МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ
Циклова комісія (кафедра) комп’ютерної інженерії та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему

АРХІТЕКТУРА ТА РЕАЛІЗАЦІЯ VPN

Виконав:
студент групи 1П-21
Спеціальності
121 Інженерія програмного забезпечення
Антон ШЕПІЛЬ
Керівник:
Наталя ФАЛЬЧЕНКО

Черкаси 2025

АНОТАЦІЯ

У кваліфікаційній роботі на тему "Архітектура та реалізація VPN" проведено комплексне дослідження теоретичних та практичних аспектів побудови віртуальних приватних мереж (VPN), що є актуальними у зв’язку з постійним зростанням потреби в захисті інформації під час передавання даних через публічні мережі, зокрема Інтернет. У теоретичній частині роботи розглянуто основні функції VPN, їх призначення, переваги використання, а також ключові архітектурні підходи до реалізації VPN-з’єднань. Особливу увагу приділено протоколам, що забезпечують функціонування VPN: PPTP, L2TP/IPsec, OpenVPN, IKEv2 та WireGuard. Описано їх технічні особливості, механізми аутентифікації, підтримку шифрування, швидкодію та сумісність із сучасними операційними системами.

Практична частина роботи присвячена розробці власного VPN-додатку за допомогою мови програмування C# та графічного інтерфейсу на основі технології WPF (Windows Presentation Foundation). У процесі реалізації використано сервіс VPNBook як безкоштовний сервер для встановлення захищених з’єднань. Особливу увагу приділено інтеграції клієнтської частини додатку з системними ресурсами Windows, налаштуванню параметрів конфігурації OpenVPN, обробці логів, повідомлень про помилки та зручності користувацького інтерфейсу.

Також у роботі досліджено методи шифрування, що застосовуються у VPN-з’єднаннях, зокрема алгоритми AES, RSA, TLS, їх роль у забезпеченні конфіденційності, цілісності та автентичності переданих даних. Результатом кваліфікаційної роботи є працездатний програмний продукт, що демонструє основи взаємодії з VPN та може бути основою для подальшого розвитку або адаптації під інші сервери чи протоколи.

ABSTRACT

The thesis titled "Architecture and Implementation of VPN" presents a comprehensive study of both theoretical and practical aspects of virtual private network (VPN) construction, which is increasingly relevant due to the growing demand for secure data transmission over public networks, especially the Internet. The theoretical part examines the main functions of VPNs, their purpose, advantages, and architectural approaches to building secure connections. Special attention is given to VPN protocols such as PPTP, L2TP/IPsec, OpenVPN, IKEv2, and WireGuard. Their technical characteristics, authentication mechanisms, encryption support, performance, and compatibility with modern operating systems are discussed in detail.

The practical part of the research focuses on the development of a custom VPN application using the C# programming language and WPF (Windows Presentation Foundation) for the graphical user interface. The VPNBook service was used as a free VPN server for establishing secure connections. Particular emphasis was placed on integrating the client-side application with Windows system resources, configuring OpenVPN settings, handling log files and error messages, and providing a user-friendly interface.

The thesis also investigates encryption techniques used in VPN connections, such as AES, RSA, and TLS algorithms, and their role in ensuring confidentiality, integrity, and authenticity of transmitted data. The final result of the research is a fully functional software product that demonstrates the fundamentals of working with VPN connections and can serve as a foundation for further development or adaptation to other servers or protocols.

ЗМІСТ

ВСТУП

У сучасному цифровому світі передача даних через відкриті мережі, такі як Інтернет, стикається з численними загрозами безпеки. Хакери, зловмисники, а також державні та приватні організації можуть здійснювати моніторинг, перехоплення або зміну інформації, що передається через незахищені канали зв'язку. Це створює критичну потребу в надійних технологіях для захисту інформації та забезпечення конфіденційності користувачів у мережі. Одним із найефективніших рішень для вирішення цієї проблеми є віртуальні приватні мережі (VPN).

VPN-технології дозволяють створювати захищені канали зв’язку між користувачами та віддаленими ресурсами, використовуючи механізми шифрування і автентифікації. Це забезпечує не лише конфіденційність переданих даних, але й їхню цілісність і доступність, що є ключовими принципами інформаційної безпеки. VPN застосовуються як у корпоративному середовищі, де захищені комунікації між офісами компанії та віддаленими співробітниками є критично важливими, так і серед звичайних користувачів, які прагнуть захистити свої особисті дані від стеження та зловмисних атак.

Завдяки VPN користувачі можуть безпечно підключатися до публічних Wi-Fi-мереж, що особливо актуально для мандрівників та людей, які часто працюють у кафе, готелях або аеропортах. Крім того, VPN дозволяє обходити географічні обмеження та цензуру, що накладаються деякими урядами або сервісами потокового відео, соціальних мереж і вебсайтів. Це робить VPN необхідним інструментом для громадян країн із жорсткими обмеженнями доступу до інформації, а також для бізнесів, які хочуть забезпечити безпечний доступ до своїх внутрішніх ресурсів з будь-якої точки світу.

Предмет дослідження – технології розробки VPN.

Об’єкт дослідження – система організації безпеки в інформаційному просторі.

Метою кваліфікаційної роботи є розробка архітектури та реалізація VPN, що дозволить детально розглянути всі аспекти роботи цієї технології. У процесі виконання роботи будуть досліджені принципи функціонування VPN, їхні основні типи (наприклад, PPTP, L2TP/IPSec, OpenVPN, WireGuard), механізми автентифікації, алгоритми шифрування, а також можливості оптимізації продуктивності та зниження затримок при передачі даних через VPN-мережу.

Актуальність теми зумовлена не лише зростаючим рівнем кіберзагроз, а й посиленням державного контролю за інтернет-трафіком. Останнім часом багато урядів почали активно впроваджувати засоби моніторингу та цензури, що ускладнює доступ до вільної інформації та ставить під загрозу конфіденційність особистих даних користувачів. Крім того, блокування інтернет-ресурсів для користувачів із певних країн змушує їх шукати способи обходу цих обмежень. Використання VPN дозволяє не тільки зберегти анонімність в мережі, але й забезпечити безпечний обмін даними в умовах постійного стеження, збору інформації про користувачів та інших загроз цифрової безпеки.

Таким чином, дослідження та розробка VPN-системи є важливим і актуальним завданням, що дозволяє підвищити рівень безпеки користувачів в Інтернеті та забезпечити їм свободу доступу до інформації незалежно від зовнішніх обмежень.

РОЗДІЛ 1 ОГЛЯД ТА АНАЛІЗ МЕТОДІВ І ЗАСОБІВ РОЗРОБКИ ПРОГРАМНОГО СЕРЕДОВИЩА VPN

1.1 Огляд існуючих інтернет-ресурсів, що характеризують предметну область

Розвиток технологій віртуальних приватних мереж (VPN) супроводжується активним обговоренням і аналізом у наукових та технічних джерелах, а також на різних інтернет-ресурсах. Основними інформаційними платформами, що містять матеріали про VPN, є офіційні сайти розробників відповідного програмного забезпечення, наукові журнали, тематичні форуми та блогові платформи [4].

Серед найвідоміших інтернет-ресурсів, які висвітлюють питання VPN, можна виділити офіційні веб-сайти розробників VPN-рішень. Вони містять детальну технічну документацію, описи алгоритмів шифрування, особливості протоколів та рекомендації щодо впровадження.

1.1.1 Характеристика існуючих застосунків в області розробки VPN

1. OpenVPN – широко відомий проєкт з відкритим кодом, підтримуваний спільнотою. Ілюстрація логотипу застосунку OpenVPN зображена на рис. 1.1

Рисунок 1.1 - OpenVPN

2. WireGuard – інноваційний VPN-протокол, орієнтований на простоту та продуктивність. Ілюстрація логотипу протоколу WireGuard зображена на рис. 1.2

Рисунок 1.2. - Протокол WireGuard

3. Cisco VPN – корпоративне рішення, розроблене компанією Cisco Systems. Ілюстрація логотипу компанії Cisco зображена на рис. 1.3

Рисунок 1.3. – Cisco

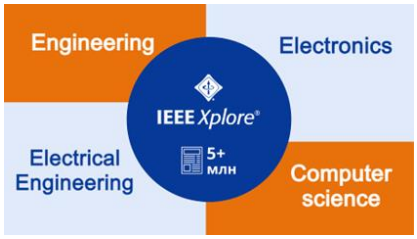
4. NordVPN – один із найпопулярніших комерційних VPN-сервісів, орієнтований на захист конфіденційності. Ілюстрація логотипу застосунку NordVPN зображена на рис. 1.4

Рисунок 1.4. - NordVPN

1.1.2 Наукові та технічні журнали, що публікують дослідження в області VPN.

Дослідження в області VPN друкуються у різноманітних технічних, наукових та науково-популярних журналах, статтях, книгах. Перелік сучасних джерел представлений у таблиці 1.1.

Таблиця 1.1 Наукові та технічні джерела що публікують дослідження в області VPN:

№	Назва	Особливості
1	IEEE Xplore 	провідне джерело наукових статей у галузі інженерії та мережевої безпеки
2	ACM Digital Library 	платформа з публікаціями щодо обчислювальних наук і кібербезпеки
3	Springer та Elsevier	великі академічні платформи з широким спектром досліджень

Перелічені ресурси надають широку спектр інформаціїї про інструменти і методи, які використовуються у VPN-технологіях.

1.1.3 Форуми та спільноти розробників

1. Stack Overflow – технічна спільнота, де обговорюються питання VPN і кібербезпеки.

Ілюстрація логотипу форуму stackoverflow зображена на рис 1.5

Рисунок 1.5. - Платформа StackOverflow

2. GitHub – платформа для зберігання та обговорення коду VPN-проєктів. Ілюстрація логотипу платформи GitHub зображена на рис 1.6

Рисунок 1.6. - Платформа GitHub

3. Reddit – майданчик для обговорень і обміну досвідом серед користувачів і фахівців. Ілюстрація логотипу форуму reddit зображена на рис 1.7

Рисунок 1.7. - Платформа Reddit

1.1.4 Комерційний підхід до статей про VPN

При аналізі інтернет-ресурсів, присвячених VPN, важливо враховувати не лише технічний аспект, а й комерційну складову подання інформації. Багато статей та оглядів VPN-продуктів можуть містити приховану або явну рекламу, що може впливати на об'єктивність поданих даних [5].

Формати комерційних статей:

- Партнерські огляди – багато сайтів публікують огляди VPN-сервісів із партнерськими посиланнями, отримуючи комісійні за залучених користувачів. Такі огляди можуть бути упередженими й акцентувати увагу на перевагах певного сервісу без згадки про його недоліки.
- SEO-оптимізовані статті – маркетингові статті, орієнтовані на залучення трафіку через пошукові системи, часто містять ключові слова та посилання на просувані VPN-сервіси.

- Порівняльні рейтинги – рейтинги VPN-сервісів, розміщені на посередницьких сайтах, часто формуються не на основі реальних тестів, а за домовленістю з компаніями.
- Брендований контент – статті, опубліковані на відомих технологічних ресурсах, можуть бути спонсоровані самими VPN-провайдерами, що знижує їхню об'єктивність.

1.1.5 Методи перевірки об'єктивності інформації

Щоб уникнути упереджених думок при вивченні VPN-технологій, рекомендується:

- Спиратися на незалежні дослідження та технічну документацію.
- Порівнювати інформацію з кількох джерел.
- Аналізувати авторів статей – перевіряти їхню експертність і наявність комерційних інтересів.
- Використовувати спеціалізовані форуми та репозиторії, де обговорюються реальні аспекти роботи VPN-сервісів.

Таким чином, комерційний підхід до висвітлення VPN-тематики відіграє значну роль у формуванні громадської думки про продукти та сервіси, тому важливо вміти критично оцінювати подану інформацію.

