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

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

# ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
**за першим (бакалаврським) рівнем вищої освіти**
на тему
**“Програмне забезпечення системи Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури”**



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

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

м. Кропивницький

**Центральноукраїнський національний технічний університет**

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань_12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

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

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

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

*Нікітіну Богдану Олександровичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури*

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

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

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури*

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

*1. Призначення та область використання.*
*2.Перегляд аналогічних існуючих систем.*
*3. Опис і обґрунтування проектних рішень.*
*4. Етапи програмування системи.*
*5. Впровадження системи в промислову експлуатацію.*
*6. Висновки*

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

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

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

## КАЛЕНДАРНИЙ ПЛАН

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



Дата видачі завдання
« 27 » лютого 2026 р.

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

Завдання прийнято до виконання
« 27 » лютого 2026 р.

Підпис здобувача
_____ Нікітін Б.О.
(прізвище та ініціали)

# АНОТАЦІЯ

**Нікітін Б.О. Програмне забезпечення системи Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2026.**

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури.

Метою розробки є програмне забезпечення системи Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури.

Результат роботи – програмна реалізація системи Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури.

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

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

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

Програму розроблено в середовищі Python.

**Ключові слова**: комп'ютерна інженерія, Virtual Desktop Infrastructure

# ABSTRACT

**Nikitin B.O. Software for the Virtual Desktop Infrastructure system for updating computing infrastructure. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.**

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for the Virtual Desktop Infrastructure system for updating computing infrastructure.

The purpose of the development is software for the Virtual Desktop Infrastructure system for updating computing infrastructure.

The result of the work is a software implementation of the Virtual Desktop Infrastructure system for updating computing infrastructure.

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

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

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

The program was developed in the Python environment.

**Keywords:** computer engineering, Virtual Desktop Infrastructure

**ЗМІСТ**

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

| | | |
|---|---|---|
| АВН | – | автоматичний визначник номера |
| АТМ | – | автоматична телефонна мережа |
| ВМ | – | віртуальна машина |
| ЛОМ | – | локальна обчислювальна мережа |
| ОП | – | оперативна пам'ять |
| ОС | – | операційна система |
| ПК | – | персональний комп’ютер |
| СЗД | – | системи зберігання даних |
| ЦОД | – | центр обробки даних |
| ЦП | – | центральний процесор |
| AES | – | алгоритм шифрування |
| DNS | – | сервер домених імен |
| IP | – | Internet Protocol |
| LAN | – | локальна мережа |
| MAC | – | Media Access Control – управління доступом до носія |
| NAT | – | Network Address Translation |
| TCP | – | Transmission Control Protocol |
| UDP | – | User Datagram Protocol – протокол користувальницьких дейтаграмм |
| UPS | — | блок безперебійного живлення |
| VDI | | Virtual Desktop Infrastructure |

# ВСТУП

**Актуальність теми**. VDI (Virtual Desktop Infrastructure) – це технологія для створення віртуальної IT-інфраструктури робочих столів. Вона дозволяє розвертати повноцінні робочі місця на базі одного сервера або групи серверів, на якому працюють безліч віртуальних ПК.

Персональні комп'ютери є основним робочим місцем співробітників. Забезпечуючи високу гнучкість у налаштуваннях і установці програмного забезпечення, вони, проте вимагають більших операційних витрат на підтримку, обслуговування, ліцензування

Головною метою впровадження VDI є:

– Підвищення безпеки.

– Зниження витрат на адміністрування робочих місць.

– Підвищення гнучкості, висока швидкість розгортання й зміни робочих місць.

– Багатофункціональне використання.

**Мета й завдання дослідження.** Метою роботи є програмне забезпечення системи Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури.

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

– Огляд існуючих систем Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури.

– Дослідження системи Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури.

– Програмна реалізація системи Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури.

**Мета й завдання дослідження.** Метою роботи є програмне забезпечення системи Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури.

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

– Огляд існуючих систем Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури.

– Дослідження системи Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури.

– Програмна реалізація системи Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури.

**Практична цінність отриманих результатів** полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

# 1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

## 1.1 Призначення системи

Замість установки кожному користувачеві комп'ютера, в VDI персональні комп'ютери консолідовані у віртуальному середовищі. Таким чином, всі обчислення відбуваються на групі серверів, а всі дані зберігаються на системі зберігання. Для кожного користувача автоматично створюється віртуальна машина, із шаблона або призначається адміністраторами виділена. Дані шаблони можна обновляти, що дозволяє встановлювати ПЗ, відновлення відразу для всіх користувачів, як локальних, так і віддалених. З іншого боку, дана архітектура дозволяє протягом декількох хвилин створити нове місце для співробітника, із уже готового шаблону.

Віртуальні комп'ютери, з якими користувачі працюють, нічим не відрізняються від звичайних. Інша лише схема роботи. Всі обчислення відбуваються на групі серверів, найчастіше, захищеної від відмови. Користувачі підключаються зі своїх тонких клієнтів, протоколи підключення дозволяють працювати навіть із «важкими» графічними застосунками без відчутних затримок для співробітника. З іншої сторони VDI працює й у випадку поганих каналів Інтернет і часто застосовується у віддалених офісах. На відміну від термінальних служб VDI не вимагає підтримки ПЗ, будь-яке програмне забезпечення буде працювати в середовищі VDI.

Впровадження VDI дозволяє спростити адміністрування комп'ютерів, у тому числі у віддалених офісах, що скорочує витрати на підтримку інфраструктури.

## 1.2 Область застосування

VDI – це комплекс засобів для побудови інфраструктури віртуальних робочих ПК (або віртуальних робочих столів), що дозволяє з різних пристроїв у будь-який момент одержати доступ до робочого стола з будь-якої точки світу (при наявності доступу до Internet як клієнта, так і інфраструктури). Залежно від наявності різних компонентів, можливо велика кількість опцій, таких як локальний доступ до стола (без з'єднання з робочим столом у конкретний момент часу), швидке розгортання нових станцій, режим автентифікації anonymous (безлімітне використання одного аккаунта необмеженим числом користувачів), гнучкі політики й багато чого іншого.

Віртуалізація робочих станцій може ділитися на два більших блоки:

– Віртуалізація робочих столів.

– Віртуалізація застосунків.

Два цих етапи можуть проходити незалежно друг від друга, а також у будь-якому порядку, що дозволить перейти на віртуальні машини досить швидко й безболісно для компанії. На даний момент з'єднуватися з Horizon можна за допомогою настільного ПК, робочих станцій, ноутбуків і нетбуків, великого спектра продукції Apple (MacBook's, IPad, IPhone), тонких клієнтів і нульових клієнтів, а також за допомогою деяких моделей Smartphone.

По своїй суті, ми маємо повністю функціональний користувальницький робочий стіл, що встановлений не в співробітника на ПК, а в серверній.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

# 2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

**Огляд VMware Horizon**

Компанія VMware випустила продукт VMware Horizon (стара назва VMware View, Virtual Infrastructure) уже порівняно давно, майже 6 років тому, але багато хто дотепер або знають про нього тільки на словах, і жодного разу не бачили в дії, або взагалі ніколи про нього не чули.

Що ж це таке, VDI (Virtual Desktop Infrastructure), у чому його основні переваги й відмінності від термінального доступу?

У цьому розділі будуть коротко викладені всі можливості, компоненти й параметри продуктів View.

**VDI**

Для віртуалізації робочих столів нам буде необхідний продукт VMware Horizon, що містить у собі:

- VMware Horizon Connection Server.
- VMware Horizon Composer.
- VMware Horizon Replica Server.
- VMware Horizon Transfer Server.
- VMware Horizon Security Server.

VMware Horizon Connection Server (колишній View Manager) – це і є основний сервер, що створює віртуальне середовище для робочих столів, а також забезпечує роботу інших компонентів View. Установлюватися Connection Server може або на окрему фізичну, або на віртуальну машину.

VMware Horizon Composer – це компонент для створення так званого “золотого образа” (Parent Image) – ізюминка рішення. На основі «золотого образа» буде відбуватися майже миттєве розгортання необхідного нам кількості столів. Установлюється Composer на машину з VCenter Server.

VMware Horizon Replica Server – сервер View, що буде приєднаний, як ще один View Connection Server у консоль керування View. Це потрібно для того, щоб розпаралелити трафік підключень клієнтів до робочих столів. На основі тегів (tags) можливо вхід у певні пули налаштувати через певні сервери View. Репліку потрібно ставити на окрему ВМ, таких реплік може бути скільки завгодно, ніяких зайвих ліцензій здобувати не потрібно.

**Horizon 7**

VMware Horizon Transfer Server – продукт, за допомогою якого ми будемо використовувати режим local mode (режим локального використання робочого стола без діючого з'єднання).

VMware Horizon Security Server – сервер, що захищає середовище View від зовнішнього миру, простими словами firewall для VMware Horizon. Через Horizon Security Server можна організувати доступ до віртуальних робочих столів через інтернет.

Ліцензується VMware Horizon по робочих столах. Ми можемо створити набагато більше робочих столів і записів користувачів, але використовуватися в певний момент часу зможе тільки придбана кількість віртуальних робочих столів.

В VMware Horizon є кілька редакцій.

Також, якщо користувач уже використовує продукцію VMware, і в нього вже є VMware vSphere і VMware vCenter Server, він може придбати версію Horizon Add-on, щоб не переплачувати за вже куплені vSphere і vCenter.

При придбанні будь-якої версії Horizon користувачеві надаються наступні компоненти:

– VMware Agent – установлюється на ВМ, що ми будемо використовувати як робочий стіл.

– VMware Client – програма, за допомогою якої нами буде зроблене з'єднання з робочим столом (якщо не застосовуються тонкі й нульові клієнти).

– VMware Horizon Administrator – універсальна веб-консоль, з під якої буде здійснене керування інфраструктурою View (ярлик буде доступний за замовчуванням на робочому столі операційної системи, на якій установлений VMware Horizon Connection Server).

**PCoIP**

Передача робочого стола нашому клієнтові може здійснюватися по засобах двох протоколів:

– PCoIP.

– RDP.

PCoIP застосовується для швидких мереж, де потрібне шифрування даних. При слабкому каналі передачі даних, а також, якщо не потрібне шифрування краще вибрати RDP.

Основна перевага PCoIP перед RDP у тім, що він видає картинку стола відразу, а RDP використовує порядкове промальовування.

Для досягнення оптимальної якості відображення віртуального робочого стола ми рекомендуємо Вам використовувати нульові клієнти PCoIP на базі чипа Teradici.

Для автентифікації в VMware View також існує підтримка SmartCard 1024 і 2048 біт (512 не підтримується).

Також існує функція Location Based Printing, що дозволяє прив'язати MAC-Адреси клієнтів до певних принтерів, що спростить налаштування печатки.

При необхідності виїхати у відрядження, де обмежений доступ до інтернет, Horizon пропонує використання локального режиму (local mode), що дозволить перевантажити дані про машину на будь-який laptop або інший пристрій і працювати зі своїм робочим столом поза залежністю від наявності зв'язку. Після повернення дані синхронізуються й робота триває вже зі звичайною машиною в ЦОДі.

Всі налаштування облікових записів, політики входу, налаштування режимів і інші параметри керування Horizon налаштовуються за допомогою служби Active Directory.

Для різних потреб можливе налаштування різних варіантів видачі робочого стола кінцевому користувачеві:

– Dedicated mode – режим твердого призначення кожному користувачеві його стола (застосовується в компаніях, коли в кожного користувача повинна бути своє незмінне індивідуальне середовище. місця на жорсткому диску буде використовуватися більше).

– Floating mode – режим плаваючих столів, коли користувач може потрапити на будь-який вільний стіл (використовується, коли всі комп'ютери мають ідентичні параметри, а користувачам лише необхідні певні програми й свій профіль. це заощадить місце на диску.

– Multiple mode – використовується для можливості входу на необмежене число машин з одним логіном/паролем (зручно в демонстраційних цілях, щоб не створювати постійно нові облікові записи).

– Kiosk mode – режим, що дозволяє одержувати доступ з одного/декількох заздалегідь певних фізичних пристроїв користувачам у режимі anonymous (наприклад, платіжний термінал).

За віртуалізацію застосунків відповідає продукт VMware ThinApp.

**VMware ThinApp**

ThinApp – це безагентна система, що “упаковує” разом з необхідним додатком всі дані реєстру/системи, необхідні для роботи застосунку, і конвертує це все в єдиний файл. При чому, ми зможемо заздалегідь створити всі необхідні нам налаштування для потрібного застосунку, що дозволить за кілька хвилин розгорнути вже настроєну програму на необмеженій кількості робочих столів.

Це буде зручно в наступних випадках:

– Якщо ми працюємо на різних версіях ОС Windows, при цьому з одним пакетом програм. Тепер нам буде потрібно всього лише покласти необхідні

файли на будь-який носій, і наш пакет застосунків буде завжди доступний без необхідності установки й додаткових налаштувань.

