

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДЗ «ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ІМЕНІ ТАРАСА ШЕВЧЕНКА»**

Навчально-науковий інститут математики та інформаційних  
технологій

(назва факультету, інституту)

Інформаційних технологій та систем

(назва кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

**БАКАЛАВРА**

(освітньо-кваліфікаційний рівень)

на тему: **РОЗРОБКА КЛІЄНТСЬКОГО ДОДАТКУ ВІДДАЛЕНОГО  
ДОСТУПУ НА ПЛАТФОРМІ .NET**

Виконав: студент 4 курсу  
напряму підготовки (спеціальності)  
121 «Інженерія програмного  
забезпечення»

(шифр і назва напряму підготовки, спеціальності)

Нєвєдров Д. О.

(прізвище та ініціали)

Керівник Донченко В. Ю.

(прізвище та ініціали)

Рецензент Козуб Ю. Г.

(прізвище та ініціали)

Полтава – 2024 року

## **ЗМІСТ**

<b>ВСТУП.....</b>	<b>3</b>
<b>РОЗДІЛ 1. АНАЛІЗ ТЕХНОЛОГІЇ ВІДДАЛЕНОГО ДОСТУПУ ДО РЕСУРСІВ КОМП'ЮТЕРУ .....</b>	<b>5</b>
1.1. Поняття та принципи функціонування віддаленого доступу.....	5
1.2. Аналіз протоколів віддаленого доступу .....	8
1.3. Аналіз додатків віддаленого доступу.....	16
Висновки до розділу .....	22
<b>РОЗДІЛ 2. РОЗРОБКА КЛІЄНТСЬКОГО ДОДАТКУ ВІДДАЛЕНОГО ДОСТУПУ НА ПЛАТФОРМІ .NET .....</b>	<b>24</b>
2.1. Характеристика платформи .NET та обґрунтування інструментів розробки додатку .....	24
2.2. Розробка програмного додатку.....	29
2.3. Тестування програмного додатку.....	42
Висновок до розділу .....	50
<b>ВИСНОВКИ .....</b>	<b>52</b>
<b>СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ .....</b>	<b>54</b>
<b>ДОДАТОК.....</b>	<b>57</b>

## ВСТУП

В даний час можливість повноцінного доступу до віддаленого комп'ютера, що знаходиться в мережі інтернет або локальної мережі, є актуальною як серед домашніх користувачів, так і серед корпоративних. Застосування технології віддаленого робочого в даний момент дуже широке: допомога користувачам (технічна підтримка), доступ до робочого столу комп'ютера з дому (або навпаки), відрядження, спостереження за роботою користувачів (наприклад, вчитель у навчальному класі).

Віддалений доступ - дуже широке поняття, яке включає в себе різні типи і варіанти взаємодії комп'ютерів, мереж і додатків. Існує величезна кількість схем взаємодії, які можна назвати віддаленим доступом, але їх об'єднує використання глобальних каналів або глобальних мереж при взаємодії. Крім того, для віддаленого доступу, як правило, характерна несиметричність взаємодії, тобто з одного боку є центральна велика мережа або центральний комп'ютер, а з іншого – окремий віддалений термінал, комп'ютер або невелика мережа, які повинні отримати доступ до інформаційних ресурсів центральної мережі.

Віддалений доступ може здійснюватися в межах однієї операційної системи або з різними платформами на клієнті й сервері, при якому користувач отримує можливість віддалено працювати з комп'ютером так само, як би він керував ним за допомогою локально підключеного терміналу. У цьому режимі він може запускати програми на віддаленому комп'ютері і бачити результати їх виконання. Тому саме розробка засобів віддаленого управління стала основною метою роботи.

**Об'єктом дослідження** є технології віддаленого доступу до ресурсів комп'ютера.

**Предметом дослідження** є клієнтський додаток віддаленого доступу на платформі .NET.

**Метою роботи** є аналіз технологій віддаленого доступу та розробка клієнтського додатку на платформі .NET.

Відповідно до предмета, мети було визначено основні **завдання дослідження**:

- Дослідження поняття та принципів функціонування віддаленого доступу;
- аналіз протоколів віддаленого доступу;
- аналіз додатків віддаленого доступу;
- огляд платформи .NET та обґрунтування інструментів розробки додатку;
- розробка програмного додатку віддаленого доступу на платформі .NET.

Для вирішення завдань дослідження використано такі **методи дослідження**: *теоретичні*: аналіз наукової літератури, узагальнення та систематизація теоретичних положень про технології віддаленого доступу до ресурсів комп'ютеру; *емпіричні*: порівняний аналіз додатків віддаленого доступу та можливостей інструментів розробки програмних додатків; *експериментальні*: тестування розробленої системи.

До складу роботи входять два розділи, в яких проаналізовано технології віддаленого доступу та обґрунтовано вибір протоколів RDP і VNC для клієнтського додатку.

Практичним результатом роботи є розроблений клієнтський додаток віддаленого доступу на платформі .NET.

## **РОЗДІЛ 1. АНАЛІЗ ТЕХНОЛОГІЇ ВІДДАЛЕНОГО ДОСТУПУ ДО РЕСУРСІВ КОМП'ЮТЕРУ**

### **1.1. Поняття та принципи функціонування віддаленого доступу**

*Віддалений доступ* являє собою функцію, що дозволяє користувачеві підключатися до комп'ютера через Інтернет за допомогою іншого ПК. Умовою для застосування такої опції є включений комп'ютер, до якого потрібно підключитися, а також встановлена і запущена функція віддаленого доступу. Здійснити таке з'єднання можна за допомогою будь-якого ПК, приєднаного до мережі [3].

Опція відкриває можливість користувачу використовувати свій комп'ютер віддалено, і забезпечує наступні додаткові можливості:

- Повний або частковий доступ до робочого столу, дисків і файлів користувача. Додатково налаштовується можливість виконання операцій видалення, копіювання і т. д.
- Організація голосового чату при наявності відповідного обладнання. Немає необхідності в додатковій телефонного зв'язку.
- Обмін файлами будь-якого формату без обмеження в обсязі.
- Запуск презентаційних матеріалів в режимі конференції.
- Відстеження і запис використання комп'ютера користувачами.
- Збір інформації про відвідані сайти, вхідних і вихідних електронних листах, використанні месенджерів.

Існують кілька видів віддаленого доступу:

- комп'ютер-мережа, що дозволяє контролювати роботу локальної мережі офісу або інтернет-кафе;
- термінал-комп'ютер, який спрощує зв'язок користувача з системою, наприклад, платіжні термінали банків;
- комп'ютер-комп'ютер, встановлює зв'язок між двома віддаленими комп'ютерами;
- мережа-мережа, відмінний інструмент при необхідності взаємодії між віддаленими корпоративними мережами.

Одним з видів віддаленого доступу є доступ до *віддаленого робочого столу (Remote Desktop)* - це термін, яким позначається режим управління, коли один комп'ютер отримує права адміністратора по відношенню до іншого, віддаленого. Зв'язок між пристроями відбувається в реальному часі за допомогою Інтернет або локальної мережі.

Рівень доступу в режимі віддаленого адміністрування визначається конкретними завданнями і може бути змінений за потребою. наприклад:

- в одному випадку, підключення до робочої сесії дає можливість повного контролю і взаємодії з віддаленим комп'ютером, при якому допускається запуск на ньому програм і маніпуляції з файлами;
- в іншому, віддалений доступ до робочого столу дозволяє лише вести спостереження за процесами, без втручання в роботу його системи.

Доступ до віддаленого робочого столу здійснюється на базі архітектури «сервер - термінал», що дозволяє працювати з ресурсами сервера так само, як і з локальними ресурсами: виконувати команди, працювати з файловою системою, запускати додатки і т.п.

Початково, термінал - це кінцевий мережевий пристрій, підключений до обчислювальної системи і призначений для введення і виведення даних. Команди, що приймаються з пристроєм введення терміналу (клавіатури), передаються на віддалений сервер, де і виконуються. Результати обробки повертаються і відображаються на пристрої виведення терміналу (дисплеї). Згодом були розроблені емулятори терміналів - спеціальні програми, що виконують ті ж завдання.

Таким чином, можна виділити два основних типи терміналів:

Реальний, або фізичний термінал - як правило має на увазі пристрій, обчислювальні можливості якого обмежені можливістю відображати те, що йому передано з мережі (як максимум - повноекранну графіком).

Віртуальний термінал - мережевий додаток (програма), що виконує

функції фізичного терміналу. З боку віддаленого хоста така програма, нічим не відрізняється від реального терміналу.

Термінальний доступ - доступ до інформаційної системи (або ПК тощо), організований так, що локальна машина-термінал не виконує обчислювальної роботи, а лише здійснює перенаправлення вводу інформації (від миші і клавіатури) на центральну машину (термінальний сервер) і відображає графічну інформацію на монітор (рис. 1.1). Причому вся обчислювальна робота в термінальній системі виконується на центральній машині.



Рис. 1.1. Технологія термінального доступу

Віддалене адміністрування - попередньо встановлена функція практично в кожній відомій сьогодні операційній системі, одночасно з цим, існує досить велика кількість програм, які роблять цей процес більш зручним, додають в стандартні версії нові функції.

Так, в операційній системі Windows вбудована можливість використання віддаленого робочого столу із застосуванням статичних IP-адресів на віддаленому комп'ютері. Налаштування віддаленого робочого столу (рис. 1.2) потребує прав адміністратора.

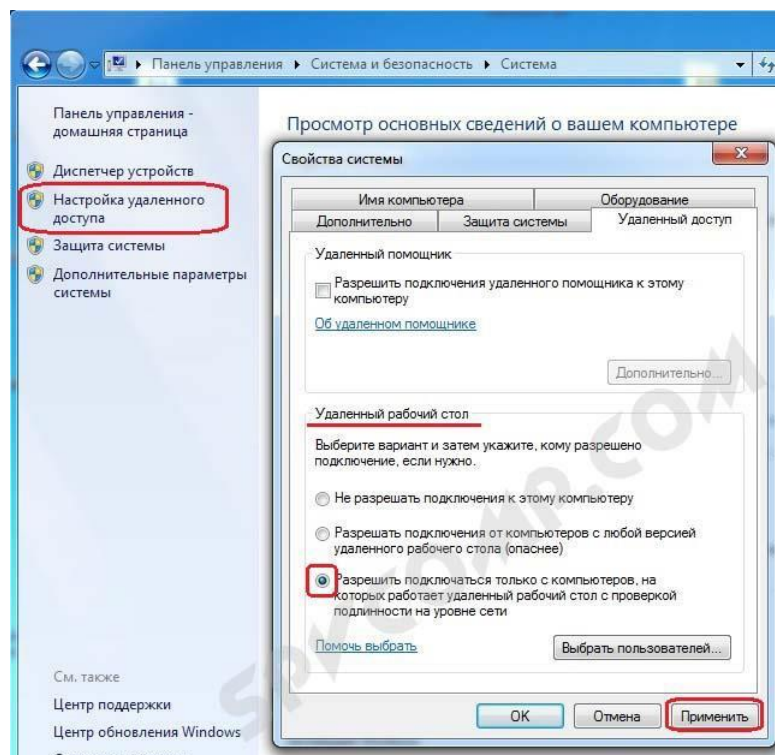


Рис. 1.2. Приклад вікна налагодження віддаленого робочого столу

Таким чином, застосування можливостей віддаленого доступу дає більш широкі можливості в корпоративному використанні ресурсів персонального комп'ютеру.

## 1.2. Аналіз протоколів віддаленого доступу

Принцип роботи протоколів віддаленого доступу полягає в наступному: для виконання програм на сервері емулюється пристрій виведення зображення (GDI). Програма виконує звичайний вивід на екран, а протоколи упаковують ці дані у зручний для передачі по мережі формат, і передають клієнту. Клієнт здійснює форматування команд і виконує на локальному екрані користувача. У зворотному порядку всі дії користувача (рухи миші, натискання клавіш) передаються від програми на тонкому клієнті до сервера, де відповідний драйвер імітує ситуацію як би дії приходили з локальної миші та клавіатури.

### Протокол RDP

RDP (англ. Remote Desktop Protocol) – протокол прикладного рівня, який використовується для забезпечення віддаленої роботи користувача з сервером, на якому запущений сервіс термінальних підключень Microsoft Windows Remote Desktop Services CAL (RDS, попередня назва Terminal Services) [21].



Клієнти існують практично для всіх версій Windows (включаючи Windows CE та Mobile), Linux, FreeBSD, Mac OS X [5]. Протокол RDP реалізований у вигляді COM-компонента (Component Object Model – модель компонентних об'єктів) Microsoft RDP Client Control, який поставляється в складі операційної системи MS Windows. Протокол є прикладним, що базується на TCP. За замовчуванням використовується порт TCP 3389.

Microsoft передбачає два режими використання протоколу RDP:

- для адміністрування (режим віддаленого адміністрування). RDP в режимі адміністрування, використовується всіма сучасними операційними системами Microsoft Windows
- для доступу до терміналів сервера (режим TerminalServer) [8].

Режим доступу до сервера терміналів можливий тільки в серверних версіях Windows.

Серверні версії Windows підтримують одночасно два віддалених підключення і один локальний вхід в систему, в той час як клієнтські - тільки один вхід (локальний або віддалений) .

Можливість віддаленого доступу до сервера терміналів відкривається тільки після установки відповідних ліцензій на License server (рис. 1.3).

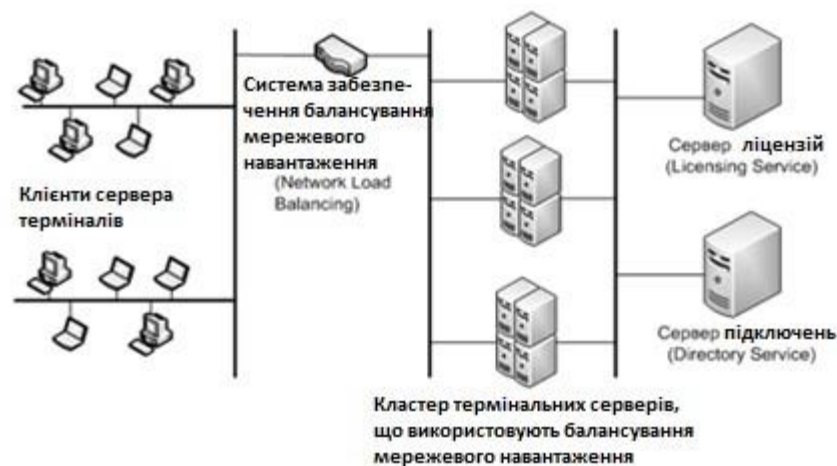


Рис. 1.3. RDP в режимі доступу до сервера терміналів

При використанні кластера термінальних серверів і балансування навантаження потрібна установка спеціалізованого сервера підключень

(Session Directory Service). Даний сервер індексує сесії користувачів, що дозволяє виконувати вхід, а також повторний вхід на термінальні сервери, що працюють в розподіленому середовищі.

Після установки з'єднання на транспортному рівні ініціалізується RDP-сесія, в рамках якої узгоджуються різні параметри передачі даних. Передача виведення за допомогою примітивів є пріоритетною для протоколу RDP, так як значно економить трафік; а зображення передається лише в тому випадку, якщо інше неможливо з яких-небудь причин (не вдалося узгодити параметри передачі примітивів при установці RDP-сесії) [21]. RDP клієнт обробляє отримані команди і виводить зображення за допомогою своєї графічної підсистеми (рис. 1.4).