1.2 Дослідження реалізації механізму VPN

Віртуальні приватні мережі використовують різні механізми для створення безпечного з’єднання між користувачем і сервером. Основою роботи VPN є тунелювання даних через загальнодоступні мережі з використанням криптографічних методів для забезпечення їхньої конфіденційності [6].

1.2.1 Основні протоколи VPN

Сучасні VPN-системи використовують кілька основних протоколів для захисту даних:

1. PPTP (Point-to-Point Tunneling Protocol) – один із найстаріших VPN-протоколів, що забезпечує базове шифрування, але має низьку продуктивність. Через вразливості безпеки його використовують рідко. Схема роботи протоколу PPTP зображена на рис. 1.8

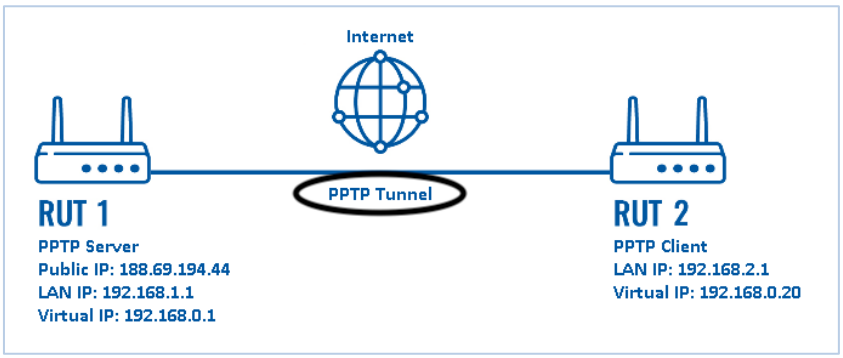


Рисунок 1.8. - Схема роботи протоколу PPTP

2. L2TP/IPSec (Layer 2 Tunneling Protocol + Internet Protocol Security) – покращена версія PPTP з додатковим шифруванням IPSec, що підвищує рівень безпеки, але вимагає значних обчислювальних ресурсів.

   Схема роботи протоколу L2TP/IPSec зображена на рис. 1.9

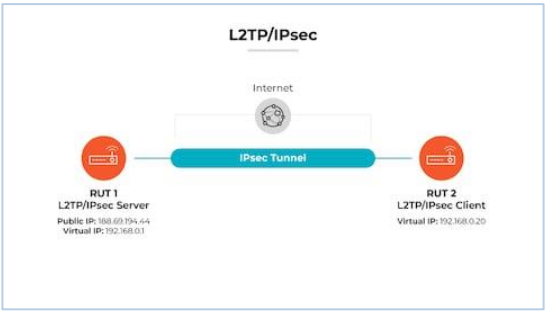


Рисунок 1.9. - Ілюстрація Layer 2 Tunneling Protocol + Internet Protocol Security

3. OpenVPN – один із найбільш безпечних і гнучких протоколів, що використовує SSL/TLS для шифрування трафіку. Завдяки відкритому

вихідному коду активно розвивається спільнотою. Схема роботи протоколу OpenVPN зображена на рис. 1.10

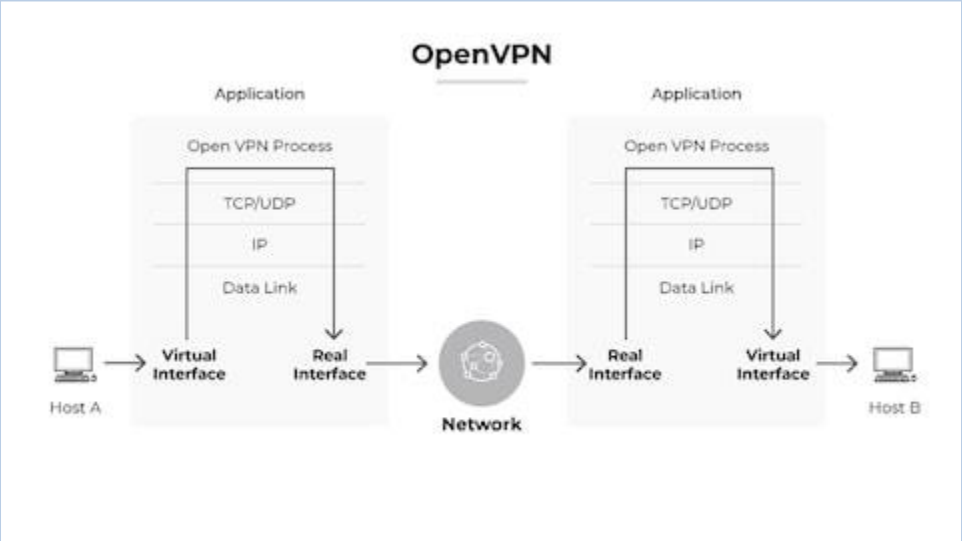


Рисунок 1.10. - Ілюстрація протоколу OpenVPN

4. WireGuard – сучасний легковаговий VPN-протокол, що забезпечує високу швидкість роботи, мінімальні затримки та надійний захист. Схема роботи протоколу WireGuard зображена на рис. 1.11

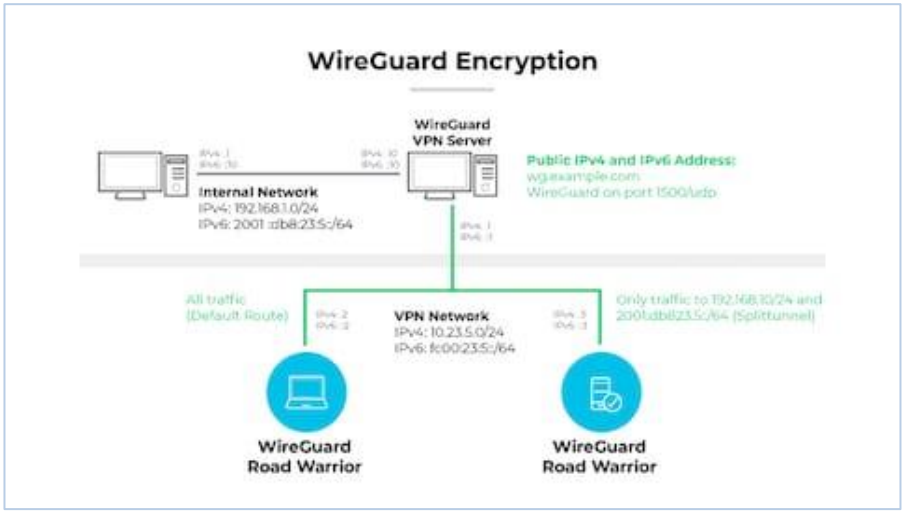


Рисунок 1.11. - Ілюстрація протоколу WireGuard

1.2.2 Алгоритми шифрування у VPN

Забезпечення безпеки у VPN досягається завдяки використанню криптографічних алгоритмів. Найпоширенішими з них є:

1. AES (Advanced Encryption Standard) – використовується в більшості сучасних VPN через свою високу стійкість до атак. Схема роботи алгоритму зображена на рис. 1.12.

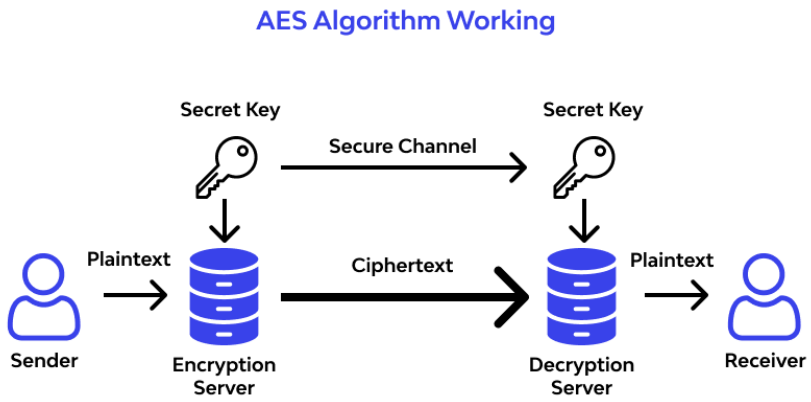


Рисунок 1.12. Ілюстрація протоколу Advanced Encryption Standard

2. ChaCha20 – застосовується в WireGuard і забезпечує ефективне шифрування з високою швидкістю. Схема роботи алгоритму зображена на рис. 1.13.

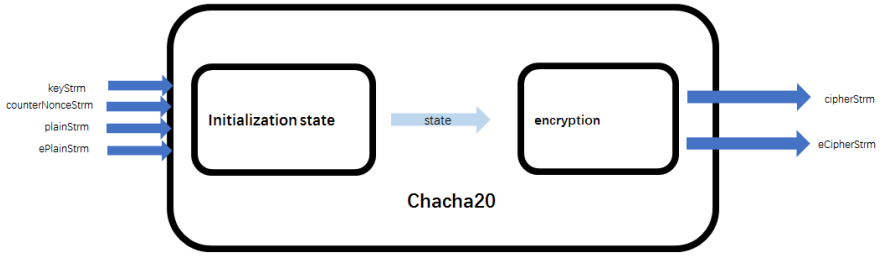


Рисунок 1.13. - Ілюстрація протоколу ChaCha20

3. RSA (Rivest-Shamir-Adleman) – алгоритм асиметричного шифрування, що використовується для встановлення захищеного з'єднання. Схема роботи алгоритму зображена на рис. 1.14.

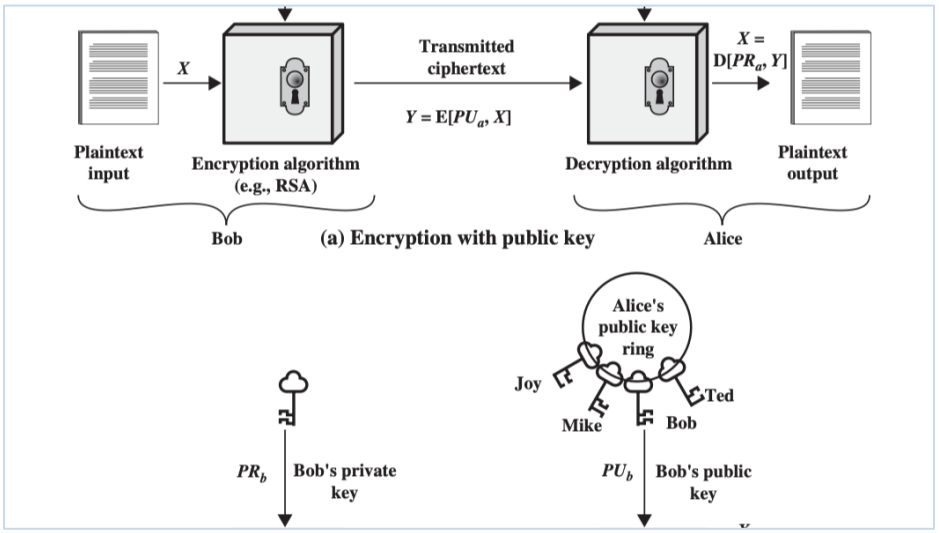


Рисунок 1.14. - Ілюстрація алгоритму Rivest-Shamir-Adleman

4. ECDH (Elliptic Curve Diffie-Hellman) – забезпечує безпечний обмін ключами між клієнтом і сервером. Схема роботи алгоритму зображена на рис. 1.15.