– Якщо ми одночасно використовуємо кілька версій того самого застосунку. Наприклад, одночасно використовуємо Word 2003 і Word 2010.

Ліцензує ThinApp або по числу користувачів, що використовують застосунок, або по числу пристроїв, з яких застосунок запущений. Також можна придбати ThinApp у комплекті з VMware View Premier.

Якщо ми хочемо використовувати ThinApp окремо від View, нам необхідно буде придбати ThinApp Suite, у комплекті якого вже є 50 стартових ліцензій. Якщо ж ми будемо використовувати View і ThinApp разом, нам досить придбати View Premier, у якому ThinApp буде ліцензуватися по 250$ за кожне користувальницьке підключення.

Продукти віртуалізації робочих столів можуть використовуватися повсюдно, однак найбільше поширення вони одержують у наступних напрямках:

– Навчальні заклади – у ЗВО країни View буде затребуваний особливо, тому що різні заходи, пов'язані з необхідністю швидкого розгортання підготовлених робочих станцій для кожного окремого студента – одна з можливостей VMware View.

– Банки – у великих банках, при необхідності розгортання новому співробітникові Desktop'а зі спеціальним програмним пакетом, раніше системним адміністраторам знадобилося б біля одного робочого дня. Тепер, щоб розгорнути машини будь-якій кількості співробітників, буде досить 10-20 хвилин.

– Віддалений доступ/робота вдома – при роботі вдома вся інформації буде залишатися в межах компанії, при цьому співробітники одержать повноцінне робоче місце, настроєне з усіма вимогами компанії.

– Страхові компанії – де є страхові агенти. Вони ніколи не сидять в офісі, а найчастіше, і не є штатними співробітниками страхових компаній, але повинні мати доступ до корпоративних ресурсів, пошті й іншому. При цьому бажано забезпечити їм комфортну роботу й уникнути витоків інформації.

– Компанії із власним Call центром – багато однотипних робочих столів, співробітники можуть бути географічно розподілені, легеня центральне керування.

VMware Horizon може бути затребуваний і в інших випадках (постійні відрядження, спільна робота й т.д.). Все залежить тільки від потреб і можливостей до придбання пакета Horizon.

**Навіщо і як впроваджують VMware Horizon 7**

87% опитаних знаходить, що найбільш важливою перевагою для компанії буде скорочення витрат на адміністрування робочих станцій.

Будь ласка, майте на увазі, що для деяких питань було кілька варіантів відповідей, тому не обов'язково в сумі повинне вийти 100% (наприклад, як у першому питанні).

**1. Які переваги VDI найбільш актуальні для вашої компанії?**

– Скорочення витрат на адміністрування робочих станцій – 87%.

– Прискорення й спрощення впровадження нових застосунків – 71%.

– Скорочення часу простою користувальницьких ПК – 65%.

– Розвантаження служби технічної підтримки – 71%.

– Стандартизація доступу до корпоративних ресурсів – 68%.

87**%** опитаних знаходить, що найбільш важливою перевагою для компанії буде скорочення витрат на адміністрування робочих станцій.

Скорочення витрат на адміністрування закладено в самій архітектурі VMware Horizon **7**.

Віртуальні робочі столи, особливо створені на базі зв'язаних клонів, практично ідентичні один одному:

– Віртуальне «устаткування» робочих столів ідентично і являє собою віртуалізоване устаткування сервера, на якому ці столи перебувають.

– У віртуальних системах установлені ті самі драйвери.

– На них установлені операційні системи з однаковими патчами й версіями.

– На віртуальних робочих столах установлені ті самі застосунки з однаковими версіями.

– Відновлення операційної системи або застосунку робиться централізовано для «золотого» образа й застосовується одночасно до всім робочим столам.

– Розходження віртуальних робочих столів – це користувальницькі дані, які зберігаються окремо від системи.

Таким чином, і проблеми, які виникають із віртуальними робочими столами загальні.

Загальна проблема, як правило, при грамотному адмініструванні вирішується більш якісно і швидше, ніж десятки або сотні різнорідних проблем, що виникають щодня.

**2. За рахунок чого, на Вашу думку, відбудеться найбільш істотна економія, скорочення вартості володіння IT?**

– Зниження витрат на періодичне відновлення парку ПК – 82%.

– Зниження витрат на адміністрування, конфігурування й підтримки - 66%.

– Скорочення супутніх витрат (електроенергія, оренда та ін.) – 34%.

**82%** опитаних відзначили найважливішу економію в зниженні витрат на періодичне відновлення парку ПК.

Дійсно, вартість тонкого або нульового клієнта для доступу до віртуальних робочих столів нижче вартості сучасного персонального комп'ютера, а строк його служби істотно вище.

Тонкий клієнт підходить для будь-якої операційної системи, установленої на віртуальному робочому столі.

Потужність його процесора й оперативна пам'ять достатня для виконання покладених на нього завдань, а саме: передачі й прийому зображення. Перехід на наступну версію операційної системи не вимагає зміни тонких клієнтів, а відсутність рухливих частин робить тонкий клієнт майже «вічним».

**3. Які будуть найбільш істотні плюси від переходу на VDI у вашій компанії?**

– Збільшення продуктивності праці співробітників – 67%.

– Підвищення оперативності при роботі із замовниками – 33%.

– Позиціонування компанії, як високотехнологічної – 46%.

Це дивно, але **67%** опитаних уважають, що найбільш істотним плюсом буде збільшення продуктивності праці співробітників.

Ми можемо припустити, що мова йде про технічну сторону продуктивності.

Наприклад, періодичне перезавантаження персонального комп'ютера, зниження швидкості його роботи згодом, установка або переустановка (у тому числі за допомогою служби підтримки) застосунків, втрата даних або крадіжка ноутбука – все це в різному ступені забирає час і знижують продуктивність.

Перефразувавши, можна сказати, що віртуальні робітники столи знижують втрати продуктивності.

**4. Які завдання ставить перед собою ваша компанія?**

– Підвищення мобільності користувачів (доступ до корпоративної системи з будь-якої точки) – 44%.

– Централізація корпоративних ресурсів (або забезпечення користувачів філій доступом до корпоративних ресурсів) – 67%.

– Міграція на наступну версію операційної системи (наприклад, Windows 10) – 33%.

– Заміна застаріваючого парку ПК – 62%.

– Підвищення безпеки даних, скорочення розкрадань інформації -61%.

– Планування передачі адміністрування ПК на аутсорсинг – 8%.

– Запобігання втрат даних і резервне копіювання користувальницьких даних – 66%.

– Забезпечення доступу до застосунків у компанії з високою текучкою кадрів (або тимчасових користувачів) – 31%.

– Забезпечення користувачів високопродуктивними робочими станціями для рішення інженерних або інших комплексних завдань – 28%.

Вдобавок до вищеописаного, значне кількість опитаних уважають, що запобігання втрат даних і резервне копіювання користувальницьких даних – найбільш важливе завдання в сфері IT (66%).

Всі віртуальні робочі столи «фізично» перебувають на серверах, а не хаотично розкидані.

Звичайні засоби резервного копіювання дозволяють зберігати користувальницькі дані й самі віртуальні робочі столи.

Втрати даних користувача, пов'язані з використовуваним їм пристроєм ПК, ноутбуком, планшетом, тонким клієнтом, повністю виключаються.

Централізація корпоративних ресурсів і керування ПК починає виходити на перший план. З моменту минулого дослідження, цей пункт набрав 5%.

**5. Скільки робочих столів, на Вашу думку, мало б зміст віртуалізувати у вашій компанії?**

– Менш 50 – 20%.

– Від 50 до 100 – 33%.

– Від 100 до 200 – 23%.

– Від 200 до 500 – 15%.

– Більше 500 – 9%.

Цифри не показові й, що скоріше говорять про розмір компанії, але все-таки 71% опитаних уважають, що віртуалізувати має сенс від 50 до 500робочих столів.

**6. Як би Ви характеризували підрозділ вашої компанії, що має сенс у першу чергу переводити на віртуальні робочі столи?**

– Тимчасові/змінні співробітники (наприклад, оператори центра обробки викликів або телемаркетинг) – 47%.

– Співробітники, залучені в процес продажів і з різною періодичністю які зустрічаються із клієнтами в офісі або за його межами – 41%.

– Допоміжний персонал, бек-офіс – 47%.

– Виробничі, наукові, інженерні підрозділи – 41%.

**47%** опитаних уважає, що в першу чергу треба віртуалізувати тимчасових/змінних співробітників (наприклад, оператори центра обробки викликів або телемаркетинг, студентів і т.д.

Від себе додамо, що для цих категорій користувачів дуже добре підійдуть float робітники столи й persona management).

Хоча в цьому питанні спостерігається рідкісна єдність і кожному представлених варіантів відповідей: менеджери на виїзді, бек-офіс, і, увага, виробничі/наукові підрозділи, було віддано більше 40%голосів.

На наш погляд, виробничі, наукові й інженерні підрозділи можуть вимагати високої якості й швидкості відображення віртуального робочого стола, тому ми наполягаємо на застосуванні в таких підрозділів нульових клієнтів на базі Teradici, наприклад PCoIP клієнт для VMware Horizon 7| Купити клієнт VMware.

**7. До якої сфери діяльності ставиться Ваша компанія?**

– Виробництво – 21%.

– Торгівля – 24%.

– Дослідницька діяльність і наука – 6%.

– Утворення – 8%.

– Державна структура – 16%.

– Фінанси – 21%.

– Страхування – 2%.

– Охорона здоров'я – 2%.

– Видобуток або переробка – 2%.

Тут варто відзначити галузі аутсайдери: страхування, охорона здоров'я, видобуток/переробка.

У порівнянні із західними показниками перші дві повинні бути в лідерах.

Причини низького інтересу з боку охорони здоров'я ясні, але страхування, а вуж тим більше видобуток/переробка – це, мабуть, тема для окремого дослідження.

**8. Чи плануєте Ви використовувати VDI?**

– Ні, у найближчій перспективі не плануємо – 3%.

– Плануємо провести пилотний проект – 36%.

– Хотіли б впровадити в найближчі 1-3 роки – 15%.

– Плануємо впроваджувати в цьому або наступному році – 33%.

– Уже використовується – 12%.

Варто відзначити, що VDI уже використовується в 12% опитаних компаній.

**9. Якщо Ви вже використовуєте або використовували VDI, які недоліки є в рішення VDI?**

– Первісні витрати – 81%.

– Продуктивність – 22%.

– Візуальний дискомфорт – 31%.

**81%** опитаних відзначили високі первісні витрати на вровадження VDI.

С іншої сторони, впровадження VDI найкраще починати в той момент, коли назріває заміна комп'ютерного парку.

А готуватися до впровадження VDI потрібно вже зараз, тобто заздалегідь (ми з підтримаємо Вас у цьому шляхетному починанні).

У цьому випадку «високі» первісні витрати перетворюються в «порівнянні», а в деяких випадках в «низькі».

**10. Які основні проблеми при керуванні фізичними ПК?**

– Різнорідність комп'ютерного парку (різне залізо, різні операційні системи) – 85%.

– Необхідність постійного відстеження версій службових програм (наприклад, антивірусів або патчів ОС) – 63%.

– Необхідність використання різноманітних користувальницьких застосунків, відновлення й конфлікти версій і релізів – 63%.

– Облік ліцензійного ПЗ – 73%.

**85%** уважають самим хворим місцем різнорідність комп'ютерного парку (різне залізо, різні операційні системи).

Інші пункти також є проблемними й не сильно уступають «лідерові».

**11. Який середній строк використання ПК у Вашій компанії?**

– Менш 2-х років – 5%.

– 2-3 роки – 18%.

– 4-5 років – 58%.

– Більше 5 років – 18%.

**12. Який середній строк використання серверів і систем зберігання у Вашій компанії?**

– Менш 3-х років – 11%.

– 3-4 роки – 27%.

– 5-6 років – 41%.

– Більше 6-ти років – 21%.

Сервера/СЗД у середньому служать трохи довше. Наприклад, у переважній більшості випадків, термін служби ПК становить 4-5 років.

А сервера й СЗД можуть експлуатуватися й 6 і 7 років.

Оскільки віртуальні робочі столи базуються на серверах, а не на ПК, те й апгрейд для них необхідний у середньому на 2 роки рідше. Вдобавок до пункту 2, де ми визначилися, що й прикінцеві пристрої (тонкі клієнти) заміняються істотно рідше, одержуємо, що за період в 15 років використання звичайних ПК компанія зробить 3 апгрейда, а при використанні віртуальних робочих столів – усього 2.

Ми знаходимо цю перевагу дуже важливим при обґрунтуванні впровадження VDI.

**13. З якої причини ПК і сервера виводяться з використання у Вашій компанії?**

– Вихід з ладу – 35%.

– Зниження продуктивності або відсутність підтримки нових операційних систем – 46%.

– Корпоративна політика – 18%.

**46%** опитаних виводять ПК із використання через зниження продуктивності або відсутності підтримки нових операційних систем.

