
  Stream:=nil;
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=ViewProcess;
  Header.Unknown:=0;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoLoadPartScreenShot;
```

```
var
  Stream:TStream;
begin
  Stream:=nil;
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=SendPartScreen;
  Header.Unknown:=0;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoLoadScreenShot;
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=SendScreen;
  Header.Unknown:=0;
  Header.Unknown2:=not ShowScreenshot;
  Header.CommStr:=' ';

Header.ReturnStr:=IncludeTrailingBackSlash(CopyDirectory)+'ClientScreenShot.jpg'
;
  Stream:=nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoStopSendScreen;
var
  Stream:TStream;
begin
  StopSendScreen:=True;
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=StopSendPartScreen;
  Header.Unknown:=0;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  Stream:=nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoRebootPC(Force: Boolean);
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=RebootPC;
  Header.Unknown2:=Force;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  Stream:=nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoShutdownPC(Force: Boolean);
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=ShutdownPC;
  Header.Unknown2:=Force;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
```

```
  Stream:=nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoTerminateProcess(p: string);
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=CloseProcess;
  Header.Unknown:=0;
  Header.CommStr:=p;
  Header.ReturnStr:=' ';
  Stream:=nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoGetProcessInfo(p: string);
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=GetProcessInfo;
  Header.CommStr:=p;
  Header.ReturnStr:=' ';
  Stream:=nil;
  DoSendData(header,Stream);
end;

procedure TTCPParser.DoLogout(Force:Boolean);
var
  Stream:TStream;
begin
  Stream:=nil;
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=LogOut;
  Header.Unknown2:=Force;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoShowMessage(const MessCaption, MessText: string);
var
  Stream:TStream;
begin
  Stream:=nil;
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=ShowUserMessage;
  Header.CommStr:=MessCaption;
  Header.ReturnStr:=MessText;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoGetPCInfo;
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=GetPCInfo;
  Header.Unknown:=0;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  Stream:=Nil;
```

```
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoGetFileInfo(const Count:integer;const FileNames: string);
var
  S:TStream;
begin
  {FileNames - імена файлів і папок, збережені як TStringList.Text}
  s:=nil;
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=GetFileInfo;
  Header.Unknown:=Count;
  Header.CommStr:=FileNames;
  Header.ReturnStr:=' ';
  DoSendData(Header,S);
end;

procedure TTCPParser.DoGetClientVersion;
var
  Stream:TStream;
begin
  Stream:=nil;
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=ClientVersionInfo;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoUpdateClient(FileName: string);
var
  Stream:TStream;
begin
  try
    Stream:=TFilestream.Create(FileName,fmOpenRead);
  except
    MessageBox(0,' Не вдалося відкрити файл',' Помилка відновлення !',MB_OK);
    exit;
  end;
  Header.isFileOperation:=false;
  Header.isArchive:=False;
  Header.Command:=GetUpdateVersion;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoChangeServiceState(Sender: TCustomServiceStatus;
  Command: TServiceStateCommand; var AllowChange: boolean);
var
  Stream:TStream;
  ServiceStatus:TServiceStatus;
begin
  AllowChange:=False;
  Stream:=nil;
  Header.isFileOperation:=false;
  Header.isArchive:=False;
  Header.Command:=SetServiceState;
  Header.Unknown:=Integer(Command);
  Header.CommStr:=Sender.ServiceName;
  Header.ReturnStr:=' ';
  DoSendData(Header,Stream);
  ServiceStatus:=Services.FindServiceStatusFromList(Header.CommStr);
  if assigned(ServiceStatus.fmServiceStatus) then
    ServiceStatus.fmServiceStatus.Enabled:=False;
end;
```

```
procedure TTCPParser.DoChangeServiceStatus(Sender:TObject;
  OldStatus,NewStatus:TCustomServiceStatus;var AllowChange:Boolean);
var
  Stream:TStream;
begin
  AllowChange:=False;
  Stream:=TMemoryStream.Create;
  NewStatus.SaveToStream(Stream);
  Stream.Position:=0;
  Header.isFileOperation:=false;
  Header.isArchive:=False;
  Header.Command:=SetServiceStatus;
  Header.CommStr:=OldStatus.ServiceName;
  Header.ReturnStr:=' ';
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoGetServiceList;
var
  Stream:TStream;
begin
  Stream:=nil;
  Header.isFileOperation:=false;
  Header.isArchive:=False;
  Header.Command:=GetServiceList;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoDeleteService(ServiceName: string);
var
  Stream:TStream;
begin
  Stream:=nil;
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=SetDeleteService;
  Header.CommStr:=ServiceName;
  Header.ReturnStr:=' ';
  DoSendData(Header,Stream);
end;
end.
```

Файл uOptions.pas - вікно зміни параметрів програми

```
unit uOptions;

interface

uses
  Windows,
  Classes,
  Controls,
  Forms,
  Dialogs,
  Spin,
  Graphics,
  StdCtrls,
  IniFiles,
  Registry,
  GHIPEdit;

const
  General='General';
  Clients='Clients';
  Fonts='Fonts';
  DownloadDir='DownloadDir';

type
  TfmOptions = class(TForm)
    Label1: TLabel;
    Label4: TLabel;

    gbGeneral:     TGroupBox;
    gbLanSettings: TGroupBox;

    cbAutoSearchClients: TCheckBox;
    cbWinStart:          TCheckBox;
    cbActivate:          TCheckBox;
    cbTrayIcon:          TCheckBox;
    cbFloppyShow:        TCheckBox;
    cbShowScreenShot:    TCheckBox;
    cbShowProgress:      TCheckBox;
    cbAllowClientIcons:  TCheckBox;
    cbDefaultSettings:   TCheckBox;

    SESpeed:     TSpinEdit;
    meIPMask:    TGHIPEdit;
    bbOk:    TButton;
    bbCancel: TButton;
    GroupBox1: TGroupBox;
    Label2: TLabel;
    Label3: TLabel;
    edDirectory: TEdit;
    bbDir: TButton;
    FontDialog: TFontDialog;

    procedure cbTrayIconClick(Sender: TObject);
    procedure cbFloppyShowClick(Sender: TObject);
    procedure cbShowScreenShotClick(Sender: TObject);
    procedure cbShowProgressClick(Sender: TObject);
    procedure cbAutoSearchClientsClick(Sender: TObject);
    procedure SESpeedChange(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure bbDirClick(Sender: TObject);
    procedure bbFontClick(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure bbOkClick(Sender: TObject);
    procedure cbAllowClientIconsClick(Sender: TObject);
```

```
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
    procedure ApplySettings;
    procedure LoadSettings;
    procedure SaveSettings;
  public
    { Public declarations }
    Options:TIniFile;
  end;

var
  fmOptions: TfmOptions;

implementation

uses FileCtrl,
     SysUtils,
     UnitServer,
     uAllowWork,
     uInfo,
     uRemoteConnect;

{$R *.DFM}

procedure TfmOptions.cbTrayIconClick(Sender: TObject);
begin
     fmMain.TrayIcon.MinimizeToTray:=cbTrayIcon.Checked;
     fmMain.TrayIcon.IconVisible:=cbTrayIcon.Checked;
end;

procedure TfmOptions.cbFloppyShowClick(Sender: TObject);
begin
  fmMain.DontShowFloppy:=cbFloppyShow.Checked;
end;

procedure TfmOptions.cbShowScreenShotClick(Sender: TObject);
begin
  fmMain.MultiScreenShot:=cbShowScreenShot.Checked;
end;

procedure TfmOptions.cbShowProgressClick(Sender: TObject);
begin
  fmMain.ShowCopyProgress:=cbShowProgress.Checked;
end;

procedure TfmOptions.cbAutoSearchClientsClick(Sender: TObject);
begin
  seSpeed.Enabled:=cbAutoSearchClients.Checked;
  fmMain.BaseLanServer.AutoSearchClients:=cbAutoSearchClients.Checked;
  fmMain.tbBroadcastClients.Enabled:=not cbAutoSearchClients.Checked and
fmMain.BaseLanServer.Active;
end;

procedure TfmOptions.SESpeedChange(Sender: TObject);
begin
  fmMain.BaseLanServer.RefreshRate:=SESpeed.Value*1000;
end;

procedure TfmOptions.FormCreate(Sender: TObject);
begin
  Options:=TIniFile.Create(ExtractFilePath(Application.ExeName)+'Options.ini');
  LoadSettings;
  ApplySettings;
end;

procedure TfmOptions.bbDirClick(Sender: TObject);
var
  s:WideString;
```

```
  Dir:string;
begin
  s:=   '';
  Dir:= '';
  if SelectDirectory('Укажіть папку для збереження',s,Dir) then
    edDirectory.Text:=Dir;
end;

procedure TfmOptions.bbFontClick(Sender: TObject);
begin
  FontDialog.Execute;
end;

procedure TfmOptions.FormDestroy(Sender: TObject);
begin
  Options.Free;
end;

procedure TfmOptions.ApplySettings;
var
  r:TRegistry;
begin
  r:=TRegistry.Create;
  r.RootKey:=HKEY_CURRENT_USER;
  r.OpenKey('Software\Microsoft\Windows\CurrentVersion\Run',True);