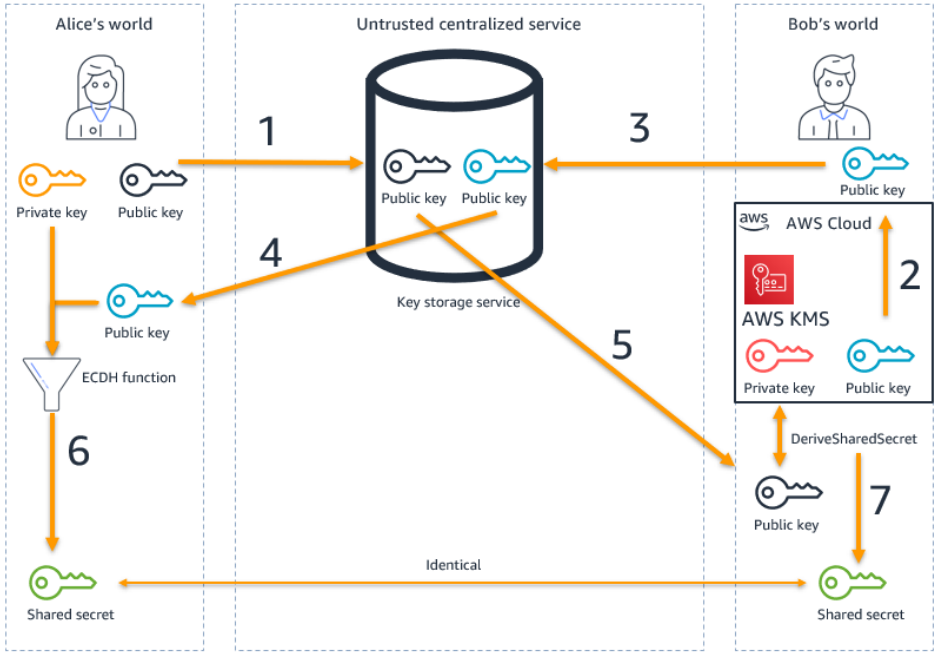


Рисунок 1.15. - Ілюстрація алгоритму Elliptic Curve Diffie-Hellman

1.2.3 Методи аутентифікації у VPN

Для забезпечення захисту доступу до VPN використовуються різні методи аутентифікації:

- аутентифікація за допомогою логіна і пароля – базовий метод, що має середній рівень захисту;
- двофакторна аутентифікація (2FA) – підвищує рівень безпеки шляхом додаткової перевірки користувача;
- сертифікатна аутентифікація – використовує цифрові сертифікати для підтвердження особи;
- аутентифікація на основі токенів – передбачає використання одноразових кодів або апаратних ключів.

1.2.4 Загальні принципи роботи VPN

Загальні принципи роботи VPN наведені нижче [7]:

1. Користувач підключається до VPN-сервера через клієнтську програму.
2. Трафік шифрується та направляється через зашифрований тунель.
3. VPN-сервер розшифровує отримані дані та передає їх до кінцевого пункту.
4. Відповідь із зовнішньої мережі проходить аналогічний шлях у зворотному порядку.

Технологія роботи VPN зображена на рис.1.16

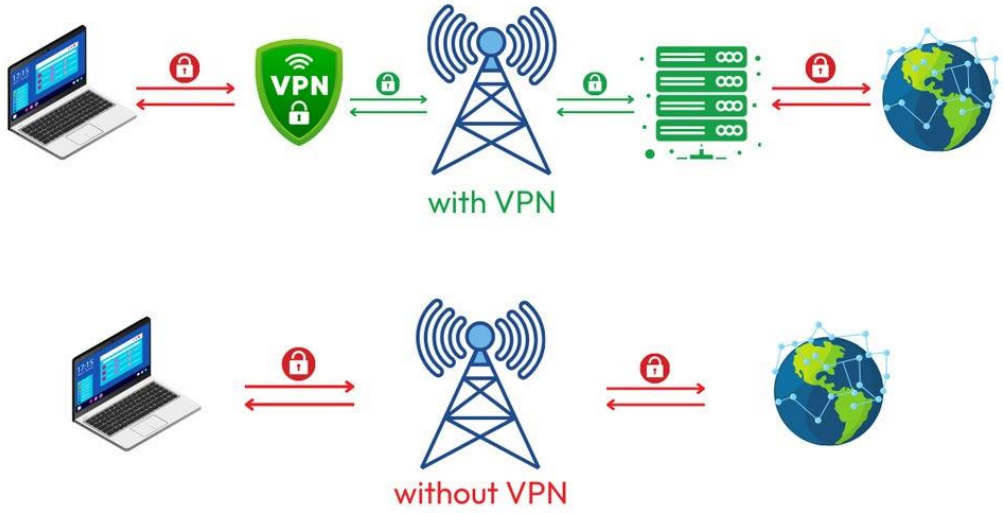


Рисунок 1.16. - Ілюстрація роботи VPN

Висновки до першого розділу

У першому розділі було проведено аналіз основних інформаційних ресурсів, що висвітлюють технології VPN, а також розглянуто механізми їх реалізації. Віртуальні приватні мережі застосовують різні протоколи, алгоритми шифрування та методи аутентифікації для забезпечення безпечного з'єднання між користувачем і мережею.

Аналіз існуючих онлайн-ресурсів показав, що сфера VPN активно розвивається, а новітні технології, такі як WireGuard, стають дедалі популярнішими завдяки високій продуктивності та простоті налаштування. Дослідження механізмів реалізації VPN дозволило визначити ключові аспекти безпеки, які необхідно враховувати під час розробки та впровадження цих систем.

Отримані результати стануть основою для подальшого вивчення аспектів безпеки VPN, а також розробки оптимальних рішень для їх ефективного використання.

РОЗДІЛ 2 РОЗРОБКА ФУНКЦІОНАЛЬНОЇ МОДЕЛІ СЕРЕДОВИЩА VPN

2.1 Теоретичні дослідження способів реалізації функціональної моделі середовища VPN

При підключенні локальної мережі організації до глобального Інтернету з’являються два основних вектори загроз: можливість несанкціонованого перехоплення даних під час їх передачі та спроби доступу до внутрішніх ресурсів мережі. Для забезпечення безпеки даних, які передаються через відкриті мережі, застосовуються такі ключові заходи: взаємна перевірка автентичності учасників обміну, шифрування переданої інформації та перевірка її цілісності [3].

Одним із сучасних підходів до організації захищеної передачі є застосування технології віртуальних приватних мереж (VPN), що створюють захищений канал або «тунель» у межах відкритої мережі. Цей тунель забезпечує надійний захист від втручання сторонніх осіб. Основні переваги VPN полягають у простій апаратній реалізації, відсутності потреби у використанні дорогих виділених каналів, можливості роботи через публічний Інтернет і збереженні високої швидкості передавання даних [3].

Існують чотири основні архітектурні моделі побудови VPN-мереж, кожна з яких має свої особливості:

1. LAN-VPN (локальна VPN-мережа): забезпечує захист даних в межах однієї організації. Реалізується через обмеження доступу до ресурсів, використання паролів, журналювання подій, шифрування конфіденційної інформації та забезпечення безпеки на рівні операційної системи.
2. Intranet-VPN (внутрішньокорпоративна VPN): дозволяє захищено з'єднувати різні підрозділи компанії, що мають географічну віддаленість. Така система використовує потужні засоби шифрування, гарантує

безперебійну роботу важливих служб і додатків, забезпечує гнучке управління користувачами [3].

3. Internet-VPN (VPN з віддаленим доступом): застосовується для організації безпечного підключення мобільних працівників і віддалених офісів до корпоративної мережі через Інтернет. Важливими складовими тут є система автентифікації, ідентифікації користувачів та централізоване управління захисними ресурсами [3].
4. Extranet-VPN (міжкорпоративна VPN): орієнтована на забезпечення безпечного обміну інформацією з зовнішніми партнерами, філіями та постачальниками. Характеризується використанням стандартизованих рішень VPN, що функціонують у різнорідних мережевих середовищах і здатні забезпечити захист навіть мультимедійного трафіку.

VPN-технології реалізуються двома основними технічними способами:

1. Через створення приватних логічних каналів типу Frame Relay або ATM, які забезпечують захищені з'єднання в межах загальної інфраструктури [3].
2. Через тунелювання IP-пакетів, коли дані шифруються та інкапсулюються у нові пакети для захищеної передачі через відкриті мережі. Це дозволяє не лише захищати вміст пакета, а й приховувати структуру внутрішньої мережі [1].
3. Шифрування заголовків, окрім вмісту, є важливим елементом захисту, оскільки воно унеможливлює аналіз мережевої топології зловмисниками. Після досягнення вузла-призначення, зашифровані пакети розшифровуються та передаються у внутрішню мережу [3].

Тунелювання також виконує функції забезпечення автентичності та цілісності даних. Воно дозволяє об'єднувати мережі з різною архітектурою та протоколами, формуючи єдиний захищений простір для обміну інформацією.

Тунельна схема представлена на рис. 2.1

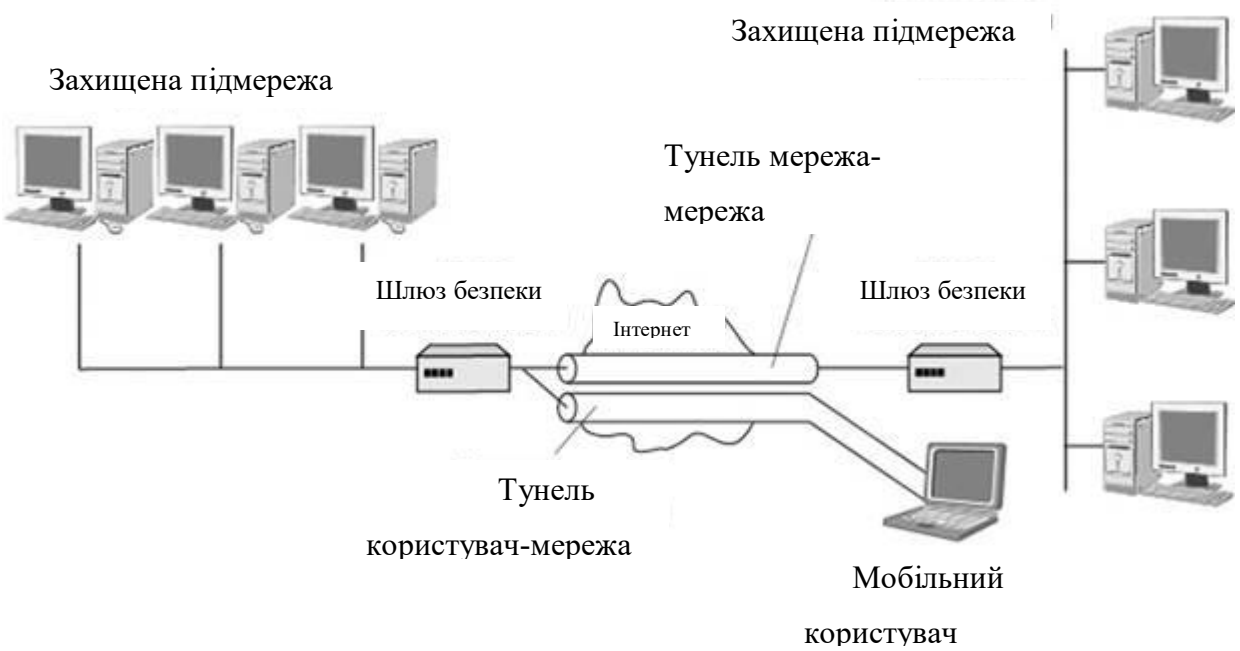


Рисунок 2.1. - Тунельна схема організації VPN-мережі

Засоби побудови VPN:

- інструменти для створення VPN можуть бути різного типу:
- програмне забезпечення, яке додає функціональність VPN до стандартної ОС;
- апаратно-програмні комплекси із спеціалізованою ОС реального часу;
- маршрутизатори чи комутатори з вбудованою підтримкою VPN;
- міжмережеві екрани з розширеною функціональністю для тунелювання.

VPN-тунелі можуть обслуговувати як цілі локальні мережі через шлюзи безпеки, так і окремих користувачів за допомогою спеціалізованого клієнтського ПЗ [2].