Згодом персональні комп'ютери «засмічуються» тимчасовими даними й записами реєстру, що знижує їхня продуктивність.

У віртуальних робочих столах це недолік зведений до мінімуму.

Тимчасові файли можуть перебувати на тимчасових диска, а періодичні відновлення користувальницьких робочих столів дозволяє акуратно встановлювати нові застосунки й стежити за їхньою кількістю і якістю.

Тонкі й нульові клієнти позбавлені цих недоліків. Таке устаткування можна використовувати «до кінця».

У випадку виходу з ладу тонкого або нульового клієнта не відбудеться нічого страшного, необхідно буде тільки поміняти один пристрій доступу на інше.

Ніякого перенесення даних і втрати часу.

**14. Яким чином організована робота мобільних/віддалених користувачів?**

– Такі користувачі не повинні мати доступ до корпоративних ресурсів (наприклад, по політиці безпеки) – 12%.

– Користувачі не мають доступ до корпоративних ресурсів, хоча це могло б бути корисним – 20%.

– Користувачі мають доступ до обмеженого набору корпоративних ресурсів, наприклад: пошта, внутрішній портал – 45%.

– Користувачі мають доступ до повної функціональності. До всіх застосунків, папкам і т.д. – 16%.

– Користувачі мають доступ до повної функціональності, а в моменти відсутності зв'язку можуть продовжувати роботу в автономному режимі. Після появи зв'язку дані синхронізуються – 7%.

**45%** опитаних забезпечують користувачам доступ до обмеженого набору корпоративних ресурсів, наприклад: пошта, внутрішній портал.

7% опитаних забезпечують доступ користувачів до повної функціональності, а в моменти відсутності зв'язку користувачі можуть працювати в автономному режимі.

Після появи зв'язку дані синхронізуються.

Це функціональність VMware Horizon Local Mode.

**15. Яким образом, Ви вважаєте, було б ефективно забезпечити доступ до віртуальних робочих столів?**

– За допомогою апаратних тонких клієнтів, що базуються на операційних системах, таких як Linux, Windows Embedded і т.д. – 63%.

– За допомогою нульових клієнтів, що базуються на технології Teradici PCoIP – 53%.

– За допомогою нульових клієнтів, що базуються на FPGA і працюють по власних протоколах – 11%.

– За допомогою програмних клієнтів, установлених на ПК загального призначення й що дозволяють перетворити ПК у тонкий клієнт – 48%.

**63%** уважають, що апаратні тонкі клієнти, що базуються на операційних системах або нульових клієнтах – оптимальний вибір для роботи з VMWare Horizon 7.

Однак варто відзначити, що 48% у той же час готові «переобладнати» використовувані ПК у тонкі клієнти.

Для цього переозброєння може використовуватися Vmware Horizon Client для Linux/Windows або комплексні продукти, такі як Wyse PC Extender.

Ми рекомендуємо використовувати нульові клієнти на базі Teradici PCoIP.

Ціна на них трохи вище, ніж на тонкі клієнти, але при цьому якість і швидкість роботи з віртуальним робочим столом значно краще.

Якщо в нульових клієнтів ця якість порівнянна зі звичайними ПК, то тонкі клієнти помітно гальмують роботу.

## 2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

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

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

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

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

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

**Переваги та недоліки Python**

Переваги:

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

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

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

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

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

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

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

Недоліки:

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

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

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

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

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

Застосування Python:

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

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

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

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

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

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

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

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

Яким фахівцям потрібно володіти Python:

– Фахівець з даних.

– Аналітик даних.

– Інженер даних.

– Інженер з машинного навчання.

– Журналіст даних.

– Архітектор даних.

– Повний стек веб-розробника.

– Backend-розробник.

– DevOps-інженер.

– Інженер-програміст.

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

## 2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

# 3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

## 3.1 Опис функціонування системи

Virtual Desktop Infrastructure (VDI) часто просувається як економічна заміна традиційних настільних систем. Реальна економічна вигода залежить від багатьох умов (наприклад, від масштабу реалізації), тому найчастіше стимулом до впровадження VDI стають інші її переваги – зокрема, підвищення рівня інформаційної безпеки. При використанні віртуальних робочих місць дані залишаються на сервері (на клієнт передається тільки зображення екрана), що перешкоджає їхній крадіжці й витоку. Крім того, на сервері їх простіше захистити від вірусів і дій зловмисників.

Розглянемо 3 можливих варіанти впровадження VDI.

**VDI з підтримкою 3D-графіки**

Завдяки появі таких технологій, як NVIDIA Grid, у віртуальні робочі місця можуть бути трансформовані й графічні робітники станції.

Ідея позбутися від робочих станцій і сконцентрувати обчислювальні ресурси в серверній виникла ще десять років тому. Однак на той момент її практичне втілення було неможливо через технічні обмеження. Апаратна підтримка графіки у віртуальних середовищах тоді була відсутня, тому кожному конструкторові довелося б виділяти окремий сервер. Крім того, оскільки протоколи доступу були недостатні оптимізовані, для «важких» графічних застосунків були потрібні канали з малою затримкою й великою пропускною здатністю.

Відновлення парку робочих місць дало привід повернутися до цієї ідеї, причому вирішальним доводом стали підтримка мобільних робочих місць і забезпечення безпеки даних, що зберігаються на них. Конструкторам замовника доводиться їздити у відрядження по всій країні. Варіант із високопродуктивними

мобільними робочими станціями був відкинутий, оскільки професійні ноутбуки важать 4 кг і більше – брати їх із собою незручно, та й коштують вони недешево. До того ж відкритим залишалося питання забезпечення безпеки даних на випадок поломки, крадіжки або втрати.

Як рішення запропонували систему 3D VDI з використанням віртуальних графічних процесорів на базі технології NVIDIA Grid для візуалізації графіки. Це дозволило забезпечити підтримку мобільних робочих місць і вирішити проблему ІБ: на клієнт передаються не дані, а тільки зображення екрана. Стаціонарні робочі місця теж поступово будуть замінені на тонкі клієнти.

Необхідною умовою для будь-якого впровадження VDI є збереження колишнього рівня продуктивності (або, у найгіршому разі, незначне її зниження). Однак удалося не тільки уникнути втрат, але й домогтися поліпшення показників. Раніше на початку кожного робочого дня конструкторам доводилося близько 40 хв чекати завантаження складання моделі, тепер ця процедура займає 10 хв. Ця стало можливим завдяки тому, що інфраструктура VDI і високопродуктивні кластери, на яких складання зберігаються й обробляються, перебувають у тому самому центрі обробки даних. Пропускна здатність каналів між віртуальними машинами з робочим місцем VDI і кластером зросла багаторазово: з колишніх 100 Мбіт/с – 1 Гбіт/с до 20 Гбіт/с для кожного сервера.

Ще одним цікавим моментом є реалізація програмно-визначаємого сховища. До використання SDS для підтримки бізнес-критичних завдань багато хто дотепер ставляться зі сторожкістю, а на момент старту проекту подібні впровадження ще тільки починалися. Всі сервери, на яких була розгорнута VDI, об'єднали в кластер високої доступності VMware і створили на його основі програмно-визначаєме сховище в vSAN.

Було протестовано програмне сховище на демостенді. Його використання дозволило вирішити проблему затримок при початковому завантаженні (boot-шторм). У випадку програмних сховищ дана проблем вирішується на рівні архітектури завдяки об'єднанню декількох серверів. VMware vSAN входить до

складу Horizon, тому додаткових ліцензій здобувати не довелося – досить було додати диски в сервери.

На VDI переведені й офісні користувачі. Кластер, на якому розгорнуті віртуальні робочі місця, складається з восьми серверів. Два з них мають по двох відеокарти й служать для підтримки робочих місць конструкторів, інші призначені для офісних користувачів. Обслуговуванням всієї системи займаються два чоловіки.

**VDI на велику кількість робочих місць**

На момент ухвалення рішення про централізацію IT-устаткування, установлене в регіональних офісах, по більшій частині вже застаріло, і було потрібно замінити колосальна кількість пристроїв. Після проведення фінансового аналізу стало ясно, що централізація інфраструктури у двох катастрофостійких центрах обробки даних у Києві й переведення співробітників на віртуальні робочі місця набагато вигідніше, ніж повномасштабне відновлення комп'ютерного парку. Як показало подальший розвиток подій, рішення виявилося вірним: проект окупився в перший же рік.

Офіси банку перебувають по всій країні – від Ужгорода до міст Кіровоградської області, так що забезпечення необхідної продуктивності стало непростим технічним завданням. За підсумками пілотного проекту було обране рішення Citrix, оскільки використовувані їм протоколи виявилися «краще оптимізовані під реалії». Проте для забезпечення нормальної роботи співробітників потрібні були й подальше налаштування протоколу, і відновлення каналів зв'язку.

Кругова затримка пакета, наприклад між Києвом і Кропивницьким, становить порядку 12 мс – це майже непомітно для користувачів. Офіси на міст Кіровоградської області підключалися по супутниковому зв'язку, а затримка досягала 40 мс, але навіть у цих умовах можна було працювати с віртуальним десктопом (тепер до міст Кіровоградської області прокладене оптичне волокно). Втім для нормальної роботи VDI важлива не стільки затримка, скільки

стабільність параметрів каналу. Приміром, при використанні каналів одного з операторів користувачі скаржилися на зависання сеансів, при цьому послу перемикання на резервний канал усе працювало нормально.

Безумовно, сам протокол теж довелося оптимізувати – зокрема, налаштувати політики відображення, щоб і якість зображення бути прийнятним для роботи, і навантаження на канал мінімальної. Так, частота кадрів, які відображаються на екрані, була скорочена до 15. На роботу із простими офісними програмами це ніяк не вплинуло, а от вимоги до каналу вдалося скоротити у два рази. У результаті для роботи зі звичайними офісними застосунками, наприклад Excel, тепер досить пропускної здатності 250 Кбіт/с на один користувача.

Адаптація потрібна була не тільки для протоколів, але й для деяких застосунків. Так, під час запуску Office 2010 відображається екранна заставка (splashscreen), з появою якої утилізація зростала до 3 Мбіт/с, тому подібні графічні ефекти довелося відключити. Схожа проблема виникла з рідером PDF від ABBEY: при прокручуванні сторінок він перемальовував не тільки чорні пікселі, але й білі, що невиправдано збільшувало навантаження на канал. Для рішення цієї проблеми довелося залучати розроблювача.

Інфраструктура VDI розділена між двома центрами обробки даних, усього для підтримки 14 тис. віртуальних робочих столів задіється більше півтори сотень серверів Huawei RH1288 V3 і чотири системи зберігання Hitachi VSP G600 середнього класу. Обчислювальні потужності розбиті на модулі, кожний з яких складається із двох кластерів по 12 серверів і розрахований на підтримку 2000 користувачів. Один модуль реалізований у вигляді територіально розподіленого кластера й забезпечує підтримку користувачів навіть при повному виході з ладу одного із ЦОД (час перемикання – 15 хв).

Впровадження VDI дозволило спростити обслуговування кінцевих систем. Так, відновлення 14 тис. робочих місць займає всього два вечори (це робиться в неробочий час, щоб не переривати робочий процес), а основна частина цього часу йде на попереднє тестування.

**Просто VDI**

Як відзначалося, головна вигода від впровадження VDI не обов'язково фінансова. Так, корпорації було потрібно забезпечити швидке розгортання нових робочих місць. Попутно впровадження VDI дозволило спростити їхню атестацію на відповідність правилам роботи з конфіденційною інформацією – це виявилося менш працезатратно й помітно дешевше, ніж атестація окремих комп'ютерів.

Якщо два розглянутих вище проекти базуються на рішеннях VMware і Citrix, багаторічних безумовних лідерів цього сегмента, то для даного проекту було обране рішення китайської компанії Huawei. Воно реалізує весь необхідний функціонал за менші гроші.

У якості одного з можливих варіантів розглядалися системи на базі Open Source, вибрати їх не ризикнули, оскільки вимоги до відказостійкости бізнесу високі, а серйозного досвіду впровадження таких рішень немає. Проектів VDI на базі устаткування Huawei на українському ринку теж поки небагато – навіть не вся документація переведена на англійський, не говорячи про українську, але фахівці уже мали досвід співробітництва з вендором і були впевнені в тому, що вся необхідна підтримка їм буде зроблена: зокрема, проблеми із пробросом USB-пристроїв при віддаленому доступі були дозволені оперативно.

Проект обійшовся не дешевше, ніж розгортання звичайних робочих місць. Однак якщо раніше на підготовку робочого місця йшло кілька годин, причому найчастіше IT-фахівці займалися цим у невизначений час, те тепер досить 10-15 хв. Для реалізації одного робочого місця вистачило скромних ресурсів: 2 vCPU, 4 Гбайт оперативної пам'яті й 60 Гбайт на диску (з обліком ОС і застосунків). На першому етапі проекту всі віртуальні машини зберігалися у вигляді повної копії. На наступному минулому реалізовані зв'язані клони (linked clone), що дозволило закупити менший обсяг дискового простору, чим при першому впровадженні.

