Київський національний торговельно-економічний університет

Кафедра комп'ютерних наук

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЕКТ

на тему:

«Розробка програмного модулю підвищення надійності комп'ютерної мережі Волинської митниці ДФС**»**

Студентки 2 курсу, 6 групи,
факультету обліку, аудиту та
інформаційних систем, денної форми
122 «Комп'ютерні науки» _____ Ждань Катерина Василівна

Науковий керівник
кандидат фізико-математичних наук
доцент кафедри комп'ютерних наук _____ Самойленко Ганна Тимофіївна

Гарант освітньої програми
Доктор фізико-математичних наук
професор кафедри комп'ютерних наук _____ Пурський Олег Іванович

Київ - 2019

ЗМІСТ

					КНТЕУ-122-2019		
Зм.	*Аркуш*	*№ документу*	*Підпис*	*Дата*			
Зав. кафедрою		*Пурський О.І.*		*03.12.2019*	*Розробка програмного модулю підвищення надійності комп'ютерної мережі Волинської митниці ДФС*	*Сторінка*	*Сторінок*
Керівник		*Самойленко Г.Т.*		*03.12.2019*		*5*	*81*
Гарант		*Пурський О.І.*		*03.12.2019*		*Кафедра комп'ютерних наук 2м-6*	
Розробив		*Ждань К.В.*		*03.12.2019*	*Зміст*		
Перевірив		*Самойленко Г.Т.*		*03.12.2019*			

АНОТАЦІЯ

магістерського дипломного проекту Ждань Катерини Василівни
на тему: «Розробка програмного модулю підвищення
надійності комп'ютерної мережі митниці»

Даний дипломний проект присвячений розробці програмного модулю підвищення надійності комп’ютерної мережі Волинської митниці ДФС. У зв’язку зі стрімкою автоматизацією роботи митниці важливим є питанням захисту оброблюваної інформації.

У роботі проаналізовано технічні та програмні засоби збереження цілісності інформаційної системи установи. Визначено чинники, що впливають на коректну та безперервну роботу локальної мережі. Запропоновано методи, що дозволяють контролювати та обмежувати дії користувачів ПК. Спроектовано та розроблено програмне забезпечення в якому реалізовано дані методи. Протестовано та налагоджено модуль підвищення надійності мережі. Апробовано на комп’ютерах установи.

Ключові слова: надійність, безпека, комп’ютерна мережа, людський фактор, розробка.

					КНТЕУ-122-2019		
Зм.	Аркуш	№ документу	Підпис	Дата			
Зав. кафедрою		Пурський О.І.		03.12.2019	Розробка програмного модулю підвищення надійності комп’ютерної мережі Волинської митниці ДФС	Сторінка	Сторінок
Керівник		Самойленко Г.Т.		03.12.2019		6	81
Гарант		Пурський О.І.		03.12.2019		Кафедра комп’ютерних наук 2м-6	
Розробив		Ждань К.В.		03.12.2019	Анотація		
Перевірив		Самойленко Г.Т.		03.12.2019			

ABSTRACT

A master's diploma project of Zhdan Kateryna
entitled «Development of a software module enhancement
reliability of custom`s computer network»

This diploma project is devoted to the development of a software module for improving the reliability of the computer network of the Volyn Customs DFS. Due to the rapid automation of customs work, it is important to protect the information processed.

This work analyzes the technical and software tools for maintaining the integrity of the institution's information system. The factors that affect the correct and uninterrupted operation of the LAN are identified. Methods have been identified to control and restrict the actions of PC users. Software is designed and developed that implements these methods. Network reliability enhancement module tested and debugged. Tested on institution computers.

Keywords: reliability, security, computer network, human factor, development.

ВСТУП

Життя сучасного суспільства не можливе без повсюдного застосування інформаційних технологій. Комп'ютери обслуговують банківські системи, контролюють роботу атомних реакторів, розподіляють енергію, стежать за розкладом потягів, космічними кораблями. Сьогодні комп'ютерні системи і телекомунікації визначають надійність систем оборони і безпеки країни, реалізують сучасні інформаційні технології, забезпечуючи зберігання інформації, її обробку, доставку і надання споживачам.

Однак саме висока ступінь автоматизації, до якої прагне сучасне суспільство, ставить його в залежність від рівня безпеки використовуваних інформаційних технологій. Масове застосування комп'ютерних систем, що дозволило вирішити задачу автоматизації процесів обробки постійно наростаючих обсягів інформації, зробило ці процеси надзвичайно уразливими по відношенню до агресивних дій і поставило перед споживачами інформаційних технологій нову проблему, - проблему інформаційної безпеки.

