МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КОМПЛЕКСНА КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Інформаційна технологія віддаленого керування інформаційно-телекомунікаційною системою. Клієнтська частина»

Завідувач

випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Петров С.О.**

Студента групи ІН.м – 91н **Шипіка М.Д.**

СУМИ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2021 г.

ЗАВДАННЯ

до комплексної дипломної роботи

Студента четвертого курсу, групи ІН.м-91н спеціальності "Інформатика" денної форми навчання Шипіка Максима Дмитровича.

Тема: " **Інформаційна технологія віддаленого керування інформаційно-телекомунікаційною системою. Клієнтська частина**"

Затверджена наказом по СумДУ

№_____ от _____2021 г.

Зміст пояснювальної записки: 1) інформаційний огляд; 2) постановка завдання; 3) вибір програмних засобів для рішення поставленої задачі; 4) розробка клієнтської частини системи; 5) аналіз готової системи.

Дата видачі завдання "_____"_____2021 г.

Керівник випускної роботи _____ Петров С.О.

Завдання прийняв до виконання _____ Шипік М.Д.

РЕФЕРАТ

Записка: 45 стор., 2 рис., 1 табл., 5 додатків, 16 джерел.

Об’єкт дослідження — системи віддаленого керування, методи та інструменти їх реалізації.

Мета роботи — інтеграція інструментів віддаленого керування до Telegram, дослідження можливостей подібних інтеграцій та потенціал використання API.

Методи дослідження — реалізація клієнтської частини системи віддаленого керування комп’ютером з використанням C#/.NET.

Результати — розроблено клієнтську частину Telegram-бота для віддаленого виконання команд cmd на комп’ютері. Зроблено висновки щодо потенціалу використання API та інтеграцій різних систем.

СИСТЕМА ВІДДАЛЕНОГО КЕРУВАННЯ, TELEGRAM-БОТ,
C#/.NET., WPF, ВЕБХУК, API.

Зміст

ВСТУП

Актуальність. У зв’язку з пандемією COVID-19 гостро постало питання віддаленого доступу, роботі на віддалені, сервісів доставки та багато інших змін, що стосуються як повсякденного життя, так і нестандартних завдань. Адаптивність, як невід’ємна складова еволюції, допомогла людству зреагувати на складні умови та швидко пристосуватися до них. Інформаційні технології посіли особливе місце у процесі адаптації, швидко провели необхідні вдосконалення та забезпечили плавний перехід до нових умов. У аспекті інформаційних систем складена ситуація змусила знову та по-новому поглянути на питання зв’язку, віддаленого керування, безконтактні системи, а також використання сучасних технологій у медицині. Враховуючи усі ці фактори було вирішено дослідити системи віддаленого керування, їх походження, розвиток, сучасні досягнення, розглянути та проаналізувати наявні технології, систематизувати та порівняти отримані дані та зробити висновки для подальшого покращення життя як розробників, так і користувачів в майбутньому.

Частиною комплексної роботи є реалізація серверної частини системи віддаленого керування.

Мета цієї роботи – дослідити рівень розвинутості систем віддаленого керування, систематизувати отримані дані та на їх основі розробити зручний сервіс для віддаленого керування персональним комп’ютером використовуючи сучасні тенденції.

Завдання:

- Ознайомитися із історією віддаленого керування;
- Оглянути сучасні системи віддаленого керування;
- Ознайомитися із готовими рішеннями поставленої проблеми;
- Проаналізувати їх проблеми та переваги;
- Обрати необхідні для розробки ресурси;
- Розробити клієнтську частину системи.

1. ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Історія систем віддаленого керування

Першою спробу дистанційного керування вважається робота сербського фізика Ніколи Тесли. У 1898 році вчений продемонстрував підводний човен, що керувався дистанційно за допомогою радіохвиль.

Продовжуючи праці Тесли, Леонардо Торез Куведо у 1903 році продемонстрував роботу його робота, що виконував прості команди на віддаленні за допомогою радіохвиль.

Як відомо, більшість винаходів людства стали можливі завдяки військовим розробкам. Дистанційне керування не виняток, тож із 30-х років XX століття почався активний розвиток систем дистанційного керування для військових цілей. У 1932 році була створена модель аероплана на радіокеруванні.

У другій половині XX століття дистанційне керування почало активно з’являтися у повсякденному житті людей у найрізноманітніших його сферах. Технології завжди намагалися зроби життя людей простішим, а бажання виконувати якомога більше дій, не підіймаючись з місця не лишало розумів як користувачів, так і розробників. Основним прикладом можна вважати пульт дистанційного керування до мультимедійних систем. За період розвитку технологій було винайдено багато рішень для дистанційного керування. Серед них були системи на дротах, з використанням радіохвиль, світлових сигналів, ультрафіолетового та інфрачервоного світла, звуку, ультразвуку і т.д.

1.2 Сучасні системи віддаленого керування

На сьогоднішній день людство не може уявити собі життя без сучасних досягнень науки та техніки. Інформаційні технології стали невід’ємною частиною людського побуту. Щодня з’являються нові рішення, що роблять наше життя простіше та економлять час. Головним поштовхом до розвитку систем дистанційного розвитку став факт розвитку та поширення систем

комунікації, таких як інтернет, мобільний інтернет, Bluetooth, Wi-Fi та інші, знайомі кожному користувачу системи. Таким чином системи віддаленого керування отримали змогу до майже безмежного розширення сфер свого застосування.

Основні приклади застосування дистанційного керування:

- Космічна техніка;
- Зв’язок та комунікація;
- Охоронні системи;
- Комп’ютерна техніка;
- Мультимедійна техніка;
- Військова техніка;
- Громадський транспорт;
- Під час Будівництва;
- Електроенергетика;
- Медицина;
- Лабораторне обладнання.

Можливості використання систем віддаленого керування не обмежуються вищевказаним списком, тому що кожного дня покращуються існуючі та створюються нові. На сьогоднішній день з використанням сучасних технологій можливо як регулювати штори за допомогою голосу, так і безпечно працювати з радіоактивними відходами з допомогою дистанційно керуємих роботів.