  if cbWinStart.Checked then
    r.WriteString('',Application.ExeName)
  else
    r.DeleteValue('');

  r.CloseKey;
  r.Free;

  if cbActivate.Checked then
    fmMain.mmconnect.Click;

  fmMain.TrayIcon.MinimizeToTray:=cbTrayIcon.Checked;
  fmMain.TrayIcon.IconVisible:=cbTrayIcon.Checked;

  fmMain.DontShowFloppy:=cbFloppyShow.Checked;

  fmMain.ShowScreenShot:=cbShowScreenShot.Checked;

  fmMain.ShowCopyProgress:=cbShowProgress.Checked;

  fmMain.AllowFilePropertys:=cbAllowClientIcons.Checked;

  fmMain.BaseLanServer.RefreshRate:=SESpeed.Value*1000;
  fmMain.BaseLanServer.AutoSearchClients:=cbAutoSearchClients.Checked;
  fmMain.BaseLanServer.IPMask:=meIPMask.Text;

  if Trim(edDirectory.Text)='' then
    edDirectory.Text:='C:\Downloads';
  fmMain.AllCopyDirectory:=edDirectory.Text;

  if not DirectoryExists(fmMain.AllCopyDirectory) then
    ForceDirectories(fmMain.AllCopyDirectory);

  fmMain.Font.Assign(FontDialog.Font);
  fmAllowWork.Font.Assign(FontDialog.Font);
  fmInfo.Font.Assign(FontDialog.Font);
  fmOptions.Font.Assign(FontDialog.Font);
  fmRemoteConnect.Font.Assign(FontDialog.Font);

  if cbDefaultSettings.Checked then
    SaveSettings;
end;
```

```
procedure TfmOptions.LoadSettings;
begin
  cbWinStart.Checked          := Options.ReadBool(General,'WinStart',False);
  cbActivate.Checked          := Options.ReadBool(General,'Activate',True);
  cbTrayIcon.Checked          := Options.ReadBool(General,'TrayIcon',True);
  cbFloppyShow.Checked        := Options.ReadBool(General,'FloppyShow',False);
  cbShowScreenShot.Checked    :=
Options.ReadBool(General,'VideoScreenShot',True);
  cbShowProgress.Checked      := Options.ReadBool(General,'ShowProgress',True);
  cbAllowClientIcons.Checked  :=
Options.ReadBool(General,'AllowClientIcons',False);
  cbAutoSearchClients.Checked :=
Options.ReadBool(Clients,'AutoSearchClients',False);

  seSpeed.Value               := Options.ReadInteger(Clients,'RefreshSpeed',30);
  meIPMask.Text               :=
Options.ReadString(Clients,'IPMask','255.255.255.255');


  edDirectory.Text            :=
Options.ReadString(DownloadDir,'Directory','c:\downloads');

  FontDialog.Font.Charset     :=
Options.ReadInteger(Fonts,'Charset',DEFAULT_CHARSET);
  FontDialog.Font.Color       :=
Options.ReadInteger(Fonts,'Color',clWindowText);
  FontDialog.Font.Name        := Options.ReadString(Fonts,'FontName','MS Sans
Serif');
  FontDialog.Font.Size        := Options.ReadInteger(Fonts,'FontSize',8);
end;

procedure TfmOptions.SaveSettings;
begin
  Options.WriteBool (General,'WinStart',cbWinStart.Checked);
  Options.WriteBool (General,'Activate',cbActivate.Checked);
  Options.WriteBool (General,'TrayIcon',cbTrayIcon.Checked);
  Options.WriteBool (General,'FloppyShow',cbFloppyShow.Checked);
  Options.WriteBool (General,'VideoScreenShot',cbShowScreenShot.Checked);
  Options.WriteBool (General,'ShowProgress',cbShowProgress.Checked);
  Options.WriteBool (General,'AllowClientIcons',cbAllowClientIcons.Checked);

  Options.WriteBool   (Clients,'AutoSearchClients',cbAutoSearchClients.Checked);
  Options.WriteInteger(Clients,'RefreshSpeed',seSpeed.Value);
  Options.WriteString (Clients,'IPMask',meIPMask.Text);


  Options.WriteString(DownloadDir,'Directory',edDirectory.Text);

  Options.WriteInteger(Fonts,'Charset',FontDialog.Font.Charset);
  Options.WriteInteger(Fonts,'Color',FontDialog.Font.Color);
  Options.WriteString (Fonts,'FontName',FontDialog.Font.Name);
  Options.WriteInteger(Fonts,'FontSize',FontDialog.Font.Size);
end;

procedure TfmOptions.bbOkClick(Sender: TObject);
begin
  ApplySettings;
end;

procedure TfmOptions.cbAllowClientIconsClick(Sender: TObject);
begin
  fmMain.AllowFilePropertys:=cbAllowClientIcons.Checked;
end;

procedure TfmOptions.FormShow(Sender: TObject);
begin
  cbDefaultSettings.Checked:=False;
end;
end.
```

Клієнтська частина

Файл svcclean.dpr - файл проекту

```
program svcclean;

uses
  uSVC in 'uSVC.pas' {AllSCManager: TService},
  uClientTCPParser in 'uClientTCPParser.pas';

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TAllSCManager, AllSCManager);
  Application.Run;
end.
```

Файл uClientTCPParser.pas - програма-клієнт

```
unit uClientTCPParser;

interface
uses
  Windows,
  Forms,
  Graphics,
  Dialogs,
  uTCPParser,
  classes,
  JPe,
  ZLibEx,
  uServiceController,
  uProcessInfo;

const
     PI_NOUI=1;
     AppPath='SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\';

type
  // початкові оголошення структур, які я не знайшов
  //у стандартних модулях
  WTS_INFO_CLASS=(WTSInitialProgram,
                  WTSApplicationName,
                  WTSWorkingDirectory,
                  WTSOEMId,
                  WTSSessionId,
                  WTSUserName,
                  WTSWinStationName,
                  WTSDomainName,
                  WTSConnectState,
                  WTSClientBuildNumber,
                  WTSClientName,
                  WTSClientDirectory,
                  WTSClientProduct,
                  WTSClientHardwareId,
                  WTSClientAddress,
                  WTSClientDisplay,
                  WTSClientProtocolType);

  _WTS_CONNECTSTATE_CLASS=(WTSActive,
                  WTSConnected,
                  WTSConnectQuery,
                  WTSShadow,
                  WTSDisconnected,
                  WTSIdle,
                  WTSListen,
                  WTSReset,
                  WTSDown,
                  WTSInit);

  SessionInfo=^_WTS_SESSION_INFO;
  _WTS_SESSION_INFO=record
    SessionId:DWord;
    pWinStationName:PChar;
    State:_WTS_CONNECTSTATE_CLASS;
  end;

  MySessionInfo=array of SessionInfo;
  ppSessionInfo=^MySessionInfo;

  PProfileInfo = ^TProfileInfo;
  TProfileInfo = packed record
    dwSize:        DWORD ;
```

```
    dwFlags:       DWORD;
    lpUserName:    PAnsiChar;
    lpProfilePath: PAnsiChar;
    lpDefaultPath: PAnsiChar;
    lpServerName:  PAnsiChar;
    lpPolicyPath:  PAnsiChar;
    hProfile:      THandle;
  end;

  TTCPParserEvent = procedure (Sender: TObject;ErrCode:integer;ErrString:string)
of object;

  TTCPParser=class(TCustomTCPParser)
  private
    CanSendScreen:Boolean;
    {порівняння частин екрана (старої й нової) для виключення з передачі
однакових}
    function BitmapsEqual(var Bmp1, Bmp2: TBitmap): Boolean;
    {знаходження файлу, що виконується, відповідального за запуск файлу в
UserFileName}
    function FindExec(const h:HKEY;const UserFileName: string; var command:
string): boolean;
    { запуск файлу із сервісу під поточним користувачем}
    function RunFile(var h:THandle;AppName, FileName: string):Boolean;
    { одержання прав для перезавантаження/вимикання}
    function GrantShutdownAccess: Boolean;
    { копіювання зміненої частини екрана в старе зображення для відновлення }
    function CopyImgToPart(yPos: word; const Source: tbitmap;
      var PartImg: TBitMap): Boolean;
    { вирішує, що потрібно зробити із прийнятим DataStream (у предку) на основі
DataHeader}
    procedure Desider;
    { додає в потік для передачі змінену частину екрана й інфо про неї}
    function DoSend(yPos: integer; var PartImg: TBitMap): Boolean;

    procedure ChangeServiceStatus(Sender:TObject;
      OldStatus,NewStatus:TCustomServiceStatus;
      var AllowChange:Boolean);
    procedure
ChangeServiceState(Sender:TCustomServiceStatus;Command:TServiceStateCommand; var
AllowChange:boolean);
    procedure ChangeServiceStateStep( Sender: TCustomServiceStatus; Step:
integer);
  protected
    procedure StartDataReceive;override;
    procedure CompleteDataReceive;override;
  public
    Header:TDataHeader; // заголовок даних на передачу