У цей час на VDI переведено 150 користувачів, надалі кількість віртуальних робочих столів планується подвоїти.

**Три плюси, один мінус**

Незважаючи на різний масштаб і різні завдання, у цих трьох проектів є ряд загальних рис, хоча й не все з них можна назвати позитивними.

По-перше, приводом до реалізації проекту VDI стала потреба відновлення застарілих робочих місць.

По-друге, впровадження VDI дозволило спростити обслуговування кінцевих робочих місць, прискорити їхнє розгортання й оптимізувати штат IT-персоналу.

По-третє, для всіх замовників немаловажним доводом на користь VDI виявилося підвищення рівня захисту даних.

Однак жоден із проектів не обійшовся без технічних проблем, що, втім, свідчить не стільки про неготовність рішення VDI до практичного використання, скільки про недостатню пристосованість – як і раніше – існуючої інфраструктури й застосунків до інших обчислювальних моделей.

## 3.2 Розробка структурної схеми

Інфраструктура віртуальних робочих столів Virtual Desktop Infrastructure (VDI) – дуже актуальна тема в наші дні. VDI – це форма віртуалізації настільних систем, у якій всі елементи робочого стола користувача розміщені в центрі обробки даних. Користувач віддалено підключається до свого робочого стола з якого-небудь клієнтського пристрою; при цьому ні робочий стіл користувача, ні застосунки, ні дані на його пристрою локально не зберігаються

**Робоче середовище настільної системи**

Найбільш важлива сторона використання комп'ютера – застосунку, які виконують певні функції й обробляють дані, як локальні, так і зберігаються на серверах. Операційна система – основний інструмент для виконання застосунків і керування даними. Користувачі часто налаштовують своє робоче середовище за допомогою спеціальних фонових рисунків, екранних заставок, ярликів і посилань

на вибрані веб-сторінки. Хоча ці налаштування адміністраторові можуть здатися не варті уваги, користувачі можуть годинами шукати загублені застосунки або дані або повторно створювати свої ярлики. Таким чином, підтримка робочого середовища користувачів дуже важлива.

Три головні частини, що утворять налаштування робочого середовища: користувальницькі дані й налаштування, застосунки й операційна система. Устаткування, на якому встановлена операційна система, також має величезне значення.

У більшості робітничих середовищ ці компоненти тісно переплетені один з одним. Операційна система встановлюється локально на настільному комп'ютері користувача; застосунки встановлюються безпосередньо в операційну систему, при цьому вносяться зміни у файлову систему, реєстр і інші системні компоненти; користувальницькі дані й налаштування зберігаються в локальній файловій системі. Оскільки ми звичайно встановлюємо застосунки в системі, користувач налаштовує операційну систему й застосунки, і в користувача є дані, хоча насправді як застосунку, так і користувальницькі дані й налаштування прив'язані до рівня операційної системи.

Таке тісне зв'язування створює ряд проблем:

– Через зберігання даних винятково на клієнтському комп'ютері виникає ризик їхньої втрати у зв'язку з виходом з ладу або збоєм устаткування, ушкодженням диска або ненавмисним видаленням. Крім того, локальні дані будуть недоступні з іншого комп'ютера. Ці ж проблеми ставляться й до користувальницьких налаштувань і системи.

– Збій у роботі операційної системи або устаткування комп'ютера вимагає складних процедур для відновлення інформації й налаштувань. Необхідно мати в наявності списки встановлених застосунків до того, як буде замінені устаткування або із система, після чого всі застосунки також повинні бути з.

– Збій у роботі застосунку приводить до складних процесів пошуку несправностей і їхнього усунення, тому що необхідно внести зміни в різних частинах системи.

– Розгортання застосунків і їхніх відновлень може бути дуже складного й потребуючого тривалого часу.

Рішення всіх перерахованих проблем полягає у віртуалізації кожного з рівнів, що робить їхній незалежними друг від друга. Це рішення формує гнучке середовище, що легко розвертати, підтримувати й можна зробити доступної з будь-якого комп'ютера.

Бажання розділити елементи системи для підвищення її гнучкості досить загальне: у багатьох продуктах використовується модульний принцип побудови. Того ж самого ми хочемо для користувальницьких систем – щоб операційна система, застосунки й користувальницькі налаштування були окремими блоками, які збираються в єдине ціле залежно від середовища користувача, що зареєструвався в системі. Цим середовищем може бути локальний робочий стіл, операційна система в середовищі VDI, сесія в службі терміналів або службі віддалених робочих столів. Оскільки ці компоненти є окремими блоками, відсутня затримка, пов'язана з очікуванням їхньої установки або налаштування. Компоненти розташовані шарами друг над іншому для створення повнофункціонального середовища настільної системи.

У даному розділі ми розглянемо технології, які дозволяють розділяти ці звичайно що сильно переплітаються рівні для складання «на лету» конструкції, що створює завершене робоче середовище настільної системи незалежно від того, де зареєструвався користувач.

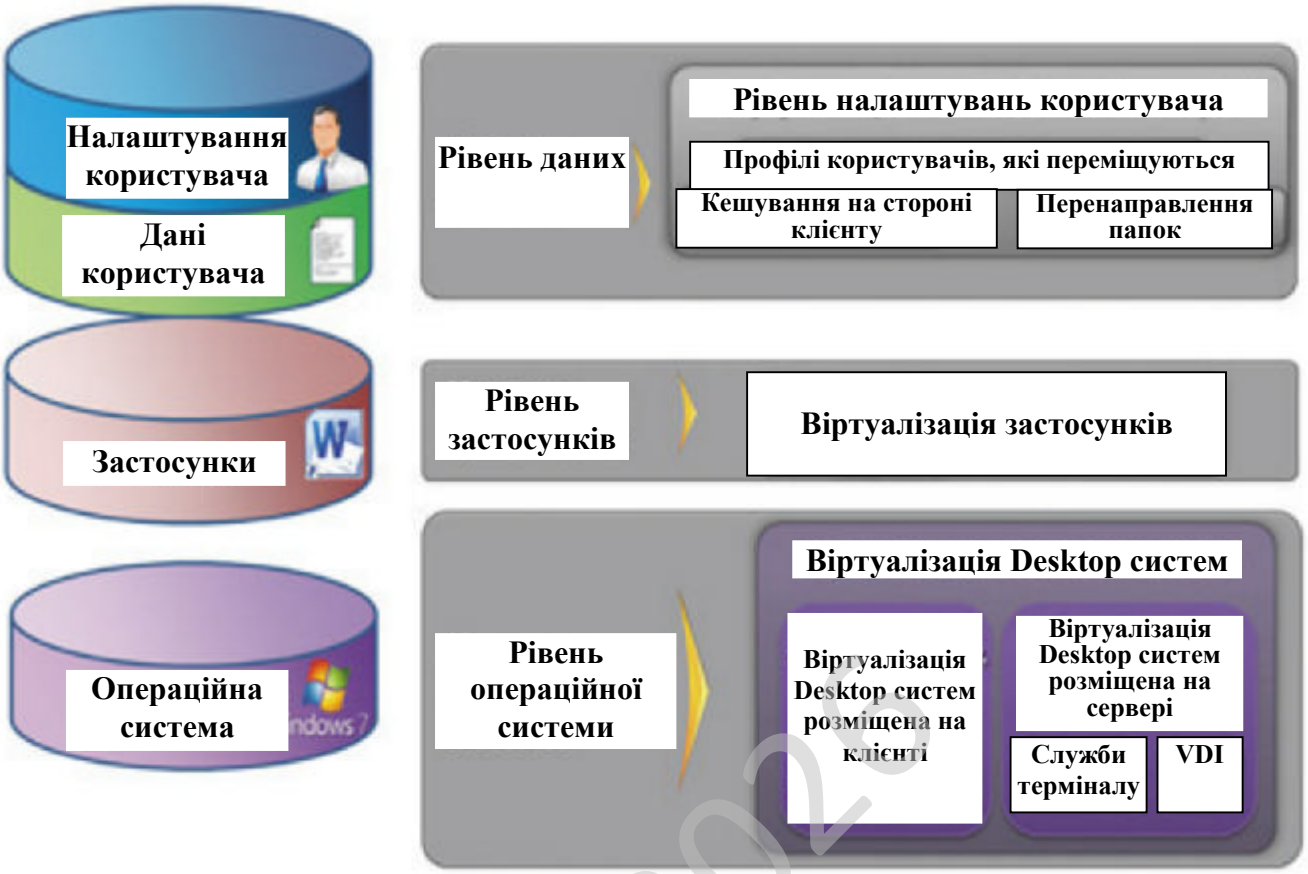


Рисунок 3.1 – Структурна схема системи

**Користувальницькі дані й налаштування**

Ціль складається в добуванні налаштувань і даних користувача з робочого середовища, так щоб ці налаштування й дані були захищені й доступні незалежно від точки підключення користувача до мережі – наприклад, його настільний комп'ютер, настільний комп'ютер іншого співробітника, віддалена сесія в системі віртуалізації подань, такий як служби терміналів (служби віддаленого робочого стола), система XenDesktop від Citrix або клієнтська операційна система в інфраструктурі VDI. У системі Windows уже давно існують технології для абстрагування користувальницьких налаштувань і даних, але в Windows 10/11 ці технології розвинені до такого рівня, що вони не роблять негативного впливу на середовище користувача й навіть підвищують доступність налаштувань і даних.

Спочатку розглянемо налаштування користувача. У кожного користувача є на комп'ютері профіль, що звичайно зберігається в системі Windows Vista і більше пізніх версіях у папці C:\Users. Цей профіль складається з файлів і папок, у тому числі файлу ntuser.dat, що містить специфічну для користувача інформацію реєстру. Хоча цей файл невеликий, він зберігає масу налаштувань, зроблених користувачем. Профіль також містить посилання «Вибране» браузера Інтернету, документи, результати пошукових запитів і інші види інформації. Обговоримо їх трохи нижче.

Для формування однакового середовища профіль користувача повинен бути доступний незалежно від місця підключення користувача до мережі. Таким чином, профіль повинен зберігатися в мережі. Ця можливість, називана «переміщуваний профіль», уже давно доступний в Windows.

У минулому компанії неохоче застосовували переміщувані профілі через незручність застосування. При завершенні сеансу роботи із системою профіль користувача просто вивантажувався на мережевий ресурс, що приводило до тривалих затримок у процесі завершення. В Windows 10/11 реалізована фонова синхронізація переміщуваних профілів. Ця можливість за замовчуванням відключена, але її можна включити за допомогою групових політик. Процес фонової синхронізації зберігає профіль користувача в певні моменти часу або через певний інтервал. Періодична синхронізація даних означає, що буде потрібно менше часу для завершення сеансу.

Інша причина того, що переміщувані профілі можуть викликати проблеми, полягає в тому, що до Windows 10/11 користувальницькі дані були частиною профілю, а це приводило до утворення дуже більших профілів. У дійсності переміщувані профілі не розроблялися з метою реплікації даних користувачів. Навіть із удосконаленням Windows 10/11 користувальницькі дані повинні бути відділені від профілів – це приводить до питання перенапрямку папок.

У користувачів є кілька місць для зберігання даних, наприклад папки «Документи» і «Картинки», які за замовчуванням є підпапками

користувальницького профілю. Якщо використовуються переміщувані профілі, ці папки й ті, що містяться в них дані реплікуються як частина загального процесу реплікації профілю, що, як ми встановили, не є оптимальним. Перенапрямок папок – це альтернативна технологія, що дозволяє налаштувати розміщення стандартних папок у мережі. Наприклад, коли користувач відкриває папку «Документи», він у дійсності відкриває мережевий ресурс, хоча користувач цього навіть не зауважує. Перенапрямок папок налаштовується за допомогою групових політик.

Функція «Автономні файли» (також відома як кешовані на стороні клієнта) дозволяє зберігати локальну копію мережевих даних користувача на спеціально настроєних комп'ютерах, для яких потрібен доступ до даних навіть при відключенні від мережі (наприклад, ноутбуки). Автономні файли синхронізують зміни в даних на рівні змін у файлах при відновленні роботи мережі. Така дельта-реплікація означає, що реплікуються тільки змінені частини файлів, а не файли цілком.

**Застосунки**

Коли ми налаштовуємо комп'ютер, першу дію, що ми виконуємо після установки операційної системи й відновлень до неї, – це установка застосунків, таких як Microsoft Office, бізнес-застосунки, модулі безпеки й інші види програм. Установка програм звичайно займає досить багато часу, тому що розгортання й налаштування вимагають відновлень у файловій системі, додавання параметрів реєстру й реєстрації ресурсів. Як правило, тільки застосунки масштабу організації встановлюються разом з розгортанням операційної системи. Застосунки, специфічні для підрозділів і окремих користувачів, можуть бути встановлені при першій реєстрації в системі, що вносить додаткові затримки.