Як один з найцікавіших прикладів сучасного дистанційного керування можна привести набираючу популярність систему «розумного будинку». Як і більшість сучасних систем, «розумний будинок» має двосторонній зв’язок, тобто явним чином передає користувачу інформацію у відгук. Система має центральний контролер, що керує усіма частинами, системи моніторингу (різноманітні датчики температури, вологості, мікрофони, фотоелементи і т.д.), системи виконання (акустичні системи, «розумні розетки», побутова

техніка з відповідними функціями, електромеханічні системи регулювання і т.д.) та органи керування. Таким чином користувач має можливість не тільки полегшити собі життя знаходячись у будинку, а й керувати ним за його межами. Отже проблема невимкненої праски може назавжди відійти у минуле, а шанс крадіжки значно зменшиться.

1.3 Системи віддаленого керування персональним комп'ютером

Ідея віддаленого керування персональним комп'ютером почала зароджуватися разом із створенням персонального комп'ютера.

Однією з перших вдалих спроб віддаленого керування була програма Carbon Copy, розроблена Meridian Technology. Carbon Copy надавала віддалений сумісний доступ до екрану, передачу файлів та чат-вікна.

Інший приклад – програма pcAnywhere, що дозволяла керувати персональним комп'ютером з встановленим хостом pcAnywhere та із відомим паролем.

SSH (Secure Shell) – Мережевий протокол для віддаленого керування операційною системою. Реалізує безпечну передачу даних на відміну від схожих протоколів Telnet та rlogin. Для встановлення такого зв'язку між девайсом та комп'ютером необхідні SSH-клієнт та SSH-сервер.

Такий метод віддаленого підключення перевірений часом, дозволяє виконати більшу частину дій на віддаленому комп'ютері, але вимагає від користувача знання роботи із консольними оболонками системи.

Операційні системи створюють власні системи віддаленого керування. Windows віддалений робочий стіл або віддалене керування MacOS – основні приклади таких систем.

З часом на ринку програмних продуктів з'являлися й інші програми, що реалізували віддалене керування комп'ютером. Наведемо порівняння деяких з них.

Таблиця 1.1 Порівняння існуючих програм віддаленого керування

Застосунок	**Показ екрану**	**Віддалений доступ**	**Обмін повідомлення**	**Спільний доступ**	**Відео-зв'язок**	**Передача файлів**	**Платформа/ ОС**
Ammyy Admin	Так	Так	Так	Так	Ні	Так	Windows
BeAnywhere	Так	Так	Так	Так	Ні	Так	Windows, Java, Mac, Android
Chrome Remote Desktop	Так	Так	Так	Так	Так	Ні	Chrome OS, Linux (beta), OS X, iOS, Windows, Android
Glance	Так	Ні	Так	Так	Так	Ні	(No download) Windows, Mac, Linux, iPhone, Android
Goverlan Reach	Так	Так	Так	Так	Так	Так	Windows, Mac, Linux, iPhone, Android
CloudBerry Remote Assistant от	Так	Так	Так	Так	Так	Ні	Windows

CloudBerry Lab							
IBM Lotus Sametime	Так	Так	Так	Так	Так	Так	Windows, Linux, Mac
LogMeIn	Так	Так	Ні	Ні	Ні	Так	Windows, Mac, iPhone, iPad, Android
Mikogo	Так	Так	Так	Так	Так	Так	Windows, Mac, iPhone, iPad, Android
Nefsis	Так	Так	Так	Так	Так	Так	Windows
Netviewer	Так	Так	Так	Так	Так	Так	Windows, Mac
NoMachine	Так	Так	Так	Ні	Ні	Так	Windows, Mac, Linux, Raspberry, iOS, Android
RAdmin	Так	Так	Так	Ні	Ні	Так	Windows
RealVNC	Так	Так	Так	Так	Ні	Так	Windows, Mac, Linux, Raspberry Pi, iOS, Android, Chrome

							OS, Solaris, HP-UX, AIX
Skype	Так	Так	Так	Ні	Так	Так	Windows, Mac, (в Linux — без можливості показу)
Splashtop	Так	Так	Ні	Так	Ні	Так	Windows, Mac, iPhone, iPad, Android, Linux, Chrome OS, Chrome browser, FireTV, FireTV stick
TeamViewer	Так	Так	Так	Так	Так	Так	Windows, Mac, Linux, iPhone, Android
Techinline	Так	Так	Так	Так	Ні	Так	Windows
TigerVNC	Так	Так	Ні	Так	Ні	Так	Windows, Linux, Mac
TightVNC	Так	Так	Ні	Так	Ні	Так	Windows

WebEx	Так	Так	Так	Так	Так	Так	Windows, Linux, Mac, Unix, Solaris, iPhone
Wire	Так	Ні	Так	Ні	Так	Так	Windows, Linux, Mac, Unix, iPhone, Android
Zoom	Так	Так	Так	Так	Так	Так	Android, iOS, Windows, Linux, Mac

Найближче до теми роботи знаходиться Віддалений робочий стіл Chrome.

Основною його перевагою є простота у реалізації підключення. Вище було згадано, що тенденцією розвитку є синхронізація девайсів. GoogleLLC. активно реалізують таку синхронізацію, тому створення можливості керування девайсами було лише питанням часу.

Для роботи з Chrome Remote Desktop необхідно виконати наступні кроки:

- Встановити Chrome Remote Desktop на комп’ютер, до якого необхідно підключитися
- У «Сервісах» обрати «Віддалений робочий стіл Chrome»
- Дозволити віддалені підключення
- Ввести PIN-код
- Виконати вхід до аканту Google з іншого девайсу
- Підключитися до віддаленого робочого столу використовуючи розширення браузера(для персональних комп’ютерів) або додатку(для смартфонів) ввівши відповідний PIN-код.