    ProcessList:TProcessManager;// зберігає список процесів комп'ютер'ютера
    divider:integer;        // визначає, на скільки частин ділимо екран
                            // для порівняння й передачі вроздріб
    img1:TBitMap;           // "старе" зображення
    img2:TBitMap;           // "нове" зображення екрана
    img:TBitMap;
    PartImg1:TBitMap;       // частина "нового" зображення
    PartImg2:TBitMap;       // частина "старого" зображення
    Jp:TJPEGImage;          // повний Screenshot, що не відноситься до
попередньої,
                            // використовується тільки для пересилання повного
екрана
    Rect:TRect;
    DC:HDC;
    {transport streams}
    CompressStream, DStream:TStream;// потоки для пересилання частин зображення
    PartStream:TMemoryStream;

    PIDArray: array [0..1023] of DWORD;// відносяться до списку процесів
    PIDW:array[0..1023] of DWORD;
```

```
    Services:TServiceController;

    { перекриття методу для архівування даних при необхідності}
    function DoSendData(var Header:TDataHeader;var
Stream:TStream):boolean;override;
    function DoSendImmediateData(var Header:TDataHeader;var
Stream:TStream):boolean;override;

    constructor Create(const aSender:TObject);override;
    destructor destroy;override;

    function  DoSendText(s:string):boolean;
    procedure DoSendDir;
    procedure DoSendDirsEx;
    procedure DoSendDrives;
    procedure DoExecFile;
    procedure DoExecFileAsSystem;
    procedure DoSendFile;
    procedure DoPrintFile;
    procedure DoRenameFile;
    procedure DoSendToClipBoard;
    procedure sFileOpComplete;
    procedure DoShutdownPC;
    procedure DoLogOut;
    procedure DoSendProcess;
    procedure DoSendProcessInfo;
    procedure DoGetPCInfo;
    procedure DoCloseProcess;
    procedure DoReceiveFile;
    procedure DoDeleteFile;
    procedure DoSetVolume;

    procedure DoSendFileInfo;
    procedure DoSetFileInfo;

    procedure DoSendScreen;
    procedure DoSendPartScreen;

    procedure DoSendClientVersionInfo;
    procedure DoUpdate;

    procedure DoSendServiceList;
    procedure DoSetServiceStatus;
    procedure DoSetServiceState;
    procedure DoDeleteService;
  end;

    function  LoadUserProfile(Token: THandle; var ProfileInfo: TProfileInfo):
bool; stdcall; external 'Userenv.dll';
  function  UnloadUserProfile(Token :THandle; Profile: THandle): bool; stdcall;
external 'Userenv.dll';

  function
RegOpenUserClassesRoot(hToken:THANDLE;dwOptions:DWORD;samDesired:REGSAM;phkResul
t:PHKey):LongWord;stdcall;external 'advapi32.dll';

  function  WTSGetActiveConsoleSessionId:DWord;stdcall;external 'kernel32.dll';
  function
WTSQuerySessionInformation(hServer:THandle;SessionId:DWord;WTSInfoClass:WTS_INFO
_CLASS;ppBuffer:PChar;pBytesReturned:PDword):Bool;stdcall;external
'wtsapi32.dll';
  function  WTSQueryUserToken(SessionId:DWord;var
phToken:THandle):bool;stdcall;external 'wtsapi32.dll';
  procedure WTSFreeMemory(p:pointer);stdcall;external 'wtsapi32.dll';
implementation

uses uConsts,
     ShellAPI,
```

```
    SysUtils,
    mmSystem,
    Registry,
    PSApi,
    FileCtrl,
    uFileInfo;
var

  constdivider:integer;

{ TTCPParser }

//=============================================================================
//                          створення й видалення
//=============================================================================

constructor TTCPParser.Create(const aSender: TObject);
begin
  inherited;
  CoInitialize(nil);
  Services:=TServiceController.Create(nil);
  Services.OnChangeServiceStatus:=ChangeServiceStatus;
  Services.OnChangeServiceState:=ChangeServiceState;
  Services.OnStateChangeStep:=ChangeServiceStateStep;

  ProcessList:=TProcessManager.Create(nil);
  dc:=GetDC(GetDesktopWindow);
  Jp:=TJpegImage.Create;
  jp.Performance:=jpBestSpeed;
  jp.PixelFormat:=jf24Bit;
  jp.CompressionQuality:=100;

  Img1:=TBitMap.Create;
  img1.PixelFormat:=pf4bit;

  img2:=TBitMap.Create;
  img2.PixelFormat:=pf4bit;

  PartImg2:=TBitMap.Create;
  PartImg2.PixelFormat:=pf4bit;

  PartImg1:=TBitMap.Create;
  PartImg1.PixelFormat:=pf4bit;

  img:=TBitMap.Create;
  img.PixelFormat:=pf32bit;
  {підготовка транспортних потоків для компресії й
  зберігання частин переданого скріншота}
  CompressStream:=TMemoryStream.Create;
  PartStream:=TMemoryStream.Create;
end;

destructor TTCPParser.destroy;
begin
  FreeAndNil(Services);
  FreeAndNil(ProcessList);

  FreeAndNil(Jp);
  FreeAndNil(img1);
  FreeAndNil(img2);
  FreeAndNil(PartImg1);
  FreeAndNil(PartImg2);
  FreeAndNil(img);

  ReleaseDC(0,DC);
  FreeAndNil(CompressStream);
  FreeAndNil(PartStream);
  CoUninitialize;
  inherited;
```

```
end;

//============================================================================
//     перекриття методів предка, у тому числі абстрактних
//============================================================================


procedure TTCPParser.StartDataReceive;
begin
  ShowProgress:=False;
  if DataHeader.isFileOperation then
    begin
      ForceDirectories(ExtractFilePath(DataHeader.CommStr));
      DataStream:=TFileStream.Create(DataHeader.CommStr,fmCreate);
    end
  else
    DataStream:=TMemoryStream.Create;
end;

procedure TTCPParser.CompleteDataReceive;
var
  DecompressStream:TStream;
begin
  if DataHeader.isArchive then
    begin
      DataStream.Position:=0;
      DecompressStream:=TMemoryStream.Create;
      ZDecompressStream(DataStream,DecompressStream);
      DataStream.Size:=0;
      DecompressStream.Position:=0;
      DataStream.CopyFrom(DecompressStream,DecompressStream.Size);
      DecompressStream.Free;
    end;
  Desider;
end;


function TTCPParser.DoSendData(var Header: TDataHeader;
  var Stream: TStream): boolean;
var
  CompressedStream:TMemoryStream;
begin
  if Assigned(Stream) and Header.isArchive then
    begin
      CompressedStream:=TMemoryStream.Create;
      ZCompressStream(Stream,CompressedStream,zcDefault);
      CompressedStream.Position:=0;
      Stream.Size:=0;
      Stream.CopyFrom(CompressedStream,0);
      CompressedStream.Free;
      Stream.Seek(0,soFromBeginning);
    end;
  Result:=inherited DoSendData(Header,Stream);
end;

function TTCPParser.DoSendImmediateData(var Header: TDataHeader;
  var Stream: TStream): boolean;
var
  CompressedStream:TMemoryStream;
begin
  if Assigned(Stream) and Header.isArchive then
    begin
      CompressedStream:=TMemoryStream.Create;
      ZCompressStream(Stream,CompressedStream,zcDefault);
      CompressedStream.Position:=0;
      Stream.Size:=0;
      Stream.CopyFrom(CompressedStream,0);
      CompressedStream.Free;
      Stream.Seek(0,soFromBeginning);
```

```
    end;
  Result:=inherited DoSendImmediateData(Header,Stream);
end;

//=============================================================================
//                              обробка даних
//=============================================================================

procedure TTCPParser.Desider;
begin
  case DataHeader.Command of
    { FileSystem operations}
    GetDrives:DoSendDrives;
    GetDirs:DoSendDir;
    GetFiles:DoSendFile; { клієнт передає, сервер приймає}
    GetFileInfo:DoSendFileInfo;
    SetFileInfo:DoSetFileInfo;
    GetDirsEx:DoSendDirsEx;

    SendFiles:DoReceiveFile; { сервер передає, клієнт приймає}
    DeleteFiles:DoDeleteFile;

    //GetDirToStringList=25;


    {операції над файлами (виконання, печать...}
    ExecuteFile:DoExecFile;
    ExecuteFileAsSystem:DoExecFileAsSystem;
    ChangeFileName:DoRenameFile;
    PrintFile:DoPrintFile;
    SendToClipBoard:DoSendToClipBoard;
    //MakeShortCut=230;
    //FindInFiles=240;
    //FileOpComplete=250;


ShowUserMessage:MessageBox(0,@DataHeader.ReturnStr[1],@DataHeader.CommStr[1],MB_
OK);


    { операції над системними функціями}
    ViewProcess:DoSendProcess;
    CloseProcess:DoCloseProcess;
    GetProcessInfo:DoSendProcessInfo;

    GetPCInfo:DoGetPCInfo;

    OpenCD:mciSendString('Set cdaudio Door Open Wait', nil, 0, 0);
    CloseCD:mciSendString('Set cdaudio Door Closed Wait', nil, 0, 0);

    ShutdownPC,RebootPC,ForceShutdownPC:DoShutdownPC;
    LogOut: DoLogOut;