Надзвичайно актуальним є захист інформації та розробка комплексу заходів, спрямованих на виключення викрадення та пошкодження важливих даних. Завдання захисту полягає в підтримці цілісності, доступності та конфіденційності інформації. Саме тому, зі значним впровадженням інформаційних технологій, постає задача в аналізі та підвищенні інформаційної безпеки митниці, що оброблює та передає дані державного рівня.

Об’єктом дослідження є інформаційна система та процеси управління локальною мережею Волинської митниці Державної фіскальної служби України.

					КНТЕУ-122-2019		
Зм.	*Аркуш*	*№ документу*	*Підпис*	*Дата*			
Зав. кафедрою		*Пурський О.І.*		*03.12.2019*	*Розробка програмного модулю підвищення надійності комп’ютерної мережі Волинської митниці ДФС*	*Сторінка*	*Сторінок*
Керівник		*Самойленко Г.Т.*		*03.12.2019*		*8*	*81*
Гарант		*Пурський О.І.*		*03.12.2019*		*Кафедра комп’ютерних наук 2м-6*	
Розробив		*Ждань К.В.*		*03.12.2019*	*Вступ*		
Перевірив		*Самойленко Г.Т.*		*03.12.2019*			

Предметом дослідження є надійність та безпека інформаційної системи Волинської митниці ДФС.

Основною метою дипломної роботи є проектування та розробка програмного модулю підвищення надійності комп'ютерної мережі митниці.

При вирішенні поставлених завдань застосовувалися такі методи, як метод аналізу та синтезу, метод порівняння, метод формалізації, методи проектування та програмування, метод експерименту. А також спеціальні методи розробки ПЗ, як, наприклад, метод управління зручністю використання програмних продуктів.

Запровадження інформаційних митних технологій має правове підгрунтя на внутрішньо-державному та на міжнародно-правовому рівнях регулювання. Основним міжнародно-правовим актом, що передбачає запровадження інформаційних технологій в митній справі є Міжнародна конвенція про спрощення і гармонізацію митних процедур від 18.05.1973 року (Кіотська конвенція) до якої Україна приєдналась 05.10.2006 року.

Інформаційне забезпечення дослідження складається з українських та зарубіжних видань та Інтернет-ресурсів щодо проектування комп'ютерної мережі, інформаційної безпеки, особливостей використання середовищ розробки, специфікацій мов програмування та ін.

РОЗДІЛ 1. Опис предметної області

1.1 Основні характеристики корпоративної мережі

Комп'ютерні мережі - це об’єднання персональних комп’ютерів, розподілених на деякій території і сполучених для спільного використання ресурсів (програм, даних і апаратних компонентів).

Головною ціллю сполучення комп'ютерів у мережу є надання абонентам доступу до певних інформаційних ресурсів (наприклад, документів, програм, баз даних і т.д.), розподіленим по цих комп'ютерах, і їх спільного використання [1].

Існують мережі простого типу, що складаються з декількох комп’ютерів та повноцінні системи, що сполучають мільйони пристроїв. Будь-яка корпоративна мережа, в тому числі і комп’ютерна мережа адмінбудинку Волинської митниці ДФС, складається з великої кількості взаємозалежних елементів. Кожна частина системи виконує ті чи інші функції у рамках єдиного неперервного процесу. Елементи пов’язані між собою інформаційно, а також обмінюються між собою факсами, документами та письмовими розпорядженнями. Окрім цього, елементи взаємодіють із зовнішніми системами — структурно, функціонально та інформаційно. Ця ситуація справедлива для усіх організацій та установ, не зважаючи на їх вид діяльності [2].

Корпоративна мережа митниці слугує платформою для єдиної внутрішньої системи документообігу, оперативного збору інформації та звітів з різних пунктів пропуску Волинської митниці ДФС, централізації інформаційно-фінансових потоків і т.д [3].

Адмінбудинок — це 3-и поверхова будівля. Структура локальної мережі

Волинської митниці ДФС представлена на рис. 1.1.