Враховуючи ситуацію у світі, пов’язану із пандемією COVID-19 багато компаній перейшли на віддалений режим. Таким чином програми віддаленого керування дали змогу швидко та ефективно перейти на новий режим. Для збереження зв’язку при неможливості знаходження співробітників поруч можливість демонстрації робочого столу та виконання дій дистанційно значно спростило роботу як роботодавцям, так і співробітникам. Служби підтримки мають змогу віддалено надавати технічну допомогу клієнтам.

Вивчаючи питання різноманітних систем, що функціонують у ситуації пандемії, враховуючи системи віддаленого керування і не тільки виявилось, що значну роль під час розробки будь-якого програмного продукту має API.

1.4 API та його застосування

API (Application Programming Interface) – інтерфейс програмування, набір класів, функцій, процедур, структур та констант для зовнішнього виклику іншими програмами. Іншими словами, API – стороння програма, функціонал якої може використовувати розробник у своїх цілях, не створюючи його власноруч з початку. До того ж не потрібно знати як працює система, API якої використовується.

API використовується маже усюди, де є інформаційні технології. Наприклад, інтернет, що складається з 7 рівнів мережевої моделі OSI, технології верхніх рівнів використовують API нижніх для конвертації протоколів під час передачі будь-яких пакетів даних.

Більшість користувачів не здогадуються про використання API у додатку. Найяскравішим прикладом під час пандемії COVID-19 можна привести мобільні додатки доставки. Під час замовлення користувач може провести оплату через API банківських систем, а потім обрати місце доставки за допомогою API карт (в першу чергу – API Google Maps). Таким чином користувачеві не потрібно окремо встановлювати додаткових додатків, витрачати зайвий час на пошук та вибір способу розрахунку, повідомлення коректної адреси і т.д. У свою чергу розробникам не потрібно додатково

створювати свою платіжну систему, систему надання адреси та інші, це значно спрощує процес розробки.

1.5 Вибір мессенджера

Так як у ситуації пандемії COVID-19 більшість діяльності перейшла на дистанційний режим, то можна вважати, що мессенджери – додатки для обміну повідомленнями – стали невід'ємною частиною людської діяльності. За даними сайту statista.com на 2021 рік, WhatsApp та Facebook Messenger займають перше да друге місця за кількістю користувачів у світі, далі йдуть Weixin/WeChat та QQ, що орієнтовані на китайську аудиторію та не користуються популярністю в Україні, наступним є Telegram.

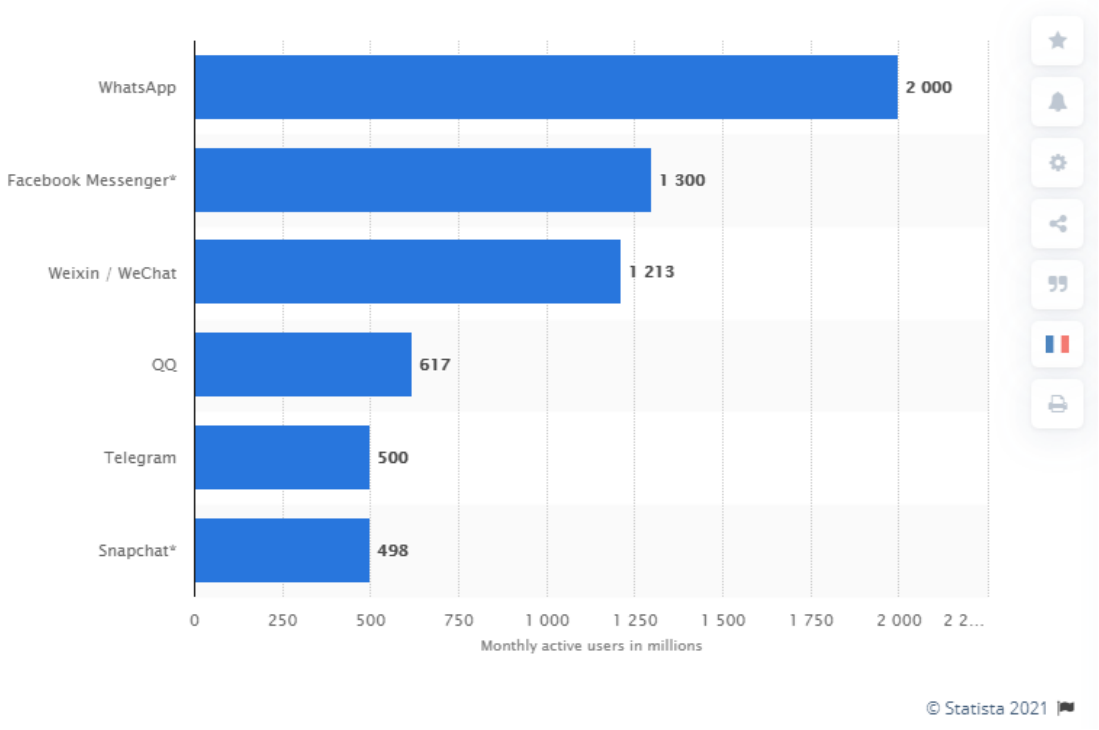


Рисунок 1.1 Популярність мессенджерів у світі

Щодо українських користувачів, за даними компанії Kantar на червень 2020 року найпопулярнішим мессенджером є Viber, але популярність набирають Facebook Messenger та Telegram.