На застосунок до часових затримок у процесі установки можуть виникнути наступні проблеми.

– **Сумісність застосунків між собою.** Зміни, внесені в операційну систему застосунками, можуть виявитися несумісними з іншими застосунками й

навіть із різними версіями тих же самих застосунків. Перед розгортанням у виробничому середовищі нового або оновленого застосунку необхідне проведення регресійного тестування, при цьому деякі комбінації застосунків можуть виявитися неможливими.

– **«Розбухання» операційної системи.** При установці кожного нового застосунку додаються нові служби, які витрачають ресурси, але не завжди реалізують корисні функції. До того ж збільшується розмір реєстру, що використовує пам'ять і приводить до з роботи системи. Навіть якщо згодом застосунки віддаляються, їхні залишки засмічують систему.

– **Відновлення застосунків.** Процеси відновлення застосунків можуть сильно відрізнятися, і це вимагає значних зусиль при плануванні й створенні інфраструктури для розгортання.

Всі ці проблеми ставляться тільки до випадку, коли застосунки встановлюються в операційній системі. Представимо, що користувачі реєструються в системі на різних комп'ютерах, у віддалених сесіях, у середовищі VDI – і їм усім потрібні різні застосунки. Установка всіх застосунків, які коли-або можуть знадобитися користувачам, на всіх розгорнутих операційних системах просто непрактична й може привести до непомірно набряклих операційних систем, підтримка яких може доставити чимало турбот, тому що відновлення будь-якого застосунку повинне бути застосоване до кожного екземпляра встановленої операційної системи. Використання традиційних методів установки програм при реєстрації користувачів у різні середовища практично неможливо через тимчасові затримки, не говорячи вже про додаткові проблеми видалення застосунків при завершенні роботи користувача.

Одне з рішень – це традиційна віртуалізація подань, у якій застосунки встановлюються на термінальних серверах, а потім виконуються на цих серверах, у те час як вікно застосунку відображається на локальному робочому столі користувача. Це рішення вимагає солідної серверної інфраструктури для розміщення застосунків, що виконуються, і не допускає автономного запуску

застосунків. Крім того, як і раніше залишається необхідність установки всіх застосунків на декількох термінальних серверах. Проте це рішення може підійти для деяких застосунків, наприклад таких, котрим потрібний доступ до більших обсягів даних, розміщених у центрі обробки даних. Коли такий застосунок працює в центрі обробки даних у середовищі віртуалізації подання, мережевий трафік при доступі до даних обмежений мережею центра обробки даних, звичайно дуже швидкої. Запуск того ж застосунку локально на комп'ютері користувача вимагає передачі всіх даних по мережі, що використовує значну частину смуги пропущення й сповільнює роботу застосунку.

Ще одне ефективне рішення для застосунків – це віртуалізація. Істотне розходження між віртуалізацією застосунків і віртуалізацією подань полягає в тому, що в першому випадку застосунку в дійсності виконуються в локальному екземплярі операційної системи. І, що більше важливо, без необхідності установки в локальній операційній системі, завдяки тому що середовище віртуалізації дозволяє застосункам працювати без внесення змін у локальну операційну систему.

Згадаємо, що при установці застосунків вносяться зміни у файлову систему (тобто в папку C:\Program Files містяться виконуються файли, що, застосунку й бібліотеки DLL, вносяться зміни до реєстру, установлюються служби). Віртуалізація застосунків здійснюється шляхом перехоплення всіх цих змін у системі за допомогою особливого процесу віртуалізації, серіалізації, (sequencing, термін з Microsoft App-V), що здійснює перетворення процедури установки застосунку у двійковий потік, що використовується при віртуалізації застосунку. Ці зміни зберігаються на різних віртуальних рівнях, таких як рівень файлової системи, рівень реєстру, рівень служб, шрифтів, компонентів OLE і конфігурації, які потім можуть бути завантажені в єдине віртуальне середовище при запуску застосунку. Для самого застосунку всі ці файли, реєстр і служби виглядають як частина локально встановленої операційної системи, але в дійсності застосунок має справа з віртуальними рівнями, які йому надає

середовище віртуалізації застосунків. Ніяких змін у локально встановлену систему не вноситься.

Помітимо, що, хоча застосунки не можуть записувати в операційну систему нічого, крім користувальницьких налаштувань і даних, вони можуть зчитувати інформацію з локальної системи.

App-V – це рішення Microsoft з віртуалізації застосунків. За замовчуванням App-V працює як потокова технологія. При першому запуску віртуалізованного застосунку клієнт App-V підключається до потокового сервера App-V, що посилає клієнтові частина двійкового потоку застосунку, необхідну для початкового запуску. Ця частина двійкового потоку називається основним функціональним блоком (Feature Block 1) і звичайно становить від 10 до 20% загального двійкового потоку. Дана частина доставляється клієнтові дуже швидко – для Office 365 інтервал між клацанням користувача по значку застосунку й запуском вікна застосунку становить біля трьох секунд.

Процес віртуалізації застосунку в App-V визначає, що необхідно помістити в основний функціональний блок, за допомогою дійсного запуску застосунку при його віртуалізації на етапі серіалізації (sequencing) і моніторингу необхідних частин потоку. Всі необхідні елементи містяться в основний функціональний блок, а частина, що залишилася, – у додатковий функціональний блок (Feature Block 2). Після запуску застосунку додатковий функціональний блок передається клієнтові у фоновому режимі. Двійковий потік кешується в локальній клієнтській операційній системі, таким чином, якщо застосунок запускається знову, потік повторно по мережі не передається.

Хоча я згадав, що віртуалізація застосунку не вносить змін у локальну операційну систему, мабуть, що вона кешує у системі потік даних. Однак потік кешується в одному файлі в профілі All Users, і цей єдиний кеш-файл розділяється між всіма користувачами й застосунками. Застосунку більше нічого не записують у файлову систему й не вносять ніяких змін у налаштування системи або до реєстру. Цей кеш також робить віртуалізований застосунок

доступним, навіть якщо комп'ютер не в мережі. У середовищі VDI ми можемо розмістити даний кеш на мережевому ресурсі, щоб він був загальним для всіх клієнтських віртуальних машин середовища VDI. Такий підхід рятує від необхідності мати для кожного клієнта власний кеш App-V, що дозволяє заощаджувати місце на диску й прискорює початковий запуск застосунку. Потік – це спосіб доставки застосунків у технології App-V (за замовчуванням). Однак App-V підтримує також створення файлу MSI, що містить повний потік для інших технологій розгортання, таких як групові політики й Microsoft System Center Configuration Manager (SCCM). Для доставки потоку App-V можна навіть використовувати традиційні мережеві загальні папки й служби Microsoft IIS. Для різних організаційних і інфраструктурних завдань доступна безліч варіантів. Помітимо, що такі базові операції, як вирізання й вставка, зв'язування й впровадження об'єктів (OLE), як і раніше працюють між віртуалізованними застосунками, але для більше глибокої інтеграції можна створити залежності між ними. Так зване формування динамічних пакетів Dynamic Suite Composition дозволяє окремим віртуалізованним застосункам взаємодіяти один з одним.

Віртуалізація застосунків дозволяє вирішити кілька проблем розгортання, приклади наведені нижче:

– Застосунки можуть бути швидко розгорнуті для кожного користувача й у реальному часі, що дозволяє адміністраторам поширювати застосунки на настільні системи так, як потрібно.

– Проблеми сумісності між застосунками вирішені, тому що окремі віртуалізованні застосунки «не бачать» один одного. Застосунку мають власні віртуальні файлові системи, реєстри й т.д. Така ізоляція також рятує від необхідності регресійного тестування при запуску нових або оновлених застосунків.

– Установка відновлень стає дуже простою. Віртуалізованний застосунок обновляється тільки один раз. App-V передає зміни в потік для всіх клієнтів, без яких-небудь дій з боку користувачів.

– Операційна система не розбухає, тому що застосунки не встановлюються в локальній операційній системі.

**Операційна система**

Організації, що здійснюють перехід на Windows 10/11, можуть зштовхнутися із проблемою сумісності застосунків. Якщо немає нової версії застосунку, сумісної з Windows 10/11, або аналогічного продукту, то як варіант можна використовувати віртуалізацію операційної системи. На цей випадок існує кілька підходів.

В Windows 10/11 реалізована технологія, що дозволяє виконувати застосунку, не сумісні з Windows 10/11, у локальній віртуальній машині із системою XP. Вікно застосунку відображається на робочому столі користувача в Windows 10/11, непомітно для користувача. У цьому випадку в середовищі Windows 10/11 виконується окремий екземпляр попередньої версії операційної системи, яким необхідно управляти. Технологія віртуалізації корпоративних настільних систем Microsoft Enterprise Desktop Virtualization (MED-V) від Microsoft, що є частиною пакета оптимізації настільних систем Microsoft Desktop Optimization Pack (MDOP), спрощує цей процес за рахунок як розподіли й відновлення віртуальних машин, так і керування ярликами на робочих столах і посиланнями URL для оглядача Microsoft Edge Такий підхід забезпечує й рішення проблеми сумісності оглядача Microsoft Edge в Windows 10/11.

Ще один підхід до віртуалізації клієнта – віртуалізація всієї настільної операційної системи користувача в центрі обробки даних і надання локальному клієнтові віддаленого підключення до клієнтських операційних систем, розміщеним у центрі обробки даних. Цей локальний клієнт може бути «тонким» пристроєм, що застарів комп'ютером з Windows Fundamentals (операційною системою на основі Windows, розробленої для тих, хто використовує старі комп'ютери й операційні системи), а також будь-яким іншим пристроєм або операційною системою, що підтримує протокол RDP (у випадку однорідного рішення на платформі Microsoft). Достоїнство цього підходу полягає в тому, що, оскільки робоче середовище користувача повністю розміщений у центрі обробки

даних, конфіденційні дані в дійсності ніколи не залишають межі центра обробки даних. Крім того, робітник стіл доступний користувачеві незалежно від того, звідки він підключається до мережі, у тому числі при підключенні з домашнього комп'ютера. Ці рішення також дуже ефективні при плануванні відновлення після збоїв.

**Збираємо всі разом**

Коли користувач уперше реєструється в новій операційній системі, або в локальній настільній системі, або в сесії служб терміналів, або в клієнтській операційній системі, розміщеної в середовищі VDI, виконуються наступні дії.

– Користувач реєструється в системі з обліковим записом служби каталогів Active Directory (AD).

– Після успішної перевірки дійсності створюються ті частини профілю, які не абстраговані за допомогою перенапрямку папок. цей мінімальний обсяг інформації завантажується дуже швидко. Потім на застосунок до перенапрямку папок виробляються налаштування користувача в локальній сесії. Таким чином, користувачеві стають доступні всі його дані, вибрані посилання й т.д.

– Клієнт App-V підключається до сервера керування App-V для визначення набору застосунків, які застосовуються до користувача, що зареєструвався в системі, і створює всі ярлики на робочому столі й у головному меню, а також налаштовує прив'язку системи до відповідних типів файлів.

– Тепер у користувача є повнофункціональне робоче середовище й він може негайно запускати застосунку й використовувати дані.

Віртуалізація настільних систем і інфраструктура VDI – не те саме. VDI підсилює технології віртуалізації настільних систем для можливості розміщення клієнтських операційних систем у центрі обробки даних. Хоча VDI може виявитися найкращим рішенням для деяких компаній, будь-яке настільне середовище може одержати виграш від тої або іншої форми віртуалізації настільних систем – від віртуалізації застосунків, віртуалізації користувальницьких налаштувань або віртуалізації операційної системи. При плануванні архітектури настільних систем, особливо на базі Windows 10/11,

обов'язково розглянете віртуалізацію настільної системи як частина цієї архітектури. Додаткова попередня робота дозволяє одержати переваги в довгостроковій перспективі. Віртуалізація настільних систем не тільки забезпечує динамічне й гнучке середовище, але й скорочує витрати на інфраструктуру й керування.

### 3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема розробленої системи Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури.

З рисунка ми бачимо, що розроблена система функціонально складається з двох основних блоків:

– серверної частини;

– клієнтської частини.

Розглянемо ці функціональні частини більш докладно.

Серверна частина включає в себе наступні модулі. Інтерфейс користувача, який дозволяє користувачу наочно працювати з програмою використовуючи, ті або інші функціональні модулі. До цих модулів відносяться наступні:

– Модуль відслідковування подій на клієнті Virtual Desktop Infrastructure.

– Модуль призначення дій над клієнтом Virtual Desktop Infrastructure та часу їх виконання.

– Модуль перегляду вмісту жорстких дисків клієнту Virtual Desktop Infrastructure.

– Модуль виконаня файлових операцій (завантаження, відправка, видалення файлів).

– Модуль відправки повідомлень на клієнт Virtual Desktop Infrastructure.

Усі вищеперераховані модулі, взаємодіють з однієї сторони з інтерфейсом користувача, а з іншої сторони з модулем відправки інформації запитів на клієнт та обробки відповідей. Цей модуль взаємодіє вже безпосередньо з клієнтською частиною розробленої системи.