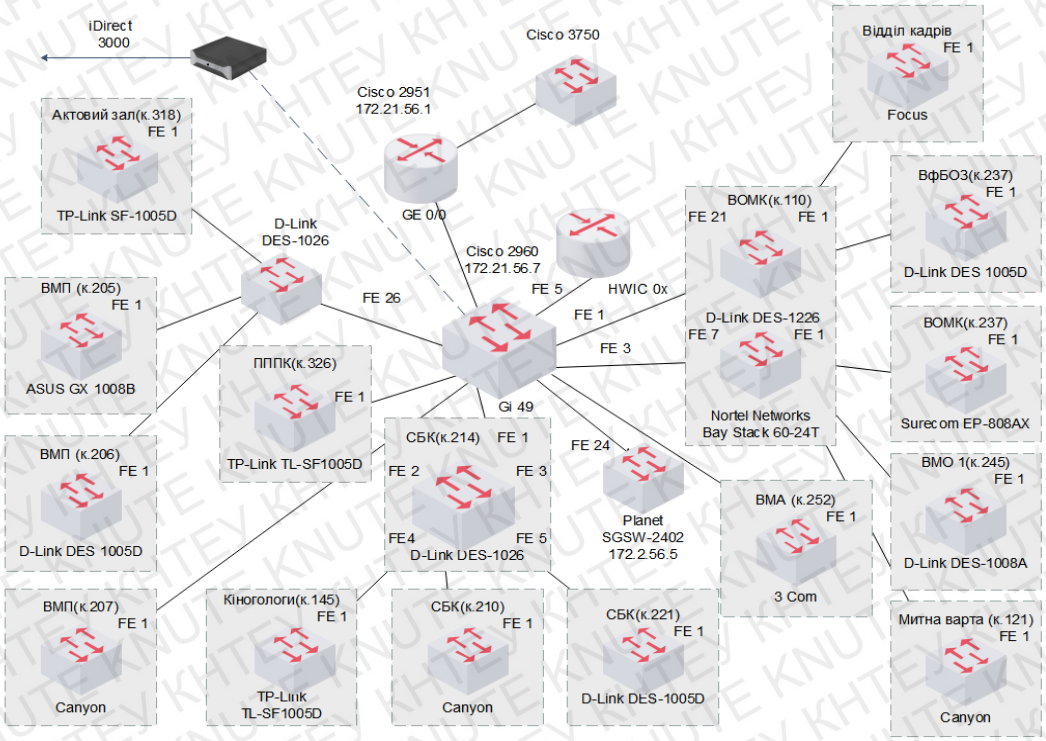


Рис. 1.1 Структура локальної мережі

Локальній мережі притаманна клієнт-серверна архітектура. Топологія локальної мережі – є змішаною топологією Star Bus (зірка на шині). У ній використовується один кабель, який ще називається магістраллю, до якого підключені декілька мереж з топологією зірка. На кінцях магістралі розташовані мережеві узгоджувачі - термінатори.

Локальна мережа будівлі спроектована по найпоширенішому на сьогоднішній день протоколу Ethernet. Протокол працює на фізичному та канальному рівнях моделі OSI. За строго технічним визначенням Ethernet — сімейство протоколів стандарту IEEE 802.3. Мережа побудована за технологією Gigabit Ethernet 1000Base-T. Gigabit Ethernet (GbE) — термін, який описує набір технологій, що використовуються для передачі пакетів Ethernet зі швидкістю 1 Гбіт/с. . Він визначений в документі IEEE 802.3-2005. Фізичний шар стандарту у нашому випадку це вита пара(1000Base-T). Стандарт IEEE802.3ab вийшов у 1999 році. Основна мета Gigabit Ethernet полягає в підвищенні швидкості передачі даних.

Швидкість передачі становить 1000Мб/с. Максимальна довжина сегменту в метрах становить 100 метрів. А типи кабелю: UTP/STP cat 5,5e,6,7.

Локальна мережа адмінбудинку Волинської митниці складається зі 118 робочих станцій, файлового сервера, сервера бази даних, поштового сервера, та сервера системи безпеки. До складу локальної мережі ще відносяться маршрутизатори, комутатори, концентратори, мости, шлюзи, кабельна система та програмне забезпечення.

Сервером називають спеціальний комп'ютер, що виконує основні сервісні функції, такі як: управління мережею, збір, обробка, зберігання і надання інформації користувачам КМ. Доцільно розділити сервери за їх функціональним призначенням у зв’язку з великим числом сервісних функцій.

Сервер бази даних використовуються для обробки запитів користувачів на мові SQL. При цьому СУБД знаходиться на сервері, до якого і підключаються клієнтські програми. Сервер бази даних Волинської митниці ДФС на базі IBM PowerLinux 7R2 16-core 4.228 GHz Power7+ 256Gb RAM. На сервері розміщена програма “Інспектор-2006”.