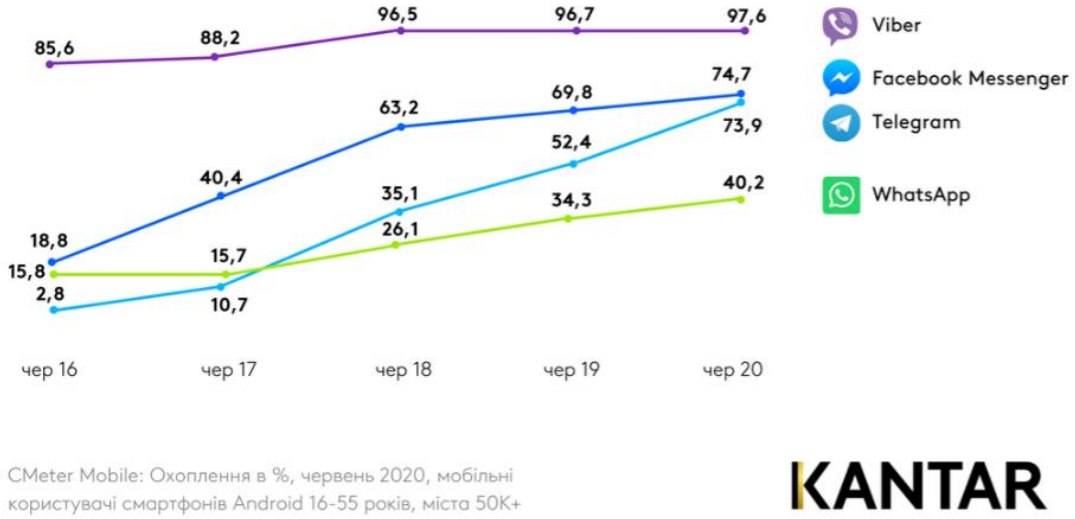


Рисунок 1.2 Динаміка кількості користувачів мессенджерів в Україні

Розглянувши та порівнявши можливості створення власних додатків-ботів з використанням вищезгаданих мессенджерів було обрано Telegram.

1.6 Постановка задачі

Враховуючи вищезгадане та сучасні тенденції було прийняте рішення створити сервіс віддаленого керування комп’ютером, що потребує від користувача найменшу кількість необхідних операцій.

Було поставлено задачу створення програмного додатку для реагування на команди надісланні Telegram-ботом. Цей програмний додаток має містити можливість самоідентифікації комп’ютера. Для встановлення зв’язку комп’ютер-смартфон необхідно також обрати метод ініціалізації комп’ютерів в системі. Для цього було обрано C#/.NET.

Таким чином маємо задачі:

- Реалізувати C# програму із формою
- Встановити зв’язок між комп’ютером та ботом
- Реалізувати відповідну реакцію комп’ютера до кожної команди

2 ТЕХНІЧНІ ЗАСОБИ РЕАЛІЗАЦІЇ

2.1 Комп’ютерна частина C#/.NET

«Мова C# (вимовляється “Сі-шарп”) – це багатопарадигмова об’єктно-орієнтована та компонентно-орієнтована мова програмування зі строгою типізацією, розроблена для платформи .NET Framework. Використання мови визначене стандартами ECMA-334 та ISO/IEC 23270:2006. Розроблена командою Microsoft Research під керівництвом Андерса Гейлсберга.» [1, с. 9]

«Платформа .NET Framework визначає середовище для підтримки, створення й виконання платформонезалежних гетерогенних додатків. Особливостями даної платформи є незалежне від мови середовище виконання (Common Language Runtime, CLR) та бібліотека класів .NET.

Після компіляції створюється не виконуваний файл написаний у машинних кодах, а набір команд записаних проміжною мовою MSIL (Microsoft Intermediate Language). При запуску цього додатку в операційній системі його код написаний у MSIL транслюється JIT-компілятором (Justin-Time compiler - оперативний компілятор) в машинні коди, зрозумілі процесору комп’ютера.

Як не важко здогадатися, такий додаток буде працювати лише в операційних системах з установленою платформою NET Framework відповідної версії.» [2, с. 8]

Для створення віконного додатку на C#/.NET було обрано пакет інструментів WPF.

WPF (Windows Presentation Foundation) – частина екосистеми платформи .NET та являє собою підсистему для створення графічних інтерфейсів. Особливістю WPF на відміну від Windows Forms є використання Direct X.

Таким чином знову можна спостерігати черговий приклад використання API високих рівнів, так як WPF використовує набір функцій закритої бібліотеки Direct X.

Основні переваги WPF:

- Використання декларованої розмітки XAML;
- Адаптивні розміри елементів;
- Додаткові елементи, що були складно реалізуємі у Windows Forms (наприклад тривимірні об’єкти або використання шаблонів);
- Висока взаємодія із Windows Forms;
- Гнучкість у створенні власних елементів;
- Апаратне прискорення графіки.

Також WPF має деякі недоліки, як наприклад складність при обробці великої кількості тривимірних об’єктів, незначне збільшення розміру файлів проекту, але в контексті задачі даної роботи вибір залишився на розглянутій технології.

Середовище Microsoft Visual Studio має широкий інструментарій для зручного створення додатків. У тому числі графічний інтерфейс для створення віконних програм. Генерування коду для створення розмітки проходить автоматично. Розглянемо елементи XAML, що будуть використані під час реалізації:

- TextBox – текстове поле, використане для введення логіну та паролю;
- Button – кнопка для збереження даних на сервері;
- Label – «ярлик», незмінний текст, що повідомляє користувачеві назви полів для введення та зворотній зв’язок про відправлені дані.

Також додатково вказуються назва та розмір вікна за допомогою параметрів Title, Height та Width.

2.2 Клас Process та його застосування

Для виконання дій у командному рядку Windows використовується клас Process.

Клас Process у платформі .NET із простору імен System.Diagnostics використовується для керування відкритими процесами та починати нові.

Розглянемо властивості класу Process:

- Handle – дескриптор процесу;
- Id – ідентифікатор процесу в рамках сеансу ОС;
- MachineName – ім’я комп’ютера;
- Modules – колекція модулів, завантажених в рамках процесу;
- ProcessName – ім’я процесу;
- StartTime – час запуску процесу;
- VirtualMemorySize64 – об’єм пам’яті, виділений на виконання процесу.

Розглянемо методи класу Process:

- CloseMainWindow() – закрити вікно процесу;
- GetProcesses() – отримати масив усіх активних процесів;
- GetProcessesByName() – отримати процеси використовуючи ім’я;
- Kill() – зупинити процес;
- Start() – запустити процес.