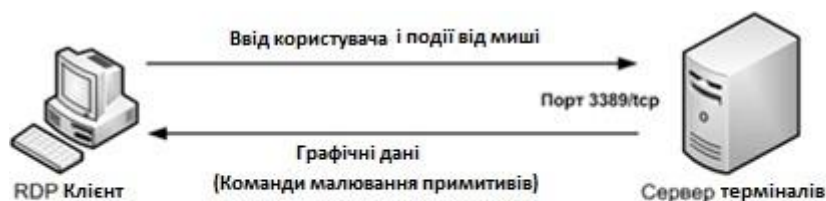


Рис. 1.4. Принцип роботи RDP

RDP підтримує кілька віртуальних каналів в рамках одного з'єднання, які можуть використовуватися для забезпечення додаткового функціоналу:

- використання принтера або послідовного порту;
- перенаправлення файлової системи;
- підтримка роботи з буфером обміну;
- використання аудіо-підсистеми.

Характеристики віртуальних каналів узгоджуються на етапі встановлення з'єднання.

Робота протоколу при стандартних налаштуваннях стиснення і глибині кольору 32 біт займає максимум 160 Кбайт / сек ширини пропускання каналу, і дана цифра зменшується зі зниженням глибини кольору. При глибині кольору 8 біт для роботи протоколу необхідно максимум 60 Кбайт / сек [6].

Забезпечення безпеки при використанні RDP (рис.1.5). Специфікація протоколу RDP передбачає використання одного з двох підходів до забезпечення безпеки:

- Standard RDP Security (вбудована підсистема безпеки);
- Enhanced RDP Security (зовнішня підсистема безпеки).



Рис. 1.5. Забезпечення безпеки при використанні RDP

*Standard RDP Security.* При цьому підході аутентифікація, шифрування і забезпечення цілісності реалізується засобами, закладеними в RDP протокол. Розглянемо ці етапи докладніше.

Аутентифікація сервера виконується наступним чином:

1. при старті системи генерується пара RSA ключів;
2. створюється сертифікат (Proprietary Certificate) відкритого ключа;
3. сертифікат підписується RSA-ключем, вбудованим в операційну систему (будь RDPклієнт містить відкритий ключ даного вбудованого RSA- ключа);
4. клієнт підключається до сервера терміналів і отримує підписаний сертифікат;
5. клієнт перевіряє сертифікат і отримує відкритий ключ сервера (даний ключ використовується в подальшому для узгодження параметрів шифрування).

Аутентифікація клієнта проводиться при введенні імені користувача та

пароля.

В якості алгоритму шифрування обраний потоковий шифр RC4. Залежно від версії операційної системи доступні різні довжини ключа від 40 до 168 біт.

При установці з'єднання після узгодження довжини генерується два різних ключа: для шифрування даних від клієнта і від сервера.

Цілісність повідомлення досягається застосуванням алгоритму генерації MAC (Message Authentication Code) на базі алгоритмів MD5 і SHA1.

Починаючи з Windows 2003 Server, для забезпечення сумісності з вимогами стандарту FIPS (Federal Information Processing Standard) 140-1 можливе використання алгоритму DES для шифрування повідомлень і алгоритму генерації MAC, що використовує тільки SHA1, для забезпечення цілісності.

*Enhanced RDP Security.* В даному підході використовуються зовнішні модулі забезпечення безпеки: TLS 1.0 і CredSSP [5].

Протокол TLS можна використовувати, починаючи з версії Windows 2003 Server, але тільки якщо його підтримує RDP-клієнт. Підтримка TLS додана, починаючи з RDP-клієнта версії 6.0

При використанні TLS, сертифікат сервера можна генерувати засобами Terminal Services або вибрати існуючий сертифікат зі сховища Windows. Протокол CredSSP є поєднання функціоналу TLS, Kerberos і NTLM.

Розглянемо основні переваги протоколу CredSSP [5]:

- перевірка дозволу на вхід в віддалену систему до установки повноцінного RDP-з'єднання, що дозволяє економити ресурси сервера терміналів при великій кількості підключень;
- надійна аутентифікація і шифрування по протоколу TLS;
- використання одноразового входу в систему (Single Sign On) за допомогою Kerberos або NTLM.

## **Протокол VNC**

VNC (англ. Virtual Network Computing) - система віддаленого доступу, що базується на концепції RFB (англ. Remote FrameBuffer, віддалений кадровий

буфер) [8] (рис. 1.6).

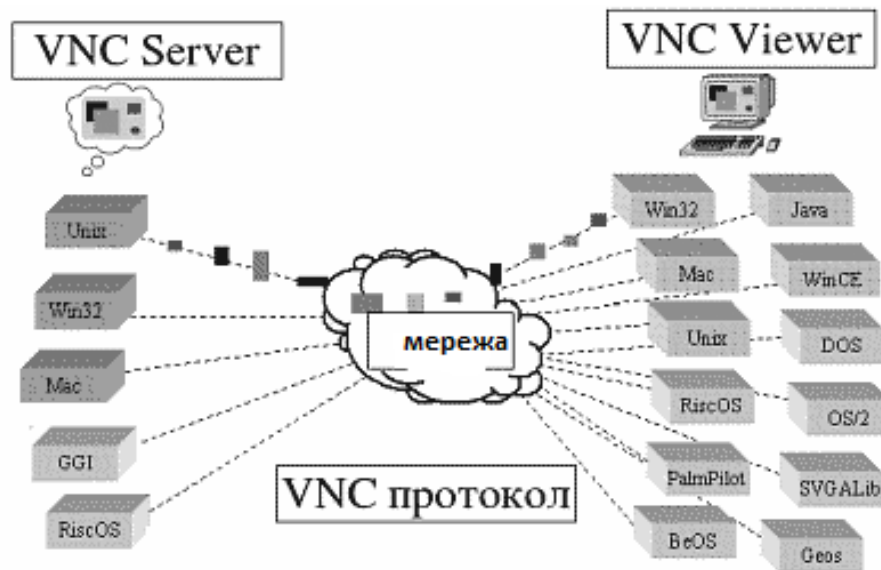


Рис. 1.6. VNC система віддаленого доступу

Система VNC є платформонезалежною: VNC-клієнт, званий VNC viewer, запущений на одній операційній системі, може підключатися до VNC-сервера, що працює на будь-якій іншій ОС. У протоколі VNC реалізований наступний принцип роботи: дані про натискання клавіш і рух миші, що виконуються на власному комп'ютері, передаються по мережі на віддалений комп'ютер і сприймаються ним як дії з його власними клавіатурою і мишкою. Схема виведення з віддаленого комп'ютера на екран користувача ґрунтується на графічних примітивах (прямокутник піксельних даних виводиться в заданій координатах позиції) в своїй примітивній формі споживає більшу частину пропускної можливості каналу. Існують різні кодування - методи визначення найбільш ефективного способу передачі цих прямокутників. Протокол VNC дозволяє клієнту і серверу «домовитися» про те, яке кодування буде використане.

Найпростіший метод кодування, що підтримується всіма клієнтами і серверами - «raw encoding», при якому пікселі передаються в порядку зліва-направо, зверху-вниз, і після передачі початкового стану екрану передаються тільки пікселі, що змінилися. Цей метод працює дуже добре при незначних змінах зображення на екрані (руху покажчика миші по робочому столу, набір

тексту під курсором), але завантаження каналу стає дуже високою при одночасному зміні великої кількості пікселів, наприклад, при перегляді відео в повноекранному режимі.

Для роботи потрібна ширина пропускання каналу становить від 32 Кбіт/сек до 2 Мбіт / сек. Для комфортної роботи в кольоровому режимі при дозволі екрану 1024x768 швидкість каналу повинна бути 1-2Мбіт / сек.

Канал використовується повністю тільки при оновленні великих ділянок екрану, при друку тексту трафік помітно менше, а в решту часу канал практично не використовується. Якщо при передачі по каналу виникають великі затримки передачі пакетів (повільні канали, супутниковий зв'язок, великі відстані), це викликає погіршення часу реакції на натискання клавіш і рух миші, що значно знижує комфортність роботи.

*Забезпечення безпеки при використанні VNC.* Метод забезпечення безпеки (security type) визначається на стадії «рукоштовування», після узгодження версії протоколу, який буде використовуватися. Специфікація протоколу описує наступні методи забезпечення безпеки:

- відсутність аутентифікації і шифрування трафіку;
- VNC аутентифікація - шифрування трафіку не використовується, однак пароль не передається у відкритому вигляді, а використовується алгоритм «виклик відповідь» з DES-шифруванням (ефективна довжина ключа становить 56-біт).

Також багато сучасних реалізації VNC підтримують розширення стандартного протоколу, які реалізують шифрування і / або стиснення VNC трафіку, розмежування за списками доступу ACL і різні методи аутентифікації.

Даним протоколом надаються такі технології і можливості [9]:

- обмін файлами;
- вагоме збільшення швидкості відтворення екрану віддаленого комп'ютера, в локальній мережі;
- підключення плагіна шифрування для поліпшення безпечного

обміну даними;

- підтримка доменної аутентифікації;
- підтримка чату з віддаленим комп'ютером для обміну повідомленнями;
- ViewerToolbar, JavaViewer з підтримкою передачі файлів;
- авто масштабування для підгонки розміру зображення віддаленого комп'ютера;
- підтримка декількох моніторів;
- Repeater / Proxy-support;
- друк файлів з VNC-сервера на принтер за замовчуванням, підключений до комп'ютера VNC клієнта;
- можливість підключення VNC клієнта через різні web-проксі сервера і фільтри, що робить його використання таким же простим як і використання браузера.

### **Протокол ICA**

ICA (англ. Independent Computing Architecture) - це закритий протокол для сервера додатків, розробленого компанією Citrix Systems. Протокол визначає специфікацію обміну даними між сервером і клієнтами, але не вбудований ні в одну з платформ. ICA виконує завдання, багато в чому схожі з X Window System [10].

Протокол ICA забезпечує стандартну підтримку графічного виведення і введення даних з клавіатури, миші. Базовий механізм передачі між сервером і клієнтом графічних даних і даних, що вводяться з клавіатури / миші, по протоколу ICA майже ідентичний протоколу RDP.

*Забезпечення безпеки при використанні ICA.* Так як специфікація протоколу є закритою, слід говорити про шифрування в продуктах на основі даного протоколу. Наприклад, Citrix Metaframe поставляється тільки з мінімальними функціями шифрування для ICA-підключень і підтримує тільки режим найпростішого шифрування, який задіює простий експортований алгоритм з менш ніж 40-розрядним ключем. Якщо необхідно організувати

захист для підключень ICA, то слід придбати додатковий пакет Secure ICA для Meta Frame. У Secure ICA використаний алгоритм шифрування RSA RC5 і підтримуються 40-, 56- і 128-розрядні ключі.

Для шифрування комунікація між сервером і клієнтом ICA може використовуватися CitrixSSL Relay, який забезпечує шифрування по протоколу Secure Sockets Layer / Transport LayerSecurity (SSL / TLS) [10].

На підставі проведеного аналізу було вирішено обрати для розробки проекту протоколи RDP і VNC, оскільки їх застосування в одному клієнтському додатку дозволить реалізувати віддалений доступ не тільки до комп'ютерів з операційною системою Windows, а також дасть можливість застосовувати різні платформи на клієнті і сервері віддаленого доступу. Крім того, дані протоколи забезпечують високий рівень безпеки, також для цих протоколів існує доступний інтерфейс програмування, що спрощує розробку клієнтських додатків.

### **1.3. Аналіз додатків віддаленого доступу**

В даний час існує величезна кількість різних інструментів, які дозволяють дистанційно керувати комп'ютером або сервером. Кожен з них має свої плюси і мінуси, різні функціональні можливості, платні або безкоштовні версії, а також мобільні або тільки десктопні варіанти.

Проведемо порівняльний аналіз найбільш популярних програм віддаленого доступу, - *TeamViewer*, *LiteManager*, *Ammy admin*, *RAdmin*, виділимо їх переваги та недоліки. Аналіз основних порівняльних характеристик подано у таблиці 1.1.

#### **TeamViewer**

Одна з найпопулярніших програм для віддаленого доступу, її можна швидко завантажити і встановити або відразу запустити, без установки. При запуску програма відображає вікно з ID і паролем для доступу до комп'ютера, а також TeamViewer дозволяє підключитися до іншого комп'ютера задавши його ID і пароль (рис. 1.7).



Таблиця 1.1

## Порівняльний аналіз програм віддаленого доступу

Характеристики	Назва програм віддаленого доступу			
	TeamViewer	LiteManager	Ammy admin	RAdmin
Протокол	Власне ПЗ	Власне ПЗ, RDP	Власне ПЗ, RDP	Власне ПЗ
Режими роботи	Клієнт & Сервер	Клієнт & Сервер & Gateway	Клієнт & Сервер	Клієнт & Сервер &
Вбудоване шифрування	AES-256	AES-256, RSA-2048	AES-256, RSA	AES-256
Передача файлів	+	+	+	+
Передача звуку	+	+	+	+
Багатоклієнтний режим	+	+	+	+
Віддалений помічник	+	+	+	+
Запит прав доступу	+	+	+	+
Linux клієнт	+	-	-	+
Mac OS клієнт	+	+	-	-
MS Windows клієнт	+	+	+	+
Android клієнт	+	+	-	-
Фіксований ID	+	+	+	-

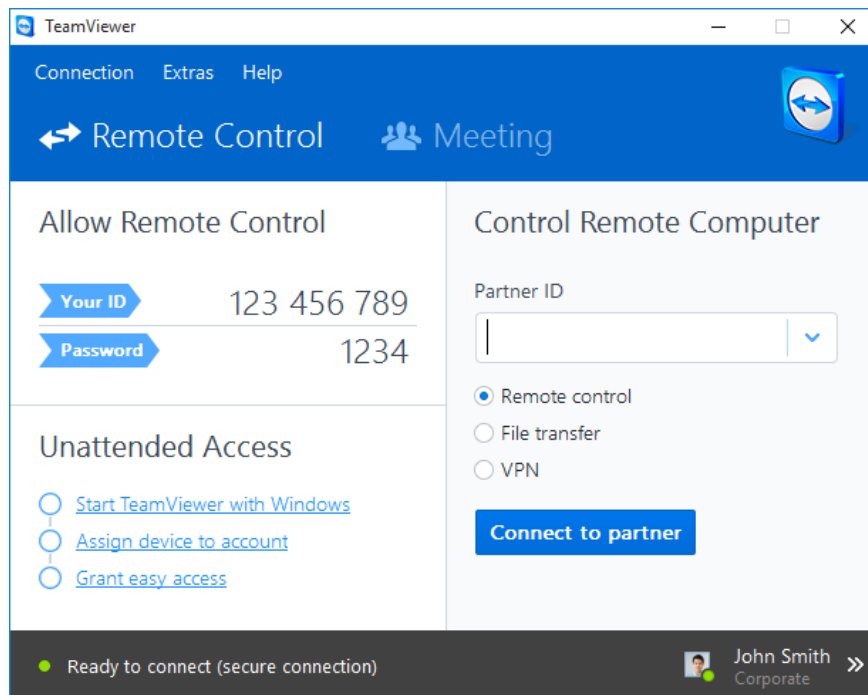


Рис. 1.7. Діалогове вікно програми TeamViewer

### *Переваги.*

У програмі є кілька основних режимів роботи - це віддалене управління, передача файлів, чат, демонстрація свого робочого столу. Програма дозволяє налаштувати цілодобовий доступ до комп'ютера, це буде зручно для системного адміністрування. Є версії для всіх мобільних платформ, для різних операційних систем. Простий і цілком зрозумілий інтерфейс плюс ряд додаткових утиліт для розширення функціоналу програми, будуть корисні для служб віддаленої підтримки.

### *Недоліки.*

Програма є безкоштовною тільки для некомерційного використання. При роботі з програмою більше 5 хвилин можуть виникнути труднощі, наприклад TV може заблокувати сеанс віддаленого підключення, розпізнавши його як комерційне використання. Для цілодобового віддаленого доступу або адміністрування декількох комп'ютерів, комп'ютерної мережі, доведеться платити за додаткові модулі програми. Вартість програми висока.