У середовищі Windows NT для реалізації VPN-технології застосовується протокол PPTP, який інкапсулює пакети PPP у IP-дейтаграми, використовуючи

криптографію RSA RC4. Таке рішення є доступним за вартістю і підходить для невеликих підприємств [3].

Приклад побудови VPN за допомогою Cisco описаний нижче:

Одним із лідерів на ринку рішень для побудови VPN є компанія Cisco Systems. Вона використовує власну ОС Cisco IOS (версії 12.x), а також протоколи L2TP (розроблений на базі Cisco L2F та Microsoft PPTP) і IPSec. Застосування клієнта Cisco VPN Client дозволяє ефективно організовувати захищені з'єднання «точка-точка» між віддаленими пристроями та маршрутизаторами Cisco, покриваючи всі основні сценарії використання VPN у сучасних мережах [3].

2.2 Моделювання предметної області для розробки програмного середовища

Розроблений VPN-клієнт є десктопним застосунком для ОС Windows, створеним за допомогою фреймворку WPF (Windows Presentation Foundation) мовою програмування C# [8].

Його головне призначення - автоматизація процесу підключення користувача до безкоштовного VPN-сервісу VPNBook за допомогою системного VPN-клієнта Windows.

Архітектура програми побудована за структурою MVVM – це сучасний підхід до розробки WPF-додатків, що забезпечує чіткий поділ відповідальностей між рівнями:

- Model: Реалізує логіку: формує команду для запуску VPN-підключення, виконує її через cmd.exe, контролює стан з'єднання.
- ViewModel: Реагує на дії користувача, обробляє команди (наприклад, "Підключитися"), передає дані у модель.
- View: Представлення користувацького інтерфейсу, розроблене за допомогою XAML.

UML-діаграма класів додатку зображена на рис. 2.2

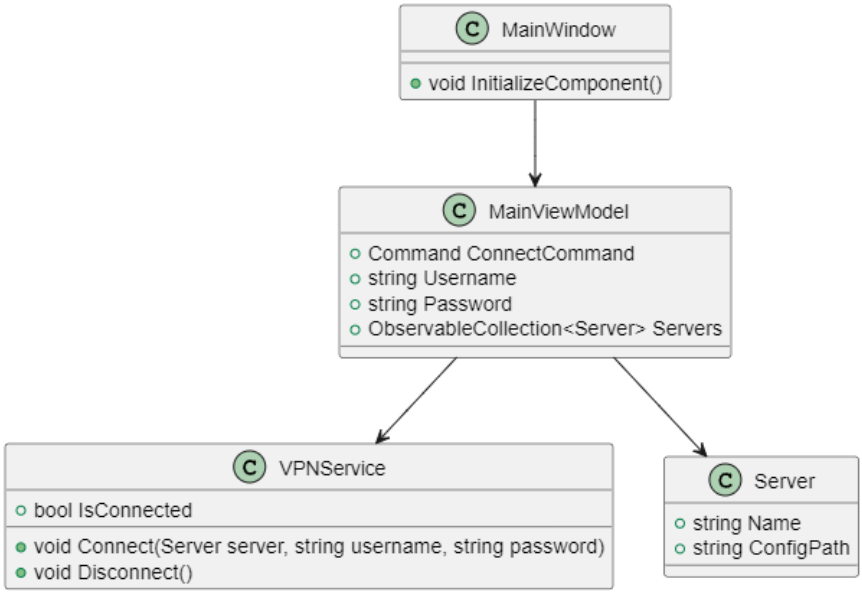


Рисунок 2.2. - UML-діаграмма класів додатку

Користувач взаємодіє з інтерфейсом, де може вибрати сервер, після чого натискає кнопку "Підключитися". Далі програма формує та виконує команду запуску VPN через .pbk (Phone Book File), що містить конфігурацію мережевого підключення.

Процес підключення описаний наступним алгоритмом:

1. Користувач запускає програму.
2. У вікні вибирає сервер (конкретний.pbk файл).
3. Після натискання кнопки "Підключитися", ViewModel викликає метод підключення в моделі.
4. Model формує команду запуску:
5. Системна утиліта rasdial.exe виконує підключення.
6. Програма аналізує результат виконання та інформує користувача про статус (успіх чи помилку).

UML-діаграмма послідовності додатку зображена на рис. 2.3

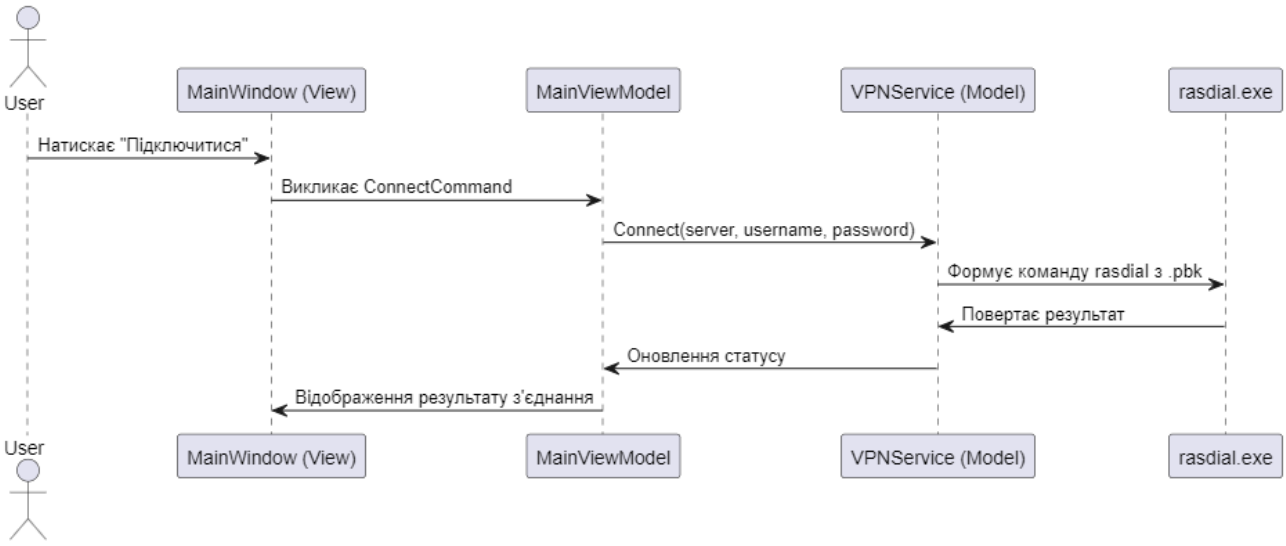


Рисунок 2.3 - UML-діаграмма послідовності додатку

При розробці VPN в кваліфікаційній роботі враховані такі особливості реалізації:

- використання rasdial.exe дозволяє інтегруватися з вбудованим VPN-клієнтом Windows, що спрощує налаштування на стороні користувача.
- .pbk файли містять усю необхідну інформацію про конфігурацію VPN-з'єднання, включно з IP-адресою сервера, протоколом, портами та типом автентифікації.
- програма не потребує сторонніх бібліотек чи драйверів VPN, надаючи нативне рішення.

Висновок до другого розділу

У результаті проведеного моделювання предметної області VPN-додатку було детально проаналізовано архітектуру програмного середовища, основні функціональні компоненти та принципи їхньої взаємодії. На відміну від традиційних VPN-клієнтів, які часто базуються на сторонніх бібліотеках або окремих сервісах типу OpenVPN, реалізований додаток використовує штатні

інструменти операційної системи Windows для ініціалізації підключення - зокрема, утиліту rasdial.exe та .pbk-файли (Phone Book), що містять усі необхідні налаштування для встановлення захищеного з’єднання.

Завдяки застосуванню архітектурного патерну MVVM (Model-View-ViewModel) досягнуто високої модульності та гнучкості програми. Поділ на рівні View, ViewModel та Model дозволив ефективно організувати кодову базу, спростити тестування окремих компонентів та забезпечити легку масштабованість у разі подальшого розширення функціоналу.

На основі побудованих UML-діаграм - зокрема діаграми класів та діаграми послідовності - вдалося візуально відобразити структуру додатку та процес підключення до VPN-сервера. Це дає можливість не лише краще зрозуміти внутрішню логіку роботи системи, але й служить зручною документацією для інших розробників, які можуть підтримувати чи вдосконалювати продукт у майбутньому.

Запропоноване рішення має низку переваг:

- використання вбудованих засобів windows без необхідності інсталяції додаткового ПЗ;
- простий, інтуїтивно зрозумілий інтерфейс користувача;
- повна автоматизація процесу підключення через командний рядок;
- гнучкість щодо зміни vpn-серверів завдяки підтримці конфігурацій у вигляді .pbk-файлів.

РОЗДІЛ 3 РОЗРОБКА І ТЕСТУВАННЯ ПРОГРАМНОГО СЕРЕДОВИЩА VPN

3.1 Інструменти розробки додатку

Розробка програмного середовища VPN здійснювалась із використанням платформи .NET та технології WPF (Windows Presentation Foundation). Основною мовою програмування виступає C#, а архітектурний шаблон реалізовано за принципом MVVM (Model-View-ViewModel), що забезпечує чітке розділення логіки, даних та інтерфейсу користувача.

Загальна структура додатку побудована навколо MVVM-шаблону:

- Model - містить структури для зберігання даних про VPN-сервери;
- ViewModel - реалізує логіку роботи додатку, управління командами, перемиканням вкладок, підключенням до VPN;
- View - графічне представлення інтерфейсу, описане в XAML.

Центральним класом є MainViewModel, що координує взаємодію між вікнами (вкладками) та ViewModel-компонентами. Повну структуру класів, зв’язків та дерева проєкту наведено у Додатку А.

Головна логіка програми зосереджена у ViewModel-класах, зокрема:

- MainViewModel - відповідає за управління головним вікном, перемикання вкладок інтерфейсу, реалізацію команд згортання/розгортання вікна.
- ProtectionViewModel - реалізує процес підключення до VPN. А саме:
    1. Зчитування збереженого пароля з локального файлу;
    2. Формування команди для cmd.exe з використанням утиліти rasdial;
    3. Обробка результату виконання команди;
    4. Відключення від сервера.

- SettingsViewModel - надає функціонал для збереження VPN-пароля у файл.
- RelayCommand - реалізація інтерфейсу ICommand для обробки взаємодії з елементами керування.

Ці компоненти детально описано у Додатку Б.

Дані про VPN-сервери представлені класом ServerModel, який містить такі властивості як: назва країни, IP-адреса, логін, пароль, а також автоматично формує посилання на зображення прапору. Ця модель забезпечує взаємодію з користувацьким інтерфейсом і використовується у списках серверів. Опис моделі представлено в Додатку В.

Інтерфейс додатку реалізовано у вигляді окремих XAML-файлів для кожного представлення:

- MainWindow.xaml - головне вікно додатку, яке містить загальну структуру інтерфейсу, включаючи меню, панель навігації та динамічний вміст
- ProtectionView.xaml - відображення списку серверів та кнопки підключення
- SettingsView.xaml - поле для введення пароля та кнопка збереження.

Кожне з вікон має прив’язку до відповідного ViewModel-класу. Повний код View представлено у Додатку Г.

У додатку реалізовано окремі XAML-словники для візуального оформлення елементів інтерфейсу:

- стиль кнопки підключення (ConnectButtonStyle);
- стиль пунктів меню (MenuButton);
- стиль списку серверів (ServerListTheme);
- стиль кнопок керування вікном (TitleButton).

Ці словники надають гнучке налаштування зовнішнього вигляду та дозволяють легко змінювати тематику оформлення. Стилі описано у Додатку Д.

Логіка підключення до VPN:

Процес підключення до VPN реалізовано без використання сторонніх бібліотек - виключно через системну команду rasdial. Це дозволяє запускати VPN-сесію за допомогою командного рядка Windows у фоновому режимі. Алгоритм роботи та приклад формування команд розглянуто у Додатку Е.