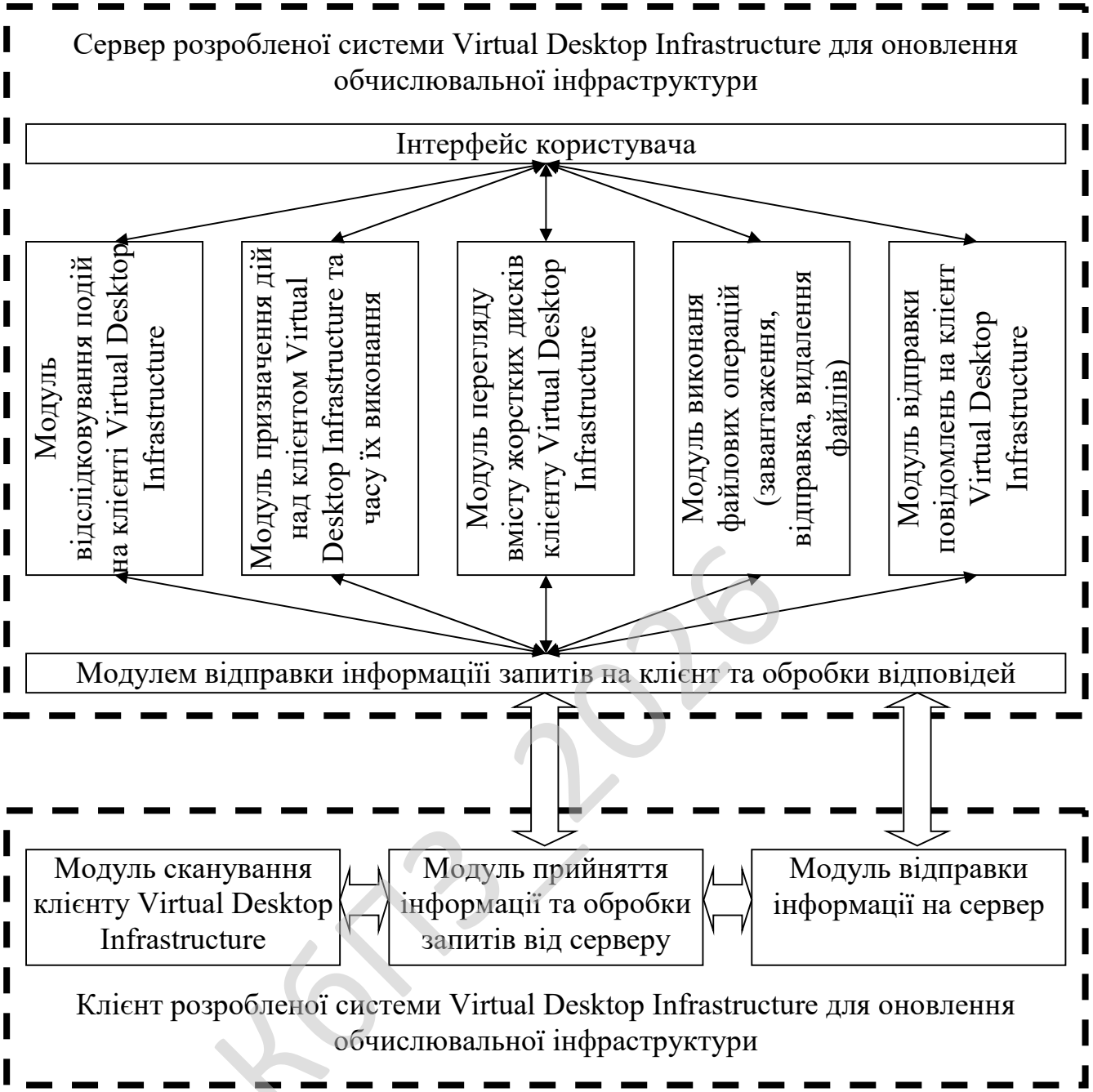




Рисунок 3.2 – Функціональна схема системи

Відповідно перейдемо до розгляду клієнтської частини розробленої програми віддаленого управління комп'ютером через мережу.

Клієнтська частина складається з наступних функціональних блоків:

– Модуль прийняття інформації та обробки запитів від серверу.

– Модуль відправки інформації на сервер.

– Модуль сканування клієнту Virtual Desktop Infrastructure.

Перший та другий модуль безпосередньо взаємодіють з серверною частиною, а точніше з таким її функціональним блоком, як модуль відправки інформації і запитів на клієнт та обробки відповідей.

### 3.4 Розробка діаграми процесів

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

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

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

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

При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі.

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграми потоків даних містять чотири типи елементів:

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

– Сховища даних (репозиторії).


Рисунок 3.3 – Діаграма взаємодії процесів

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 бакалаврської 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.

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

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

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

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

Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи. Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані. Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми.

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

Блок-схеми показують весь процес роботи системи з підсистемами та частково доказують правильність вибраних проектних рішень. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає високого рівня декомпозиції задач на класи.

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.

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних програм і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

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

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

– мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до

серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

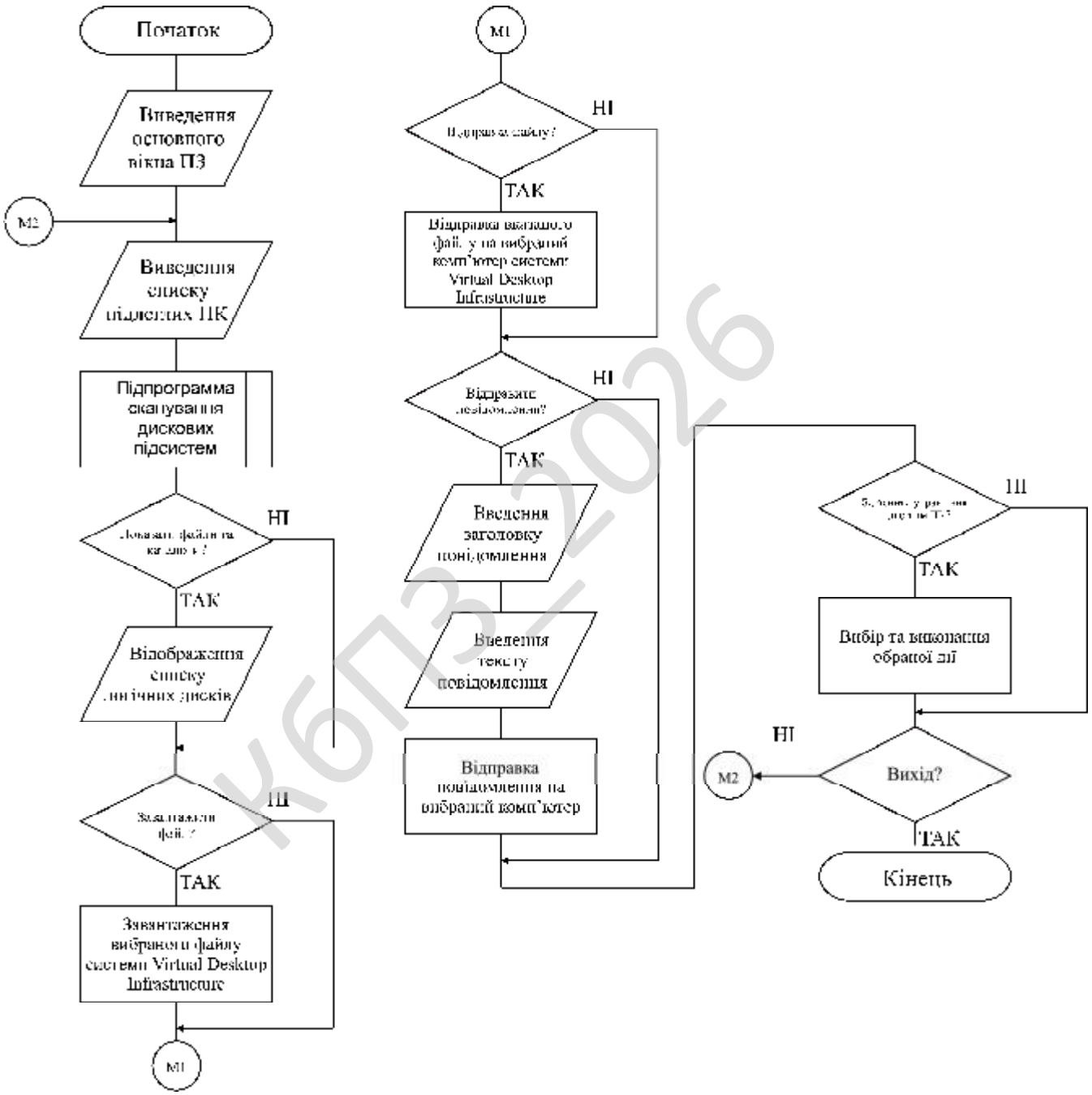


Рисунок 4.1 – Блок-схема основної програми

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

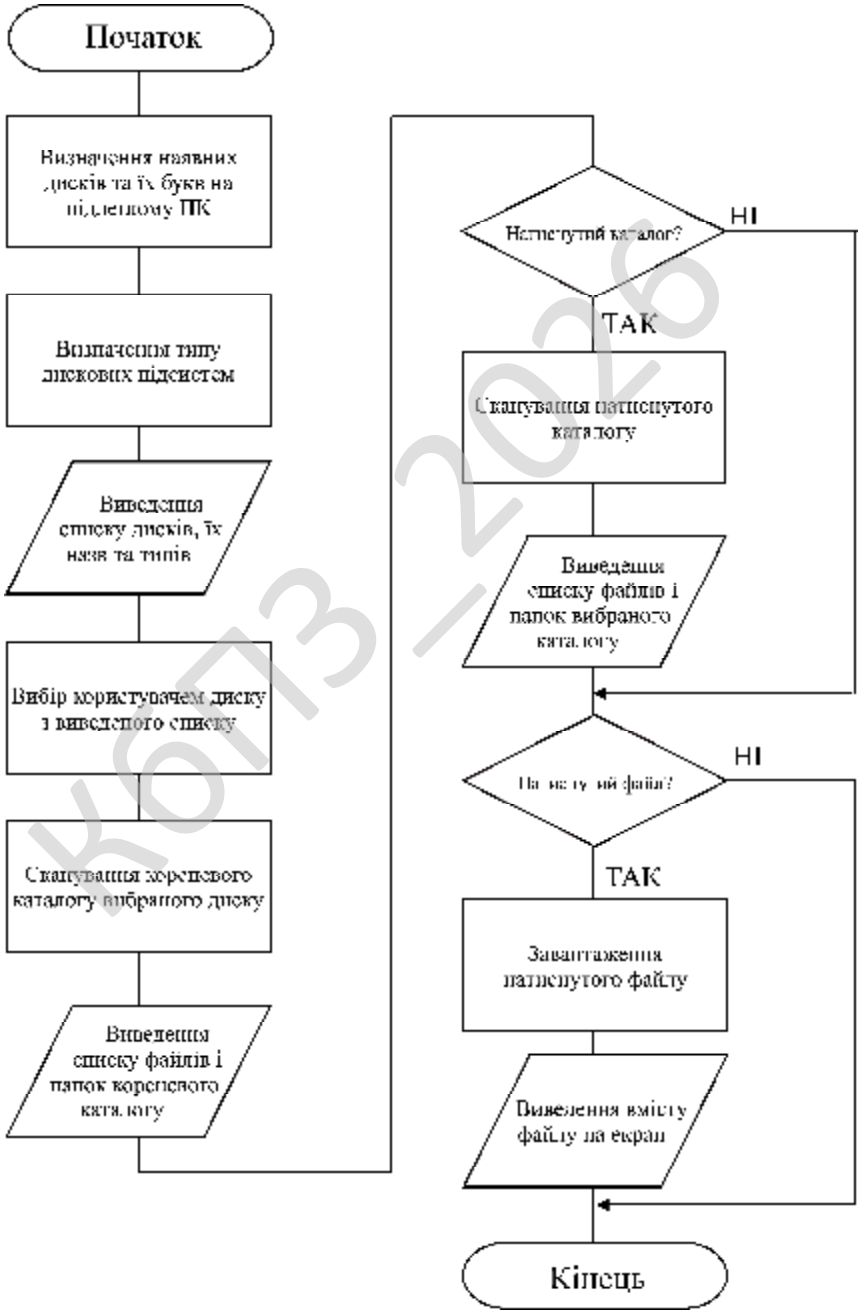


Рисунок 4.2 – Блок-схема роботи підпрограми

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

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

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

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

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

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

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

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

Типовим прикладом клієнт-серверної взаємодії є WWW. Існує величезна кількість веб-серверів, на яких розміщується та чи інша інформація. У найпростішому випадку ця інформація являє собою набір веб-сторінок, які можуть зберігатися на сервері у вигляді файлів, розмічених за допомогою мови

розмітки HTML. Але ситуація, як правило, є складнішою; значна частина веб-ресурсів на сучасному етапі є динамічними, тобто вони не існують в заздалегідь підготовленому вигляді, а створюються безпосередньо в процесі обробки запиту від користувача.