### *Підсумок.*

Дана програма ідеально підходить для разового віддаленого підключення або використання її нетривалий період часу. Зручно

використовувати з мобільних платформ, але не адмініструвати велику кількість комп'ютерів. За додаткові модулі доведеться доплачувати.

### LiteManager

Проста, але досить таки потужна по можливостях програма, складається з двох частин, перша це Server який потрібно встановити або запустити на віддаленому комп'ютері і Viewer, який дозволяє керувати іншим комп'ютером. Для роботи програма вимагає трохи більше навичок і досвіду від користувача, хоча робота сервером навіть простіше ніж в TeamViewer, сервер можна один раз встановити і більше не яких дій від користувача не потрібно, ID буде завжди постійний, його навіть можна задати самому, що дуже зручно для запам'ятовування. Версія LiteManager Free є безкоштовною для особистого та комерційного використання (рис. 1.8).

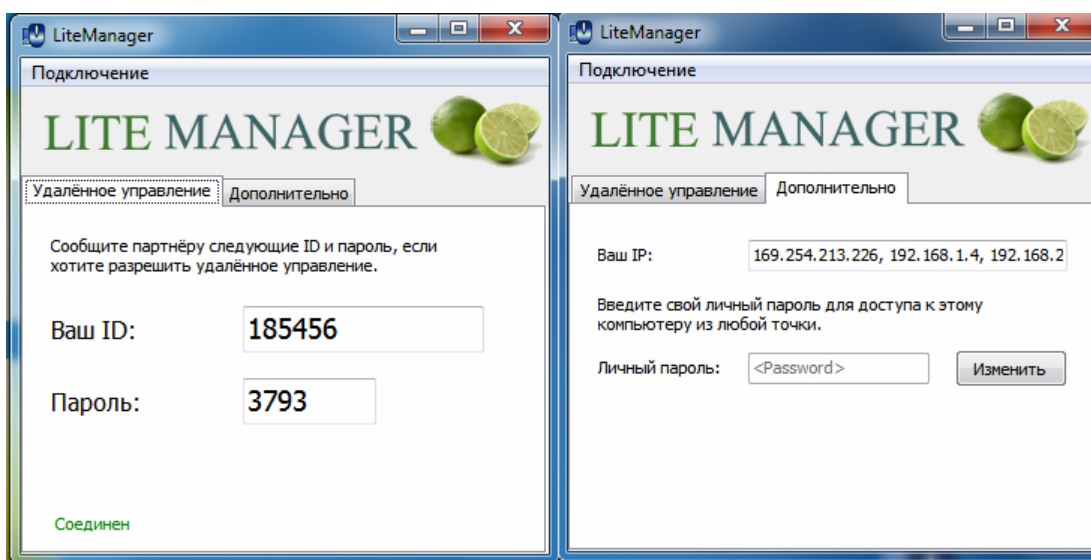


Рис. 1.8. Діалогове вікно програми LiteManager

### *Переваги.*

У програмі крім основних режимів віддаленого доступу (віддаленого управління, передачі файлів, чату, диспетчера задач, редактора реєстру) є специфічні функції, наприклад: інвентаризація, запис екрану, віддалена установка. Програма безкоштовна для використання на 30-ти комп'ютерах, її можна використовувати для цілодобового доступу без будь-яких додаткових модулів. Відсутні будь-які обмеження по часу роботи. Є можливість налаштування свого власного ID сервера для налаштування корпоративної

служби підтримки. У програмі немає жодних обмежень за часом роботи і блокувань.

#### *Недоліки.*

Бракує клієнта під мобільні платформи або інші системи, є обмеження на 30 комп'ютерів в безкоштовній версії, для адміністрування більшої кількості необхідно придбати ліцензію. Деякі, специфічні режими роботи доступні тільки в Pro версії.

#### *Підсумок.*

Програма LiteManager підійде для надання віддаленої підтримки, для адміністрування декількох десятків комп'ютерів абсолютно безкоштовно, для настройки власної служби віддаленої підтримки. Вартість програми найнижча в своєму сегменті і ліцензія не обмежена за часом.

#### **Ammy admin**

Програма в основному аналогічна TeamViewer, але більш простий варіант. Присутні тільки основні режими роботи - перегляд і управління, передача файлів, чат. Програма може працювати без установки, безкоштовна для некомерційного використання (рис. 1.9).

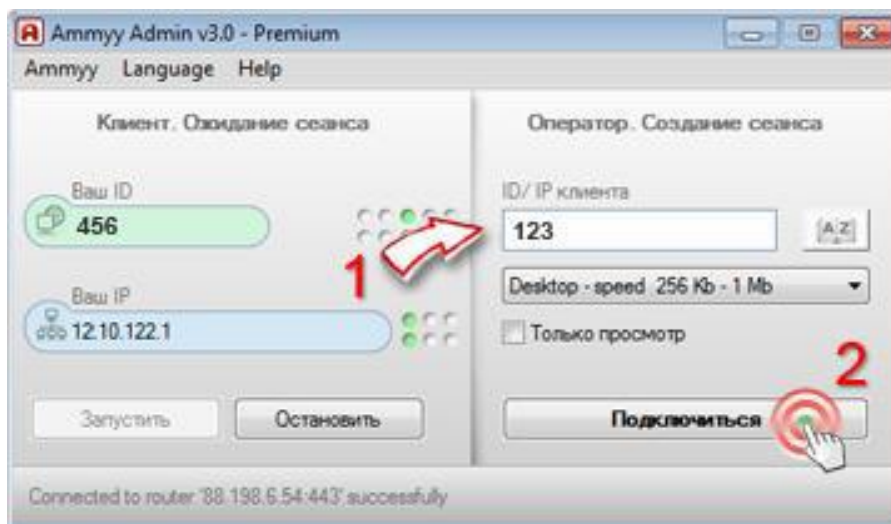


Рис. 1.9. Діалогове вікно програми Ammy admin

#### *Переваги.*

Проста і легка програма, можна працювати як в Інтернеті, так і в локальній мережі, потребує мінімальних налаштувань і не вимагає якихось особливих умінь і навичок. За порівняння з TeamViewer м'якша ліцензійна

політика.

### *Недоліки.*

Мінімум функцій для віддаленого управління, адмініструвати великий парк комп'ютерів буде складно. При довгому використанні, більше 15 годин на місяць, сеанс роботи може бути обмежений або заблокований, платна для комерційного використання.

### *Підсумок:*

Дана програма більше підійде для разового підключення до комп'ютера і не сильно складних завдань, наприклад в якості надання допомоги не досвідченому користувачеві в налаштуванні комп'ютера.

## **RAdmin**

Одна з перших програм віддаленого управління і відома у своєму колі, більше призначена для системного адміністрування, основний акцент зроблено на безпеці. Програма складається з двох частин: компонент сервера і клієнта. Вимагає установки, не досвідченому користувачеві буде не просто з нею розібратися, програма призначена в основному для роботи з IP адресою, що не зовсім зручно для надання техпідтримки через Інтернет. Програма платна, але має безкоштовний тестовий період (рис. 1.10).

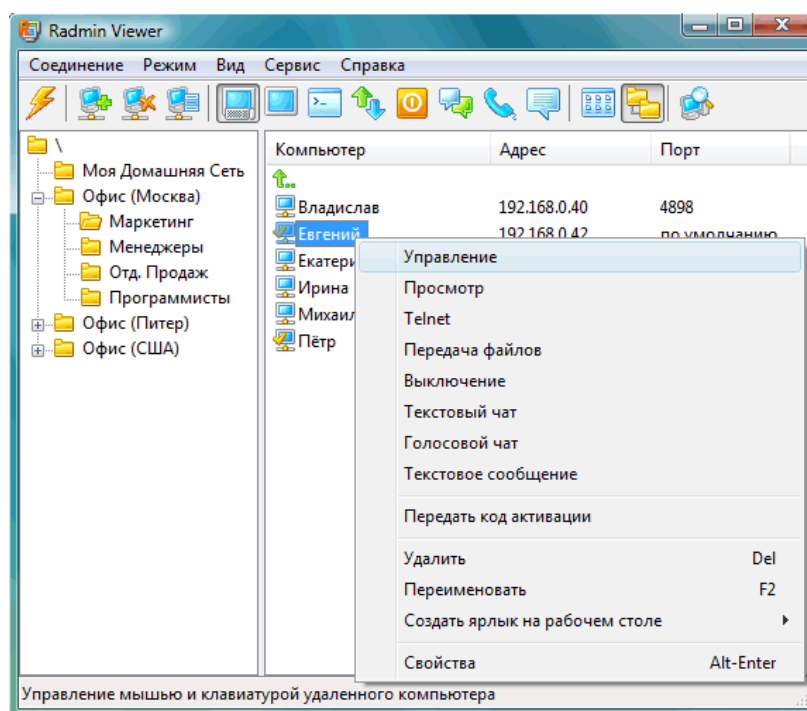


Рис. 1.10. Діалогове вікно програми RAdmin

### *Переваги.*

У програми висока швидкість роботи, особливо в хорошій мережі, завдяки відео драйверу захоплення робочого столу. Крім того, перевагою є підвищена надійність і безпека. Вбудована технологія Intel AMT, що дозволяє підключатися до BIOS віддаленого комп'ютера і налаштовувати його. Реалізовано тільки основні режими роботи віддалене управління, передача файлів, чат і т.д.

### *Недоліки.*

Майже немає можливості для роботи без IP адреси, тобто з'єднуватися з ID. Відсутній клієнт для мобільних систем. Немає безкоштовної версії, тільки тестовий період 30 днів. Для роботи з програмою необхідні навички досвідченого користувача. При підключенні відео драйвер може відключати графічну оболонку Aero.

### *Підсумок.*

Програма більше підійде для системних адміністраторів для адміністрування комп'ютерів і серверів в локальній мережі. Для роботи через Інтернет, можливо, доведеться налаштувати VPN тунель.

Таким чином, проведений аналіз довів, що сьогодні існує доволі велика кількість програм, що забезпечують віддалений доступ та віддалене адміністрування. Але, більшість з них потребує наявності (придбання) ліцензії для довготривалої роботи з програмою. Тому розробка власного додатку віддаленого доступу може вирішити цю проблему.

### **Висновки до розділу**

В першому розділі дипломної роботи було досліджено теоретичні основи віддаленого доступу до ресурсів комп'ютера. Встановлено, що віддалений доступ являє собою функцію, що дозволяє користувачеві підключатися до комп'ютера через Інтернет за допомогою іншого ПК. Одним з видів віддаленого доступу є доступ до віддаленого робочого столу (Remote Desktop) - це термін, яким позначається режим управління, коли один комп'ютер отримує права адміністратора по відношенню до іншого,

віддаленого. Зв'язок між пристроями відбувається в реальному часі за допомогою Інтернет або локальної мережі.

Доступ до віддаленого робочого столу здійснюється на базі архітектури «сервер - термінал», що дозволяє працювати з ресурсами сервера так само, як і з локальними ресурсами: виконувати команди, працювати з файловою системою, запускати додатки і т.п.

Проаналізовано принципи роботи протоколів віддаленого доступу: RDP, VNC, ICA. На підставі проведеного аналізу було вирішено обрати для розробки проекту протоколи RDP і VNC, оскільки їх застосування дозволить реалізувати віддалений доступ не тільки до комп'ютерів з операційною системою Windows, а також дасть можливість застосовувати різні платформи на клієнті і сервері віддаленого доступу. Крім того, дані протоколи забезпечують високий рівень безпеки.

Проведений аналіз додатків (TeamViewer, LiteManager, Ammy admin, RAdmin) довів, що сьогодні існує доволі велика кількість програм, що забезпечують віддалений доступ та віддалене адміністрування. Але, більшість з них потребує наявності (придбання) ліцензії для довготривалої роботи з програмою. Тому розробка власного додатку віддаленого доступу може вирішити цю проблему.

## РОЗДІЛ 2. РОЗРОБКА КЛІЄНТСЬКОГО ДОДАТКУ ВІДДАЛЕНОГО ДОСТУПУ НА ПЛАТФОРМІ .NET

### 2.1. Характеристика платформи .NET та обґрунтування інструментів розробки додатку

Платформа .NET Framework - це технологія, яка підтримує створення і виконання нового покоління додатків. При розробці платформи .NET Framework враховувалися наступні цілі.

- ✓ Забезпечення узгодженого об'єктно-орієнтованого середовища програмування для локального збереження і виконання об'єктного коду, для локального виконання коду, розподіленого в Інтернеті, або для віддаленого виконання.
- ✓ Забезпечення середовища виконання коду, що мінімізує конфлікти при розгортанні програмного забезпечення та управлінні версіями.
- ✓ Забезпечення середовища виконання коду, що гарантує безпечне виконання коду, включаючи код, створений невідомим або не повністю довіреним стороннім виконавцем.
- ✓ Забезпечення середовища виконання коду, що виключає проблеми з продуктивністю середовищ виконання сценаріїв або коду, що інтерпретується.
- ✓ Забезпечення єдиних принципів розробки для різних типів додатків, таких як додатки Windows і веб-додатки.
- ✓ Взаємодія на основі промислових стандартів, яке гарантує інтеграцію коду платформи .NET Framework з будь-яким іншим кодом.

*Архітектура .Net Framework.* Платформа складається з двох частин. Основою є виконуючого середовища Common Language Runtime (CLR), яка може виконувати як звичайні програми, так і серверні додатки. Серед CLR управляє пам'яттю, виконанням потоків, виконанням коду, перевіркою безпеки коду, компіляцією і іншими системними службами. Ці зособи є внутрішніми



для керованого коду, який виконується в середовищі CLR.

Друга, не менш важлива частина, це бібліотека класів Framework Class Library (FCL), що містить в собі безліч компонентів для роботи з базами даних, мережею, введенням / висновком, файлами, призначеним для користувача інтерфейсом і т.д. Це дозволяє розробнику не займатися низькорівневим програмуванням, а використовувати вже готові класи.

*Важливі частини бібліотеки класів.*

Windows Forms - відповідає за розробку графічного інтерфейсу. Фактично є обгорткою над Win32 API.

ADO.NET - надає доступ даними. В основному використовується для роботи з базами даних.

ASP.NET - технологія розробки веб-сайтів, веб-додатків і веб-сервісів.

Language Integrated Query (LINQ) - реалізація мови запитів, що нагадує по синтаксису SQL в програмах на .Net.

Windows Presentation Foundation (WPF) - система створення графічних інтерфейсів, що використовує мову розмітки XAML. На відміну від Windows Forms використовує графічну технологію DirectX, що забезпечує швидшу роботу за рахунок апаратного прискорення графіки.

Windows Communication Foundation (WCF) - система обміну даними між додатками .Net. Використовується для створення розподілених додатків.

*Технологія ADO.NET.* ADO .NET (ActiveX Data Objects .NET) є набором класів, що реалізують програмні інтерфейси для полегшення підключення до баз даних з програми незалежно від особливостей реалізації конкретної системи управління базами даних і від структури самої бази даних, а також незалежно від місця розташування цієї самої бази - зокрема, в розподіленій середовищі (клієнт-серверний додаток) на стороні сервера.

В цій технології закладена можливість роботи програми в стані "розриву" з'єднання з базою даних. Додатки підключаються до бази даних тільки на невеликий проміжок часу. З'єднання встановлюється тільки тоді, коли клієнт з віддаленого комп'ютера запрошує на сервері дані.

В об'єктній моделі ADO.NET можна виділити декілька рівнів ( рис. 2.1).