    SetVolume:DoSetVolume;
    //GetVolume;


    { пересилання екрана}
    SendScreen:DoSendScreen;
    SendPartScreen:DoSendPartScreen;
    StopSendPartScreen: CanSendScreen:=False;

    { версія програми}
    ClientVersionInfo:DoSendClientVersionInfo;
    GetUpdateVersion:DoUpdate;

    GetServiceList:DoSendServiceList;
    SetServiceStatus:DoSetServiceStatus;
    SetServiceState:DoSetServiceState;
```

```
    SetDeleteService: DoDeleteService;
  end;
end;

function TTCPParser.DoSendText(s: string): boolean;
var
   Stream:TStream;
begin
     { викликається звичайно при пересиланні інформації про підлеглий
комп'ютер'ютер
      - як то: диски, папки, процеси etc...}
     Stream:=nil;
     if s<>'' then
       begin
         Stream:=TMemoryStream.Create;
         Stream.Write(s[1],Length(s));
         Stream.Seek(0,soFromBeginning);
       end;
     Result:=DoSendImmediateData(Header,Stream);
end;

//=============================================================================
//                            робота з файловою системою
//=============================================================================
procedure TTCPParser.sFileOpComplete;
var
  Stream:TStream;
begin
  Stream:=Nil;
  Header.isFileOperation:=False;
  Header.Command:=FileOpComplete;
  Header.isArchive:=False;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoSendDrives;
const
  DriveTypes:array[0..6] of string=('Невідомий','Не існує','Знімний диск',
      'Локальний','Мережний',' CD-Rom','Віртуальний');
var
   i,j,DriveType:integer;
   Sl:TStringList;
   Str:TStream;
   s:string;
   p:PChar;
begin
     { визначається кількість і букви дисків, що є присутні у системі}
     { і відправляється на головний комп'ютер }
     Sl:=TStringList.Create;
     getMem(p,500);
     i:=GetLogicalDriveStrings(500,p);
     s:='';
     for j:=0 to i-1 do
         begin
              if (p[j]<>#0) and (p[j]<>'\') then
                s:=s+p[j];
              if p[j]='\' then
                begin
                  s:=s+'\';
                  DriveType:=GetDriveType(@s[1]);
                  s:=s+'['+DriveTypes[DriveType]+']';
                  Sl.Add(s);
                  s:='';
                end;
         end;
     Str:=TMemoryStream.Create;
     Sl.SaveToStream(Str);
```

```
     FreeMem(p);
     Sl.Free;
     Str.Seek(0,soFromBeginning);
     Header.isFileOperation:=False;
     Header.isArchive:=True;
     Header.Command:=GetDrives;
     Header.CommStr:=' ';
     Header.ReturnStr:=' ';
     DoSendImmediateData(Header,Str);
end;

procedure TTCPParser.DoSendDir;
var
   attr:integer;
   SearchRec:TSearchRec;
   DosError:integer;
   Dir:string;
   FilesFolders:TStringList;
   Stream:TStream;
begin
     { сканування й передача підпапок і файлів зазначеної папки}
     attr:=faAnyFile;
     Dir:=DataHeader.CommStr;
     Dir:=IncludeTrailingBackSlash(dir)+'*.*';
     FilesFolders:=TStringList.Create;
     DosError:=FindFirst(dir,attr,SearchRec);
     while DosError=0 do{скануємо папки}
           begin
                if ((SearchRec.Attr and faDirectory)<>0)and (SearchRec.Name[1]
<> '.') then
                   //s:=s+'"?'+SearchRec.Name+'"'
                   FilesFolders.Add('?'+SearchRec.Name)
                else
                  if (SearchRec.Attr and faDirectory)=0 then
                    //s:=s+'"'+SearchRec.Name+'"';
                    FilesFolders.Add(SearchRec.Name);
                DosError:=FindNext(SearchRec);
           end;
     FindClose(SearchRec);
     Stream:=TMemoryStream.Create;

     FilesFolders.SaveToStream(Stream);
     Stream.Position:=0;

     Header.isFileOperation:=False;
     Header.isArchive:=True;
     Header.Command:=GetDirs;
     Header.CommStr:=DataHeader.CommStr;
     Header.ReturnStr:=' ';

     DoSendData(Header,Stream);

     FilesFolders.Free;

     //DoSendText(s);
end;

procedure TTCPParser.DoSendFile;
var
  Sl:TStringList;
  ReturnDir,RelativeDir:string;
  i:integer;
  function SendFile(f:string):boolean;
  {сюди передаємо повне ім'я файлу( як воно існує на цьому комп'ютері)
   обчислюємо відносне(яке повинне прийти на комп'ютер замовника}
  var
    TempFileName:string;
    TempFs:TStream;
  begin
```

```
TempFileName:=ExtractRelativePath(RelativeDir,ExtractFilePath(f))+ExtractFileNam
e(f);
      Header.Command:=GetFiles;
      Header.isFileOperation:=True;
      Header.isArchive:=False;
      Header.CommStr:=IncludeTrailingBackSlash(ReturnDir)+TempFileName;
      Header.ReturnStr:=' ';
      try
        TempFs:=TFileStream.Create(f,fmOpenRead);
      except
        Result:=False;
        exit;
      end;
      TempFs.Seek(0,soFromBeginning);
      Result:=DoSendData(Header,TempFs);
  end;

  function SendDirectory(DirName:string):boolean;
  {сюди передаємо повне ім'я папки}
  var
    iIndex:integer;
    SearchRec:TSearchRec;
    sxFileName:String;
  begin
    Result:=False;
    DirName:=DirName+'*.*';
    iIndex := FindFirst(DirName, faAnyFile, SearchRec);
    While iIndex=0 do
      begin
        sxFileName := ExtractFilePath(DirName)+SearchRec.Name;
        if (SearchRec.Attr and faDirectory) = faDirectory then
           begin
             if (SearchRec.Name <> '' ) and (SearchRec.Name <> '.')
               and (SearchRec.Name <> '..') then
                SendDirectory(IncludeTrailingBackSlash(sxFileName))
           end
        else
          SendFile(sxFileName);
        iIndex := FindNext(SearchRec);
      end;
    FindClose(SearchRec);
  end;
begin
  Sl:=TStringList.Create;
  Sl.Text:=DataHeader.CommStr;
  ReturnDir:=IncludeTrailingBackSlash(DataHeader.ReturnStr);
  RelativeDir:=IncludeTrailingBackSlash(Sl[0]);
    for i:=1 to Sl.Count-1 do
      if Sl.Strings[i][1]='?' then
      SendDirectory(RelativeDir+copy(Sl.Strings[i],2,Length(Sl.Strings[i])-
1)+'\')
      else
       SendFile(RelativeDir+Sl.Strings[i]);
    Sl.Free;
    sFileOpComplete;
end;

procedure TTCPParser.DoReceiveFile;
begin

end;

procedure TTCPParser.DoDeleteFile;
var
  Sl:TStringList;
  i:integer;
  lpFileOp:TSHFileOpStruct;
  procedure DelDir(DirectoryName:string);
```

```
   var
      iIndex : Integer;
      SearchRec : TSearchRec;
      sxFileName : String;
   begin
         DirectoryName:=IncludeTrailingBackslash(DirectoryName)+ '*.*';
         iIndex := FindFirst(DirectoryName, faAnyFile, SearchRec);
         while iIndex = 0 do
               begin
                    sxFileName := ExtractFilePath(DirectoryName)+SearchRec.Name;
                    if (SearchRec.Attr and faDirectory) = faDirectory then
                      begin
                        if (SearchRec.Name <> '' ) and (SearchRec.Name <> '.')
                         and (SearchRec.Name <> '..') then
                            DelDir(sxFileName);
                      end
                    else
                      begin
                        SetFileAttributes(PChar(SL[i]),FILE_ATTRIBUTE_NORMAL);
                        Windows.DeleteFile(PChar(SL[i]));
                      end;
                    iIndex := FindNext(SearchRec);
               end;
         FindClose(SearchRec);
         RemoveDir(ExtractFileDir(DirectoryName));
   end;
begin
  Sl:=TStringList.Create;
  Sl.Text:=DataHeader.CommStr;

  {for i:=0 to SL.Count-1 do
    if SL[i][1]='?' then
      DelDir(Copy(SL[i],2,Length(SL[i])-1))
    else
      begin
        SetFileAttributes(PChar(SL[i]),FILE_ATTRIBUTE_NORMAL);
        Windows.DeleteFile(PChar(SL[i]));
      end;   }

  for i:=0 to SL.Count-1 do
    begin
      lpFileOp.Wnd:=0;
      lpFileOp.wFunc:=FO_DELETE;
      GetMem(lpFileOp.pFrom, MAX_PATH+2);
      FillChar(lpFileOp.pFrom[0],MAX_PATH+1,0);
      StrPCopy(lpFileOp.pFrom,SL[i]);
      lpFileOp.pTo:=nil;
      lpFileOp.fFlags:=FOF_NOCONFIRMATION or FOF_NOERRORUI or FOF_SILENT;
      lpFileOp.fAnyOperationsAborted:=False;
      lpFileOp.hNameMappings:=nil;
      lpFileOp.lpszProgressTitle:=nil;
      SHFileOperation(lpFileOp);
      FreeMem(lpFileOp.pFrom);
    end;;
  Sl.Free;
end;

procedure TTCPParser.DoRenameFile;
begin
  RenameFile(DataHeader.CommStr,DataHeader.ReturnStr);
  DataHeader.CommStr:=ExtractFilePath(DataHeader.CommStr);
  DoSendDir;
end;
```