Програма "Інспектор-2006" розроблена за архітектурою клієнт — сервер та працює під керуванням операційної системи Microsoft Windows 2003. Серверна (СУБД Microsoft SQL Server 2000 та база даних) і клієнтська частини можуть бути встановлені як на один, так і на різні комп'ютери, причому кількість клієнтських установок обмежується тільки потужністю сервера [4].

Файловий сервер - сервер, що зберігає інформацію у вигляді файлів і представляє користувачам доступ до неї. Файл-сервер забезпечує і певний рівень захисту від несанкціонованого доступу. На митниці використовуються 2 файлових сервери - Dell PowerEdge 1950, 2 процесора Intel Quad-Core L5420 2.5GHz, 16GB DRAM, 2x 73GB 15K SAS.

Сервер застосунків - сервер, що виконує деякі прикладні програми. Термін також відноситься і до програмного забезпечення, що встановлено на такому

сервері і забезпечує виконання прикладного ПЗ[5].

Як сервер застосунків використовується сервер DELL PowerEdge R720. Це двосокетний сервер з процесорами Intel Xeon E5 в 2U корпусі. Встановлено 4 шт. по 8Gb RAM та 2 шт. по 300GB 2.5″ SAS 6Gbps HS 10K гнучких дисків та 4 шт. по 100Gb додаткових гнучких дисків [6].

Мережевий диск (мережевий драйвер) - призначений логічний диск (папка), який служить для зберігання «загальних» файлів, доступних для всіх користувачів, на інших персональних комп'ютерах, включених в загальну локальну мережу. Всередині локальної мережі є 2 мережевих диски. Диск X ємністю 1Тб. є диском, доступним для усіх працівників установи. Диск G такої ж ємності є унікальним для кожного відділу митниці. Ніхто окрім працівників певного відділу не має доступу до службової інформації, що знаходиться на ньому. Це дуже зручно, адже налагоджена робота підрозділів установи забезпечує коректне функціонування усієї митниці.

Ядро (Cisco 3500 series) комп’ютерної мережі поділене на локальну мережу (Cisco 2960) та систему відеоспостереження. Через маршрутизатор (Cisco 3911) дані з ядра відправляються до ДФС в Київ, де є 2 вузли (1 основний, 2-ий резервний). Усі митні пости та митниця на пряму з’єднана з ДФС та її вузлами.

Комп’ютерна мережа адмінбудинку Волинської митниці ДФС - це система зв’язку через кабельне середовище. Для передачі інформації у комп’ютерній мережі використовують мережеві кабелі: виті пари. На кінцях кабелів розташовані спеціальні сполучні роз’єми 8-контактні конектори 8P8C, відомі більшості під назвою RJ-45.

Концентратор, або хаб, – багатопортовий повторювач, що дозволяє обслуговувати відразу декілька комп'ютерів в мережі. В межах будівлі знаходиться 7 концентраторів (2 на 1-у поверсі, 3 на 2-у поверсі та 2 на 3-у поверсі). При з’єднанні комп’ютерів за допомогою даного пристрою використовується розкладка “нормально”.

При каскадуванні хабів або при підключенні комп’ютер - комп’ютер (без хабів) використовується розкладка “uplink” на одному кінці кабелю та “нормально” на іншому.

1.2 Способи та результати втручання користувачів у надійність та відмовостійкість комп’ютерної мережі

На сьогоднішній день зараження ПК небезпечним програмним забезпеченням можливе не лише технічним шляхом. Доволі активно для досягнення цієї цілі експлуатуються методи соціальної інженерії (фішинг, рекламні банери, розсилка спаму). Використовуючи такі людські якості, як цікавість, неуважність, і бажання отримати безкоштовні бонуси, зловмисники викликають з боку користувача дії, що полегшують проникнення шкідливих програм до системних файлів і операційної системи [7].

Абонент КМ може зіштовхнутись з тим, що отриманий на електронну пошту лист заохочує його перейти на сайт, де знадобиться введення пароля чи інших особистих даних для вирішення неіснуючих загроз власним фінансовим рахункам у банку або платіжній системі. Електронні листи з невідомих адрес можуть містити вкладення з безкоштовним софтом, який, насправді, не є таким. Іноді достатньо навести курсор на спамові розсилки для того щоб запустити небезпечний скрипт.