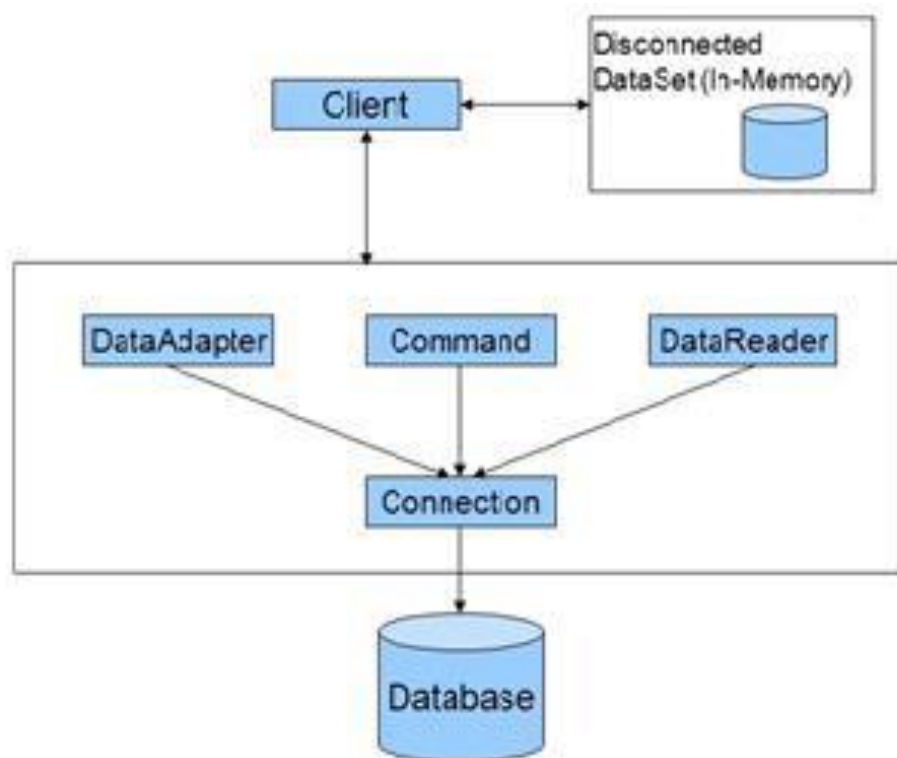


Рис. 2.1. Модель технології ADO.NET

**Рівень даних.** Це базовий рівень, на якому розташовуються самі дані. На даному рівні забезпечується фізичне зберігання інформації на магнітних носіях і маніпуляція з даними на рівні вихідних таблиць (вибірка, сортування, додавання, видалення, оновлення).

**Рівень бізнес-логіки.** Це набір об'єктів, що визначають, з якою базою даних належить встановити зв'язок і які дії необхідно буде виконати з даними, що містяться в неї. Для встановлення зв'язку з базами даних використовується об'єкт `FbDataConnection`. Для зберігання команд, що виконують будь-які дії над даними, використовується об'єкт `FbDataAdapter`. Для зберігання результатів вибірки використовується об'єкт `DataSet`.

**Рівень додатку.** Це набір об'єктів, що дозволяють зберігати і відображати дані на комп'ютері кінцевого користувача. Для зберігання інформації використовується об'єкт `DataSet`, а для відображення даних є досить великий набір елементів управління (`DataGrid`, `TextBox`, `ComboBox`, `Label` і т. д.).

**Мови програмування та платформа .NET.** Однією з основних ідей Microsoft .NET є сумісність програмних частин, написаних різними мовами.

Наприклад, служба, написана на C ++ для Microsoft .NET, може звернутися до методу класу з бібліотеки, написаної на Delphi; на C # можна написати клас, успадкованих від класу, написаного на Visual Basic .NET, а виключення, створене методом, написаним на C #, може бути перехоплено і оброблено в Delphi. Кожна бібліотека (збірка) в .NET має відомості про свою версію, що дозволяє усунути можливі конфлікти між різними версіями збірок. Тому визначимо основні мови, що поставляються разом з Microsoft Visual Studio:

- C #
- Visual Basic .NET
- JScript .NET
- C ++ / CLI - нова версія Managed C ++
- F # - член сімейства мов програмування ML, включений в VS2010 / VS2012 / VS2015 / VS2017

Найбільш популярною мовою програмування на платформі .Net є C#. C # - це об'єктно-орієнтована мова зі строгою типізацією, що дозволяє розробникам створювати різні безпечні і надійні додатки, що працюють на платформі .NET Framework. C # можна використовувати для створення клієнтських додатків Windows, XML-веб-служб, розподілених компонентів, додатків клієнт-сервер, додатків баз даних і т. Д. Visual C # надає розвинений редактор коду, зручні конструктори користувацького інтерфейсу, інтегрований відладчик і багато інших засобів, які спрощують розробку додатків на мові C # для платформи .NET Framework.

Програми C # виконуються на платформі .NET Framework, яка інтегрована в Windows і містить віртуальне загальномовне середовище виконання (середу CLR) і уніфікований набір бібліотек класів.

Вихідний код, написаний на мові C # компілюється в проміжну мову (IL), яка відповідає специфікаціям CLI. Код на мові IL і ресурси, в тому числі точкові малюнки і рядки, зберігаються на диск у вигляді виконуваного файлу (зазвичай з розширенням .exe або .dll). Такий файл називається складанням. Збірка містить маніфест з інформацією про типи, версії, вимог безпеки, мовою

і регіональних параметрах для цієї збірки.

При виконанні програми C # Серед CLR завантажує збірку і виконує різні дії в залежності від відомостей, збережених в маніфесті. Якщо виконуються всі вимоги безпеки, серед CLR виконує JIT-компіляцію з коду на мові IL в інструкції машинного мови. Також середовище CLR виконує інші операції, наприклад автоматичну збірку сміття, обробку винятків і управління ресурсами. Код, що виконується середовищем CLR, іноді називають "керованим кодом", щоб підкреслити відмінності цього підходу від "некерованого коду", який відразу компілюється в машинний мову для певної системи.

На наступній схемі (рис. 2.2) показано зв'язок між файлами вихідного коду C #, бібліотеками класів .NET Framework, збірками і середовищем CLR

Таким чином, застосування мови програмування C# дозволить найбільш ефективно реалізувати можливості .NET Framework для розробку додатку віддаленого доступу.

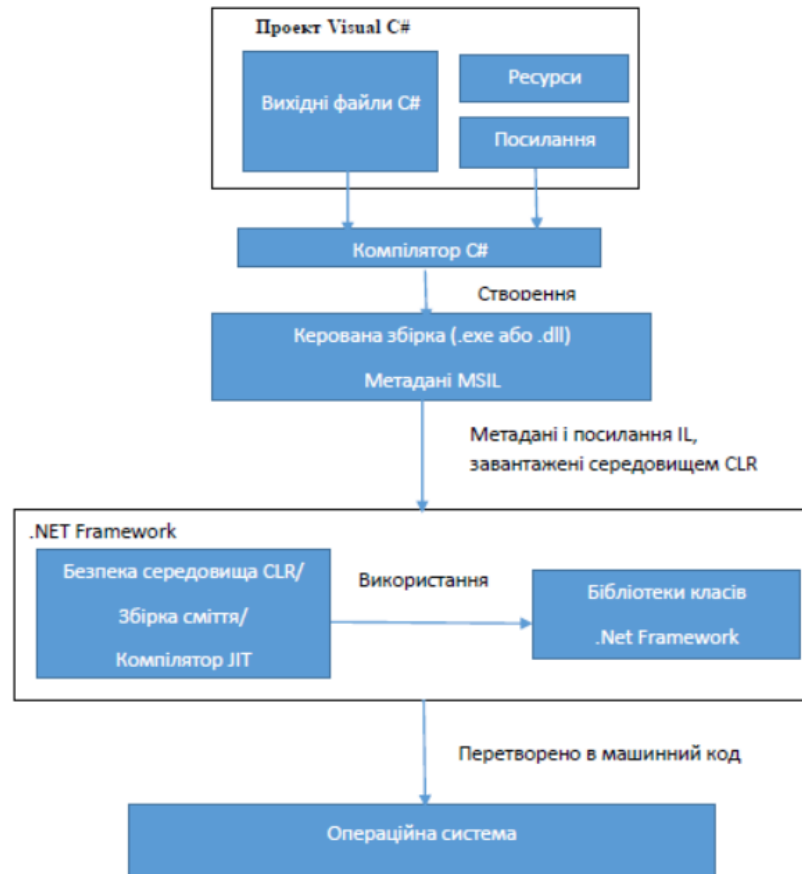


Рис. 2.2. Зв'язок між додатком C # та платформою .NET Framework

## 2.2. Розробка програмного додатку

Розробка програмного проекту доступу до віддаленого робочого столу відбувається згідно технічного завдання, яке подано у додатку А.

### Постановка задачі

Після аналізу всіх вимог були сформульовані цілі:

1. Розробити клієнт-додаток, що дозволяє:

- здійснювати віддалений доступ за протоколами RDP і VNC.
- підключатися до хосту, дані про який зберігаються в файлі (\* .rdp).
- спостерігати і керувати віддаленим комп'ютером в мережі.
- зберігати дані про вузол підключення в базі даних.

Клієнт повинен бути побудований так, щоб реалізовувати варіанти використання, зображені на рис. 2.3

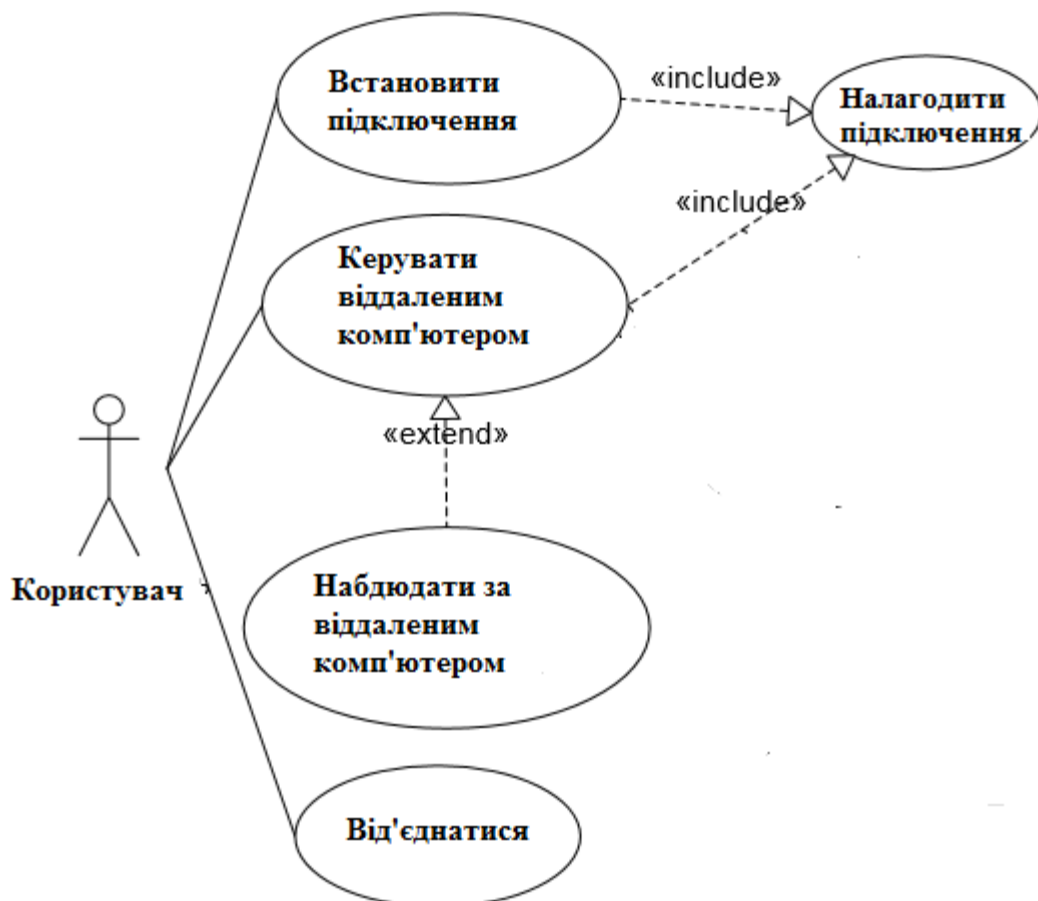


Рис. 2.3. Функціональна схема клієнт-додатку

Під час розробки використовувати наступні засоби програмування:

- мова програмування C # і середовище розробки Microsoft Visual

Studio 2010;

- для реалізації інтерфейсної частини додатку застосовується бібліотека Windows Forms на основі технології .Net Framework 4.
- в реалізації моделі оступу до даних застосовується технологія ADO.NET.

### **Архітектура проекту**

Для реалізації клієнта було вирішено взяти за основу трирівневу архітектуру, кожен шар якої буде відповідати за свою логічну частину:

- Рівень відображення
- Рівень логіки - управління підключенням
- Рівень протоколу

Рівень відображення відповідає тільки за візуалізацію поточного екрану віддаленого комп'ютера і дозволяє користувачеві здійснювати дії миші і клавіатури, які будуть передані на нижній рівень.

Завдання рівня логіки полягає в управлінні всім процесом роботи програми в цілому і кожного його компонента. В першу чергу мова йде про встановлення підключення до віддаленого комп'ютеру.

На самому нижньому рівні відбувається логіка дії протоколів. Концептуальна модель класів клієнтського додатку відображена у UML-

діаграмі (рис. 2.4). Основні класи проекту - клас RemoteRDPClient та клас RemoteVNCCClient беруть участь в реалізованій ієрархії класів і є нащадками абстрактного класу RemoteAbstractWidget.

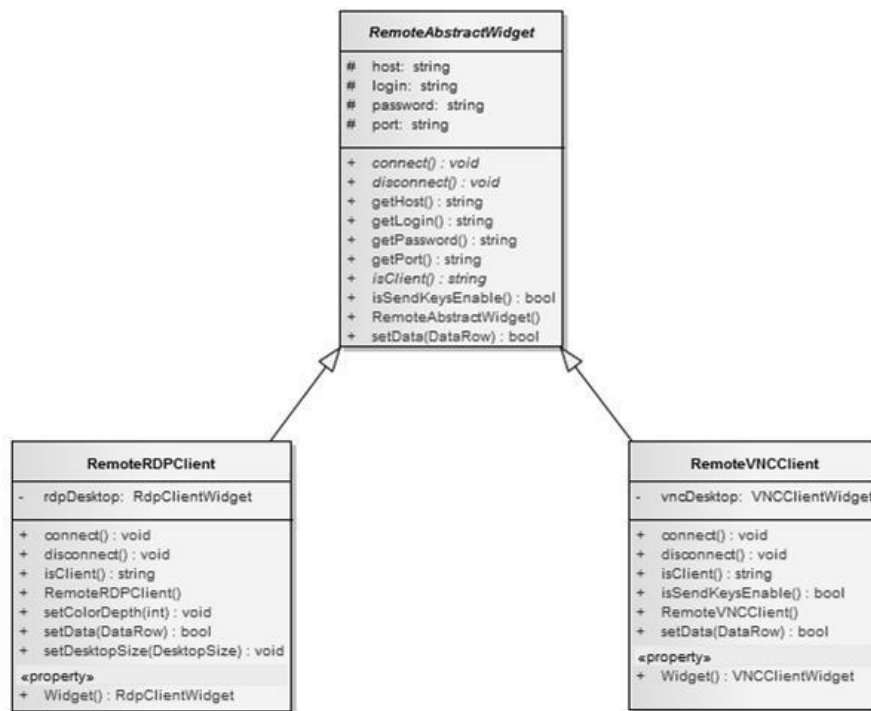


Рис. 2.4. Ієрархічна модель класів

Клас RemoteAbstractWidget є абстракцією поведінки класів RemoteRDPCClient та RemoteVNCClient. Код класу RemoteAbstractWidget наведено нижче:

```

public abstract class RemoteAbstractWidget
{
    protected string
    password; protected
    string login; protected
    string host; protected
    string port;
    public
    RemoteAbstractWidget() { }
    public abstract void connect();
    public abstract void disconnect();
    public abstract string isClient();
    public virtual bool setData(DataRow row)
    {

```

Лістинг 2. 1

```

45, true);

    host = Convert.ToString(row["ip"]); port =
Convert.ToString(row["port"]); login =
Convert.ToString(row["login"]);

    password = Helper.XorString(Convert.ToString(row["password"]),
    return true;
}

    public virtual bool isSendKeysEnable()
    {
        return false;
    }
    public string getPassword()
    {
        return password;
    }
    public string getLogin()
    {
        return login;
    }
    public string getPort()
    {
        return port;
    }
    public string getHost()
    {
        return host;
    }
}

```

### **Реалізація інтерфейсу додатка**

Інтерфейсна частина додатку повинна складатися з форми, яка містить наступні основні елементи управління:



- Рядок меню.
- Панель інструментів.
- Список.
- Картотека.
- Рядок стану.