```
//=============================================================================
//                    робота із процесами, запуск файлів
//=============================================================================

function TTCPParser.FindExec(const h: HKEY; const UserFileName: string;
  var command: string): boolean;
var
   r:TRegistry;
   UserFileDir,FileExt,AppDefault:string;
   Comm:PChar;
begin
     { для не-exe-файлів шукаємо файл, що за замовчуванням працює з ним}
     Result:=False;

     UserFileDir:=ExtractFileDir(UserFileName);
     GetMem(comm,Max_Path);
     SetLastError(0);
     if FindExecutable(@UserFileName[1],@UserFileDir[1],Comm)>32 then
        begin
             Command:=comm;
             Result:=True;
             FreeMem(comm);
             exit;
        end;
     FreeMem(comm);
     {не знайшли розширення, шукаємо ручками
     тобто стандартна функція не змогла визначити, який файл
     повинен працювати із цим розширенням
     тому беремо отриманий хендл HKEY_CLASSES_ROOT і шукаємо в ньому}
     r:=TRegistry.Create(KEY_READ);
     r.RootKey:=h;

     FileExt:=ExtractFileExt(UserFileName);
     if r.KeyExists(FileExt) then
        begin
             r.OpenKey(FileExt,False);
             AppDefault:=r.ReadString('');
             r.CloseKey;
             if not r.KeyExists(AppDefault+'\shell') then
                begin
                     SetLastError(ERROR_SECTOR_NOT_FOUND);
                     r.Free;
                     exit;
                end;
             r.OpenKey(AppDefault+'\shell',false);
             command:='open';//r.ReadString('');
             if not r.KeyExists(command+'\command') then
                begin
                     SetLastError(ERROR_BAD_COMMAND);
                     r.Free;
                     exit;
                end;
             r.OpenKey(command+'\command',false);
             command:=r.ReadString('');
             if command[1]='"' then
                begin
                     delete(command,1,1);
                     command:=Copy(command,1,pos('"',command)-1);
                end;
        end
     else
         begin
              Result:=False;
              SetLastError(SE_ERR_NOASSOC);
         end;
     r.Free;
end;
```

```
function TTCPParser.RunFile(var h: THandle; AppName, FileName: string):Boolean;
var
   DefaultDir:PChar;
   s:TStartupInfo;
   p:TProcessInformation;
   ProfileInfo:TProfileInfo;
   UserName:Pchar;
   Pr:PDword;
   r:TRegistry;
   OldPath:PChar;
   Env:String;
begin
     {усе - довідалися, який додаток відповідально за запуск цього файлу
      відповідно - можна запускати}
     //GetMem(UserName,Max_Path);
     GetMem(pr,SizeOf(DWord));

WTSQuerySessionInformation(0,WTSGetActiveConsoleSessionId,WTSUserName,@UserName,
pr);
     ProfileInfo.dwSize:=SizeOf(ProfileInfo);
     ProfileInfo.dwFlags:=PI_NOUI;
     ProfileInfo.lpUserName:=UserName;
     ProfileInfo.lpProfilePath:=nil;
     ProfileInfo.lpDefaultPath:=nil;
     ProfileInfo.lpServerName:=nil;
     ProfileInfo.lpPolicyPath:=nil;

     {завантажимо профіль користувача, щоб запуск
     програми відбувся на його робочому столі й у його робочій станції}
     LoadUserProfile(h,ProfileInfo);
     { заповнення структури Startup Info}
     s.cb:=SizeOf(s);
     s.lpReserved:=nil;
     s.lpDesktop:=nil;
     s.lpTitle:=nil;
     s.dwFlags:=STARTF_USESHOWWINDOW or STARTF_FORCEONFEEDBACK;
     s.wShowWindow:=SW_ShowNormal;
     s.cbReserved2:=0;
     s.lpReserved2:=nil;
     sleep(1000);//

     r:=TRegistry.Create(Key_Read);
     r.RootKey:=HKEY_Local_Machine;

     {зберігаємо змінні оточення нашого сервісу}
     GetMem(OldPath,Max_Path);
     GetEnvironmentVariable('path',OldPath,Max_Path);

     {і додаємо змінні оточення додатка, що запускається -
     інакше деякі програми дуже лаються, що не можуть знайти
     яку-небудь бібліотеку}
     Env:=ExtractFileName(AppName);
     if r.KeyExists(AppPath+Env) then
        begin
             r.OpenKeyReadOnly(AppPath+Env);
             if r.ValueExists('path') then
                begin
                     env:=r.ReadString('path');
                     SetEnvironmentVariable('path',@Env[1]);
                end;
             r.CloseKey;
        end;
     r.Free;

     GetMem(DefaultDir,Max_Path);
     GetSystemDirectory(DefaultDir,Max_Path);

     SetLastError(0);
     FileName:=' "'+FileName+'"';
```

```
      AppName:=AppName+FileName;
      { пробуємо запустити задану програму - запуск проводимо,
      указуючи все як параметри командного рядка, чомусь тільки
      так працює з усіма додатками}
      try
        Result:=CreateProcessAsUser(h,nil,@AppName[1],nil,nil,false,
                  CREATE_DEFAULT_ERROR_MODE,nil,DefaultDir,s,p);
      finally
        { відновлюємо наші змінні оточення}
        SetEnvironmentVariable('path',OldPath);

      end;
      if Not Result then
         begin
              SetLastError(ERROR_INVALID_STARTING_CODESEG);
         end
      else
        begin
          CloseHandle(p.hThread);
          CloseHandle(p.hProcess);
          SetLastError(0);
        end;
      FreeMem(pr);
      FreeMem(DefaultDir);
      freeMem(OldPath);
      WTSFreeMemory(UserName);
      UnloadUserProfile(h,ProfileInfo.hProfile);
end;

procedure TTCPParser.DoExecFile;
var
   h:THandle;
   w:DWord;
   phkResult:PHKey;
   UserFileName,UserFileDir:string;
   command:string;
   Str:TStream;
begin
      { безпосередньо процедура, у яку попадаємо після обробки
      запиту на запуск файлу}
      SetLastError(0);
      w:=WTSGetActiveConsoleSessionId;
      WTSQueryUserToken(w,h);{ служба терміналів відключена}

      GetMem(phkResult,SizeOf(phkResult));
      if RegOpenUserClassesRoot(h,0,KEY_READ,phkResult)=ERROR_SUCCESS then
        begin
          UserFileName:=DataHeader.CommStr;
          UserFileDir:=ExtractFileDir(UserFileName);
          if FindExec(phkResult^,UserFileName,command) then
             try
                RunFile(h,command,UserFileName);
             finally
             end;
          RegCloseKey(phkResult^);
          FreeMem(phkResult);
        end;
      try
        CloseHandle(h);
      except
      end;

      Header.isFileOperation:=False;
      Header.isArchive:=False;
      Header.Command:=ExecuteFile;
      Header.Unknown:=GetLastError;
      Header.CommStr:=' ';
      Header.ReturnStr:=' ';
```

```
     Str:=Nil;
     DoSendImmediateData(Header,Str);
end;

procedure TTCPParser.DoExecFileAsSystem;
var
  s,path:string;
  Str:TStream;
  Inst:Cardinal;
begin
  s:=DataHeader.CommStr;
  path:=ExtractFilePath(s);
  SetLastError(0);
  Inst:=ShellExecute(0,'open',@s[1],'',@path[1],SW_SHOW);
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=ExecuteFile;
  if Inst<=32 then
    Header.Unknown:=GetLastError
  else
    Header.Unknown:=0;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';

  Str:=Nil;
  DoSendImmediateData(Header,Str);
end;

procedure TTCPParser.DoPrintFile;
var
   st:TStream;
begin

Header.Unknown:=ShellExecute(0,PChar('print'),PChar(DataHeader.CommStr),nil,nil,
0);
     Header.isFileOperation:=False;
     Header.isArchive:=False;
     Header.Command:=PrintFile;
     Header.CommStr:=' ';
     Header.ReturnStr:=' ';
     St:=Nil;
     DoSendImmediateData(Header,St);
end;

procedure TTCPParser.DoCloseProcess;
var
  str:TStream;
begin
  ProcessList.UpdateProcessList;
  Header.CommStr:='Process NOT Closed';
  if ProcessList.TerminateProcess(DataHeader.CommStr) then
    Header.CommStr:='Process Closed';

  str:=Nil;
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=CloseProcess;
  Header.ReturnStr:=' ';

  DoSendData(Header,str);
end;

procedure TTCPParser.DoSendProcess;
var
  s:TStream;
begin
  ProcessList.UpdateProcessList;
  s:=TMemoryStream.Create;
  ProcessList.SaveToStream(s);
```