Комп'ютерні віруси завдають шкоди розміром в декілька мільярдів доларів кожного року, спричиняючи системні критичні помилки, зупиняючи налагоджену роботу великих сайтів та веб-додатків, знищуючи або модифікуючи файли, додаючи час відклику. Наслідки зараження ПК шкідливим програмним забезпеченням можуть бути незворотніми. До первинних ознак зараження відносять: неспроможність завантажити комп’ютер, зникнення файлів, форматування HDD, неспроможність завантажити файли, незрозумілі системні повідомлення, звукові ефекти і т. д.

До вторинних ознак належать: уповільнення роботи комп’ютера, зменшення вільної пам’яті, затримка при виконанні програм, незрозумілі зміни в файлах, зміна

дати модифікації файлів без причини, незрозумілі помилки write-protection, помилки при інсталяції і запуску Windows, неспроможність зберігати документи Word в інші каталоги, повільна робота дисків, виникнення файлів невідомого походження [8].

1.3 Огляд існуючих способів та засобів захисту комп'ютерної мережі від впливу людських чинників

Важливим є використання спеціального ПЗ, що захищає мережу від різного виду загроз. Застосування спеціалізованих продуктів, які будуть підтримувати і по можливості автоматизувати необхідні заборонні і контрольні процедури можуть стосуватися не тільки рядових співробітників, але і привілейованих користувачів. Наприклад, якщо заборонено використовувати флешки, то впровадити продукт для контролю зовнішніх пристроїв, який буде дійсно це забороняти, а також повідомляти про порушників.

Керівництво організації в свою чергу повинно було дотримуватися деяких рекомендацій по засобам захисту та використовувати такі модулі як:

- антивіруси (Kaspersky, Symantec, G DATA і ін.);
- захисні мережеві екрани (Entensys, Kerio і ін.);
- спеціалізовані засоби захисту від DDoS attack (Attack Killer, Qrator і ін.);
- технології захисту від вразливостей (Appercut, Checkmarx, Fortify і ін.);
- спеціалізовані засоби захисту від цільових атак (Attack Killer, FireEye і ін.).

Серед ПЗ захисту установи є антивірус.

Надзвичайно важливими є засоби архівування інформації. Інколи резервні копії інформації доводиться виконувати при загальній обмеженості ресурсів розміщення даних, наприклад власникам персональних комп'ютерів. У цих випадках використовують програмну архівацію. Архівація є злиттям декількох файлів і каталогів в єдиний файл – архів, одночасно зі скороченням загального обсягу вихідних файлів шляхом усунення надмірності, але без втрат інформації, тобто з можливістю точного відновлення вихідних файлів.

Криптографічні методи захисту інформації використовуються для обробки, зберігання та передачі інформації на носіях і по мережах зв'язку. при передачі даних на великі відстані єдино надійним способом шифрування є криптографічний захист інформації . Шифрування також знижує можливості повноцінного огляду. Тому все більше підприємств починають використовувати штучний інтелект та машинне навчання. За їхньою допомогою легше виявляти нетипову активність і великі обсяги шифрованого веб-трафіку. Отриману інформацію досліджують фахівці з інформаційної безпеки.

Дієвим є використання проксі-серверів. Даний сервер забезпечує захист комп'ютерної мережі від несанкціонованого доступу: наприклад, можна налаштувати проксі-сервер так, що локальні комп'ютери будуть звертатися до зовнішніх ресурсів тільки через нього, а зовнішні комп'ютери не зможуть звертатися до локальних взагалі. Проксі-сервер дозволяє користувачу виконувати непрямі запити до мережевих сервісів[9]. Проте цей метод не є достатньо дієвим проти атак на більш високих рівнях - наприклад, на рівні додатку (віруси, код Java і JavaScript).

Міжмережеві екрани, що також називаються брандмауерами або фаєрволами, інспектують і фільтрують весь прохід трафіку мережевого / транспортного рівнів між глобальними та локальними мережами. Це дозволяє різко знизити загрозу несанкціонованого доступу ззовні в корпоративні мережі, проте не усуває цю небезпеку повністю [10].

VPN (віртуальна приватна мережа) дозволяє передавати секретну інформацію через мережі, в яких можливе прослуховування трафіку сторонніми людьми. Використовувані технології: PPTP, PPPoE, IPSec [11].