Після всіх налаштувань розміщених компонентів на головній формі форма прийняла вид, представлений на рис. 2.5.

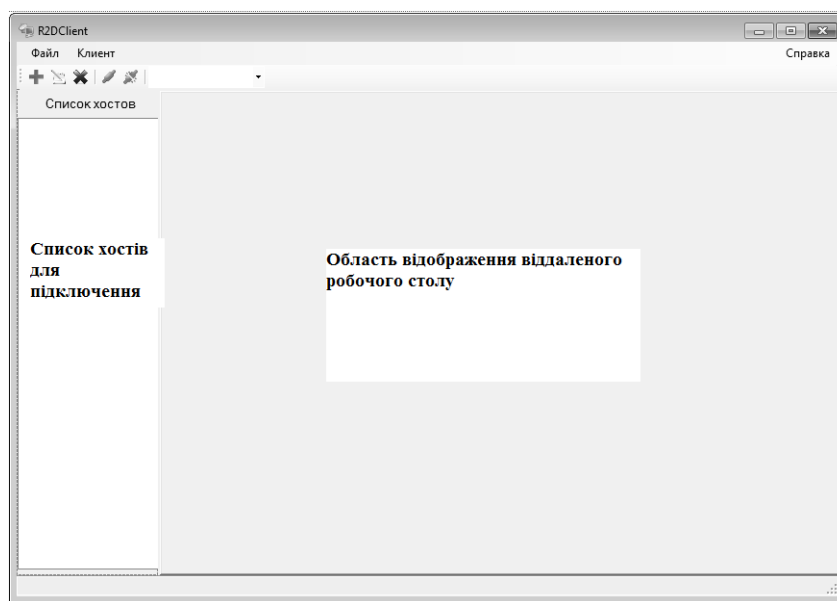


Рис. 2.5. Головна форма проекту

### Реалізація рівня відображення

У центральній області форми при ініціалізації підключення і в залежності від обраного типу підключення програмними засобами було реалізовано розміщення на формі наступних компонентів:

- AxMSTSCLib.AxMsRdpClientNotSafeForScripting
- VncSharp.RemoteDesktop

Компонент AxMSTSCLib.AxMsRdpClientNotSafeForScripting та VncSharp.RemoteDesktop відповідають за відображення віддаленого робочого стола відповідно до типу підключення RDP або VNC

Ієрархія класів, зазначена на рис. 2.4, реалізована для уніфікації компонентів AxMSTSCLib.AxMsRdpClientNotSafeForScripting і VncSharp.RemoteDesktop, тому що в цих компонентах визначені різні, не

подібні між собою методи підключення. Це дозволило використовувати поліморфні властивості в подальшій програмній реалізації.

Приклад коду, що реалізує відображення віддаленого столу при підключенні за протоколом VNC наведено нижче:

Лістинг 2.2

```
using VNCCClientWidget = VncSharp.RemoteDesktop;

public RemoteVNCCClient() : base()
{
    vncDesktop = new VNCCClientWidget();
    vncDesktop.Enabled = true;
    vncDesktop.Visible = true;
    vncDesktop.AutoScroll = true;
    vncDesktop.Dock = System.Windows.Forms.DockStyle.Fill;
    vncDesktop.Location = new System.Drawing.Point(0, 0);
    vncDesktop.Name = "vncDesktop";
    vncDesktop.TabIndex = 0;
    vncDesktop.GetPassword = base.getPassword;
    vncDesktop.SetScalingMode(true);
}
```

В цьому методі створюється об'єкт vncDesktop класу ncSharp.RemoteDesktop: Код для RDP протоколу має аналогічну структуру.

### **Реалізація рівня логіки**

#### *Реалізація технології віддаленого доступу*

Нижче представлений лістинг коду методу класу-обробнику подій, який викликається при підключенні, використовуючи RDP протокол.

Лістинг 2.3

```
private void rdpConnectMenuItem_Click (object sender, EventArgs e)
{
    RemoteRDPCClient client = new RemoteRDPCClient();
```

```

TabPage NewPage = new
TabPage(); NewPage.Text =
Convert.ToString(ipList.Items[ipList.SelectedIndex]
); NewPage.Controls.Add(client.Widget);
tabControl.TabPages.Add(NewPage);
tabControl.SelectedIndex = tabControl.TabCount -
1; DataTable table = SQL.dataTable("hostdata");
DataRow row = table.Rows[ipList.SelectedIndex];
if (client.setData(row))
{
    client.setColorDepth(RDPSettingsDlg.getColorDepth());
    client.setDesktopSize(RDPSettingsDlg.getDesktopSize()
); client.connect();
}
else disconnectButton_Click(sender, e);
}

```

У наведеному лістингу програмної реалізації методу створюється об'єкт класу RemoteRDPClient. Далі об'єкту NewPage.Text присвоюється значення адреса хоста, після чого викликом функції NewPage.Controls.Add (client.Widget) вміст вкладки заповнюється об'єктом класу RdpClientWidget. Властивість client.Widget повертає цей об'єкт. Клас RdpClientWidget є нащадком класу AxMsRdpClientNotSafeForScripting. Далі створена вкладка додається на панель вкладок, виконавши наступну інструкцію tabControl.TabPages.Add (NewPage). Після чого з бази даних отримуємо налаштування підключення (адреса хоста, ім'я користувача, пароль) і проводиться підключення з урахуванням налаштувань екрану викликом методу client.connect (). Метод connect() по протоколу RDP наведено нижче.

Лістинг 2.4

```

public override void connect()
{

```

```

if (host != null)
{
    try
    {
        vncDesktop.VncPort = Convert.ToInt32(base.getPort());
        vncDesktop.Connect(host, false, true);
        if (vncDesktop.IsConnected)
        {
            vncDesktop.SetScalingMode(true);
            vncDesktop.FullScreenUpdate();
        }
    }
    else
    {
        MessageBox.Show("Не удастся осуществить соединение.
        Проверьте правильность ввода пароля", "Ошибка соединения",
        MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
    }
    catch (VncProtocolException vex)
    {
        MessageBox.Show(string.Format("Unable to connect to VNC
        host:\n\n{0}.\n\nCheck that a VNC host is running there.", vex.Message),
        string.Format("Unable to Connect to {0}", host),
        MessageBoxButtons.OK,
        MessageBoxIcon.Exclamation);
    }
    catch (Exception ex)
    {
        MessageBox.Show(string.Format("Unable to connect to
        host. Error was: {0}", ex.Message),
        string.Format("Unable to Connect to {0}", host),
        MessageBoxButtons.OK,

```

```

        MessageBoxIcon.Exclamation);
    }
}

```

Наведений у лістингу 2.3 та 2.4 код дозволяє здійснити з'єднання з віддаленим хостом, використовуючи протокол RDP.

Аналогічним чином реалізовано підключення з використанням протоколу VNC. Виклик метода connect() (лістинг 2.5) відбувається у обробнику подій vncConnectMenuItem\_Click(object sender, EventArgs e).

Лістинг 2.5

```

public override void connect()
{
    if (host != null)
    {
        try
        {
            vncDesktop.VncPort = Convert.ToInt32(base.getPort());
            vncDesktop.Connect(host, false, true);
            if (vncDesktop.IsConnected)
            {
                vncDesktop.SetScalingMode(true);
                vncDesktop.FullScreenUpdate();
            }
            Else
                MessageBox.Show("Не удается осуществить соединение.
                Проверьте правильность ввода пароля", "Ошибка соединения",
                MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        }
        catch (VncProtocolException vex)
        {

```

```

        MessageBox.Show(string.Format("Unable to connect to VNC
host:\n\n{0}.\n\nCheck that a VNC host is running there.", vex.Message),
            string.Format("Unable to Connect to {0}", host),
            MessageBoxButtons.OK,
            MessageBoxIcon.Exclamation);
    }
    catch (Exception ex)
    {
        MessageBox.Show(string.Format("Unable to connect to host.
Error was: {0}", ex.Message),
            string.Format("Unable to Connect to {0}", host),
            MessageBoxButtons.OK,
            MessageBoxIcon.Exclamation);
    }
}
}
}

```

### *Збереження даних про хости, що під'єднуються*

Для реалізації можливості зберігання даних про хости, що підключаються, була використана база даних SQLite і технологія доступу до даних ADO.NET [1, с. 10].

Для створення бази даних застосовуються SQLite — компактна вбудована СУБД. SQLite не використовує парадигму клієнт-сервер, тобто движок SQLite не є окремо працюючим процесом, з яким взаємодіє програма, а являє собою бібліотеку, з якої програма компонується, і движок стає складовою частиною програми. Таким чином, в якості протоколу обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виконується програма.

Для збереження даних про хости, що підключаються, створена таблиця

hosts.db. Модель бази даних наведена у табл. 2.1

Таблиця 2.1

**Модель бази даних hosts.db**

Назва поля	Ім'я поля	Тип даних	Властивості
Унікальний ідентифікатор	ID	INTEGER	PRIMARY KEY, UNIQUE, NOT NULL,
Адреса IP	IP	VARCHAR(64)	UNIQUE, NOT NULL,
Назва порту	PORT	VARCHAR(16)	
Логін	LOGIN	VARCHAR(64)	
Пароль	PASSWORD	VARCHAR(64)	

Для взаємодії з об'єктами ADO.NET був розроблений клас SQLiteInterface, даний клас дозволяє абстрагуватися від інтерфейсів ADO.NET і надає простий функціонал доступу і оперування з даними бази даних. Нижче на лістингу 2.6 наведена реалізація одного з методів даного класу.

Лістинг 2.6.

```
private bool openDataBase()
{
    try { if (connection.ConnectionString.Length > 0)
        connection.Open();
        else throw new System.Exception("Строка соединения с
        БД не корректна."); }
    catch (System.Exception Except) {
        MessageBox.Show(Except.Message, "Ошибка");
        connection.Close();
        return false;
    }
    return connection.State == System.Data.ConnectionState.Open
        ? true : false;
}
```

Вище наведений метод здійснює підключення до бази даних, попередньо перевіряючи рядок з'єднання з базою даних.

При запуску програми здійснюється заповнення списку даними хостів з

бази даних за допомогою об'єкта класу SQLiteInterface в конструкторі класу головної форми. Нижче на лістингу 2.7 представлений даний фрагмент коду програми.

Лістинг 2.7.

```
public MainWindow()
{
    SQL.connectionString = "Data Source=hosts.db; Version=3;";
    SQL.addTable("hostdata");
    SQL.fill("hostdata");
    DataTable table = SQL.dataTable("hostdata");
    for (int i = 0; i < table.Rows.Count; ++i)
        ipList.Items.Add(new IListItem(table.Rows[i][1]
        .ToString(), 0));
    ipList.SelectedIndex = 0;
}
```

*Підключатися до хосту, дані про який зберігаються в файлі (\* .rdp).*

RDP-файл - це текстовий файл, що містить настройки підключення до заданого сервера. Для реалізації можливості імпорту даних з \* .rdp файлів реалізований метод openMenuItem\_Click ():

Лістинг 2.8.

```
private void openMenuItem_Click(object sender, EventArgs e)
{
    OpenFileDialog myOpenFileDialog;
    myOpenFileDialog = new OpenFileDialog();
    myOpenFileDialog.Filter = "RDP File|*.rdp";
    myOpenFileDialog.Title = "Выбор файла с настройками RDP";
    myOpenFileDialog.ShowDialog();
    if (myOpenFileDialog.FileNames.Count() > 0)
    {
        RDPFile MyRdpFile = new RDPFile();
    }
}
```



```

MyRdpFile.Read(myOpenFileDialog.FileName)
; if (MyRdpFile.FullAddress != string.Empty)
{
    DataRow row =
    SQL.newRow("hostdata"); row["ip"] =
    MyRdpFile.FullAddress; row["port"] =
    string.Empty;
    row["login"] = MyRdpFile.Username;
    row["password"] =
    Helper.XorString(MyRdpFile.Password, 45, true);
    SQL.addNewRow("hostdata", row);
    ipList.Items.Add(new IListItem(MyRdpFile.FullAddress, 0));
}
}
}

```

У вище наведеному лістингу створюється об'єкт класу OpenFileDialog, який виводить на екран стандартний діалог відкриття файлу. Далі шлях до вибраного файлу передається методу Read (myOpenFileDialog.FileName) викликається об'єктом класу RDPFile, здійснюється зчитування даних з файлу, після чого даний заносяться в базу даних і відображатися в списку хостів на формі. Нижче на рис. 2.6 представлений діалог відкриття rdp файлу реалізований в програмі.

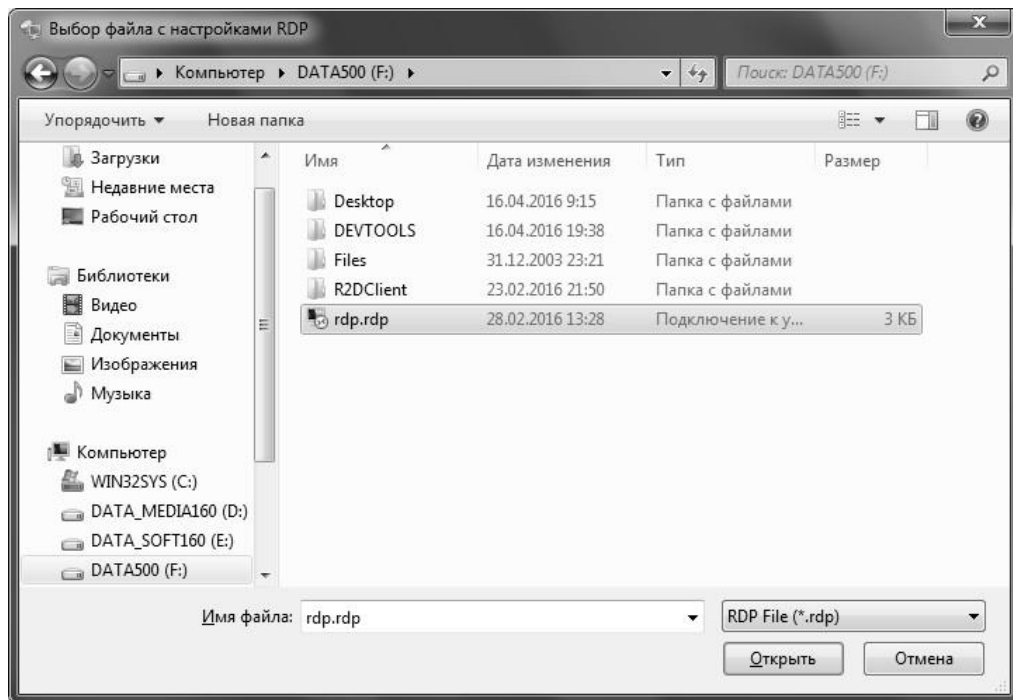


Рис. 2.6. Діалогове вікно вибору файлу

### Реалізація рівня протоколу

Реалізація цього рівня в програмі відбувається за допомогою застосування бібліотек Interop.MSTSCLib.dll і AxInterop.MSTSCLib.dll - модулі, що належить збірці бібліотек типів `MSTSCLib` для операційної системи Windows та VncSharp.dll – це OpenSource компонент, реалізація протоколу VNC для .NET Framework.