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

Але, крім браузерів, до серверів можуть звертатися і інші клієнти, а саме – автономні програми. Вони можуть передбачати взаємодію з людиною, а можуть працювати в цілком автоматичному режимі. Типовим класом таких програм є роботи, призначені для автоматичного перегляду веб-ресурсів. Зокрема, роботи є важливим елементом пошукових систем і використовуються ними для перегляду сторінок і збору інформації про них.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Прикладний програмний інтерфейс може бути також реалізацією протоколу.

Коли ППІ реалізує протокол, він може бути заснованим на проксі-методах віддалених викликів, що засновані на протоколі зв'язку. Роль ППІ може

заключатися саме в тому, щоб приховати деталі транспортного протоколу. Наприклад: RMI є ППІ, який реалізує протокол або JRMP IIOP як RMI-IIOP.

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

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

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

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

Частина цієї тенденції пов'язана з рухом Семантичного веб-ресурсу до Опису Платформ (RDF), Концепції розвитку веб-технологій інженерних онтологій.

Прикладні програмні інтерфейси у Web, що дозволяють комбінувати декількома прикладними програмними інтерфейсами в нові додатки називають гібридними.

**S.M.A.R.T.** (Self-monitoring, analysis and reporting technology – технологія самоконтролю, аналізу та звітності) – технологія оцінки стану жорсткого диска вбудованою апаратурою самодіагностики, а також механізм передбачення часу виходу його з ладу.

Технологія S.M.A.R.T. є частиною протоколу ATA (нині поширеного в інтерфейсі SATA).

Сучасні SSD накопичувачі з SATA інтерфейсом також підтримують S.M.A.R.T. Однак широко поширені USB флешки практично не підтримують S.M.A.R.T. оскільки USB Mass Storage device class заснований на іншому протоколі, SCSI, який не містить аналогічної S.M.A.R.T. функціональності.

Існує невелика кількість топових флешок, зроблених на основі SATA контролерів і перехідників SATA-USB, які працюють за специфікацією SAT (SCSI-ATA Translation). Деякі з таких перехідників підтримують трансляцію S.M.A.R.T. даних.

Перший жорсткий диск, що володіє системою самодіагностики, був представлений в 1992 році фірмою IBM в дискових масивах IBM 9337 для серверів AS / 400, що використовують IBM 0662 SCSI-2 диски.

Технологія була названа Predictive Failure Analysis (PFA). Вимірювалося кілька ключових параметрів, їх оцінка проводилася безпосередньо контролером диска. Результат був обмежений лише одним бітом: або все в порядку, або диск може незабаром вийти з ладу.

Пізніше компаніями Compaq, Seagate, Quantum і Conner була розроблена інша технологія, названа IntelliSafe. У ній був загальний протокол видачі інформації про стан жорсткого диска, але вимірювані параметри і їх пороги кожна компанія визначала самостійно.

На початку 1995 року Compaq запропонувала стандартизувати технологію. Компанії IBM, Seagate, Quantum, Conner і Western Digital (остання на той момент ще не мала системи відстеження параметрів жорсткого диска) підтримали цю

ідею. За основу була взята технологія IntelliSafe. Спільно розроблений стандарт назвали S.M.A.R.T.

Стандарт S.M.A.R.T. I передбачав моніторинг основних параметрів і запускався тільки після команди.

У розробці S.M.A.R.T. II брала участь Hitachi, яка запропонувала методику повної самодіагностики накопичувача (extended self-test), також з'явилася функція журналювання помилок.

У S.M.A.R.T. III з'явилася функція виявлення дефектів поверхні і можливість їх відновлення «прозоро» для користувача.

S.M.A.R.T. проводить спостереження за основними характеристиками накопичувача, кожна з яких отримує оцінку. Характеристики можна розбити на дві групи:

1. Параметри, що відображають процес природного старіння жорсткого диска (число оборотів шпинделя, число переміщень головок, кількість циклів включення-виключення).

2. Поточні параметри накопичувача (висота головок над поверхнею диска, число перепризначених секторів, час пошуку доріжки і кількість помилок пошуку).

Дані зберігаються в шістнадцятковому вигляді, званому raw value ( «сирі значення»), а потім перераховуються в value – значення, яке символізує надійність щодо деякого еталонного значення. Зазвичай value розташовується в діапазоні від 0 до 100.

Висока оцінка говорить про відсутність змін даного параметра або повільному його погіршенні. Низька – про можливий збій незабаром.

Значення, менше, ніж мінімальна, при якому виробником гарантується безвідмовна робота накопичувача, означає вихід вузла з ладу.

Технологія S.M.A.R.T. дозволяє здійснювати:

– моніторинг параметрів стану;

– сканування поверхні;

– сканування поверхні з автоматичною заміною сумнівних секторів на надійні.

Слід зауважити, що технологія S.M.A.R.T. дозволяє прогнозувати вихід пристрою з ладу в результаті механічних несправностей, що становить близько 60% причин поломки жорсткого диска. Передбачити наслідки стрибка напруги або механічного удару S.M.A.R.T. не здатна.

Слід зазначити, що накопичувачі не можуть самостійно повідомляти про свій стан за допомогою технології SMART, однак для цього існують спеціальні програми. Таким чином, використання технології S.M.A.R.T. неможливо без наявності наступних двох складових:

1. ПЗ, вбудованого в контролер накопичувача;

2. Зовнішнього ПЗ, вбудованого в хост.

Програми, що відображають стан S.M.A.R.T.-атрибутів, працюють за таким алгоритмом:

1. Перевірка наявності підтримки накопичувачем технології S.M.A.R.T.

2. Посилка команди запиту S.M.A.R.T.-таблиць.

3. Отримання таблиць в буфер додатки.

4. Розшифровка табличних структур, витяг номера атрибута і його числового значення.

5. Зіставлення стандартизованих номерів атрибутів їх назвами (іноді – в залежності від типу, моделі або виробника, як, наприклад, в програмі Victoria).

6. Висновок числових значень в зручному для сприйняття вигляді (наприклад, конвертація шістнадцятирічних значень в десяткові).

7. Витяг з таблиць прапорів атрибутів (ознак, що характеризують призначення атрибута в даному накопичувачі, наприклад, «життєво важливий» або «лічильник»).

8. Висновок загального стану пристрою на підставі всіх таблиць, значень і прапорів.

Жорсткі диски з підтримкою SMART версії 2 і старше, пропонують ряд різних тестів:

1. Короткий (Short). Перевіряє електричні і механічні параметри, а також продуктивність на читання. Тест, як правило, триває близько двох хвилин.

2. Довгий/розширений (Long / extended). Тест перевіряє всю поверхню диска і не має обмеження за часом. В середньому займає близько двох-трьох годин.

3. Тест транспортування (Conveyance). Швидкий тест, призначений для оцінки стану диска після транспортування диска від виробника до постачальника.

4. Вибірковий (Selective). Деякі диски дозволяють перевірити певну частину поверхні.

Журнал тестів SMART може містити результати тільки 21 останніх тестів і доступний тільки для читання. Іншими словами, скинути його штатними засобами неможливо. Журнал представляє собою таблицю з наступних колонок: порядковий номер тесту, тип тесту, результат тесту, скільки відсотків залишилося до завершення, час життя диска, LBA.

Хоча я реалізовував програму сам, було використано підходи Scrum для саморозвитку та пришвидшенню розробки, розглянемо цей метод. Scrum – підхід управління проектами для гнучкої розробки програмного забезпечення. Скрам чітко робить акцент на якісному контролі процесу розробки.

Підхід вперше описали Гіротака Такеучі та Ікуджіро Нонака в статті The New New Product Development Game (Гарвардський Діловий Огляд, січ–лют 1986). Вони відзначили, що проекти, над якими працюють невеликі, крос–функціональні команди, зазвичай систематично продукують кращі результати, і пояснили це, як «підхід регбі». У 1991 році ДеҐрейс та Шталь у книжці Злі проблеми, справедливі рішення послалися на цей підхід, як на Scrum (штовханина; сутичка навколо м'яча (у регбі)), спортивний термін, згаданий в статті Такеучі і Нонака. Кен Швабер на початку 1990–х використовував підхід який привів Scrum в його компанію.

Вперше метод Scrum було представлено на загальний огляд задокументованим, чітко сформульованим та описаним спільно Сазерлендом та Швабером на OOPSLA'96 в Остіні. Швабер та Сазерленд протягом наступних років працювали разом щоб обробити та описати весь їхній досвід та найкращі практичні зразки для індустрії в одне ціле, в ту методологію, що відома сьогодні як Scrum. Швабер об'єднав зусилля з Майком Бідлом в 2001, щоб детально описати метод в книжці Agile Software Development with SCRUM. Не зважаючи на те, що для Scrum нарікли долю управління проектами з розробки ПЗ, він може також використовуватися в роботі команд обслуговувань програмного забезпечення (software maintenance teams), або як підхід управління розробкою і супроводом програм: Scrum of Scrums.

Scrum – це кістяк процесу, який включає набір методів і попередньо визначених ролей. Головні дійові особи – ScrumMaster, той хто опікується процесами, веде їх і працює як керівник проекту, Власник Продукту, людина, що представляє інтереси кінцевих користувачів та інших зацікавлених в продукті сторін, та Команду, яка включає розробників.

Протягом кожного спринту, 15-30 денного періоду (тривалість визначається командою), працівники створюють функціональний ріст програмного забезпечення.

Набір можливостей, які імплементуються кожного спринту, приходять з етапу, що має назву product backlog (документація запитів на виконання робіт), який має найвищу пріоритетність за рівнем вимог до роботи, що повинна бути виконана.

Запити на виконання робіт (backlog items), що визначені протягом наради з планування спринту (sprint planning meeting), переміщуються в етап спринту. Протягом цієї наради Власник Продукту інформує про завдання, які він хоче, аби були виконані. Тоді Команда визначає, скільки з бажаного вони можуть зробити, щоб завершити необхідні частини протягом наступного спринту. Протягом спринту команда виконує визначений фіксований список завдань (т.з. backlog

items). Впродовж цього періоду ніхто не має права змінювати перелік запитів на виконання робіт, що слід розуміти, як заморожування вимог (requirements) протягом спринту.

Product backlog – це документ, який має список вимог до функціональності, які упорядковані згідно зі ступенем важливості. Product backlog представляє список того, що повинно бути реалізовано. Елементи цього списку називається «історіями» (user story) або елементами backlog–у (backlog items). Product backlog відкритий для редагування усім учасникам Scrum–процесу.

Обов'язкові поля:

1. ID – унікальний ідентифікатор, порядковий номер, який використовується для ідентифікації історій у разі їх перейменування.

2. Назва (Name) – стислий опис історії. Він повинен бути однозначним, щоб і розробники і product owner могли зрозуміти, про що йдеться і відрізнити одну історію від іншої.

3. Важливість (Importance) – ступінь важливості даної історії на погляд product owner 'а. Зазвичай являє собою натуральне число, іноді для цієї цілі використовуються числа Фібоначчі. Чим більше значення, тим більше пріоритет.

4. Попередня оцінка (initial estimate) – початкова оцінка об'єму робіт, необхідного для реалізації історії порівняно з іншими історіями. Вимірюється у story point'ах. Приблизно відповідає числу «ідеальних людино–днів».

5. Як продемонструвати (how to demo) – стисле пояснення того, як завершена задача буде продемонстрована у кінці спринта. Дане поле може являти собою код автоматизованого приймального тесту.

Додаткові поля. Іноді, також, використовуються додаткові поля у product backlog, в основному для того, щоб допомогти product owner'у визначитися з його пріоритетами.

Категорія (track). Наприклад, «панель управління» чи «оптимізація». За допомогою цього поля product owner може легко вибрати усі пункти категорії «оптимізація» і задати їм низький пріоритет.

Компоненти (components) – указує, які компоненти (наприклад, база даних, сервер, клієнт) будуть зачеплені при реалізації історії. Дане поле складається з групи checkbox'ів, які відмічаються, якщо відповідні компоненти потребують змін.

Ініціатор запиту (requestor). Product owner може захотіти зберігати інформацію про усіх замовників, зацікавлених у даній задачі. Це потрібно для того, щоб тримати їх у курсі діла про хід виконання робіт.

ID у системі обліку помилок (bug tracking ID) – якщо ви використовуєте окрему систему обліку помилок, тоді у описі історії корисно зберігати посилання на всі дефекти, які до неї відносяться.

Sprint backlog – містить функціональність, обрану Product Owner із Product Backlog. Всі функції розбиті по задачах, кожна з яких оцінюється командою. Кожен день команда оцінює об'єм роботи, який необхідно провести для завершення задачі.

Burndown chart – показує, скільки вже виконано і скільки ще залишається зробити.