Із функціоналу класу для реалізації поставленої задачі нам потрібен метод Start(), що запускає процес. Для повідомлення який процес та з якими даними необхідно запускати використаємо екземпляр класу ProcessStartInfo. Після цього можна надавати отриману інформацію до процесу та виконувати запуск.

2.3 Використання Wix toolset для створення інсталлера

Для зручності користувача було вирішено створити зручний інсталлер за допомогою інструментів Wix toolset.

WiX (Windows Installer XML) – набір інструментів та специфікацій для створення пакетів інсталювання Windows. Результатом створення пакету інсталювання є файл MSI.

MSI (Microsoft Installer) – підсистема операційно системи Windows, що забезпечує встановлення програм. Інформація про встановлення знаходиться у файлі з розширенням імені .msi.

У складі документа MSI знаходяться всі необхідні дані для процесу встановлення, опису програмного продукту, зв’язків між його компонентами а також файлів, що необхідно встановити, стиснутих у форматі .cab.

Cabinet(.cab) – формат архівованих даних для використання в ОС Windows, що підтримує стиснення та цифрові підписи.

У роботі використовується Wix toolset для створення інсталлера формату .msi. Особливим під час створення інсталлера є момент ініціювання функції запису файлів реєстру до системи для реалізації ідеї запуску програми одночасно із операційною системою. Це значно полегшує роботу на віддаленні, так як для початку зв’язку персонального комп’ютера та Telegram-клієнта необхідно лише увімкнути комп’ютер.

3. РЕАЛІЗАЦІЯ СИСТЕМИ. КЛІЄНТСЬКА ЧАСТИНА. C#

3.1 Форма WPF та клієнтська логіка

Для створення ПК-клієнта використовуємо середовище Microsoft Visual C# та WPF. Використовуючи конструктор форм створюємо вікно із полями:

a) Поле «Логін»
b) Поле «Пароль»
c) Кнопка «Зберегти»
d) 3 Підписи:
    1. Login
    2. Password
    3. Saved
e) Пікторгама для панелі задач (Спливаюча підказка)

Більша частина генерації автоматизована, тому в результаті роботи отримуємо файл MainWindow.xaml наступного змісту

```
<Window x:Class="Client.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Client"
        mc:Ignorable="d"
        Title="Client" Height="320" Width="270"
```

```
        Loaded="Window_Loaded">
    <Grid>
        <Label Content="Login" HorizontalAlignment="Left"
Margin="108,10,0,0" VerticalAlignment="Top" />
        <TextBox Text=""  Name="Login" HorizontalAlignment="Left"
Height="27" Margin="66,47,0,0" TextWrapping="Wrap"
VerticalAlignment="Top" Width="120" />
        <Label Content="Password" HorizontalAlignment="Left"
Margin="98,79,0,0" VerticalAlignment="Top" />
        <TextBox Text="" Name="Password" HorizontalAlignment="Left"
Height="27" Margin="66,110,0,0" TextWrapping="Wrap"
VerticalAlignment="Top" Width="120" />
        <Label Name="Saved" Content="Saved" HorizontalAlignment="Left"
HorizontalContentAlignment="Center" Margin="90,160,0,0"
VerticalAlignment="Top" Width="67" Background="Green"
Visibility="Hidden" />
        <Button Content="Save" HorizontalAlignment="Left"
Margin="66,211,0,0" VerticalAlignment="Top" Width="120" Height="33"
Click="Button_Click" />
    </Grid>
</Window>
```

Зовнішній вигляд вікна програми продемонстровано у додатку Г.

Основні процеси відбуваються у файлі MainWindow.xaml.cs.

Створюємо таймер

```
  private static Timer timer;
```

У конструкторі намагаємося знайти логін та пароль від клієнта-комп'ютера, використовуючи клас StreamReader. У разі відсутності логіна або пароля ініціалізуємо поля порожніми значеннями. Закриваємо файловий потік методом Close().

```
  public MainWindow()
          {
              InitializeComponent();
              StreamReader sr = new
StreamReader(@"Data.dat");
              string[] logPass =
sr.ReadLine()?.Split();
              Login.Text = logPass?[0] ?? "";
              Password.Text = logPass?[1] ?? "";
              sr.Close();
          }
```

Далі реалізуємо подію Button_click(), що спрацьовує при натисканні кнопки збереження. Використовуємо клас StreamWriter для запису логіну та паролю в файл Data.dat. Повідомляємо користувачу успішність збереження, роблячи видимим напис «Saved». Ініціалізуємо змінну url значенням серверу із додатком. Генеруємо повідомлення для відправлення на сервер за допомогою

об’єкту класу NameValueCollection, що містить логін та пароль. Відправляємо дані на сервер url за допомогою методу Upload(), що буде описаний далі.

```
  private void Button_Click(object sender,
RoutedEventArgs e)
          {
              StreamWriter sw = new
StreamWriter(@"Data.dat");
              sw.WriteLine(Login.Text + ' ' +
Password.Text);
              sw.Close();
              Saved.Visibility = Visibility.Visible;

              string url =
@"https://remotecomputerconrtrolbot.000webhostapp.com/c
lient.php";
              NameValueCollection collection = new
NameValueCollection()
                  {
                      { "login", Login.Text },
                      { "pass", Password.Text }
                  };
```

```
            Upload(url, collection);
        }
```

Протягом роботи програми, за допомогою таймера timer кожні 4 секунди відбувається перевірка команд викликом методу Process().

```
        private void Window_Loaded(object sender,
RoutedEventArgs e)
        {
            timer = new Timer
            {
                Interval = 4000,
                Enabled = true
            };

            timer.Elapsed += new
ElapsedEventHandler(Process);
            timer.Start();
        }
```