Таким чином, розроблений клієнтський додаток віддаленого доступу R2DClient повністю відповідає всім вимогам, що вказані у технічному завданні.

### 2.3. Тестування програмного додатку

Тестування розробленого додатку будемо здійснювати за допомогою встановленої віртуальної машини VirtualBox версія 5.2.22 [12]. Параметри віртуальної машини та операційної системи подані нижче:

- операційна система - Windows XP (32-bit);
- тип підключення - віртуальний адаптер хоста;
- IP-адреса 192.168.56.2;
- маска підмережі 255.255.255.0;
- основний шлюз 192.168.56.1;

- у вікні «Властивості системи», вкладка «Віддалені сеанси» встановити опцію «Дозволити віддалений доступ цьому комп'ютеру».

На віртуальну машину становлено VNC-сервер TightVNC. Сервер налагоджено з паролем root.

Параметри комп'ютеру, на якій встановлено клієнтський додаток віддаленого доступу R2DClient:

- операційна система Windows 8.1 (64-bit);
- підключення - VirtualBox Host-Only Ethernet Adapter;
- IP-адреса 192.168.56.1;
- маска підмережі 255.255.255.0.

### **Цілі та методи тестування**

Клієнтський додаток віддаленого доступу R2DClient повинен забезпечувати:

1. Зберігання даних про вузол підключення в базі даних.
2. Підключення до віддаленого комп'ютеру за RDP протоколом та адміністрування.
3. Підключення до віддаленого комп'ютеру за VNC протоколом та адміністрування.
4. Підключення до хосту, дані про який зберігаються в файлі (\* .rdp).

В ході тестування будемо використовувати ручний метод. Ручне тестування (manual testing) - частина процесу тестування на етапі контролю якості в процесі розробки програмного забезпечення шляхом моделювання можливих сценаріїв дії користувача.

### **Етапи тестування**

1. Зберігання даних про вузол підключення в базі даних.

На клієнтському додатку встановимо тип протоколу - RDP, IP-адресу хосту, порт, логін та пароль доступу до комп'ютеру – admin, admin. Після виконання дій дані про хост збережено в БД та відображено у лівій панелі (рис. 2.7).

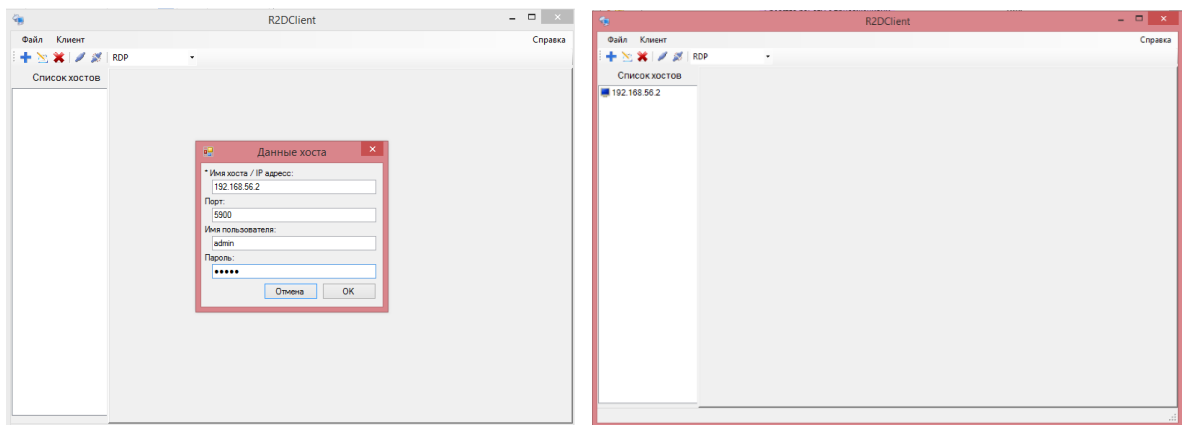


Рис. 2.7. Зберігання даних про вузол підключення в базі даних

2. Підключення до віддаленого комп'ютеру за RDP протоколом та адміністрування.

Тестування передбачає, що віртуальна машина налагоджена то запущена. Клієнтський додаток віддаленого доступу R2DClient запущено на машині клієнта. На клієнтському додатку встановимо тип протоколу - RDP, IP-адресу хосту, порт, логін та пароль доступу до комп'ютеру – admin, admin (рис. 2.8).

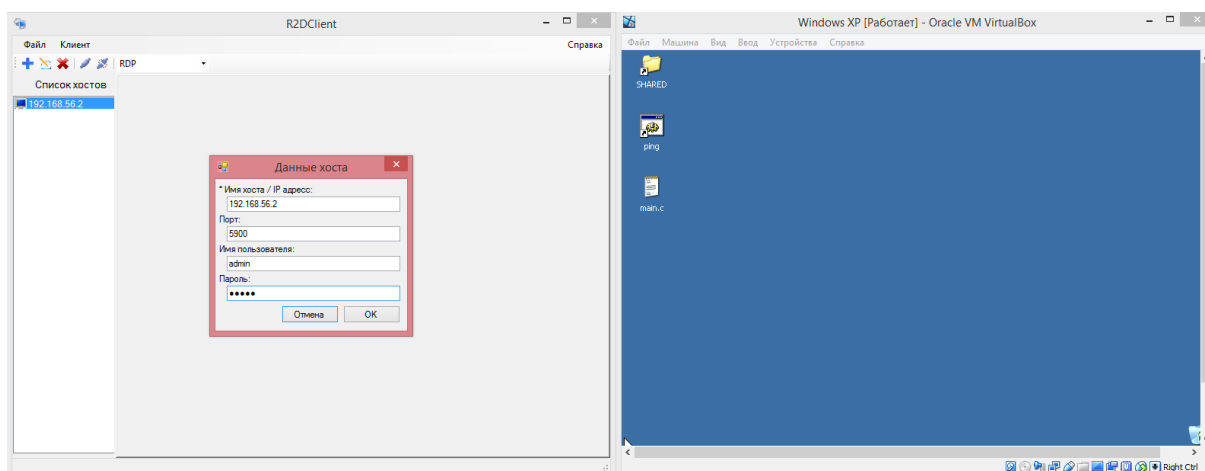


Рис. 2.8. Встановлення підключення за RDP протоколом

Після виконання операції з'єднання з хостом ми отримуємо відображення віддаленого робочого столу на клієнтському додатку (рис. 2.9)

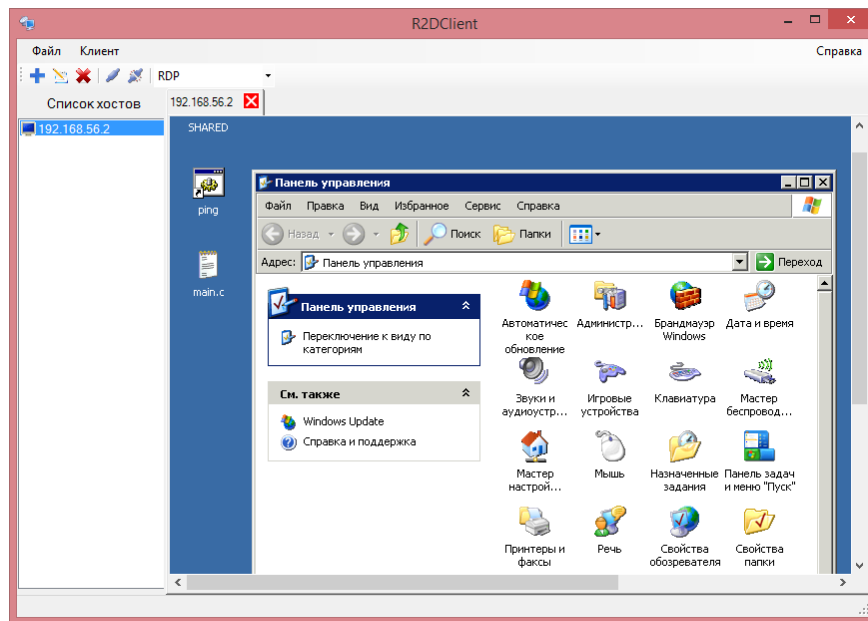


Рис. 2.9. Відображення віддаленого робочого столу на клієнтському додатку

Для тестування функцій адміністрування додамо нового користувача за допомогою опції Адміністрування – Керування комп’ютером в панелі інструментів на віддаленому комп’ютері. Початковий стан вкладки Локальні користувачі зображено на рис. 2.10.

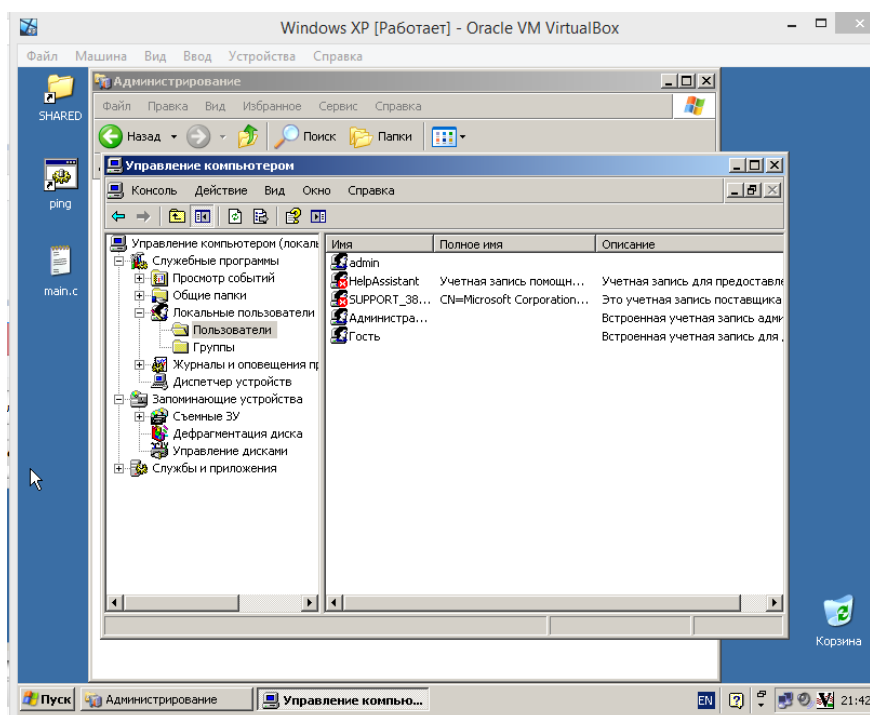


Рис. 2.10. Початковий стан вкладки Локальні користувачі

Виконаємо дії додавання користувача на клієнтському додатку (рис. 2.11).

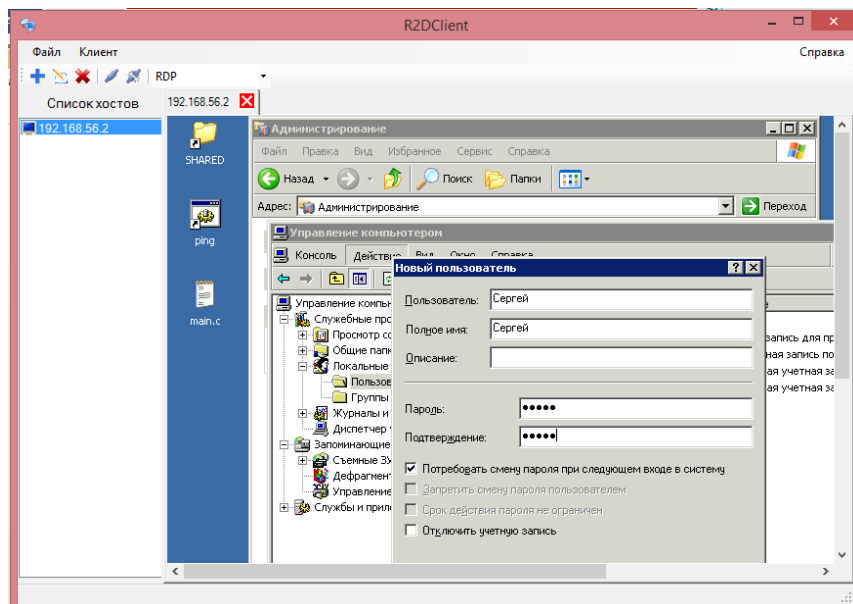


Рис. 2.11. Добавлення користувача на клієнтському додатку  
Результати віддаленого адміністрування показано на рис. 2.12.

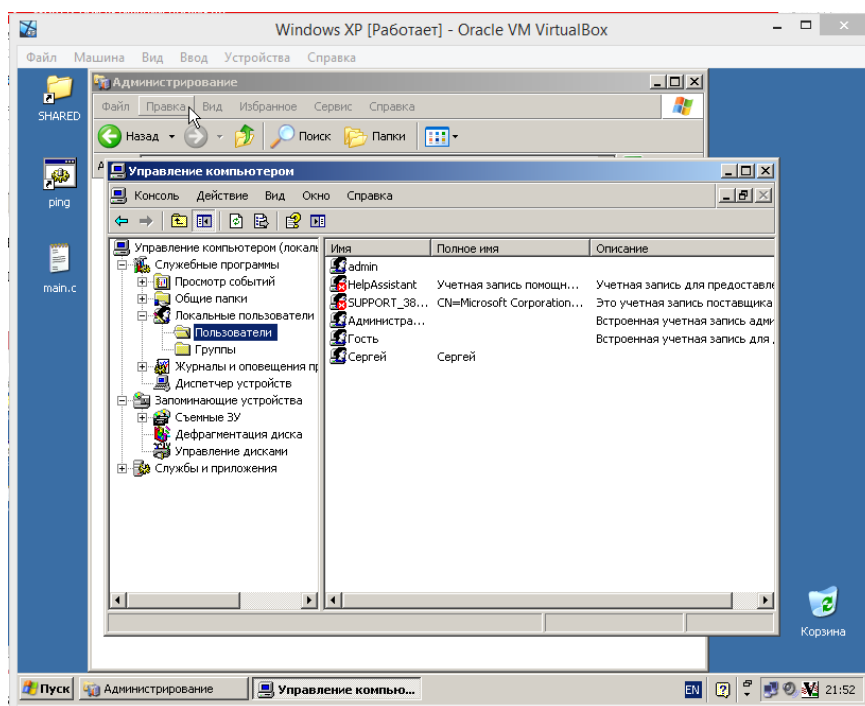


Рис. 2.12. Результати віддаленого адміністрування

3. Підключення до віддаленого комп'ютеру за VNC протоколом та адміністрування.

Запустимо VNC-сервер TightVNC на виконання. Пароль – root. На клієнтському додатку встановимо тип протоколу - VNC, IP-адресу хосту, порт, логін та пароль доступу до серверу (рис. 2.13).

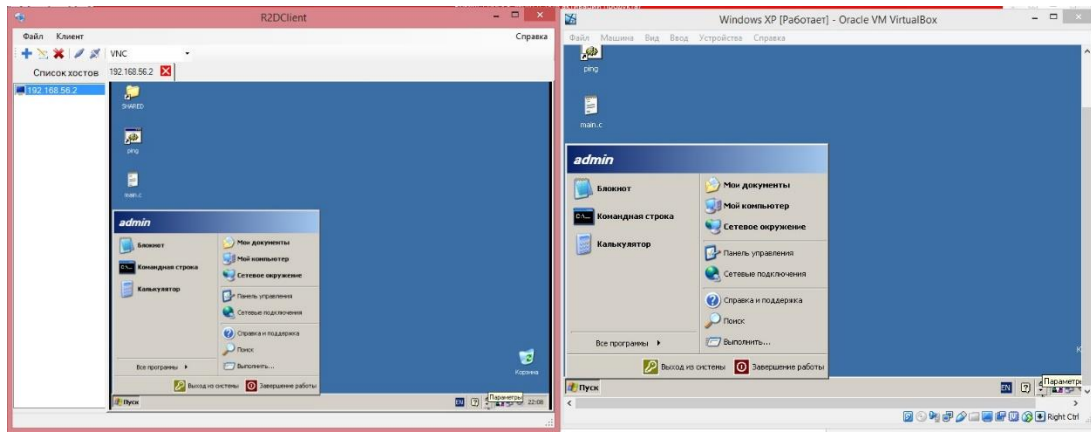


Рис.2.13. Віддалений доступ до робочого столу за VNC протоколом

Для тестування функцій адміністрування видалимо нового користувача з іменем Сергій за допомогою Облікових записів користувачів панелі інструментів (рис. 2.14-2.15).