3.2 Опис інтерфейсу розробленого середовища

Середовище надає користувачеві зручний інтерфейс для вибору VPN-сервера та підключення до нього. Всі налаштування підключення - включаючи облікові дані та вибір країни - реалізуються через використання .pbk-файлів, що дозволяє уникнути зовнішніх бібліотек на кшталт OpenVPN. На рисунку 3.1 представлено головне вікно реалізованого середовища.

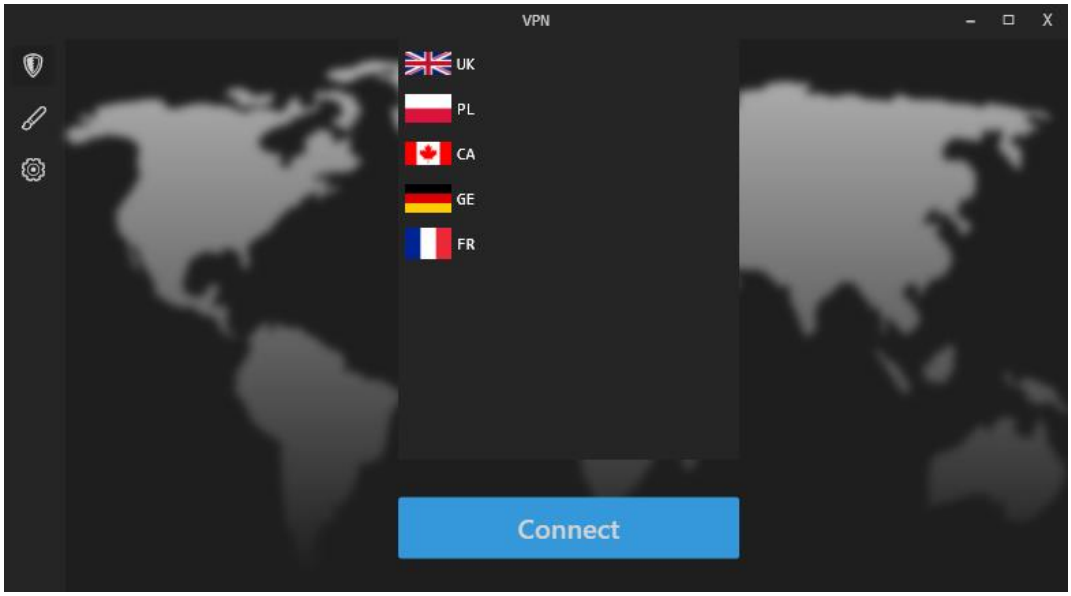


Рисунок 3.1 Головне вікно реалізованого середовища

Для підключення до середовища потребується пароль, який знаходиться на сайті сервісу vpnbook. Для його введеня було створено вікно з можливістю збереження цього пароля для подальшого підключення до сервісу через .pbk-файли. Зовнішній вигляд вікна зображено на рисунку 3.2.

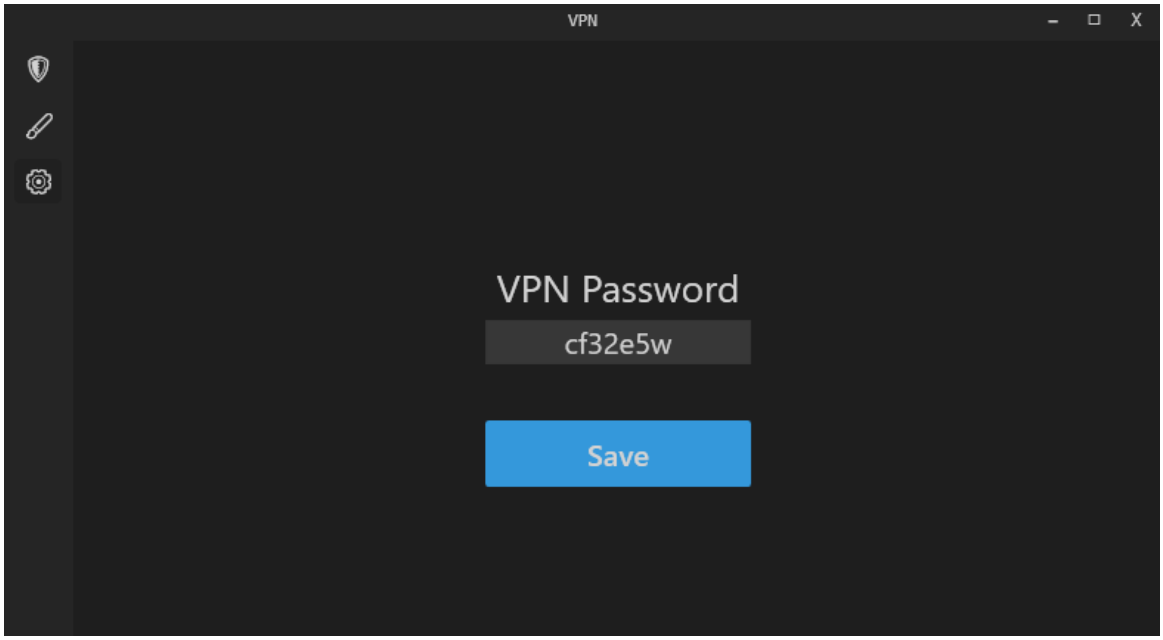


Рисунок 3.2 Вікно введення паролю для підключення

Інтерфейс не перегружений, доступний та зрозумілий для користувача використовуючи підходи мінімалізму проєктування інтерфейсу. А саме:

- не додавати елементи які користувачеві не потрібні, або які відволікають від використання програми;
- додавати мінімум кнопок для взаємодії, щоб не плутати користувача;
- візуально показувати для чого саме використовується програма та виділяти основні елементи.

3.3 Тестування додатку

Метою тестування є перевірка коректності роботи механізму підключення до VPN-сервера, реалізованого у класі ProtectionViewModel. Основна увага

приділяється сценаріям підключення до серверів, зчитування пароля, відображення статусу з’єднання та обробки можливих помилок.

Тестування проводилось у наступному середовищі:

- операційна система: Windows 10 Pro (64-bit)
- роздільна здатність дисплея: 2560×1920
- облікові дані VPN: Сервіс VPNBook
- метод підключення: системна команда rasdial через cmd.exe
- тип тестування: ручне (мануальне), функціональне

Для тестування додатку було розроблено 5 тест-кейсів. Тест TC-01(підключення до сервера з вибраною країною) – перевіряє базову функціональність - можливість підключення до конкретного сервера, обраного користувачем у списку. Характеристики тест-кейсу 01 представлені в таблиці 3.1.

Таблиця 3.1 - Характеристики тест-кейсу 01

№	Назва характеристики	Значення
1	2	3
1	ID	TC-01
2	Назва тесту	Підключення до сервера з вибраною країною
3	Мета	Перевірити коректність підключення при виборі конкретного сервера (наприклад, UK)
4	Передумови	VPN-сервер "UK" доступний, правильний пароль записано в password.txt
5	Кроки	1. Обрати сервер "UK" 2. Натиснути кнопку "Connect"
6	Очікуваний результат	Встановлюється з’єднання з сервером UK, статус: Connected!

Продовження таблиці 3.1

1	2	3
7	Фактичний результат	З’єднання встановлено, статус оновлено
8	Статус	Успішно

Тест ТС-02(підключення без вибору сервера) перевіряє поведінку програми у випадку, коли користувач не вибрав жоден сервер. Характеристики тесту 02 представлені в таблиці 3.2.

Таблиця 3.2 – Характеристики тест-кейсу 02

№	Назва характеристики	Значення
1	ID	ТС-02
2	Назва тесту	Підключення без вибору сервера
3	Мета	Перевірити, чи використовується сервер за замовчуванням, якщо нічого не обрано
4	Передумови	Жоден сервер не вибрано, пароль валідний
5	Кроки	1. Не обирати сервер 2. Натиснути "Connect"
6	Очікуваний результат	Підключення до серверу Poland, статус: Connected!
7	Фактичний результат	Підключено до Poland, статус оновлено
8	Статус	Успішно

Тест ТС-03(підключення з неправильним паролем). Симулюється ситуація, коли у файлі password.txt збережено невірний пароль. Характеристики тесту 03 представлені в таблиці 3.3.

Таблиця 3.3 - Характеристики тест-кейсу 03

№	Назва характеристики	Значення
1	ID	ТС-03
2	Назва тесту	Підключення з неправильним паролем
3	Мета	Перевірити обробку помилки при неправильних облікових даних
4	Передумови	У password.txt введено неправильний пароль
5	Кроки	1. Обрати будь-який сервер 2. Натиснути "Connect"
6	Очікуваний результат	Помилка з кодом 691, статус не Connected!
7	Фактичний результат	Підключення не виконано, помилка оброблена
8	Статус	Успішно

Тест ТС-04 (відключення після підключення). Цей сценарій перевіряє логіку повторного натискання кнопки "Connect", коли VPN-з’єднання вже активне. Характеристики тесту 04 представлені в таблиці 3.4.

Тест ТС-05 (підключення до невідомої країни) створено для перевірки стійкості логіки до помилкових або нових значень. Характеристики тесту 05 представлені в таблиці 3.5.

Таблиця 3.4 - Характеристики тест-кейсу 04

№	Назва характеристики	Значення
1	ID	TC-04
2	Назва тесту	Відключення після підключення
3	Мета	Перевірити функцію завершення активного з’єднання
4	Передумови	Встановлене VPN-з’єднання
5	Кроки	1. Натиснути "Connect" вдруге (режим відключення)
6	Очікуваний результат	З’єднання завершено, статус: Disconnected!
7	Фактичний результат	З’єднання припинено, статус оновлено
8	Статус	Успішно

Таблиця 3.5 - Характеристики тест-кейсу 05

№	Назва характеристики	Значення
1	2	3
1	ID	TC-05
2	Назва тесту	Підключення до невідомої країни
3	Мета	Перевірити обробку значення, якого немає у списку країн
4	Передумови	Сервер із країною X штучно доданий

Продовження таблиці 3.5

5	Кроки	1. Обрати сервер із кодом країни X 2. Натиснути "Connect"
6	Очікуваний результат	З’єднання з сервером Poland (сервер за замовчуванням)
7	Фактичний результат	Підключення здійснено до Poland
8	Статус	Успішно

В результаті аналізу тестування зроблені наступні висновки:

- усі тест-кейси пройдено успішно;
- програма адекватно реагує на різні сценарії - як при правильних діях користувача, так і при помилках, зокрема:

1. коректно зчитує пароль із локального файлу;
2. адаптує команду rasdial залежно від вибраного сервера;
3. обробляє ситуації з помилковими даними або відсутністю вибору.

- Механізм підключення/відключення стабільний, не спричиняє зависань чи системних збоїв, а статус з’єднання відображається у реальному часі.

Результати тестування підтвердили, що розроблений механізм підключення до VPN відповідає функціональним вимогам. Програма стабільно працює в середовищі Windows 10 та правильно реагує на різні вхідні ситуації. Надійність взаємодії з cmd.exe і системною утилітою rasdial дозволяє впевнено використовувати дане середовище у реальних умовах.

Висновки до третього розділу

У третьому розділі було реалізовано програмне середовище для підключення до VPN-сервісу на основі фреймворку WPF з використанням мови програмування C#**.** В основу архітектури додатку покладено шаблон MVVM (Model-View-ViewModel), який забезпечив ефективне розділення логіки, представлення даних та взаємодії з інтерфейсом користувача.

Підключення до VPN здійснюється шляхом виклику системної утиліти rasdial у фоновому режимі через командний рядок Windows. Такий підхід забезпечує простоту реалізації, автоматизацію процесу та повну інтеграцію з інструментами ОС.