Використовуємо екземпляр класу WebClient для перевірки наявності на сервері файлу із форматом логін.cmd.dat, що містить команду, яку має виконати комп’ютер. Використовуємо блок try{} щоб уникнути помилок під час запитів. Отримані із файлу із командою дані зберігаємо у змінну data. Ініціалізуємо змінну cmd декодуючи змінну data. Створюємо екземпляр

process класу Process та екземпляр startInfo класу ProcessStartInfo. Ініціалізуємо startInfo як схований процес, вказуємо назву процесу «cmd.exe» - стандартний командний рядок. Додаємо аргументом перемикач /C, що завершить роботу процесу після виконання команди. Додаємо до процесу process дані startInfo та виконуємо процес методом Start(). Далі реалізуємо відправлення запиту на видалення команди з серверу використовуючи описані вище інструменти url-адресу, колекцію NameValueCollection та метод Upload().

```
    private void Process(object sender, EventArgs e)
          {
              WebClient wc = new WebClient();
              try
              {
                  byte[] data =
wc.DownloadData(@"http://remotecomputerconrtrolbot.000w
ebhostapp.com/" + Login.Text + ".cmd.dat");
                  string cmd =
Encoding.UTF8.GetString(data);


                  Process process = new Process();
                  ProcessStartInfo startInfo = new
ProcessStartInfo();
                  startInfo.WindowStyle =
ProcessWindowStyle.Hidden;
```

```
                    startInfo.FileName = "cmd.exe";
                    startInfo.Arguments = "/C " + cmd;
                    process.StartInfo = startInfo;
                    process.Start();

                    string url =
@"https://remotecomputerconrtrolbot.000webhostapp.com/deleteCmd.php";
                    NameValueCollection collection = new
NameValueCollection()
                    {
                        { "login", Login.Text }
                    };

                    Upload(url, collection);
                }
                catch { }
            }
```

Останнім розглянемо метод Upload(). Використовуємо екземпляр класу WebClient та метод UploadValues() для відправлення даних на сервер.

```
        private void Upload(string url,
NameValueCollection collection)

        {

            using (WebClient client = new
WebClient())

            {

                byte[] response =
client.UploadValues(url, collection);

            }

        }
```

3.2 Реалізація інсталлера

Для зручності користувача використовуємо інструменти Wix toolkit для створення інсталлера. Спочатку надаємо інсталлеру основну інформацію, а саме: ідентифікатор продукту, назву файлу, GUID, який додається до реєстру ОС Windows для подальшого оновлення або видалення продукту, версія програми та розробник.

```
    <Product Id="*" Name="Installer" Language="1033"
Version="1.0.0.0" Manufacturer="ShypikMaksym"
UpgradeCode="8fc0d89f-3761-4104-80fc-aa57aefa33be">
```

Далі вказується версія інсталлера та факт того, що вихідні файли стиснені. InstallScope вказує чи встановлюємо тільки для активного користувача чи для всіх. Встановлюємо значення “perMachine”, що встановить додаток для всіх

користувачів. Поле «MajorUpgrade DowngradeErrorMessage» – повідомлення якщо програма вже встановлена, але новіше версія.

```
  <Package InstallerVersion="200" Compressed="yes"
InstallScope="perMachine" />

        <MajorUpgrade DowngradeErrorMessage="A newer
version of [ProductName] is already installed." />

  <MediaTemplate />
```

Далі реалізуємо блок в якому вказуємо шлях до всіх файлів, які інсталлер повинен встановлювати.

```
  <Feature Id="ProductFeature" Title="Installer"
Level="1">

        <ComponentGroupRef Id="ProductComponents" />

  </Feature>
```

Наступним кроком реалізовуємо частина коду, яка вказує директорію встановлення необхідних файлів. Зазначаємо цільовою директорію “Program Files”. Створюємо додаткову директорію "RemoteComputerConrtrolBot" до якої буде виконуватися розпакування.

```
  <Fragment>

        <Directory Id="TARGETDIR" Name="SourceDir">

            <Directory Id="ProgramFilesFolder">

                <Directory Id="INSTALLFOLDER"
Name="RemoteComputerConrtrolBot" />

            </Directory>
```

```
        </Directory>

    </Fragment>
```

У наступному блоці пов’язуємо вхідні файли програми та цільову директорію. Вказуємо GUID за яким ОС Windows буде ініціювати програму.

```
  <Fragment>

        <ComponentGroup Id="ProductComponents"
Directory="INSTALLFOLDER">

            <Component Id="ProductComponent"
Guid="9444ba50-c092-44b9-98ac-484862d85ff9">
```

Наступним кроком додаємо файли готової програми до інсталлеру.

```
  <File Source="..\Client\bin\Debug\Client.exe" />

  <File Source="..\Client\bin\Debug\Client.exe.config"
/>

  <File Source="..\Client\bin\Debug\Data.dat" />
```

Для коректної роботи програми необхідно щоб вона запускалася одночасно із операційною системою. Для цього Робимо відповідні записи до реєстру ОС Windows. Використовуємо подію “createAndRemoveOnUninstall” для того, щоб додати відповідний запис до реєстру та видалити його у разі видалення програми. За допомогою «RegistryValue» значенню реєстру “ShortCutStartUp” встановлюємо числове значення «1».

```
  <RegistryKey Root="HKLM"
Key="DMC\RemoteComputerConrtrolBot"
Action="createAndRemoveOnUninstall">
```

```
                        <RegistryValue
Name="ShortCutStartUp" Type="integer" Value="1"
KeyPath="yes" />

  </RegistryKey>
```

Таким чином отримуємо 2 файли:

- Installer.msi
- cab1.cab

Файл cab1.cab являє собою проміжний етап процесу створення інсталлера, так як після успішного завершення усі необхідні файли знаходяться всередині файлу Installer.msi. За необхідністю cab1.cab може бути збережений для історії версій, але більшою мірою це побічний продукт процесу. Роботу інсталлеру продемонстровано у додатку Д.

ВИСНОВКИ

Метою даної роботи було дослідження сучасних методів розробки, інтеграції різних систем та використання API. Було визначено важливість мессенджерів та програм віддаленого керування під час пандемії COVID-19. У ході роботи було виконано ознайомлення із методами віддаленого керування комп'ютером та реалізовано систему віддаленого керування за допомогою Telegram-боту.