ВИСНОВКИ ДО РОЗДІЛУ 1

Проаналізувавши технічні характеристики комп'ютерної мережі установи, принципи поводження абонентів з ПК митниці, а також наявне програмне та технічне забезпечення, що використовується для підтримки безперервної коректної

роботи мережі необхідно зробити наступні висновки:

- система засобів технічного та програмного забезпечення надійності та відмовостійкості є застарілою та потребує модернізації;
- надзвичайно великий вплив на налагоджену роботу локальної мережі має її абонент;
- актуальним є створення програмного модулю, що надавав би додатковий захист від необдуманих та неуважних дій користувачів;
- необхідним є підвищення комп'ютерної грамотності усіх користувачів ПК установи.

РОЗДІЛ 2. Аналіз інформаційного забезпечення предметної області

2.1 Методи дослідження предметної області

Перед початком створення певного ПЗ доцільним є виокремлення, аналіз та пошук необхідних методів створення і супроводження ПЗ комп'ютеризованих систем управління. Коректна робота програмного модулю напряму залежить від правильного вибору методів створення або ж оптимізації, що дозволяють забезпечити найкращу ефективність процесу розробки.

Для досягнення поставленої мети використовувались такі методи дослідження:

1.Аналіз та синтез застосовувались при ознайомленні з програмними та технічними можливостями локальної мережі митниці, під час опису та вивчення функціональних характеристик наявних платформ захисту мережі.

2.Порівняння використовувалось для знаходження необхідного функціонального забезпечення, яке б суттєво підвищило стійкість системи.

3.Формалізація та моделювання використовувались під час складання технічного завдання на розробку. Усі накопичені раніше знання та інформація систематизувались для ефективної розробки, тестування та налагодження програмного засобу.

4.Проектування та програмування використовувались для розроблення засобу, який реалізує усі необхідні та визначені в процесі аналізу мережі функції. Під час використання даних методів часто виникає необхідність в модернізації технічного завдання.

5.Експеримент використовувався для апробації запропонованого програмного рішення. Перевірка на практиці дозволяє емпірично ознайомитись зі специфікою програми, а також виокремити пункти, що потребують оптимізації або ж переробки під час кінцевого налагодження модулю [12].

Під час розробки подібного програмного модулю підвищення надійності комп'ютерної мережі установи використовуються і інші спеціальні методи розробки ПЗ. Серед них можна виокремити:

1.Метод експертного оцінювання властивостей повторно використовуваних компонентів ПЗ. Повторне використання є досить популярним у наш час, адже це дієвий підхід до зменшення затрат та часу на створення ПЗ шляхом застосування накопиченого досвіду [13].

2.Метод синтезу та аналізу стилів програмування. Даний метод спрямований на автоматизацію застосування стилю програмування.

3.Метод управління зручністю використання програмних продуктів. Даний метод включає перелік етапів, що виконуються під час розробки ПЗ для надання продукту зручності використання – ступеню, що в якому ПП може бути використаний певними користувачами для досягнення визначених цілей з ефективністю, економічністю та задоволеністю у певному контексті використання [14].

2.2 Технічне завдання на розробку програмного забезпечення.

Технічне завдання є основним документом, що визначає вимоги і порядок щодо розробки рішення. Визначення технічного завдання відбувається у декілька етапів.

Першим етапом є вибір найменування та його коротка характеристика. Повне найменування ПЗ, що розробляється : «Програмний модуль підвищення надійності комп'терної мережі митниці». Скорочене найменування ПЗ: «Модуль «НКМ»».

Коротка характеристика: програма призначена для обмеження та обліку дій абонентів КМ митниці.

Область застосування модулю: використання його як повнофункціонального програмного комплексу для підвищення надійності комп'ютерної мережі Волинської митниці ДФС.

Другим етапом є підстава для розробки. Розробка даного ПЗ ведеться на підставі завдання на проходження переддипломної практики освітньо-кваліфікаційного рівня «Магістр».

Третім етапом є призначення розробки. Метою проекту є розробка програми, що підвищує надійність та відмовостійкість системи, зменшує імовірність проникнення в інформаційну систему митниці шкідливого ПЗ та витік даних з локальної мережі митниці контролюючи дії користувачів ПК.

Функціональне призначення: аналіз та обмеження дій користувачів задля підвищення безпеки роботи та обміну даними користувачами локальної мережі.

Експлуатаційне призначення: комплексний модуль може застосовуватись на всіх ЕОМ подібних установ задовольняючи мінімальні вимоги щодо апаратних та програмних засобів для нормального функціонування системи.