У ході реалізації були створені ключові компоненти, кожен з яких детально описано у відповідних додатках:

Архітектура проєкту та основні класи наведені у Додатку А**;**

Логіка ViewModel-компонентів, включаючи керування вікнами, перемикання вкладок та підключення до серверів, подана у Додатку Б;

Модель VPN-сервера, що зберігає всі необхідні дані та генерує відповідні прапори країн, розглянута у Додатку В**;**

XAML-інтерфейси всіх вікон додатку представлено у Додатку Г**;**

Тематичні стилі оформлення для елементів інтерфейсу систематизовано у Додатку Д**;**

Повна логіка підключення через rasdial описана у Додатку Е**.**

Сукупність реалізованих компонентів дозволила створити надійне, масштабоване та інтуїтивно зрозуміле середовище, яке може бути легко адаптоване для інших VPN-сервісів або розширене новими функціями. Розробка підтвердила ефективність використання WPF, .NET та системних засобів Windows для побудови мережевого клієнта без залучення стороннього програмного забезпечення.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було повністю реалізовано проєкт з розробки програмного середовища для підключення до VPN-сервісу на базі фреймворку WPF. Метою дослідження стало моделювання предметної області, проєктування архітектури, реалізація функціоналу підключення до VPN, а також створення зручного користувацького інтерфейсу.

На етапі дослідження предметної області було проаналізовано принципи роботи VPN-з'єднання, методи його реалізації в середовищі Windows, а також особливості використання сервісу VPNBook, який надає безкоштовний доступ до VPN через готові файли конфігурацій. На основі цього аналізу було змодельовано структуру додатку та обрано оптимальний підхід для реалізації - з використанням вбудованої в Windows команди rasdial, що дозволяє підключатися до VPN через .pbk-файли без потреби у сторонньому програмному забезпеченні.

У результаті розробки було створено повноцінний програмний продукт, який включає:

- підтримку архітектури MVVM, що забезпечує гнучкість і масштабованість додатку;
- модуль ProtectionViewModel, який відповідає за логіку з'єднання з сервером;
- модуль SettingsViewModel, що дає змогу зберігати облікові дані для підключення;
- систему команд RelayCommand, яка зв'язує події інтерфейсу з логікою додатку;
- гнучкий інтерфейс користувача з підтримкою перемикання тем, віконного керування та ін.

Програмне середовище дозволяє користувачеві:

- вибрати одну з доступних країн-серверів (Польща, Франція, Німеччина, Канада, Велика Британія);
- ввести пароль до облікового запису VPNBook та зберегти його;
- одним натисканням підключитися або відключитися від вибраного сервера.

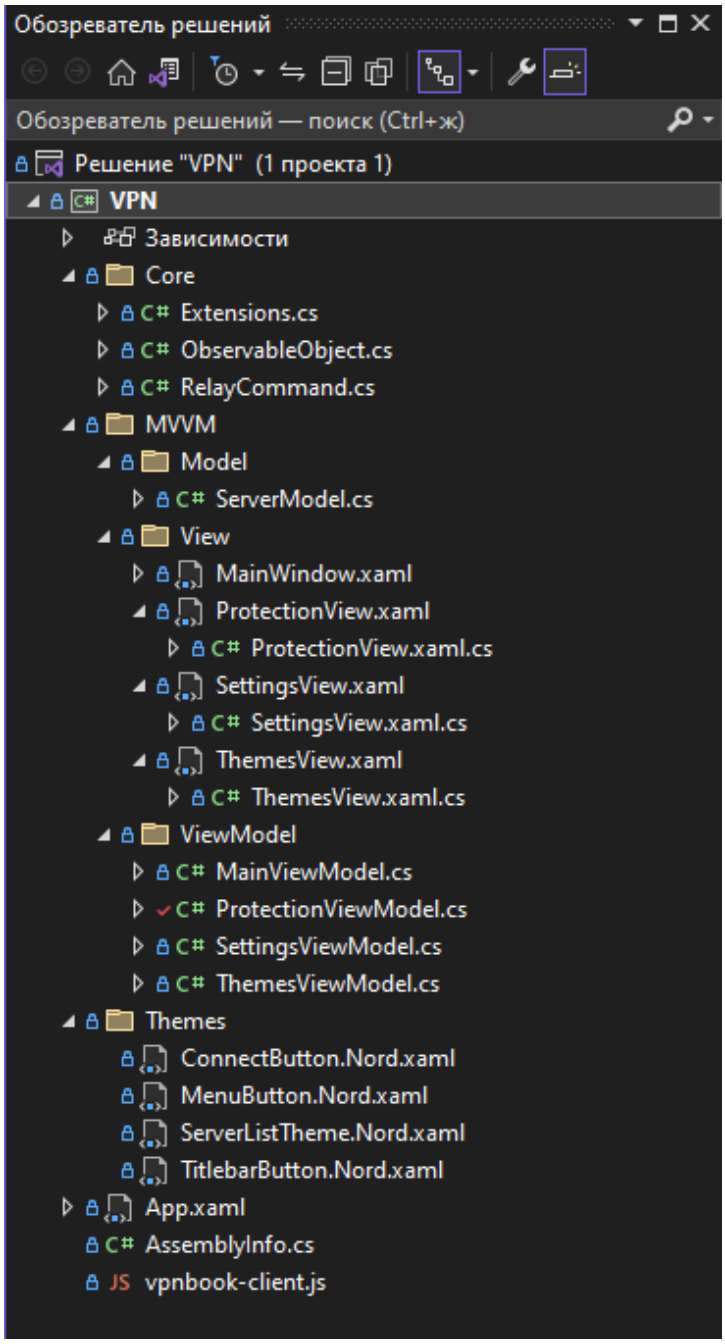
У кваліфікаційній роботі також детально описано реалізацію ключових програмних компонентів у вигляді лістингів коду з поясненнями, а також представлено UML-діаграми, що відображають структуру додатку та логіку взаємодії між його модулями. Це дозволило продемонструвати розуміння принципів об’єктно-орієнтованого проєктування та практичне вміння реалізовувати програмні системи з урахуванням архітектурних шаблонів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. VPN та мережі: методичні вказівки / уклад. В. Ю. Слюсар. Київ : КПІ ім. Ігоря Сікорського, 2022. 30 с. URL: https://ela.kpi.ua/server/api/core/bitstreams/35d4a2d2-53ed-453f-9bcd-fa883a982f53/content (дата звернення: 10.06.2025).
2. Пирогов О. В. Побудова захищених віртуальних приватних мереж VPN : бакалавр. робота. Харків : НУ «ХАІ», 2021. 55 с. URL: https://openarchive.nure.ua/server/api/core/bitstreams/40fd3817-2725-4582-a9aa-4e9e348ef8aa/content (дата звернення: 10.06.2025).
3. Концепція захищених віртуальних приватних мереж. URL: https://stud.com.ua/97437/informatika/kontseptsiya_zahischenih_virtualnih_privatnih_merezh (дата звернення: 10.06.2025).
4. What is a VPN? – Microsoft Azure. URL: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-vpn (дата звернення: 10.06.2025).
5. Что такое VPN? – NordVPN. URL: https://nordvpn.com/ru/what-is-a-vpn/ (дата звернення: 10.06.2025).
6. What is a VPN? – Kaspersky. URL: https://www.kaspersky.com/resource-center/definitions/what-is-a-vpn (дата звернення: 10.06.2025).
7. What is a VPN and why it's important? – Reddit. URL: https://www.reddit.com/r/VPN/comments/1f4whk1/what_is_a_vpn_and_why_its_important/ (дата звернення: 10.06.2025).
8. What is a VPN and how does it work? – YouTube. URL: https://www.youtube.com/watch?v=wfWxdh-_k_4&list=PLWKjhJtqVAbkq5Oh8ERRJ1aPZK2NKBSRx (дата звернення: 10.06.2025).

ДОДАТКИ

Додаток А – Архітектура додатку (структура, шаблон MVVM, організація проєкту)



Додаток Б – ViewModel (логіка, команди, перемикання вкладок)

Extentions.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;

namespace VPN.Core
{
    internal class Extensions
    {
        public static readonly DependencyProperty Icon =
            DependencyProperty.RegisterAttached("Icon",
typeof(string),typeof(Extensions),                              new
PropertyMetadata(default(string)));

        public  static  void  SetIcon  (UIElement  element,
string value)
        {
            element.SetValue(Icon, value);
        }

        public  static  string  GetIcon(UIElement  element,
string value)
        {
           return (string)element.GetValue(Icon);
        }
    }
}
```

ObservableObject.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;

namespace VPN.Core
{
    public class ObservableObject : INotifyPropertyChanged
```

Продовження додатку Б

```
      {
          public event PropertyChangedEventHandler?
PropertyChanged;
          public virtual void
OnPropertyChanged([CallerMemberName] string propertyName = null)
          {
              PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(propertyName));
          }
      }
  }


  RelayCommand.cs
  using System;
  using System.Collections.Generic;
  using System.Linq;
  using System.Text;
  using System.Threading.Tasks;
  using System.Windows.Input;

  namespace VPN.Core
  {
      public class RelayCommand : ICommand
      {
          private Action<object> execute;
          private Func<object, bool> canExecute;

          public event EventHandler CanExecuteChanged
          {
              add { CommandManager.RequerySuggested += value;
}
              remove { CommandManager.RequerySuggested -=
value; }
          }

          public RelayCommand(Action<object> execute,
Func<object, bool> canExecute = null)
          {
              this.execute = execute;
              this.canExecute = canExecute;
          }

          public bool CanExecute(object parameter)
          {
              return this.canExecute == null ||
this.canExecute(parameter);
```

```
        }

            public void Execute(object parameter)
            {
                this.execute(parameter);
            }
        }

    }
```

MainViewModel.cs

```
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;
    using System.Windows;
    using VPN.Core;

    namespace VPN.MVVM.ViewModel
    {
        class MainViewModel : ObservableObject
        {

            public RelayCommand MoveWindowCommand {get;set;}
            public  RelayCommand  ShutDownWindowCommand  {  get;
set; }
            public  RelayCommand  MaximizeWindowCommand  {  get;
set; }
            public  RelayCommand  MinimizeWindowCommand  {  get;
set; }

            public RelayCommand ShowProtectionView { get; set; }

            public RelayCommand ShowSettingsView { get; set; }

            public RelayCommand ShowThemesView { get; set; }

            private object _currentView;

            public object CurrentView
            {
                get { return _currentView; }
                set {
                    _currentView = value;
```

Продовження додатку Б

```
        OnPropertyChanged();
          }
                }

                public ProtectionViewModel ProtectionVM { get; set;
}

                public SettingsViewModel SettingsVM { get; set; }

                public ThemesViewModel ThemesVM { get; set; }

                public MainViewModel()
                {

                    ProtectionVM = new ProtectionViewModel();
                    SettingsVM = new SettingsViewModel();
                    ThemesVM = new ThemesViewModel();
                    CurrentView = ProtectionVM;

                    Application.Current.MainWindow.MaxHeight       =
SystemParameters.MaximizedPrimaryScreenHeight;
                    MoveWindowCommand = new RelayCommand(o => {
Application.Current.MainWindow.DragMove(); });
                    ShutDownWindowCommand = new RelayCommand(o => {
Application.Current.Shutdown(); });
                    MaximizeWindowCommand = new RelayCommand(o =>
                    {
                        if
(Application.Current.MainWindow.WindowState                    ==
WindowState.Maximized)


Application.Current.MainWindow.WindowState = WindowState.Normal;
                        else

Application.Current.MainWindow.WindowState                     =
WindowState.Maximized;

                    });
                    MinimizeWindowCommand = new RelayCommand(o => {
Application.Current.MainWindow.WindowState                     =
WindowState.Minimized; });

                    ShowProtectionView = new RelayCommand(o => {
CurrentView = ProtectionVM; });
                    ShowSettingsView = new RelayCommand(o => {
CurrentView = SettingsVM; });
```

```
    ShowThemesView = new RelayCommand(o => { CurrentView =
ThemesVM;});
            }
        }
    }
```