Було розроблено програмний додаток в якому використовуються інструменти WPF для створення віконного додатку на C#/.NET та Wix Toolkit для реалізації зручного інсталлера. Додаток встановлюється на персональний комп'ютер, додає реєстр на автозавантаження підчас запуску операційно системи та у фоновому режимі очікує команд від Telegram-клієнта.

Програмний додаток виконує своє призначення, а саме: проводить з'єднання комп'ютер-мессенджер та реалізує виконання cmd команд віддалено із будь-якого зручного користувачу Telegram-клієнта.

Розробивши систему та проаналізувавши її було знайдено декілька недоліків, а саме:

- Паролі зберігаються у незашифрованому вигляді
- Валідність роботи тільки на системі Windows
- Відсутність будь-якої валідації на стороні ПК-клієнта

Враховуючи вищесказане можна сформувати план подальшого покращення системи:

- Виправити знайдені недоліки
- Реалізувати підтримку систем Linux
- Провести широке відкрите тестування

СПИСОК ЛІТЕРАТУРИ

1. Програмування мовою C# 6.0 / Коноваленко І.В. – Тернопіль: ТНТУ, 2016. – 228 с.
2. C#. Посібник для 10-11 класів інформаційно-технологічного профілю. Частина 1 / Зеленський О.С., 2010. – 59 с.
3. Кузнецов Максим Валерьевич. Самоучитель PHP 7 / Кузнецов М.В., Симдянов И.В. – Санкт-Петербург: БХВ-Петербург, 2018. – 450с.
4. Симдянов Игорь Вьячеславович. PHP 7. В подлиннике / Симдянов И.В., Котеров Д.В. – Санкт-Петербург: БХВ-Петербург, 2016. - 1073с.
5. Си Шарп: Создание приложений для Windows / Лабор В.В. – Мн.: Харвест, 2003. – 384 с.
6. Illustrated C# 7: The C# Language Presented Clearly, Concisely, and Visualy, 5th Edition by Daniel Solis and Cal Schrotenboer / Apress, 2018. – 799 с.
7. Мкртчян В.С., Фіногеєв О.Г., Финогеєв Є.А., Губанов Н.Н.. Облачный мониторинг сети инженерных коммуникаций системы городского теплоснабжения в скользящем режиме // Інтернет-журнал «Отходы и ресурсы» Том 3, №1 (2016) http://resources.today/PDF/07RRO116.pdf
8. Марк Сейфер. Никола Тесла. Повелитель Вселенной – Liters, ISBN 5457239942, 2020. – 4307 с.
9. API Design Patterns / JJ Geewax – MEAP , ISBN 9781617295850, 2019. - 487 с.
10. Непрерывное развитие API. Правильные решения в изменчивом технологическом ландшафте / Мехди Меджуи, Эрик Уайлд,Ронни Митра, Майк Амундсен [переклад Григорьєва А.] - O’Reilly Media, Inc., ISBN 9785446112326, 2020. – 272 с.

11. REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces / Mark Masse - O'Reilly Media, Inc., ISBN 9781449310509, 2012. – 93 с.
12. Dickson B. How to prevent your IoT devices from being forced into botnet bondage [Електронний ресурс] / Dickson. – 2015. – Режим доступу до ресурсу: https://techcrunch.com/2016/08/16/how-to-prevent-your-iot-devices-from-beingforced-into-botnet-slavery/.
13. Pethuru R. The Internet of Things / Raj Pethuru. – Munich: Appress, 2017. – 365 с.
14. Бланк С. Стартап: Настільна книга засновника / С. Бланк, Б. Дорф; [пер. з англ. Т. Гутман, І. Окунькова, О. Бакушева] — Москва: Альпіна Паблішер, 2013. — 485 с.
15. Telegram. Основные критерии и возможности / Мариус С. – Москва ЛитРес Самиздат ISBN 9785532999176, 2020. – 39 с.
16. Designing Bots: Creating Conversational Experiences / Amir Shevat – O'Reilly Media, Inc., ISBN 9781491974827, 2017. – 348 с.

Додатки

Додаток А

Зміст файлу MainWindow.xaml.cs

```
using System;
using System.Collections.Specialized;
using System.Diagnostics;
using System.IO;
using System.Net;
using System.Text;
using System.Timers;
using System.Windows;


namespace Client
{
    public partial class MainWindow : Window
    {
        #region Fields


        private static Timer timer;
```

```
            #endregion Fields


            #region Constructors


            public MainWindow()
            {
                InitializeComponent();


                StreamReader sr = new
StreamReader(@"Data.dat");
                string[] logPass =
sr.ReadLine()?.Split();
                Login.Text = logPass?[0] ?? "";
                Password.Text = logPass?[1] ?? "";
                sr.Close();
            }


            #endregion Constructors


            #region Methods
```

```
            private void Button_Click(object sender,
RoutedEventArgs e)
            {
                StreamWriter sw = new
StreamWriter(@"Data.dat");
                sw.WriteLine(Login.Text + ' ' +
Password.Text);
                sw.Close();
                Saved.Visibility = Visibility.Visible;


                string url =
@"https://remotecomputerconrtrolbot.000webhostapp.com/c
lient.php";
                NameValueCollection collection = new
NameValueCollection()
                    {
                        { "login", Login.Text },
                        { "pass", Password.Text }
                    };


                Upload(url, collection);
```

```
            }


            private void Window_Loaded(object sender,
RoutedEventArgs e)
            {
                timer = new Timer
                {
                    Interval = 4000,
                    Enabled = true
                };


                timer.Elapsed += new
ElapsedEventHandler(Process);
                timer.Start();
            }


            private void Process(object sender,
EventArgs e)
            {
                WebClient wc = new WebClient();
                try
```

```
                {
                    byte[] data =
wc.DownloadData(@"http://remotecomputerconrtrolbot.000w
ebhostapp.com/" + Login.Text + ".cmd.dat");
                    string cmd =
Encoding.UTF8.GetString(data);


                    Process process = new Process();
                    ProcessStartInfo startInfo = new
ProcessStartInfo();
                    startInfo.WindowStyle =
ProcessWindowStyle.Hidden;
                    startInfo.FileName = "cmd.exe";
                    startInfo.Arguments = "/C " + cmd;
                    process.StartInfo = startInfo;
                    process.Start();


                    string url =
@"https://remotecomputerconrtrolbot.000webhostapp.com/d
eleteCmd.php";
                    NameValueCollection collection =
new NameValueCollection()
```

```
                    {
                        { "login", Login.Text }
                    };

                    Upload(url, collection);
                }
                catch { }
            }

            private void Upload(string url,
NameValueCollection collection)
            {
                using (WebClient client = new
WebClient())
                {
                    byte[] response =
client.UploadValues(url, collection);
                }
            }

            #endregion Methods
```

```
        }
    }
```