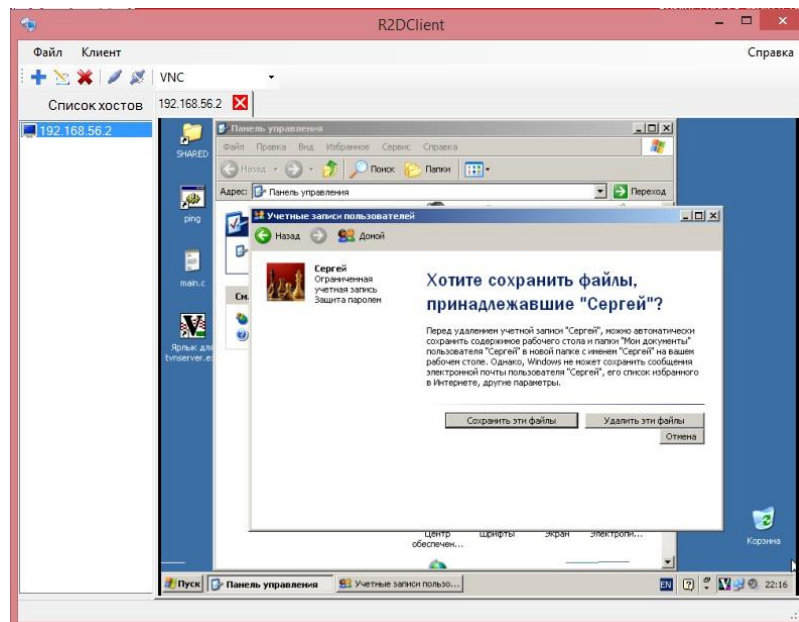


Рис. 2.14. Видалення запис користувача на клієнтському додатку

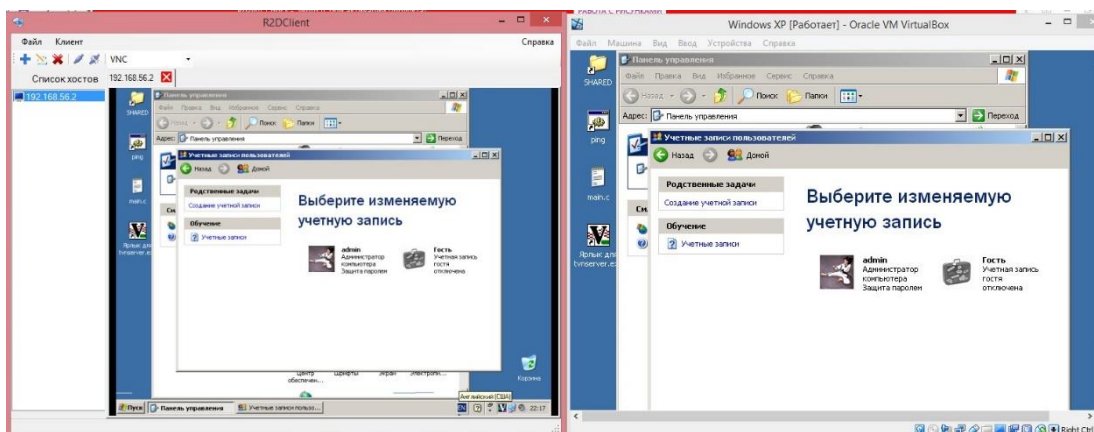


Рис. 2.15. Результаты віддаленого адміністрування

4. Підключення до хосту, дані про який зберігаються в файлі (\* .rdp). В першу чергу створимо rdp-файл через вікно Підключення до віддаленого робочо столу. Зміст rdp-файлу наведено у лістингу нижче. IP - адреса віддаленого комп'ютера виділена жирним шрифтом.

Лістинг 2.9

```
screen mode id:i:2
use multimon:i:0
desktopwidth:i:1600
desktopheight:i:900
session bpp:i:32
winposstr:s:0,3,0,0,800,600
compression:i:1
keyboardhook:i:2
audiocapturemode:i:0
videoplaybackmode:i:1
connection type:i:7
networkautodetect:i:1
bandwidthautodetect:i:1
displayconnectionbar:i:1
enableworkspacereconnect:i: 0
disable wallpaper:i:0
allow font smoothing:i:0
allow desktop composition:i:0
disable full window drag:i:1
disable menu anims:i:1
disable themes:i:0
disable cursor setting:i:0
bitmapcachepersistenable:i:1
full
address:s:192.168.56.2
```



audiomode:i:0  
redirectprinters:i:1  
redirectcomports:i:0  
redirectsmartcards:i:1  
redirectclipboard:i:1  
redirectposdevices:i:0  
drivestoredirect:s:  
autoreconnection  
enabled:i:1 authentication  
level:i:2 prompt for  
credentials:i:0 negotiate  
security layer:i:1  
remoteapplicationmode:i:0

Виконаємо команду Файл-Відкрити на клієнтському додатку. Відкриється вікно діалогу для вибору rdp-файлу (рис. 2.16). Після вибору файлу (Default.rdp) в панелі Список хостів клієнтського додатку з'явиться IP-адреса віддаленого комп'ютера, після чого можна виконувати операцію з'єднання (рис. 2.17).

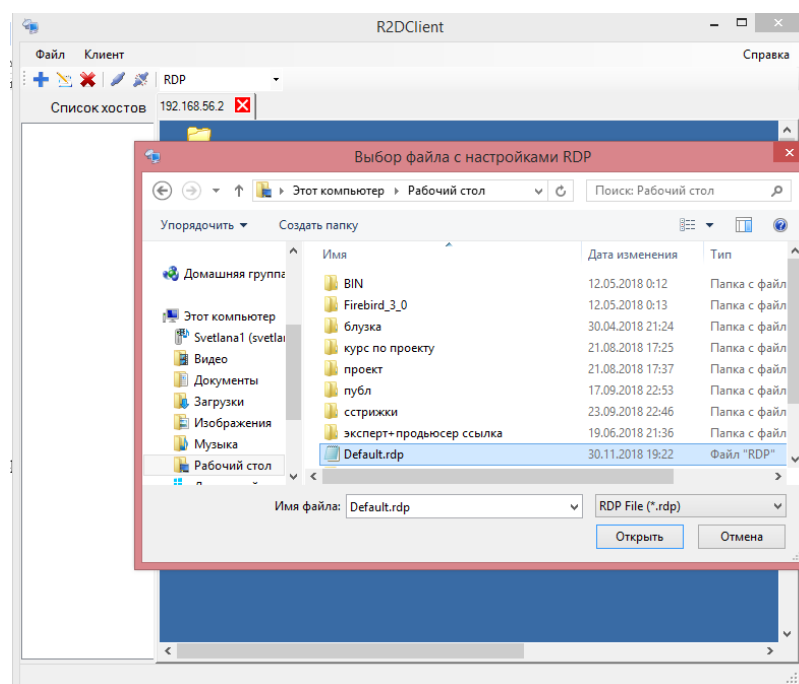


Рис. 2.16. Вікно діалогу для вибору rdp-файлу

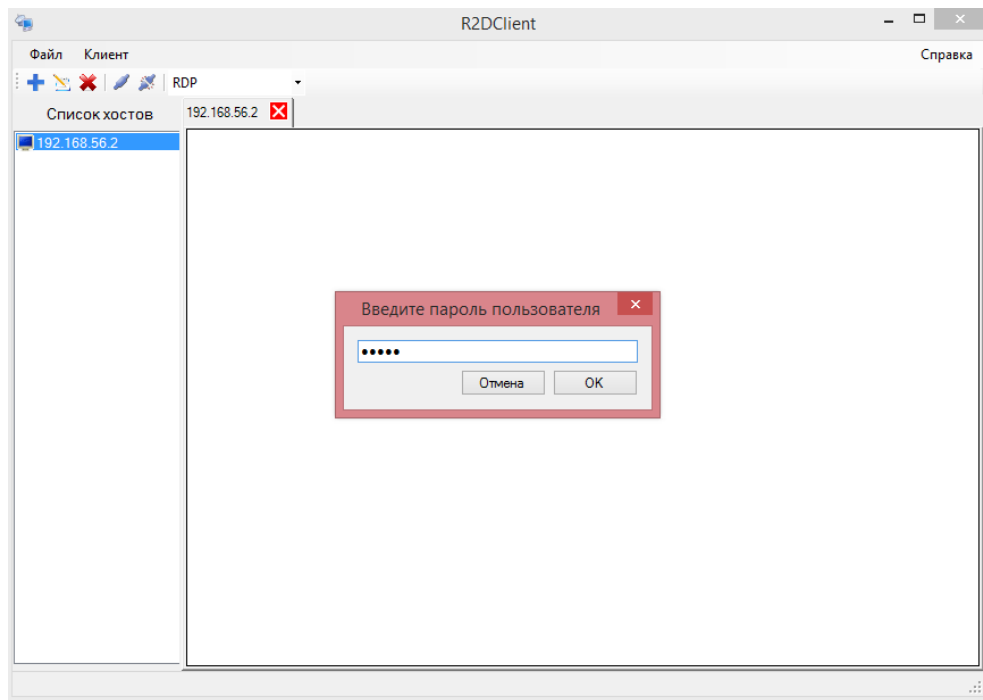


Рис. 2.17. Вікно з'єднання з віддаленим комп'ютером за допомогою rdp-файлу

Висновок: за результатами тестування розроблений додаток повністю відповідає усім цілям та вимогам.

### **Висновок до розділу**

У другому розділі бакалаврської роботи було обґрунтовано інструменти розробки додатку, а саме: обрано мову програмування C# і середовище розробки Microsoft Visual Studio 2010; для реалізації інтерфейсної частини додатку застосовується бібліотека Windows Forms на основі технології .Net Framework 4; в реалізації моделі доступу до даних застосовується технологія ADO.NET.

- Також сформульовано основні цілі та вимоги до додатку:
- здійснювати віддалений доступ за протоколами RDP і VNC.
- підключатися до хосту, дані про який зберігаються в файлі (\*.rdp).
- спостерігати і керувати віддаленим комп'ютером в мережі.
- зберігати дані про вузол підключення в базі даних.

Для реалізації клієнта було вирішено взяти за основу трирівневу архітектуру, кожен шар якої буде відповідати за свою логічну частину: рівень відображення, рівень логіки - управління підключенням, рівень протоколу.

Для реалізації можливості зберігання даних про хости, що підключаються, була використана база даних SQLite на базі технології доступу до даних ADO.NET.

Таким чином, було розроблено клієнтський додаток віддаленого доступу на платформі .Net, який відповідає всім встановленим цілям і вимогам.

## ВИСНОВКИ

Під час виконання кваліфікаційної роботи було проведено аналіз технологій віддаленого доступу та розроблено клієнтський додаток на платформі .NET

Встановлено, що віддалений доступ являє собою функцію, що дозволяє користувачеві підключатися до комп'ютера через Інтернет за допомогою іншого ПК. Одним з видів віддаленого доступу є доступ до віддаленого робочого столу (Remote Desktop), доступ до якого здійснюється на базі архітектури «сервер - термінал».

Проаналізовано принципи роботи протоколів віддаленого доступу: RDP, VNC, ICA. На підставі проведеного аналізу було вирішено обрати для розробки проекту протоколи RDP і VNC, оскільки їх застосування дозволить реалізувати віддалений доступ не тільки до комп'ютерів з операційною системою Windows, а також дасть можливість застосовувати різні платформи на клієнті і сервері віддаленого доступу.

Проведений аналіз додатків (TeamViewer, LiteManager, Ammy admin, RAdmin) довів, що програми, що забезпечують віддалений доступ та віддалене адміністрування, потребують наявності (придбання) ліцензії для довготривалої роботи з програмою. Тому розробка власного додатку віддаленого доступу може вирішити цю проблему.

Під час розробки клієнтського додатку було обґрунтовано інструменти розробки, а саме: обрано мову програмування C # і середовище розробки Microsoft Visual Studio 2010; для реалізації інтерфейсної частини додатку застосовується бібліотека Windows Forms на основі технології .Net Framework 4; в реалізації моделі доступу до даних застосовується технологія ADO.NET.

Також сформульовано основні цілі та вимоги до додатку: здійснювати віддалений доступ за протоколами RDP і VNC; підключатися до хосту, дані про який зберігаються в файлі (\* .rdp); спостерігати і керувати віддаленим комп'ютером в мережі; зберігати дані про вузол підключення в базі даних.

Для реалізації клієнта було вирішено взяти за основу трирівневу

архітектуру: рівень відображення, рівень логіки - управління підключенням, рівень протоколу.

Для реалізації можливості зберігання даних про хости була використана база даних SQLite на базі технології доступу до даних ADO.NET.

Таким чином, було розроблено клієнтський додаток віддаленого доступу на платформі .Net, який відповідає всім встановленим цілям і вимогам.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вилдермьюс Ш. Практическое использование ADO.NET. Доступ к данным в Internet. / Вилдермьюс, Шон. : Пер. с англ. — М. : Издательский дом "Вильямс", 2003. — 288 с.
2. Виртуализация Citrix XenServer, XenDesktop и XenApp [Электронный ресурс]. – Режим доступа к ресурсу: <http://www.vmggu.ru/citrix-xen>.
3. Ибе О. Сети и удаленный доступ. Протоколы, проблемы, решения / Оливер Ибе. – Пер. И. Синицын. — М.: ДМК Пресс, 2002. — 336 с.
4. Калюжный А.Д. Анализ средств сжатия видеосигнала / А.Д. Калюжный, Г.В. Табунщик // Системи обробки інформації: зб. наук. пр. – Х.: ХУПС, 2010. – Вип. 7(88). – С. 200.
5. Мазерс Т.В. Архитектура тонкого клиента в WindowsNT/2000. Реализация терминальных служб и Citrix MetaFrame / Т.В. Мазерс. – М.: Вильямс, 2001. – 800 с.
6. Норма: Терминальные решения [Электронный ресурс]. – Режим доступа к ресурсу: [http://www.norma-ts.ru/support\\_info\\_04.shtml](http://www.norma-ts.ru/support_info_04.shtml).
7. Переход на терминальные системы в условия экономического кризиса [Электронный ресурс]. – Режим доступа к ресурсу: [http://omsk.narod.ru/documents/5\\_prichin.html](http://omsk.narod.ru/documents/5_prichin.html)
8. Принцип работы Microsoft Terminal Services [Электронный ресурс]. Режим доступа: [http://technet.microsoft.com/en-us/library/cc755399\(W5.10\).aspx](http://technet.microsoft.com/en-us/library/cc755399(W5.10).aspx)
9. Решения для удаленной работы с домашним компьютером. [Электронный ресурс]. – Режим доступа: <http://compress.ru/> (дата обращения: 28.11.2016).
10. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Рихтер Дж. — 4-е изд. — СПб.: Питер, 2013. — 896 с
11. Робинсон С. C# для профессионалов / Робинсон С., Корнес О., Глинн Д. и др. — Изд-во: "Лори", Москва, 2003. — 1024 с.