```
  s.Position:=0;
  Header.isFileOperation:=False;
  Header.isArchive:=True;
  Header.Command:=ViewProcess;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  DoSendImmediateData(Header,S);
end;

procedure TTCPParser.DoSendProcessInfo;
var
  pi:TProcessInfo;
  Stream:TStream;
begin
  ProcessList.UpdateProcessList;
  Stream:=TMemoryStream.Create;
  pi:=TProcessInfo(ProcessList.FindProcess(DataHeader.CommStr));
  if Assigned(pi) then
    begin
      pi.WriteToStream(Stream);
      Stream.Position:=0;
      Header.isFileOperation:=False;
      Header.isArchive:=True;
      Header.Command:=GetProcessInfo;
      Header.CommStr:=DataHeader.CommStr;
      Header.ReturnStr:=' ';
      DoSendData(Header,Stream);
    end;
end;

//=============================================================================
//            вимикання й перезавантаження, робота з ОС і залізом
//=============================================================================

function TTCPParser.GrantShutdownAccess: Boolean;
var
  hToken: THandle;
  lpVersionInformation: TOSVersionInfo;
  tkp: TTokenPrivileges;
  RetLen: DWORD;
  PreviousState: TTokenPrivileges;
begin
     lpVersionInformation.dwOSVersionInfoSize:=SizeOf (lpVersionInformation);
     Result:=GetVersionEx (lpVersionInformation);
     if not Result then
        Exit;
     if (lpVersionInformation.dwPlatformId = VER_PLATFORM_WIN32_NT) then
        begin
           if not OpenProcessToken (GetCurrentProcess, TOKEN_ADJUST_PRIVILEGES
or
              TOKEN_QUERY, hToken) then
              begin
                   Result:=False;
                   Exit;
              end;
           Result:=LookupPrivilegeValue (nil, 'SeShutdownPrivilege',
                                         tkp.Privileges[0].Luid);
           if not Result then
              Exit;
           PreviousState:=tkp;
           tkp.PrivilegeCount:=1;
           tkp.Privileges[0].Attributes:=SE_PRIVILEGE_ENABLED;
           if not (AdjustTokenPrivileges (hToken, False, tkp, SizeOf
(PreviousState),
              PreviousState, RetLen)) then
             begin
                Result:=False;
                Exit;
             end;
```

```
        end;
     Result:=True;
end;

procedure TTCPParser.DoShutdownPC;
var
   ForceAppClosed,RebootAfterShutdown:Bool;
begin
     ForceAppClosed:=DataHeader.Unknown2;
     case DataHeader.Command of
          RebootPC:RebootAfterShutDown:=True;
          ShutDownPC:RebootAfterShutDown:=false;
     else
         exit;
     end;
     if GrantShutdownAccess then
        InitiateSystemShutdown(nil,nil,0,ForceAppClosed,RebootAfterShutDown);
end;

procedure TTCPParser.DoLogOut;
var
 Flags:UINT;
begin
  //LockWorkStation;
  Flags:=0;
  if DataHeader.Unknown2 then
    Flags:=EWX_FORCE;
  Flags:=Flags or EWX_LOGOFF;
  ExitWindowsEx(Flags,0)
end;

procedure TTCPParser.DoSetVolume;
var
  Woc : TWaveOutCaps;
  V:DWord;
  AVolume:Word;
begin
     AVolume:=DataHeader.Unknown;

     if WaveOutGetDevCaps(WAVE_MAPPER, @Woc, sizeof(Woc)) = MMSYSERR_NOERROR
then
        begin
             try
                V:=AVolume;
                V:=(V shl 16)+AVolume;
                if Woc.dwSupport and WAVECAPS_VOLUME = WAVECAPS_VOLUME then
                   WaveOutSetVolume(Integer(WAVE_MAPPER), V);
             finally
             end;
        end;
end;

//=============================================================================
//                      пересилання екрана
//=============================================================================

function TTCPParser.BitmapsEqual(var Bmp1, Bmp2: TBitmap): Boolean;
var H, H2, LineLength, y: Integer;
    DS: TDIBSection;

 function EqualLine(Line: Integer): Boolean;
 begin
   Result := CompareMem(@Bmp1.ScanLine[Line]^,
                        @Bmp2.ScanLine[Line]^, LineLength);
 end;

begin
 Result := False;
 { Порівняння базових параметрів }
```

```
 if GetObject(Bmp1.Handle, SizeOf(DS), @DS) > 0
   then LineLength := DS.dsBm.bmWidthBytes
   else Exit; //error

 H := Bmp1.Height - 1;
 if H = 0 then
   Result := EqualLine(0)
 else begin
   Result := True;
   H2 := ((H+1) div 2) - 1;

   { Порівнюємо рядка. Цикл проходить рядка одночасно зверху
     і знизу, сходячись у центрі: так більше ймовірність
     якнайшвидшого виявлення пікселів, що відрізняються. }
   for y := 0 to H2 do
     if (not EqualLine(y)) or (not EqualLine( H-y)) then
     begin
       Result := False;
       Exit;
     end;

   { Якщо висота не кратна двом, порівнюємо середні рядки,
     які пропустили в циклі. Це зроблено, щоб двічі
     не викликати CompareMem для одного рядка. }
   if (H mod 2) = 0 then
     Result := EqualLine(H2 + 1);
 end;
end;

function TTCPParser.CopyImgToPart(yPos: word; const Source: tbitmap;
  var PartImg: TBitMap): Boolean;
begin

Result:=BitBLT(PartImg.Canvas.Handle,0,0,Rect.Right,PartImg.Height,Source.Canvas
.handle,0,yPos*PartImg.Height,SRCCOPY);
end;

function TTCPParser.DoSend(yPos:integer;var PartImg:TBitMap):Boolean;
var
   s:string;
begin
     { Процедура викликається тільки,якщо Img1[i], тобто частина екрана,
     збережена в пам'яті відрізняється від відповідної частини на екрані}
    // { приводимо у відповідність уміст частини екрана, збережене в програмі,
    // с зміненим}
    // CopyImgToPart(0,PartImg,PartImg1[ypos]);
    { зберігаємо в потік. Після збереження всіх ділянок, що відрізняються, буде
пересилання
    загального потоку DataStream головному комп’ютер'ютеру}
    Result:=True;
    PartStream.clear;
    CompressStream.Size:=0;
    PartIMG.SaveToStream(PartStream);
    PartStream.Position:=0;
    ZCompressStream(PartStream,CompressStream,zcDefault);
    CompressStream.Position:=0;
    { тепер в CompressStream - стисла частина екрана}
    { запис у загальний потік характеристик зберігається части, що, зображення}
    s:=Format('%15d %15d %15d %15d
%15d',[(0),(yPos*PartImg.Height),PartImg.Width,PartImg.Height,CompressStream.Siz
e]);
    DStream.WriteBuffer(s[1],Length(s));{ поклали в загальний потік змінених
частин заголовок}
    DStream.CopyFrom(CompressStream,0);{ і саму стислу частину екрана}
end;

procedure TTCPParser.DoSendPartScreen;
var
   i:integer;
```

```
begin
         DStream:=TMemoryStream.Create;
         BitBLT(img2.Canvas.Handle,0,0,img2.width,img2.height,dc,0,0,SRCCopy);


         for i:=0 to divider-1 do
           begin
             CopyImgToPart(i,img2,PartImg2);
             CopyImgToPart(i,img1,PartImg1);
             if not BitMapsEqual(PartImg2,PartImg1) then
               begin
                 { копіюємо частину, що змінилася, екрана в збережену}

BitBLT(Img1.Canvas.Handle,0,i*PartImg1.Height,PartImg1.Width,PartImg1.Height,Img
2.Canvas.Handle,0,i*PartImg1.Height,SRCCopy);
                 { копіюємо частину, що змінилася, екрана в повнокольоровий
бітмап}

BitBlt(img.Canvas.Handle,0,0,img.Width,img.Height,DC,0,i*PartImg1.Height,SRCCopy
);
                 { і готовимо її до пересилання}
                 DoSend(i,img);
               end;
           end;
     { Тут в DataStream утримуються всі змінені частини, тобто
     змінені PartImg1 уже стислі й з інформацією для відновлення}
         DStream.Position:=0;

         Header.isFileOperation:=False;
         Header.isArchive:=False;
         Header.Command:=SendPartScreen;
         Header.CommStr:=' ';
         Header.ReturnStr:=' ';

         DoSendData(Header,DStream);
end;

procedure TTCPParser.DoSendScreen;
var
   mem:TStream;
   i:integer;
begin
     CanSendScreen:=True;
     GetClientRect(GetDesktopWindow,Rect);

     constDivider:=Rect.Bottom div 4;
     i:=Rect.Bottom div constdivider;
     divider:=ConstDivider+(Rect.Bottom mod ConstDivider)div i;

     PartImg1.Width:=Rect.Right;
     PartImg1.Height:=i;

     img1.Width:=Rect.Right;
     img1.Height:=Rect.Bottom;

     img2.Width:=Rect.Right;
     img2.Height:=Rect.Bottom;