Четвертим етапом є вимоги до функціональних характеристик. Додаток повинен надавати наступні можливості адміністратору:

- додавання користувачів на налаштування їх доступу до тих чи інших;
- можливостей роботи з ПК та зовнішніми пристроями;
- додавання заборонених сайтів та програмних комплексів;
- заборона завантажень з глобальної мережі Інтернет;
- запис історії з даними про час та адресу відвіданих користувачами сайтів;
- заборона читання/запису на зовнішні накопичувальні пристрої.

П'ятим етапом є вимоги до надійності. Модуль повинен виконувати наступні вимоги щодо надійності:

- обробляти помилкові дії користувачів і повідомляти їх про це;
- підтримувати функції захисту від несанкціонованого доступу.

Шостим етапом є вимоги до параметрів технічних засобів. Для коректної роботи програми необхідно щоб технічні засоби відповідали таким мінімальним вимогам:

- 32-розрядний(x86) або 64-розрядний(x64) процесор із тактовою частотою 1 ГГц;
- 1 гігабайт (ГБ) RAM (для 32-розрядної версії) або 2 ГБ (для 64 - розрядної);
- ОС Windows 2003/2007.

2.3. Вибір засобів розробки модулю.

Розробка програмного модулю – це процес, що складається з взаємозалежних підпроцесів або дисциплін. Якщо розглядати включення даних етапів в модель водоспаду, то їх розташування є послідовним:

1.Специфікація або ж визначення основних вимог до виконання технічного завдання.

2.Проектування ПЗ.

3.Програмування ПЗ.

4.Тестування ПЗ.

5.Налагодження ПЗ.

6.Впровадження ПЗ.

7.Супровід ПЗ [15].

В деяких випадках етапи вимагають використання різних програмних засобів, за допомогою яких можна ефективно та якісно виконати той чи інший процес. Для визначення вимог до виконання т/з та програмування ПЗ інколи вистачає тектового редактору. Для програмування достатньо спеціалізованого редактора вихідних текстів, компілятора та інтерпретатора, а для налагодження – відлагоджувача, що необхідний для виправлення вад.

Проте найоптимальнішим є використання інтегрованого середовища розробки, що включає майже усі необхідні для повного життєвого циклу програми засоби. Ці засоби є взаємодоповнюваними та взаємозалежними.

IDE (Integrated development environment) – комплексне програмне забезпечення, що мінімально складається з редактора вихідного коду, засобів автоматизації побудови та налагоджувача. Межа між інтегрованим середовищем розробки та іншими частинами більш широкого середовища розробки програмного забезпечення недостатньо чітко визначена. Іноді система управління розробкою або різні інструменти для спрощення побудови графічного інтерфейсу користувача (GUI) є інтегрованими [16].

Окремі утиліти командного рядка часто видають результат у вигляді звичайного тексту в форматі, що допускає введення іншими утилітами, завдяки чому вони виступають в якості фільтрів даних [17].

В наш час створено ряд зручних та доступних інтегрованих середовищ розробки, що значно полегшують роботу програмістів. Серед найпопулярніших можна виділити:

1) NetBeans IDE - інтегроване середовище розробки додатків, безкоштовне IDE з відкритим вихідним кодом. Призначена для професійної розробки десктоп додатків, web-додатків, корпоративних систем, програм для мобільних пристроїв. NetBeans - єдине IDE, яка влаштує і початківця розробника і професіонала [18].

NetBeans дозволяє розробляти додатки з набору модульних компонентів програмного забезпечення, що називаються модулями. NetBeans працює на Windows, macOS, Linux та Solaris. Крім розробки Java, вона має розширення для інших мов, таких як PHP, C, C ++, HTML5 та JavaScript. Програми на базі NetBeans можуть бути розширені сторонніми розробниками [19].

2) Eclipse – провідна відкрита платформа для професійних розробників. Eclipse є платформою для розробки будь-яких інтегрованих середовищ програмування і практично будь-якого клієнтського ПЗ. Налагоджувач Eclipse дозволяє заглянути в середину виконання коду. Це дозволяє ознайомитись з поетапною роботою коду. Eclipse забезпечує повну підтримку таких методів гнучкої розробки ПЗ як тестування та рефакторінг [20].

Використовуючи Eclipse IDE можна легко поєднати підтримку декількох мов та інші функції в будь-якому пакеті за замовчуванням, а Eclipse Marketplace надає практично необмежені налаштування та розширення [21].