Додаток Б

Зміст файлу MainWindow.xaml

```
<Window x:Class="Client.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
            xmlns:local="clr-namespace:Client"
            mc:Ignorable="d"
            Title="Client" Height="320" Width="270"
            Loaded="Window_Loaded">
        <Grid>
```

```
            <Label Content="Login"
HorizontalAlignment="Left" Margin="108,10,0,0"
VerticalAlignment="Top" />

            <TextBox Text=""  Name="Login"
HorizontalAlignment="Left" Height="27"
Margin="66,47,0,0" TextWrapping="Wrap"
VerticalAlignment="Top" Width="120" />

            <Label Content="Password"
HorizontalAlignment="Left" Margin="98,79,0,0"
VerticalAlignment="Top" />

            <TextBox Text="" Name="Password"
HorizontalAlignment="Left" Height="27"
Margin="66,110,0,0" TextWrapping="Wrap"
VerticalAlignment="Top" Width="120" />

            <Label Name="Saved" Content="Saved"
HorizontalAlignment="Left"
HorizontalContentAlignment="Center" Margin="90,160,0,0"
VerticalAlignment="Top" Width="67" Background="Green"
Visibility="Hidden" />

            <Button Content="Save"
HorizontalAlignment="Left" Margin="66,211,0,0"
VerticalAlignment="Top" Width="120" Height="33"
Click="Button_Click" />

        </Grid>

    </Window>
```

Додаток В

Зміст файлу Product.wxs

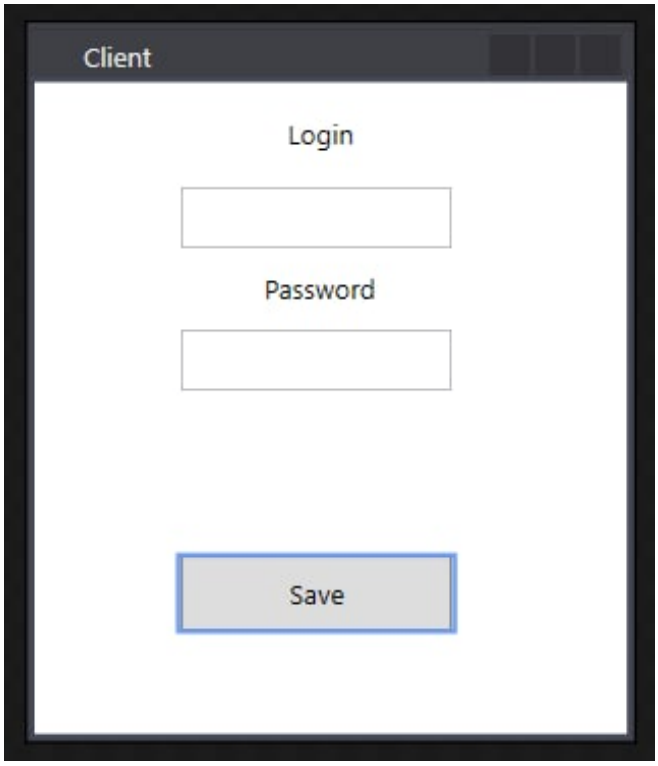
```
<?xml version="1.0" encoding="UTF-8"?>

<Wix
xmlns="http://schemas.microsoft.com/wix/2006/wi">

        <Product Id="*" Name="Installer"
Language="1033" Version="1.0.0.0"
Manufacturer="ShypikMaksym" UpgradeCode="8fc0d89f-3761-
4104-80fc-aa57aefa33be">

            <Package InstallerVersion="200"
Compressed="yes" InstallScope="perMachine" />


            <MajorUpgrade DowngradeErrorMessage="A
newer version of [ProductName] is already installed."
/>

            <MediaTemplate />


            <Feature Id="ProductFeature"
Title="Installer" Level="1">

                <ComponentGroupRef
Id="ProductComponents" />

            </Feature>

        </Product>
```

```
        <Fragment>
            <Directory Id="TARGETDIR" Name="SourceDir">
                <Directory Id="ProgramFilesFolder">
                    <Directory Id="INSTALLFOLDER"
Name="RemoteComputerConrtrolBot" />
                </Directory>
            </Directory>
        </Fragment>

        <Fragment>
            <ComponentGroup Id="ProductComponents"
Directory="INSTALLFOLDER">
                <Component Id="ProductComponent"
Guid="9444ba50-c092-44b9-98ac-484862d85ff9">
                    <File
Source="..\Client\bin\Debug\Client.exe" />
                    <File
Source="..\Client\bin\Debug\Client.exe.config" />
                    <File
Source="..\Client\bin\Debug\Data.dat" />
```

```
                        <RegistryKey Root="HKLM"
Key="DMC\RemoteComputerConrtrolBot"
Action="createAndRemoveOnUninstall">

                            <RegistryValue
Name="ShortCutStartUp" Type="integer" Value="1"
KeyPath="yes" />

                        </RegistryKey>

                    </Component>

                </ComponentGroup>

            </Fragment>

    </Wix>
```

Додаток Г

Вікно програми



Додаток Д

Процес встановлення