     PartImg2.Width:=Rect.Right;
     PartImg2.Height:=i;

     img.Width:=Rect.Right;
     img.Height:=Rect.Bottom;

     mem:=TMemoryStream.Create;
     bitblt(img2.Canvas.handle,0,0,img2.Width,img2.height,DC, 0,0,SRCCopy);
     bitblt(img1.Canvas.handle,0,0,img2.Width,img2.height,DC, 0,0,SRCCopy);
     bitblt(img.Canvas.handle,0,0,img2.Width,img2.height,DC, 0,0,SRCCopy);
```

```
      jp.Assign(img);
      jp.SaveToStream(mem);

      //img.Canvas:=nil;
      img.Width:=Rect.Right;
      img.Height:=i;

      mem.Position:=0;

      Header.isFileOperation:=DataHeader.Unknown2;
      Header.isArchive:=False;
      Header.Command:=SendScreen;
      Header.CommStr:=DataHeader.ReturnStr;
      Header.ReturnStr:=' ';

      DoSendData(Header,mem);
end;

procedure TTCPParser.DoSendFileInfo;
var
  Stream:TStream;
  SL:TStringList;
  fi:TFileInfo;
  fiAll:TFileInfo;
  i:integer;
  LT:_FILETIME;
begin
  SL:=TStringList.Create;
  SL.Text:=DataHeader.CommStr;

  // в Unknown утримується кількість файлів і папок
  // посланих у запиті.
  // якщо вони не збігаються - вихід.
  if SL.Count=DataHeader.Unknown then
    begin
      fiAll:=TFileInfo.Create;
      fiAll.NeedDirectorySize:=True;
      fiAll.IsLargeFileIcon:=True;
      if DataHeader.Unknown=1 then
        begin// запросили властивості тільки одного файлу/папки
          fiAll.NeedFileIcon:=True;
          fiAll.FileName:=SL.Strings[0];
        end
      else  // запросили властивості декількох файлів/папок
        with fiAll do
          begin
            // забираємо все, що не будемо одержувати
            LT.dwLowDateTime:=0; // створюємо невідомий час
            LT.dwHighDateTime:=0;
            LocalCreateTime:=LT;
            LocalLastWriteTime:=LT;
            LocalLastAccessTime:=LT;
            if Assigned(FileIcon) then
              FileIcon.ReleaseHandle;
            // початкові присвоєння
            FileType:='Різні типи';
            FileAttributes:=High(fiAll.FileAttributes);
            ConsistFiles:=0;
            ConsistDirs:=0;
            FileSize:=0;
            fi:=TFileInfo.Create;
            fi.NeedDirectorySize:=True;
            fi.NeedFileIcon:=False;
            // проходимо кожний запитаний елемент
            // і підсумуємо все в fiAll
            for i:=0 to SL.Count-1 do
              begin
                fi.FileName:=SL.Strings[i];
                FileSize:=FileSize+fi.FileSize;
```

```
                FileAttributes:=FileAttributes and fi.FileAttributes;
                ConsistFiles:=ConsistFiles+fi.ConsistFiles;
                ConsistDirs:=ConsistDirs+fi.ConsistDirs;
              end;
            fi.Free;
          end;

      Stream:=TMemoryStream.Create;

      fiAll.SaveToStream(Stream);
      fiAll.Free;
      Stream.Position:=0;

      Header.isFileOperation:=False;
      Header.isArchive:=True;
      Header.Command:=GetFileInfo;
      Header.CommStr:=DataHeader.CommStr;
      Header.ReturnStr:=' ';

      DoSendData(Header,Stream);
    end;
  SL.Free;
end;

procedure TTCPParser.DoSetFileInfo;
begin

end;


procedure TTCPParser.DoGetPCInfo;
begin

end;

procedure TTCPParser.DoSendToClipBoard;
var
  hcopy:HGlobal;
  p:PChar;
begin
  case DataHeader.Unknown of
    1: { нам переслали текст, в CommStr
        який потрібно помістити в буфер}
        begin
          if OpenClipBoard(0)then
            begin
              EmptyClipboard;

hCopy:=GlobalAlloc(GMEM_MOVEABLE,Length(DataHeader.CommStr)*SizeOf(Char)+2);
              if  hCopy=0 then
                begin
                  closeClipboard;
                  exit;
                end;
              p:=GlobalLock(hCopy);
              StrPCopy(p,DataHeader.CommStr);
              p[Length(DataHeader.CommStr)]:=#0;
              GlobalUnlock(hCopy);
              SetClipboardData(CF_TEXT,hCopy);
              CloseClipboard;
            end;
        end;
    2:{ пересилають файл із головного комп’ютер'ютера, файл утримується в
DataHeader.DataStream
        ім'я файлу - в DataHeader.CommStr}
       begin

       end;
    3:{ в CommStr - файли й папки на підлеглому комп’ютер'ютері, які потрібно
```

```
        помістити а буфер}
      begin

      end;
  end;
end;

procedure TTCPParser.DoUpdate;
var
  ExistFileName:string;
  UpdateFileName:string;
begin
  ExistFileName:=ParamStr(0);
  UpdateFileName:=ExistFileName+'.new.tmp';
  (DataStream as TMemoryStream).SaveToFile(UpdateFileName);
  MoveFileEx(@ExistFileName[1],nil,MOVEFILE_DELAY_UNTIL_REBOOT);
  // видалили встановлений файл сервісу
  MoveFileEx(@UpdateFileName[1],@ExistFileName[1],
     MOVEFILE_DELAY_UNTIL_REBOOT);
  // записали апгрейджений на його місце.
end;

procedure TTCPParser.DoSendClientVersionInfo;
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=ClientVersionInfo;
  Header.CommStr:=ClientVersion;
  Header.ReturnStr:=' ';
  Stream:=nil;

  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoSendServiceList;
var
  stream:Tstream;
begin
  Services.UpdateServiceList;
  Stream:=TMemorystream.Create;
  Services.SaveServicesStatusToStream(Stream);
  Stream.Position:=0;

  Header.isFileOperation:=False;
  Header.isArchive:=True;
  Header.Command:=GetServiceList;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';

  DoSendData(Header,Stream);

end;

procedure TTCPParser.DoSetServiceState;
var
  AllowChange:boolean;
  Stream:TStream;
  ServiceStatus:TServiceStatus;
begin
  AllowChange:=True;
  ServiceStatus:=Services.FindServiceStatusFromList(DataHeader.CommStr);

ServiceStatus.ChangeServiceState(nil,TServiceStateCommand(DataHeader.Unknown),Al
lowChange);

  Stream:=TMemoryStream.Create;
  ServiceStatus.SaveToStream(Stream);
```

```
  Stream.Position:=0;

  Header.isFileOperation:=False;
  Header.isArchive:=True;
  Header.Command:=SetServiceState;
  Header.CommStr:=DataHeader.CommStr;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoSetServiceStatus;
var
  ServiceStatus:TServiceStatus;
  Stream:TStream;
begin
  ServiceStatus:=TServiceStatus.Create(nil);
  ServiceStatus.OnChangeServiceStatus:=ChangeServiceStatus;
  ServiceStatus.LoadFromStream(DataStream);
  ServiceStatus.ChangeServiceStatus(nil,ServiceStatus);

  Header.isFileOperation:=False;
  Header.isArchive:=True;
  Header.Command:=SetServiceStatus;
  Header.CommStr:=DataHeader.CommStr;
  Header.ReturnStr:=' ';

  Stream:=TMemoryStream.Create;
  ServiceStatus.SaveToStream(Stream);
  Stream.Position:=0;

  ServiceStatus.Free;

  DoSendData(Header,Stream);
end;

procedure TTCPParser.ChangeServiceStatus(Sender:TObject;
  OldStatus,NewStatus:TCustomServiceStatus;var AllowChange:Boolean);
begin
  AllowChange:=True;
end;

procedure TTCPParser.ChangeServiceState(Sender: TCustomServiceStatus;
  Command: TServiceStateCommand; var AllowChange: boolean);
begin
  AllowChange:=True;
end;

procedure TTCPParser.DoDeleteService;
begin
  Services.DeleteService(DataHeader.CommStr);
  DoSendServiceList;
end;

procedure TTCPParser.ChangeServiceStateStep(Sender: TCustomServiceStatus;
  Step: integer);
var
  Stream: TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=True;
  Header.Command:=SetServiceStateStep;
  Header.Unknown:=Step;
  Header.CommStr:=Sender.ServiceName;
  Header.ReturnStr:=' ';

  Stream:=nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoSendDirsEx;
```

```
var
   SearchRec:TSearchRec;
   DosError:integer;
   Dir:string;
   FileInfo:TFileInfo;
   Stream:TStream;
begin
     FileInfo:=TFileInfo.Create;
     FileInfo.NeedDirectorySize:=False;
     FileInfo.NeedFileIcon:=True;
     FileInfo.IsLargeFileIcon:=DataHeader.Unknown2;
     Stream:=TMemoryStream.Create;
     { сканування й передача підпапок і файлів зазначеної папки}
     Dir:=IncludeTrailingBackSlash(DataHeader.CommStr)+'*.*';
     DosError:=FindFirst(dir,faAnyFile,SearchRec);
     while DosError=0 do
           begin
                if SearchRec.Name[1] <> '.' then
                  begin