SettingsViewModel.cs

```
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;
    using VPN.Core;
    using System.Windows;
    using System.IO;
    using System.Windows.Controls;

    namespace VPN.MVVM.ViewModel
    {
        public class SettingsViewModel : ObservableObject
        {
            public RelayCommand SaveCommand { get; set; }

            private string _password;
            public string Password
            {
                get => _password;
                set
                {
                    _password = value;
                    OnPropertyChanged(nameof(Password));
                }
            }
            public SettingsViewModel()
            {
                string                basePath                =
AppDomain.CurrentDomain.BaseDirectory;
                string   folderPath   =   Path.Combine(basePath,
"..\\VPN");
                string   filePath   =   Path.Combine(folderPath,
"password.txt");

                SaveCommand = new RelayCommand(o =>
                {
```

Продовження додатку Б

```
      try
                        {
                            if (!Directory.Exists(folderPath))
                            {
Directory.CreateDirectory(folderPath);
                            }
                            using (StreamWriter writetext = new
StreamWriter(filePath))
                            {

                                writetext.WriteLine(Password);
                            }

                        }
                        catch (Exception ex)
                        {
                            MessageBox.Show("Помилка при збереженні:
" + ex.Message);
                        }
                    });

                }


            }


        }
```

Додаток В – Model (модель VPN-серверів)

ServerModel.cs

```
namespace VPN.MVVM.ViewModel
{
    public class ServerModel
    {
        public int ID { get; set; }
        public string Username { get; set; }
        public string Password { get; set; }
        public string Server { get; set; }
        public string Country { get; set; }

        public string FlagSource => GetFlagSource(Country);

        private string GetFlagSource(string countryCode)
        {
            return countryCode switch
            {
                "UK" => "https://flagcdn.com/w40/gb.png",
                "PL" => "https://flagcdn.com/w40/pl.png",
                "CA" => "https://flagcdn.com/w40/ca.png",
                "GE" => "https://flagcdn.com/w40/de.png",
                "FR" => "https://flagcdn.com/w40/fr.png"
            };
        }
    }
}
```

Додаток Г – View (XAML-інтерфейс: MainWindow, ProtectionView, SettingsView)

MainWindow.xaml

```
    <Window x:Class="VPN.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:VPN"
             xmlns:extensions ="clr-namespace:VPN.Core"
             xmlns:viewmodel="clr-namespace:VPN.MVVM.ViewModel"
             mc:Ignorable="d"
             Title="MainWindow"
             Height="450" Width="800"
             WindowStyle="None"
             Background="Transparent"
             AllowsTransparency="True"
             ResizeMode="CanResize">

         <Window.DataContext>
             <viewmodel:MainViewModel/>
         </Window.DataContext>

         <DockPanel Background="#1E1E1E"
                    Margin="7">
             <Border Height="28"
                     Background="#252525"
                     DockPanel.Dock="Top">
                 <Border.InputBindings>
                     <MouseBinding MouseAction="LeftClick"
                                   Command="{Binding
MoveWindowCommand}"/>
                 </Border.InputBindings>
                 <Grid>
                     <Grid.ColumnDefinitions>
                         <ColumnDefinition/>

                         <ColumnDefinition Width="100"/>
```

Продовження додатку Г

```
        </Grid.ColumnDefinitions>
                        <TextBlock Text="VPN"
                                   Foreground="LightGray"
                                   FontFamily="Consolas"
                                   VerticalAlignment="Center"
                                   HorizontalAlignment="Center"
                                   Margin="100,0,0,0"/>
                        <StackPanel Grid.Column="1"
                                    Orientation="Horizontal"
                                    HorizontalAlignment="Right"
                                    Margin="0,0,4,0">
                            <Button Content="-"
                                    FontSize="22"
                                    Padding="0,-7,1,0"
                                    Command="{Binding
MinimizeWindowCommand}"/>
                            <Button Content="□"
                                    FontSize="16"
                                    Padding="0,-4,1,4"
                                    Command="{Binding
MaximizeWindowCommand}"/>
                            <Button Content="X"
                                    Padding="0,0,0,2"
                                    Command="{Binding
ShutDownWindowCommand}"/>

                        </StackPanel>
                    </Grid>

                </Border>
                <Border Width="48"
                        HorizontalAlignment="Left"
                        Background="#252525">
                    <Border.Style>
                        <Style>
                            <Style.Triggers>
                                <EventTrigger
RoutedEvent="Border.MouseEnter">
                                    <EventTrigger.Actions>
                                        <BeginStoryboard>
                                            <Storyboard>
                                                <DoubleAnimation
Storyboard.TargetProperty="(Border.Width)"

To="120"

Duration="0:0:.1"/>
```

Продовження додатку Г

```
     </Storyboard>
                                    </BeginStoryboard>
                                </EventTrigger.Actions>
                            </EventTrigger>

                            <EventTrigger
RoutedEvent="Border.MouseLeave">
                                <EventTrigger.Actions>
                                    <BeginStoryboard>
                                        <Storyboard>
                                            <DoubleAnimation
Storyboard.TargetProperty="(Border.Width)"

To="48"

Duration="0:0:.1"/>
                                        </Storyboard>
                                    </BeginStoryboard>
                                </EventTrigger.Actions>
                            </EventTrigger>
                        </Style.Triggers>
                    </Style>
                </Border.Style>
                <StackPanel>
                    <RadioButton Content="VPN"
                                 extensions:Extensions.Icon
="&#128737;"
                                 FontSize="16"
                                 Padding="7.6,3.5,8,0"
                                 FontWeight="Normal"
                                 Foreground="LightGray"
                                 IsChecked="True"
                                 Command="{Binding
ShowProtectionView}"/>
                    <RadioButton Content="Themes"
                                 FontSize="16"
                                 extensions:Extensions.Icon="□"
                                 Padding="6,3.5,4.7,0"
                                 FontWeight="Normal"
                                 Foreground="LightGray"
                                 IsChecked="False"
                                 Command="{Binding
ShowThemesView}"/>
                    <RadioButton Content="Settings"
                                 FontSize="16"
                                 extensions:Extensions.Icon="⚙"
                                 Padding="6,3.5,4.7,0"
```

```
      FontWeight="Normal"
                                    Foreground="LightGray"
                                    IsChecked="False"
                                    Command="{Binding
ShowSettingsView}"/>
                </StackPanel>
            </Border>
            <ContentPresenter
                Content="{Binding CurrentView}"/>
        </DockPanel>
    </Window>
```

ProtectionView.xaml

```
    <UserControl x:Class="VPN.MVVM.View.ProtectionView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
                 xmlns:local="clr-namespace:VPN.MVVM.ViewModel"
d:DataContext="{d:DesignInstance
Type=local:ProtectionViewModel}"
                 mc:Ignorable="d"
                 d:DesignHeight="450" d:DesignWidth="800"
                 Background="#1E1E1E">
        <Grid>
            <Grid.RowDefinitions>
                <RowDefinition/>
                <RowDefinition Height="100"/>
            </Grid.RowDefinitions>
            <Image Source="https://i.imgur.com/bx3bBLj.png"
                   RenderOptions.BitmapScalingMode="Fant"
                   Grid.RowSpan="2">
                <Image.OpacityMask>
                    <LinearGradientBrush EndPoint="0.5,1"
                                         StartPoint="0.5,0">
                        <GradientStop Color="Black"/>
                        <GradientStop Color="Transparent"
                                      Offset="1"/>
                    </LinearGradientBrush>
                </Image.OpacityMask>
```

Продовження додатку Г

```
        <Image.Effect>
                    <BlurEffect Radius="10"/>
                </Image.Effect>
            </Image>
            <ListView ItemsSource="{Binding Servers}"
                      Width="250"
                      Background="#252525"
                      BorderThickness="0"
                      ScrollViewer.CanContentScroll="False"
                      SelectedItem="{Binding    SelectedServer,
Mode=TwoWay}">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <TextBlock   Text="{Binding   Country}"
Foreground="White"/>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>

            <TextBlock Text="{Binding ConnectionStatus}"
                       Foreground="White"
                       Grid.Row="1"
                       HorizontalAlignment="Center"
                       VerticalAlignment="Top"/>

            <Button Width="250"
                    Height="45"
                    Content="Connect"
                    Grid.Row="1"
                    Style="{StaticResource ConnectButtonStyle}"
                    Command="{Binding ConnectCommand}"/>

        </Grid>
    </UserControl>
```

SettingsView.xaml

```
    <UserControl x:Class="VPN.MVVM.View.SettingsView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
```

Продовження додатку Г

```
     xmlns:d="http://schemas.microsoft.com/expression/blend/2008
"
                  xmlns:local="clr-namespace:VPN.MVVM.ViewModel"
                  xmlns:viewmodel="clr-
namespace:VPN.MVVM.ViewModel"   d:DataContext="{d:DesignInstance
Type=local:SettingsViewModel}"
                  mc:Ignorable="d"
                  d:DesignHeight="450" d:DesignWidth="800"
                  Background="#1E1E1E">

         <Grid            Height="250px"            Width="250px"
HorizontalAlignment="Center" VerticalAlignment="Center">
             <TextBlock Text="VPN Password"
                        VerticalAlignment="Top"
                        HorizontalAlignment="Center"
                        Foreground="LightGray"
                        FontSize="26" Margin="0,70,0,0"/>
             <TextBox Name="Password"
              Text="{Binding                                Password,
UpdateSourceTrigger=PropertyChanged}"
              VerticalAlignment="Center"
              HorizontalAlignment="Center"
              Height="30"
              Width="180"
              Foreground="LightGray"
              Background="#FF373737"
              BorderThickness="1"
              BorderBrush="#FF353535"
              TextAlignment="Center"
              FontSize="20"/>
             <Button
                     Height="45"
                     Content="Save"
                     Style="{StaticResource ConnectButtonStyle}"
                     Command="{Binding SaveCommand}"
                     Margin="35,178,35,27" Cursor="Hand"/>
         </Grid>
     </UserControl>
```

Додаток Д – Теми оформлення (XAML-словники стилів)

ConnectButton.Nord.xaml

```
     <ResourceDictionary
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
         <Style           TargetType="{x:Type           Button}"
x:Key="ConnectButtonStyle">
             <Setter Property="Background" Value="#3498db"/>
             <Setter Property="Foreground" Value="LightGray"/>
             <Setter Property="Cursor" Value="Hand"/>
             <Setter Property="FontWeight" Value="Medium"/>
             <Setter Property="FontSize" Value="20"/>

             <Setter Property="Template">
                 <Setter.Value>
                     <ControlTemplate        TargetType="{x:Type
Button}">
                         <Border    Background="{TemplateBinding
Background}" CornerRadius="2">
                             <ContentPresenter
VerticalAlignment="Center"

HorizontalAlignment="Center"

Content="{TemplateBinding Content}"/>
                         </Border>
                     </ControlTemplate>
                 </Setter.Value>
             </Setter>

             <Style.Triggers>
                 <Trigger Property="IsEnabled" Value="False">
                     <Setter               Property="Background"
Value="Gray"/>
                 </Trigger>
             </Style.Triggers>

         </Style>
     </ResourceDictionary>
```