**Планування спринта (Sprint Planning Meeting)**

Проходить на початку нової ітерації Спринта:

– Із Product Backlog обираються задачі, зобов'язання по виконанню яких за спринт приймає на себе команда;

– На основі обраних задач створюється Sprint Backlog. Кожна задача оцінюється у ідеальних людино–годинах;

– Рішення задачі не повинно займати більше 12 годин або одного дня. При необхідності задача розбивається на підзадачі;

– Обговорюється та визначається, яким чином буде реалізовано цей об'єм робіт;

– Тривалість наради обмежена зверху 4–8 годинами в залежності від тривалості ітерації, досвіду команди тощо;

– (перша частина наради) Беруть участь Product Owner + Команда: обирають задачі із Product Backlog;

– (друга частина наради) Бере участь лише команда: обговорюють технічні деталі реалізації, наповнюють Sprint Backlog.

**Щоденна нарада (Daily Scrum Meeting)**

Відбувається кожен день протягом спринта. Є «пульсом» ходу спринта. Нараді властиві наступні обмеження:

– починається точно вчасно;

– всі можуть спостерігати, але говорять тільки обрані;

– триває не більш ніж 15 хвилин;

– проводиться в одному і тому ж місці протягом одного спринта.

Протягом наради кожен член команди відповідає на 3 запитання:

– Що зроблено з моменту попередньої щоденної наради?;

– Що буде зроблено з моменту поточної наради до наступної?;

– Які проблеми заважають досягненню цілей спринта? (Над рішенням цих проблем працює ScrumMaster. Зазвичай це рішення проходить за рамками щоденної наради і у складі осіб, що безпосередньо займаються даною перешкодою.)

Демонстрація (Sprint Review Meeting):

– Проходить у кінці ітерації (спринта).

– Команда демонструє внесок функціональності до продукту всім зацікавленим особам.

– Залучається максимальна кількість глядачів.

– Усі члени команди беруть участь у демонстрації (одна людина на демонстрацію або кожен показує, що зробив за спринт).

– Обмежена 4–ма годинами в залежності від тривалості ітерації і змін у продукті.

Ретроспектива (Sprint Retrospective):

– Члени команди висловлюють свою думку про минулий спринт.

– Відповідають на два основних запитання: Що було зроблено добре у минулому спринті?; Що потрібно покращити в наступному?.

– Виконують покращення процесу розробки (вирішують питання та фіксують вдалі рішення).

– Обмежена 1-3ма годинами.

### 4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму UMAC (код автентифікації повідомлення на основі універсального гешування) – один з видів коду автентичності повідомлень (MAC).

Швидка «універсальна» функція використовується, для того, щоб гешувати вхідне повідомлення M у короткий рядок. До цього рядка потім застосовується функція XOR із псевдовипадковим значенням, у результаті чого ми одержуємо тег UMAC:

$$Tag=H_{K1}(M)\oplus F_{K2}(Nonce)$$

де K1 і K2 – секретні випадкові ключі, які мають одержувач і відправник.

Звідси видне, що безпека UMAC залежить від того, яким випадковим способом відправник і одержувач вибрали таємну геш-функцію й псевдовипадкову послідовність. При цьому значення Nonce міняється кожний такт. Через використання Nonce, приймач і передавач повинні знати час відправлення повідомлення й принцип створення значення Nonce. Замість цього можна використовувати в якості Nonce будь-яке інше неповторюване значення, наприклад порядковий номер повідомлення. При цьому даний номер не зобов'язано бути секретним, головне щоб він не повторювався.

UMAC розрахований на використання 32-х, 64-х, 92-х, і 128-бітових тегів, залежно від необхідного рівня безпеки. UMAC звичайно використовується разом з алгоритмом шифрування AES.

Функція створення ключа й псевдовипадкової послідовності

Створення псевдовипадкових байтів необхідно для роботи UHASH і при створенні тегів

**Вибір блокового шифру**

Для своєї роботи UMAC використовує блоковий шифр, вибір якого визначають наступні константи:

– BLOCLEN – довжина, у байтах, блоку з яким працює блоковий шифр.

– KEYLEN – довжина, у байтах, ключа блокового шифру.

При цьому використовується функція

– ENCIPHER(K,P) – зашифрувати рядок P з BLOCLEN байтів, використовуючи ключ K.

Приклад: якщо використовується AES з 16-байтним ключем, то BLOCLEN буде рівним 16( тому що AES працює з 16-байтними блоками).

**KDF – функція створення ключа**

Ця функція генерує послідовність псевдовипадкових байтів, використовуваних для ключових геш-функцій.

Вхід:

– K – рядок довжиною KEYLEN байт. / / Ключ блокового шифру.

– Index – ненегативне ціле число менше, чим 2 ^ 64.

– Numbytes – ненегативне ціле число менше, чим 2 ^ 64.

Вихід:

– Y – рядок довжини numbytes байт.

**PDF: функція створення псевдовипадкового числа**

Ця функція ухвалює ключ і даний час і повертає псевдовипадкове число для використання його в тегу покоління. За допомогою цієї функції можуть бути отримані числа довжиною 4, 8, 12 або 16 байт.

Вхід:

– K – рядок довжиною KEYLEN байт.

– Nonce – рядок довжиною від 1 до BLOCKLEN байт.

– Taglen – ціле число 4, 8, 12 або 16.

Вихід:

– Y – послідовність байтів довжини taglen.

**Генерація UMAC-тегів**

Генерація UMAC-тегів відбувається за допомогою UHASH функції при використанні Nonce значенні й отриманої до цього рядка. Їхня довжина може бути 4, 8, 12 або 16 байт.

Вхід:

– K – рядок довжиною KEYLEN байт.

– M – рядок довжиною менше 267 біт.

– Nonce – випадкове число від 1 до BLOCKLEN байт.

– Taglen – ціле 4, 8, 12 або 16.

Вихід:

– Тег, послідовність байтів довжиною taglen.

Алгоритм обчислення тегів:

Hashedmassage = UHASH(K, M, Taglen)

Pad = PDF(K, nonce, Taglen)

Tag = Pad xor Hashedmassage

**UMAC-32 UMAC-64 UMAC-96 UMAC-128**

Дані позначення містять у своїй назві певне значення довжини тегу:

– UMAC-32 ( K, M, Nonce) = UMAC (K, M, Nonce, 4).

– UMAC-64 ( K, M, Nonce) = UMAC (K, M, Nonce, 8).

– UMAC-96 ( K, M, Nonce) = UMAC (K, M, Nonce, 12).

– UMAC-128 (K, M, Nonce) = UMAC (K, M, Nonce, 16).

**Універсальна функція гешування(UHASH)**

UHASH – універсальна функція гешування, серцевина алгоритму UMAC. UHASH – функція працює в три етапи. Спочатку до вхідного повідомлення застосовується L1-HASH, потім до цього результату застосовується L2-HASH і, нарешті, до результату застосовується L3-HASH. Якщо при цьому довжина

вхідного повідомлення не більш 1024 біт, то L2-HASH не використовується. Тому що функція L3-hash повертає тільки слово довжини 4 байта, те якщо потрібно одержати геш довжини більше 4 байт, здійснюється кілька ітерацій даної трирівневої схеми.

**Універсальна функція**

Нехай функція гешування вибирається із класу геш-функцій H, які відображають повідомлення в D, набір усіляких образів повідомлення. Цей клас називається універсальним, якщо для яких-небудь окремих пар повідомлень, існує на безлічі H/D функцій, функція, яка відображає їх в елемент D. Зміст цієї функції в тому, що якщо третя сторона прагне замінити одне повідомлення іншим, але при цьому вважає, що геш-функція була обрана абсолютно випадково, те ймовірність не виявлення підміни стороною, що ухвалює, прагне до 1/D.

**L1-hash – перший етап**

L1-hash розбиває повідомлення на шматки з 1024 байт і до кожного шматка застосовує алгоритм гешування називаний NH. Вихідний результат алгоритму NH в 128 раз менше вхідного.

**L2-hash – другий етап**

L2-hash працює з виходом L1-hash, використовує поліноміальний алгоритм POLY. Другий етап гешування використовується, тільки якщо довжина вхідного повідомлення більше 16 мегабайт. Використання алгоритму POLY потрібно для того, щоб уникнути тимчасову атаку. На виході з алгоритму POLY виходить 16 байтне число.

**L3-hash – третій етап**

Цей етап потрібно для того щоб з вихідних 16 байтів алгоритму L2-hash одержати 4-байтне значення.

# 5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено розроблене у бакалаврської дипломної роботі система Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи: Функціональних кнопок ПЗ; Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші; Верхнього меню; Розділу обрання групи; Розділу виведення результату роботи системи.

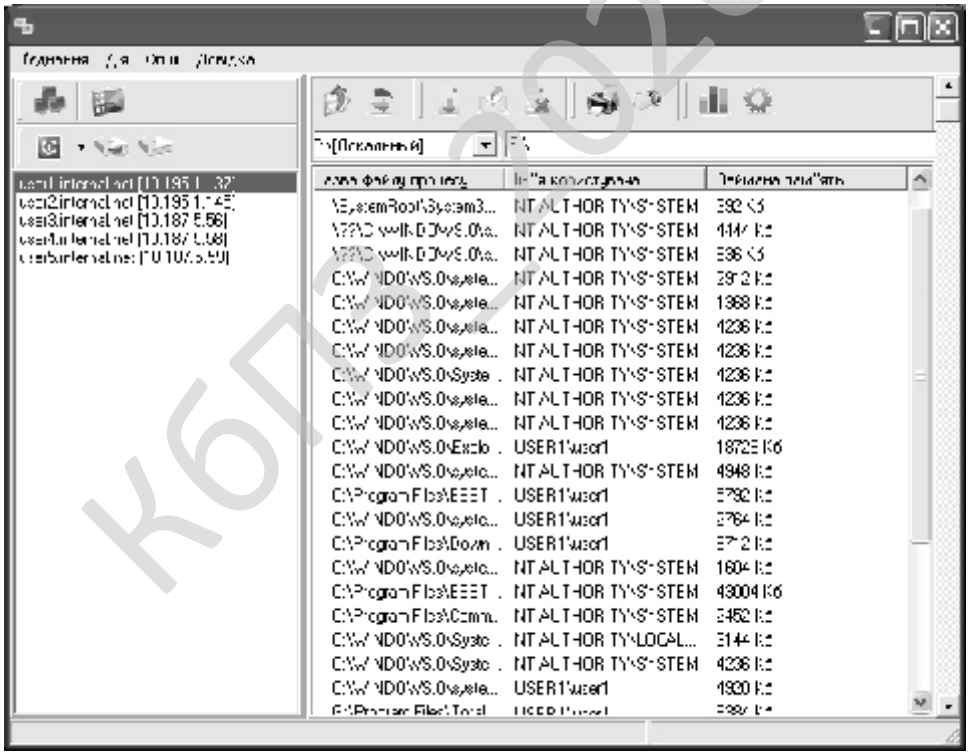

Рисунок 5.1 – Головне вікно ПЗ

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

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

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

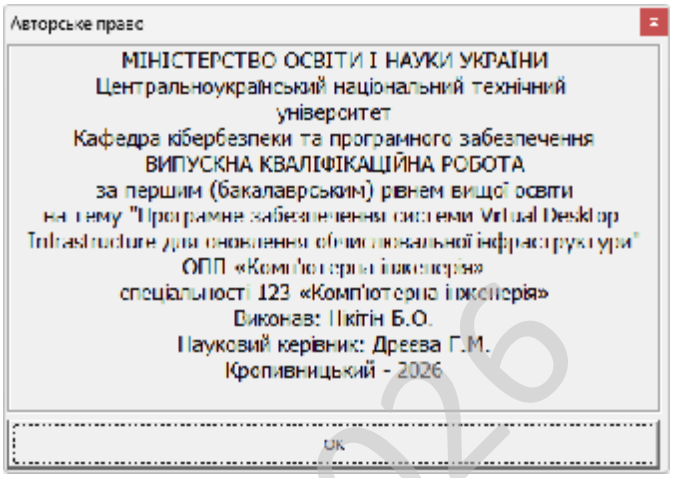


Рисунок 5.2 – Авторське право

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

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

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

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

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

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

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

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

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

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

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

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

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

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

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

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

– У програмі можуть бути пропущені деякі маршрути.

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

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

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

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

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

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

Проводилось тестування чорної скриньки.

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

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

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

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

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

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

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

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

Будь-який спосіб тестування «чорної скриньки» повинен:

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

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

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

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій.

– Помилок інтерфейсу.

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

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

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – Freeware. Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли

програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення.

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

# 6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури.

– Досліджена система Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури.

– На основі отриманих результатів досліджень створена програмна реалізація системи Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури.

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

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

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для

системи Virtual Desktop Infrastructure для оновлення обчислювальної інфраструктури. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм UMAC.

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

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

22. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and

Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

37. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у

комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

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

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

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

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

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

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

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

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

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

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

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

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

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