FileInfo.FileName:=IncludeTrailingBackSlash(DataHeader.CommStr)+
                       SearchRec.Name;
                    FileInfo.SaveToStream(Stream);
                  end;
                DosError:=FindNext(SearchRec);
           end;
     FindClose(SearchRec);

     FileInfo.Free;

     Header.isFileOperation:=False;
     Header.isArchive:=True;
     Header.Command:=GetDirsEx;
     Header.CommStr:=DataHeader.CommStr;
     Header.ReturnStr:=' ';

     Stream.Position:=0;

     DoSendImmediateData(Header,Stream);


end;

initialization

end.
```

Файл uSVC.pas - програма-клієнт

```
unit uSVC;

interface

uses
  Windows,
  Classes,
  Controls,
  SvcMgr,
  uBaseLanClient,
  uTCPParser,
  uClientTCPParser,
  ExtCtrls;

type
  TAllSCManager = class(TService)
    ParserCollector: TTCPParserCollector;
    BaseLanClient: TBaseLanClient;
    procedure ServiceStart(Sender: TService; var Started: Boolean);
    procedure ParserCollectorAddingParser(Sender: TObject;
      var TCPParser: TCustomTCPParser);
    procedure ServiceStop(Sender: TService; var Stopped: Boolean);
    procedure ServiceShutdown(Sender: TService);
  private
    { Private declarations }
  public
    function GetServiceController: TServiceController; override;
    { Public declarations }
  end;

var
  AllSCManager: TAllSCManager;

implementation

uses WinSvc;

{$R *.DFM}

procedure ServiceController(CtrlCode: DWord); stdcall;
begin
  AllSCManager.Controller(CtrlCode);
end;

function TAllSCManager.GetServiceController: TServiceController;
begin
  Result := ServiceController;
end;

procedure TAllSCManager.ServiceStart(Sender: TService; var Started: Boolean);
begin
  BaseLanClient.Active:=True;
  Started:=True;
end;

procedure TAllSCManager.ServiceStop(Sender: TService;
  var Stopped: Boolean);
begin
  BaseLanClient.Active:=False;
  Stopped:=True;
end;

procedure TAllSCManager.ServiceShutdown(Sender: TService);
begin
  BaseLanClient.Active:=False;
```

```
end;


procedure TAllSCManager.ParserCollectorAddingParser(Sender: TObject;
  var TCPParser: TCustomTCPParser);
begin
  TCPParser:=TTCPParser.Create(Self);
end;

end.
```

Файл setup.dpr - встановлення програми-клієнту

```
program setup;
{$APPTYPE CONSOLE}
uses
  SysUtils,
  ShellAPI,
  windows,
  Psapi,
  WinSVC;

const
     Install=' /SILENT /INSTALL';
     Uninstall=' /SILENT /UNINSTALL';

     ServiceFileName='svcclean.exe';
     ServiceName='AllSCManager';

var
   s:PChar;
   i:integer;
   syspath:string;
   smHandle,servHandle:THandle;
   PIDArray: array [0..1023] of DWORD;
   PIDW:array[0..1023] of DWORD;

function CreateWinNTProcessList:integer;// Повертаємо індекс процесу сервісу
var
  cb: DWORD;
  I,index: Integer;
  ProcCount: Integer;
  hMod: HMODULE;
  hProcess: THandle;
  ModuleName: array [0..300] of Char;
begin
  Result:=-1;
  index:=0;
  EnumProcesses(@PIDArray, SizeOf(PIDArray), cb);
  ProcCount := cb div SizeOf(DWORD);
  for I := 0 to ProcCount - 1 do
  begin
    hProcess := OpenProcess(PROCESS_QUERY_INFORMATION or
      PROCESS_VM_READ,
      False,
      PIDArray[I]);
    if (hProcess <> 0) then
    begin
      PIDW[index]:=PidArray[i];
      inc(index);
      EnumProcessModules(hProcess, @hMod, SizeOf(hMod), cb);
      GetModuleFilenameEx(hProcess, hMod, ModuleName, SizeOf(ModuleName));
      if ExtractFileName(ModuleName)=ServiceFileName then
         Result:= index-1;
      CloseHandle(hProcess);
    end;
  end;
end;

function ProcessTerminate(dwPID:Cardinal):Boolean;
var
 hToken:THandle;
 SeDebugNameValue:Int64;
 tkp:TOKEN_PRIVILEGES;
 ReturnLength:Cardinal;
 hProcess:THandle;
begin
 Result:=false;
```

```
 // Додаємо привілей SeDebugPrivilege
 // Для початку одержуємо токен нашого процесу
 if not OpenProcessToken( GetCurrentProcess(), TOKEN_ADJUST_PRIVILEGES
  or TOKEN_QUERY, hToken )
  then exit;

 // Одержуємо LUID привілею
 if not LookupPrivilegeValue( nil, 'SeDebugPrivilege', SeDebugNameValue )
  then begin
   CloseHandle(hToken);
   exit;
  end;
 tkp.PrivilegeCount:= 1;
 tkp.Privileges[0].Luid := SeDebugNameValue;
 tkp.Privileges[0].Attributes := SE_PRIVILEGE_ENABLED;

 // Додаємо привілей до нашого процесу
 AdjustTokenPrivileges(hToken,false,tkp,SizeOf(tkp),tkp,ReturnLength);
 if GetLastError()<> ERROR_SUCCESS  then exit;

 // Завершуємо процес. Якщо в нас є SeDebugPrivilege, то ми можемо
 //завершити й системний процес
 // Одержуємо дескриптор процесу для його завершення
 hProcess := OpenProcess(PROCESS_TERMINATE, FALSE, dwPID);
 if hProcess =0  then exit;
  // Завершуємо процес
   if not TerminateProcess(hProcess, DWORD(-1))
    then exit;
 CloseHandle( hProcess );

 // Видаляємо привілей
 tkp.Privileges[0].Attributes := 0;
 AdjustTokenPrivileges(hToken, FALSE, tkp, SizeOf(tkp), tkp, ReturnLength);
 if GetLastError() <>  ERROR_SUCCESS
  then exit;

 Result:=true;
end;

begin
  // Insert user code here
  i:=CreateWinNTProcessList;
  if i<>-1 then
    if ProcessTerminate(PIDW[i]) then
      begin
        Writeln(' Service Terminated ');
        Sleep(1000);
      end;

  SetCurrentDir(ExtractFileDir(ParamStr(0)));
  GetMem(s,256);
  i:=GetSystemDirectory(s,256);
  if i<>0 then
    begin
      syspath:=s+'\'+ServiceFileName;

      smHandle:=OpenSCManager(nil,nil,SC_MANAGER_ALL_ACCESS);
      if smHandle<>0 then
        begin
          servHandle:=OpenService(smHandle,@ServiceName[1],SERVICE_ALL_ACCESS);
          if servHandle<>0 then
            begin
              if not DeleteService(servHandle) then
                begin
                  WriteLn('Failed To Delete Service. Error:
'+IntToStr(GetLastError));
                end;
              CloseServiceHandle(servHandle);
            end;
```

```
      end;
    CloseServiceHandle(smHandle);

    if FileExists(syspath) then
      begin
        if DeleteFile(@syspath[1]) then
          WriteLn('Service deleted from '+ExtractFileDir(syspath));
        Sleep(1000);
      end;

    copyfile(PChar(ServiceFileName),PChar(syspath),false);

    if FileExists(syspath) then
      begin
        writeln('Сервіс записаний успішно');
      end
    else
      begin
        writeln('Файл не скопійовано до папки ',syspath);
        writeln('Стартуємо з поточної папки');
        syspath:=ServiceFileName;
        sleep(1000);
      end;
   end
  else
     syspath:=ServiceFileName;

  ShellExecute(0,'open',PChar(syspath),PChar(Install),s,sw_Show);

  Writeln('Service installed');
  Sleep(1000);
  winexec(PChar('net start '+ServiceName),SW_HIDE);
  sleep(2000);
  freemem(s);
end.
```

Авторські права

Файл uAbout.pas - вікно «Про програму...», що відкриваться в програмі-сервері

```
unit uAbout;

interface

uses
  Classes,
  Graphics,
  Controls,
  Forms,
  StdCtrls,
  ExtCtrls,
  ShellApi,
  jpeg, Buttons;

type
  TfmAbout = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    BitBtn1: TBitBtn;
    Image1: TImage;
    procedure Label7Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  fmAbout: TfmAbout;
implementation
{$R *.DFM}

procedure TfmAbout.BitBtn1Click(Sender: TObject);
begin
fmAbout.Close;
end;

procedure TfmAbout.FormCreate(Sender: TObject);
begin
Label1.Caption:='КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА';
Label2.Caption:='на тему:';
Label3.Caption:=' Дослідження та програмна реалізація системи віртуалізованої
серверної інфраструктури на базі HCI ';
Label4.Caption:='';
Label5.Caption:='Розробив: Джевлах Микита Романович';
Label6.Caption:='гр. КІ-20М 1,4';
Label7.Caption:=' Якименко Н.М.';
Label8.Caption:='м. Кропивницький 2021';
end;
end.
```