12. Сеть и удаленный доступ к сети [Электронный ресурс]. – Режим доступа к ресурсу: <http://mif.vspu.ru/books/w2k/gl16/gl16.html>
13. Системы удаленного управления. [Электронный ресурс]. – Режим доступа: <http://www.osp.ru/> (дата обращения: 28.11.2016).
14. Создание и настройка виртуальной машины в VirtualBox [Электронный ресурс]. – Режим доступа к ресурсу: <https://www.gotoadm.ru/create-and-settings-virtual-machine-in-virtualbox/>
15. Терминальные решения [Электронный ресурс]. – Режим доступа к ресурсу: [http://www.onix.kiev.ua/terminal\\_advantages.asp](http://www.onix.kiev.ua/terminal_advantages.asp)
16. Тонкие клиенты и терминальные системы Citrix и Windows [Электронный ресурс]. – Режим доступа к ресурсу: [http://www.uw.ru/thin\\_client/](http://www.uw.ru/thin_client/).
17. Удаленный доступ к ПК без риска // CHIP. — 2015. — №12. — [Электронный ресурс]. – Режим доступа к ресурсу: <https://ichip.ru/category/podborki>
18. Управление «Институт информатики ИжГТУ» [Электронный ресурс]. – Режим доступа к ресурсу: <http://www.pro-spo.ru/index.php>
19. Шилдт Г. С#: Учебный курс / Шилдт Г. — СПб.: Питер, 2003. — 512 с.
20. Черкасова Н.И., Сетевые операционные системы. Сетевые файловые системы и служба каталогов. Учебное пособие. [Текст] / Н.И. Черкасова – М.: МГТУ ГА, 2010. –68 с.
21. RAdmin – руководство по продажам [Электронный ресурс]. – Режим доступа: <http://old.mont.ru/docs/> (дата обращения: 28.11.2016).
22. Remote Desktop Protocol Performance [Электронный ресурс]. – Режим доступа к ресурсу: [http://download.microsoft.com/download/4/d/9/4d9ae285-3431-4335-a86e-969e7a146d1b/rdp\\_performance\\_whitepaper.docx](http://download.microsoft.com/download/4/d/9/4d9ae285-3431-4335-a86e-969e7a146d1b/rdp_performance_whitepaper.docx).
23. Security Lab by Positive Technologies [Electronic resource]. – Resource Access Mode <http://www.securitylab.ru/analytics/367591.php>
24. The RFB Protocol [Electronic resource]. – Resource Access Mode:

- <http://www.tigervnc.com/cgi-bin/rfbproto#tight-security-type>.
25. The Trusted Solution for Remote Desktop Control [Electronic resource]. – Resource Access Mode: <https://www.teamviewer.com/en/solutions/remote-desktop/#gref>
26. VNC Virtual Network Computing [Electronic resource]. – Режим доступа к ресурсу: <https://www.raspberrypi.org/documentation/remote-access/vnc/>
27. Windows Shop [Electronic resource]. Resource Access Mode: <http://www.microsoft.com/windows/buy/default.aspx>
28. What is a VNC (Virtual Network Computing)? [Electronic resource]. – Resource Access Mode: <http://www.remoteaccess.org/what-is-a-vnc/>



## ДОДАТОК

### Лістинг файлу MainWindow.cs

```
using System;
using System.Collections.Generic; using System.ComponentModel; using
System.Data;
using System.Drawing; using System.Linq; using System.Text;
using System.Windows.Forms; using RDPFileReader;
using VncSharp;
namespace R2DClient
{
    public partial class MainWindow : Form
    {
        private RDPSettingsDialog RDPSettingsDlg = new RDPSettingsDialog();
        private SQLiteInterface SQL = new SQLiteInterface(); private
List<RemoteAbstractWidget> widgets;
        public MainWindow()
        {
            InitializeComponent();
            widgets = new List<RemoteAbstractWidget>();
            cbVariantClient.SelectedIndex = 0;
            closeMenu();
            RDPSettingsDlg.setSize(Convert.ToInt32(SettingsIni.IniReadValue("RDP",
"size"))));
            RDPSettingsDlg.setColor(Convert.ToInt32(SettingsIni.IniReadValue("RDP"
, "color"))));
            SQL.connectionString = "Data Source=hosts.db; Version=3;";
            SQL.addTable("hostdata");
            SQL.fill("hostdata");
            DataTable table = SQL.dataTable("hostdata"); for (int i = 0; i <
table.Rows.Count; ++i)
```

```

ipList.Items.Add(new IListBoxItem(table.Rows[i][1].ToString(), 0));
ipList.SelectedIndex = 0;
}
private void settingsRDPMenuItem_Click(object sender, EventArgs e)
{
    RDPSettingsDlg.setSize(Convert.ToInt32(SettingsIni.IniReadValue("RDP",
"size")));
    RDPSettingsDlg.setColor(Convert.ToInt32(SettingsIni.IniReadValue("RDP",
, "color")));
    RDPSettingsDlg.ShowDialog();
}
private void deleteMenuItem_Click(object sender, EventArgs e)
{
    DataTable table = SQL.dataTable("hostdata");
table.Rows[ipList.SelectedIndex].Delete(); SQL.save("hostdata");
ipList.Items.RemoveAt(ipList.SelectedIndex);
}
private void deleteHostBotton_Click(object sender, EventArgs e)
{
    deleteMenuItem_Click(sender, e);
}
private void addHostMenuItem_Click(object sender, EventArgs e)
{
    AddHostDialog HostDlg = new AddHostDialog(SQL);
HostDlg.ShowDialog();
    string newHost = HostDlg.getNewHost(); if (newHost != string.Empty)
ipList.Items.Add(new IListBoxItem(newHost, 0));
}
private void addHostButton_Click(object sender, EventArgs e)
{

```

```

addHostMenuItem_Click(sender, e);
}

private void editHostStripMenuItem_Click(object sender, EventArgs)
{
    AddHostDialog HostDlg = new AddHostDialog(SQL); DataTable table =
SQL.dataTable("hostdata"); DataRow row = table.Rows[ipList.SelectedIndex];
HostDlg.ShowDialog(row);

    string newHost = HostDlg.getNewHost(); if (newHost != string.Empty)
    ipList.Items[ipList.SelectedIndex]=new
    IListItem(newHost, 0);
}

private void openHostEdit(object sender, EventArgs e)
{
    editHostStripMenuItem_Click(sender, e);
}

private void editHostBotton_Click(object sender, EventArgs e)
{
    editHostStripMenuItem_Click(sender, e);
}

private void rdpConnectMenuItem_Click(object sender, EventArgs e)
{
    RemoteRDPCClient client = new RemoteRDPCClient(); widgets.Add(client);
   TabPage NewPage = new TabPage();
    NewPage.Text= Convert.ToString(ipList.Items[ipList.SelectedIndex]);
    NewPage.Controls.Add(client.Widget);
tabControl.TabPages.Add(NewPage); tabControl.SelectedIndex =
tabControl.TabCount - 1;

    DataTable table = SQL.dataTable("hostdata"); DataRow row =
table.Rows[ipList.SelectedIndex];
    if (client.setData(row))

```

```

    {
        closeMenu(client); client.setColorDepth(RDPSettingsDlg.getColorDepth());
client.setDesktopSize(RDPSettingsDlg.getDesktopSize()); client.connect();
    }
    else disconnectButton_Click(sender, e);
    }

    private void vncConnectMenuItem_Click(object sender, EventArgs e)
    {
        RemoteVNCCClient client = new RemoteVNCCClient(); widgets.Add(client);
       TabPage NewPage = new TabPage();
        NewPage.Text= Convert.ToString(ipList.Items[ipList.SelectedIndex]);
        NewPage.Controls.Add(client.Widget); tabControl.TabPages.Add(NewPage);
        tabControl.SelectedIndex = tabControl.TabCount - 1;
        DataTable table = SQL.dataTable("hostdata"); DataRow row =
table.Rows[ipList.SelectedIndex];
        if (client.setData(row))
        {
            closeMenu(client); client.connect();
        }
        else
        disconnectButton_Click(sender, e);
    }

    private void openMenuItem_Click(object sender, EventArgs e)
    {
        OpenFileDialog myOpenFileDialog; myOpenFileDialog = new
OpenFileDialog(); myOpenFileDialog.Filter = "RDP File|*.rdp";
        myOpenFileDialog.Title = "Выбор файла с настройками RDP";
        myOpenFileDialog.ShowDialog();
        if (myOpenFileDialog.FileNames.Count() > 0)
        {

```

```

RDPFile MyRdpFile = new RDPFile();
MyRdpFile.Read(myOpenFileDialog.FileName);
    if (MyRdpFile.FullAddress != string.Empty)
    {
        DataRow row = SQL.newRow("hostdata");
        row["ip"] = MyRdpFile.FullAddress; row["port"] = string.Empty;
row["login"] = MyRdpFile.Username;
        row["password"] = Helper.XorString(MyRdpFile.Password,
SQL.addNewRow("hostdata", row);
        ipList.Items.Add(new IListItem(MyRdpFile.FullAddress,
    }
    }

private void closeTab(object sender, ClosingEventArgs e)
{
    if (tabControl.SelectedIndex >= 0)
    {
        RemoteAbstractWidgetaWidget= widgets[tabControl.SelectedIndex];
        aWidget.disconnect(); widgets.RemoveAt(tabControl.SelectedIndex);
tabControl.TabPages.RemoveAt(tabControl.SelectedIndex);
    }
}

private void disconnectButton_Click(object sender, EventArgs e)
{
    if (tabControl.SelectedIndex >= 0)
    {
        RemoteAbstractWidgetaWidget= widgets[tabControl.SelectedIndex];
        aWidget.disconnect(); widgets.RemoveAt(tabControl.SelectedIndex);
tabControl.TabPages.RemoveAt(tabControl.SelectedIndex);
    }
    if (tabControl.SelectedIndex != -1)

```

```

{
    RemoteAbstractWidget aWidget = widgets[tabControl.SelectedIndex];
    closeMenu(aWidget);
}
else closeMenu();
}

private void connectBotton_Click(object sender, EventArgs e)
{
    if (cbVariantClient.SelectedIndex == 0) rdpConnectMenuItem_Click(sender,
e);

    if (cbVariantClient.SelectedIndex == 1)
vncConnectMenuItem_Click(sender, e);
}

private void ctrl_alt_delVNCMenuItem_Click(object sender,
EventArgs e)
{
    RemoteAbstractWidget aWidget = widgets[tabControl.SelectedIndex];
    if (aWidget.isClient() == "VNC")
    {
        RemoteVNCClient client = (RemoteVNCClient)aWidget;
        if (client.Widget.IsConnected)
client.Widget.SendSpecialKeys(SpecialKeys.CtrlAltDel);
    }
}

private void ctrl_escVNCMenuItem_Click(object sender, EventArgs e)
{
    RemoteAbstractWidget aWidget =
widgets[tabControl.SelectedIndex];
    if (aWidget.isClient() == "VNC")
    {

```

```

        RemoteVNCClient client = (RemoteVNCClient)aWidget; if
(client.Widget.IsConnected)
        client.Widget.SendSpecialKeys(SpecialKeys.CtrlEsc);
    }
}

private void alt_f4VNCMenuItem_Click(object sender, EventArgs e)
{
    RemoteAbstractWidget aWidget= widgets[tabControl.SelectedIndex];
    if (aWidget.isClient() == "VNC")
    {
        RemoteVNCClient client = (RemoteVNCClient)aWidget; if
(client.Widget.IsConnected)
        client.Widget.SendSpecialKeys(SpecialKeys.AltF4);
    }
}

private void ctrlVNCMenuItem_Click(object sender, EventArgs e)
{
    RemoteAbstractWidget aWidget= widgets[tabControl.SelectedIndex];
    if (aWidget.isClient() == "VNC")
    {
        RemoteVNCClient client = (RemoteVNCClient)aWidget; if
(client.Widget.IsConnected)
        client.Widget.SendSpecialKeys(SpecialKeys.Ctrl, false/* don't release CTRL
*/);
    }
}

private void altVNCMenuItem_Click(object sender, EventArgs e)
{
    RemoteAbstractWidget
aWidget=
widgets[tabControl.SelectedIndex];

```

```

        if (aWidget.isClient() == "VNC")
        {
            RemoteVNCClient client = (RemoteVNCClient)aWidget; if
(client.Widget.IsConnected)
                client.Widget.SendSpecialKeys(SpecialKeys.Alt,false/* don't release ALT
*/);
        }
    }

    private void copyClipboardMenuItem_Click(object sender, EventArgs e)
    {
        RemoteAbstractWidget aWidget= widgets[tabControl.SelectedIndex];
        if (aWidget.isClient() == "VNC")
        {
            RemoteVNCClient client = (RemoteVNCClient)aWidget; if
(client.Widget.IsConnected)
            {
                client.Widget.FillServerClipboard();
                локальный хост",
                MessageBox.Show(this, "Буффер обмена скопированн на
                "Копирование буффера обмена", MessageBoxButtons.OK,
                MessageBoxIcon.Information);
            }
        }
    }

    private void closeMenu(RemoteAbstractWidget aWidget)
    {
        if (aWidget.isClient() == "RDP")
        {
            vncMenuItem.Enabled = false; sendKeysVNCMenuItem.Enabled = false;
            ctrl_alt_delVNCMenuItem.Enabled = false;

```



```

        ctrl_escVNCMenuItem.Enabled = false; alt_f4VNCMenuItem.Enabled =
false; ctrlVNCMenuItem.Enabled = false; altVNCMenuItem.Enabled = false;
copyClipboardMenuItem.Enabled = false;

        rdpMenuItem.Enabled = true; settingsRDPMMenuItem.Enabled = true;
        cbVariantClient.SelectedIndex = 0;
    }

    if (aWidget.isClient() == "VNC")
    {
        vncMenuItem.Enabled = true; vncMenuItem.Enabled = true;
sendKeysVNCMenuItem.Enabled = true; ctrl_alt_delVNCMenuItem.Enabled =
true; ctrl_escVNCMenuItem.Enabled = true; alt_f4VNCMenuItem.Enabled = true;
ctrlVNCMenuItem.Enabled = true; altVNCMenuItem.Enabled = true;
copyClipboardMenuItem.Enabled = true;

        rdpMenuItem.Enabled = false; settingsRDPMMenuItem.Enabled = false;
        cbVariantClient.SelectedIndex = 1;
    }
}

private void closeMenu()
{
    vncMenuItem.Enabled = false; sendKeysVNCMenuItem.Enabled = false;
ctrl_alt_delVNCMenuItem.Enabled = false; ctrl_escVNCMenuItem.Enabled =
false; alt_f4VNCMenuItem.Enabled = false; ctrlVNCMenuItem.Enabled = false;
altVNCMenuItem.Enabled = false; copyClipboardMenuItem.Enabled = false;
rdpMenuItem.Enabled = true; settingsRDPMMenuItem.Enabled = true;
}

private void selectTab(object sender, TabControlEventsArgs e)
{
    if (tabControl.SelectedIndex != -1)
    {
        RemoteAbstractWidget aWidget= widgets[tabControl.SelectedIndex];

```

```

        closeMenu(aWidget);
        if (aWidget.isClient() == "RDP"){
            RemoteRDPCClient client = (RemoteRDPCClient)aWidget;
tabControl.TabPages[tabControl.SelectedIndex].Update(); client.Widget.Update();
        }
        if (aWidget.isClient() == "VNC"){
            RemoteVNCCClient client = (RemoteVNCCClient)aWidget;
client.Widget.Update();
        }
        }
        else closeMenu();
    }

    private void aboutMenuItem_Click(object sender, EventArgs e)
    {
        AboutBox box = new AboutBox(); box.ShowDialog();
    }

    private void exitMenuItem_Click(object sender, EventArgs e)
    {
        for (int i = 0; i < tabControl.TabCount; ++i) disconnectButton_Click(sender,
e);
        Close();
    }

    private void tabControl_SelectedIndexChanged(object sender,
EventArgs e)
    {
    }
    }
    }
}

```