MenuButton.Nord.xaml

Продовження додатку Д

```
    <ResourceDictionary
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                    xmlns:extensions                ="clr-
namespace:VPN.Core">
        <Style TargetType="{x:Type RadioButton}">
            <Setter Property="Background" Value="Transparent"/>
            <Setter Property="Margin" Value="4,0,4,0"/>
            <Setter Property="Height" Value="38"/>
            <Setter Property="Padding" Value="0"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate        TargetType="{x:Type
RadioButton}">
                        <Border    Background="{TemplateBinding
Background}"
                                   CornerRadius="4"
                                   Margin="4">
                            <StackPanel
Orientation="Horizontal">
                                <TextBlock
Margin="{TemplateBinding Padding}"
                                           Text="{Binding
Path=(extensions:Extensions.Icon),
RelativeSource={RelativeSource Mode=TemplatedParent}}"/>
                                <ContentPresenter
VerticalAlignment="Center"

Margin="0,0,0,2"/>
                            </StackPanel>
                        </Border>

                        <ControlTemplate.Triggers>
                            <Trigger     Property="IsMouseOver"
Value="True">
                                <Setter   Property="Background"
Value="#202020"/>
                            </Trigger>

                            <Trigger       Property="IsChecked"
Value="True">
                                <Setter   Property="Background"
Value="#202020"/>
                            </Trigger>
```

```
      </ControlTemplate.Triggers>

                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>
    </ResourceDictionary>
```

ServerListTheme.Nord.xaml

```
    <ResourceDictionary
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
        <Style TargetType="{x:Type ListView}">
            <Setter Property="Background"
                    Value="#252525"/>
            <Setter Property="Foreground"
                    Value="White"/>
            <Setter Property="ScrollViewer.CanContentScroll"
                    Value="False"/>
            <Setter Property="BorderThickness"
                    Value="0"/>
            <Setter Property="FontFamily"
                    Value="Consolas"/>
        </Style>

        <Style TargetType="ListViewItem">
            <Setter Property="Cursor"
                    Value="Hand"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate>
                        <StackPanel Orientation="Horizontal"
                                    Margin="4"
                                    MinHeight="25"

Background="{TemplateBinding Background}">
                            <Image Width="34"

RenderOptions.BitmapScalingMode="Fant"
                                    Source="{Binding
FlagSource}"/>
                            <TextBlock Text="{Binding Country}"
                                       Margin="4,0,0,0"
```

Продовження додатку Д

```
     VerticalAlignment="Center"/>
                        </StackPanel>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>

            <Setter Property="Background"
                    Value="Transparent"/>
            <Style.Triggers>

                <Trigger Property="IsMouseOver" Value="True">
                    <Setter               Property="Background"
Value="#1E1E1E"/>
                </Trigger>

                <Trigger Property="IsSelected" Value="True">
                    <Setter               Property="Background"
Value="#1E1E1E"/>
                </Trigger>
            </Style.Triggers>

        </Style>

        <Style TargetType="{x:Type RepeatButton}">
            <Setter Property="Background" Value="Transparent"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate        TargetType="{x:Type
RepeatButton}">
                        <Rectangle       Fill="{TemplateBinding
Background}"
                                   Height="{TemplateBinding
Height}"
                                   Width="{TemplateBinding
Width}"/>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>

        <Style TargetType="{x:Type Thumb}">
           <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="{x:Type Thumb}">
                    <Rectangle Fill="#424242"
                                   Height="{TemplateBinding
Height}"
```

Продовження додатку Д

```
     Width="{TemplateBinding Width}"/>
                 </ControlTemplate>
             </Setter.Value>
            </Setter>
         </Style>

         <Style TargetType="{x:Type ScrollBar}">
             <Setter Property="Width" Value="12"/>
             <Setter Property="MinWidth" Value="12"/>

             <Setter Property="Template">
                 <Setter.Value>
                     <ControlTemplate        TargetType="{x:Type
ScrollBar}">
                         <Grid>
                             <Track x:Name="PART_Track"
                                    IsDirectionReversed="True">
                                 <Track.DecreaseRepeatButton>
                                     <RepeatButton
Command="{x:Static ScrollBar.PageUpCommand}"/>
                                 </Track.DecreaseRepeatButton>

                                 <Track.IncreaseRepeatButton>
                                     <RepeatButton
Command="{x:Static ScrollBar.PageDownCommand}"/>
                                 </Track.IncreaseRepeatButton>

                                 <Track.Thumb>
                                     <Thumb/>
                                 </Track.Thumb>

                             </Track>
                         </Grid>
                     </ControlTemplate>
                 </Setter.Value>
             </Setter>

         </Style>

     </ResourceDictionary>
```

TitleButton.Nord.xaml

```
     <ResourceDictionary
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
"
```

Продовження додатку Д

```
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
        <Style TargetType="{x:Type Button}">
            <Setter Property="Background" Value="Transparent"/>
            <Setter Property="Foreground" Value="LightGray"/>
            <Setter Property="BorderThickness" Value="0"/>
            <Setter Property="Width" Value="20"/>
            <Setter Property="Height" Value="20"/>
            <Setter Property="Margin" Value="4"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate        TargetType="{x:Type
Button}">
                        <Border    Background="{TemplateBinding
Background}"
                                CornerRadius="2">
                            <ContentPresenter
VerticalAlignment="Center"

HorizontalAlignment="Center"

Margin="{TemplateBinding Padding}"/>

                        </Border>

                        <ControlTemplate.Triggers>
                            <Trigger     Property="IsMouseOver"
Value="True">
                                <Setter   Property="Background"
Value="#444444">

                                </Setter>
                            </Trigger>
                        </ControlTemplate.Triggers>

                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>
    </ResourceDictionary>
```

App.xaml

```
    <Application x:Class="VPN.App"
```

Продовження додатку Д

```
     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/present
ation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                   xmlns:local="clr-namespace:VPN"
                   xmlns:view="clr-namespace:VPN.MVVM.View"
                   xmlns:vms="clr-namespace:VPN.MVVM.ViewModel"
                   StartupUri="MVVM/View/MainWindow.xaml">
         <Application.Resources>
             <ResourceDictionary>
                 <ResourceDictionary.MergedDictionaries>
                     <ResourceDictionary
Source="/Themes/TitlebarButton.Nord.xaml"/>
                     <ResourceDictionary
Source="/Themes/MenuButton.Nord.xaml"/>
                     <ResourceDictionary
Source="/Themes/ConnectButton.Nord.xaml"/>
                     <ResourceDictionary
Source="/Themes/ServerListTheme.Nord.xaml"/>
                 </ResourceDictionary.MergedDictionaries>

                 <DataTemplate                  DataType="{x:Type
vms:ProtectionViewModel}">
                     <view:ProtectionView/>
                 </DataTemplate>

                 <DataTemplate                  DataType="{x:Type
vms:SettingsViewModel}">
                     <view:SettingsView/>
                 </DataTemplate>

                 <DataTemplate                  DataType="{x:Type
vms:ThemesViewModel}">
                     <view:ThemesView/>
                 </DataTemplate>

             </ResourceDictionary>
         </Application.Resources>
     </Application>
```

Додаток Е – Логіка підключення до VPN через rasdial

ProtectionViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Shapes;
using VPN.Core;

namespace VPN.MVVM.ViewModel
{
    public class ProtectionViewModel:ObservableObject
    {
        public ObservableCollection<ServerModel> Servers { get; set; }
        private string _connectionStatus;
        bool FirstConnect = true;
        public string ConnectionStatus
        {
            get { return _connectionStatus; }
            set { _connectionStatus = value; OnPropertyChanged(); }

        }

        private string passwordForCon;

        private ServerModel _selectedServer;
        public ServerModel SelectedServer
        {
            get => _selectedServer;
            set
            {
                _selectedServer = value;
                OnPropertyChanged();
            }
        }

        public RelayCommand ConnectCommand { get; set; }
```

Продовження додатку Е

```
     public ProtectionViewModel()
             {
                 Servers                    =                    new
ObservableCollection<ServerModel>();

                 Servers.Add(new ServerModel {
                        Country = "UK"
                        });
                 Servers.Add(new ServerModel
                 {
                     Country = "PL"
                 });

                 Servers.Add(new ServerModel
                 {
                     Country = "CA"
                 });

                 Servers.Add(new ServerModel
                 {
                     Country = "GE"
                 });

                 Servers.Add(new ServerModel
                 {
                     Country = "FR"
                 });



                 ConnectCommand = new RelayCommand(o =>
                 {
                     Task.Run(() =>
                     {

                         if (FirstConnect)
                         {
                             ConnectionStatus = "Connecting..";
                             using (StreamReader readtext = new
StreamReader("..\\VPN\\password.txt"))
                             {
                                 passwordForCon                =
readtext.ReadLine();
                             }
                             var    processToConnect    =    new
Process();
```

Продовження додатку Е

```
      processToConnect.StartInfo.FileName = "cmd.exe";

processToConnect.StartInfo.WorkingDirectory                    =
Environment.CurrentDirectory;
                            if (SelectedServer == null)
                            {

processToConnect.StartInfo.Arguments  =  $@"/c  rasdial  MyServer
vpnbook {passwordForCon} /phonebook:./VPN/Poland.pbk";
                                ConnectionStatus  =  "Connecting
to default server (Poland)..";
                            }
                            else
                            {
                                switch (SelectedServer.Country)
                                {
                                    case "FR":

processToConnect.StartInfo.Arguments  =  $@"/c  rasdial  MyServer
vpnbook {passwordForCon} /phonebook:./VPN/France.pbk";
                                        ConnectionStatus      =
"Connecting to France..";
                                        break;
                                    case "UK":

processToConnect.StartInfo.Arguments  =  $@"/c  rasdial  MyServer
vpnbook {passwordForCon} /phonebook:./VPN/UnitedKingdom.pbk";
                                        ConnectionStatus      =
"Connecting to United Kingdom..";
                                        break;
                                    case "GE":

processToConnect.StartInfo.Arguments  =  $@"/c  rasdial  MyServer
vpnbook {passwordForCon} /phonebook:./VPN/Germany.pbk";
                                        ConnectionStatus      =
"Connecting to Germany..";
                                        break;
                                    case "PL":

processToConnect.StartInfo.Arguments  =  $@"/c  rasdial  MyServer
vpnbook {passwordForCon} /phonebook:./VPN/Poland.pbk";
                                        ConnectionStatus      =
"Connecting to Poland..";
                                        break;
                                    case "CA":
```

Продовження додатку Е

```
    processToConnect.StartInfo.Arguments = $@"/c rasdial
MyServer vpnbook {passwordForCon} /phonebook:./VPN/Canada.pbk";
                                        ConnectionStatus =
"Connecting to Canada..";
                                        break;
                                    default:

processToConnect.StartInfo.Arguments = $@"/c rasdial MyServer
vpnbook {passwordForCon} /phonebook:./VPN/Poland.pbk";
                                        ConnectionStatus =
"Connecting to default server (Poland)..";
                                        break;
                                }
                            }



processToConnect.StartInfo.UseShellExecute = false;

processToConnect.StartInfo.CreateNoWindow = true;

                            processToConnect.Start();
                            processToConnect.WaitForExit();

                            switch (processToConnect.ExitCode)
                            {
                                case 0:
                                    Debug.WriteLine("Success");
                                    ConnectionStatus =
"Connected!";
                                    break;
                                case 691:
                                    Debug.WriteLine("Wrong
credentials!");
                                    break;
                                default:

Debug.WriteLine($"Error:{processToConnect.ExitCode}");
                                    break;
                            }
                            FirstConnect = false;
                        }
                        else
                        {
                            ConnectionStatus =
"Disconnecting..";
                            var process = new Process();
```

Продовження додатку Е

```
     process.StartInfo.FileName = "cmd.exe";
                                process.StartInfo.WorkingDirectory
= Environment.CurrentDirectory;

process.StartInfo.ArgumentList.Add(@"/c rasdial /d");
                                process.StartInfo.UseShellExecute =
false;
                                process.StartInfo.CreateNoWindow  =
true;

                                process.Start();
                                process.WaitForExit();
                                ConnectionStatus = "Disconnected!";
                                FirstConnect = true;
                            }
                        });


                    })
                    {

                    };
                }

            }
        }
```