3) Dev-C++ – інтегроване середовище розробки на C і C ++, повнофункціональна C ++ IDE. Серед основних особливостей виділяють:

- існування переносної версії програми, що не потребує завантаження;
- вбудований менеджер проектів;
- гнучке налаштування робочого середовища, редактора і компілятора, велика кількість різних опцій;
- можлива робота з CVS (Concurrent Versions System).

4) IntelliJ IDEA - IDE, розроблена Jet Brains. Користувачам пропонується безкоштовна версію Community Edition. IntelliJ IDEA підтримує Java 8 і Java EE 7, має великий інструментарій для розробки мобільних додатків і корпоративних технологій для різних платформ.

Мови програмування, що підтримуються: AngularJS, CoffeeScript, HTML, JavaScript, LESS, Node JS, PHP, Python, Ruby, Sass, TypeScript і інші.

Особливостями є:

- інтеграція з Git;
- підтримка Google App Engine, Grails, GWT, Hibernate, Java EE, OSGi, Play, Spring, Struts і інших;
- вбудовані засоби розгортання і налагодження для більшості серверів додатків.

5) Code::Blocks – безкоштовний, кросплатформний IDE з відкритим кодом та підтримуваними мовами : C, C ++ та Fortran. IDE може бути розширене за допомогою плагінів. Будь-який функціонал можна додати встановивши або написавши таким чином. Наприклад, функції компіляції та налагодження вже надаються плагінами [22].

6) RubyMine – комерційне інтегроване середовище розробки , що підтримує такі мови програмування : CoffeeScript, CSS, HAML, HTML, JavaScript, LESS,

Ruby і Rails, Ruby і SASS.

Серед особливостей можна виокремити:

- дерево проектів, яке довзволяє швидко аналізувати код;
- інтеграція з CVS, Git, Mercurial, Perforce и Subversion;
- RubyMotion підтримує розробку під iOS.

Проаналізувавши та ознайомившись з найпопулярнішими інтегрованими середовищами необхідно виділити Microsoft Visual Studio, що впевнено лідирує на ринку IDE протягом довгого часу. Це визнання серед професійних програмістів є ознакою зручності у використанні, багатофункціональності, надійності та гнучкості. Visual Studio 2019 версії Community є безкоштовним для студентів, open-source дописувачів та приватних осіб. Версії Professional та Enterprise є комерційними, проте вони значно розширюють можливості середовища додаючи масштабованість та можливість використання командами будь-яких розмірів.

VS – потужний інструмент, який дозволяє розробляти складні програмні комплекси, що мають множинні способи взаємодії з користувачами у вигляді різних діалогових вікон, панелей інструментів, меню, кнопок, списків і т.п. Ці програми називаються віконними додатками (або Windows-додатками) і забезпечують графічні інтерфейси. [23 стр.3]

До основних особливостей та переваг при роботі з даним середовищем можна віднести:

1)Автоматично згенерований код. Будь-який проект, в будь-якому випадку, містить автоматично згенерований код, який представляє собою основу майбутньої програми. VS пропонує безліч готових до використання елементів управління, включаючи і код, необхідний для їх створення. Це економить час розробників, позбавляючи їх від необхідності кожного разу заново створювати типовий програмний код для вирішення часто завдань, що часто зустрічаються.

2)Досвід швидкого кодування (Rapid Coding Experience). Велика кількість клавіатурних комбінацій (keyboard shortcuts) та технологія Intellisense, відома як

автодоповнення оптимізують роботу програміста по кодуванню. Існує і набір засобів швидкої переробки рефакторінга (refactoring), які дозволяють швидко вдосконалити структуру коду, не відриваючись від процесу програмування [24].

3)Настроюваність і розширюваність. Багато з елементів, що утворюють середовище VS, є налаштованими. Для більш складних варіантів настройки VS пропонує спеціальний інтерфейс прикладного програмування (Application Programming Interface, API), призначений для створення власних додаткових модулів (add-ins) і розширень (extensions).

4) Статистика моніторингу продуктивності в режимі реального часу.

5) Список помилок, який спрощує налагодження.

6) Підтримка розділеного екрану та ін. [25, стр 8]

Мови та засоби програмування, що підтримуються : Ajax, ASP.NET, DHTML, JavaScript, JScript, Visual Basic, Visual C #, Visual C ++, Visual F #, XAML і інші.

Проте працюючи з певним середовищем необхідно володіти знаннями щодо його недоліків. Оскільки VS є надзвичайно важким ПЗ, для відкриття і запуску додатка необхідні значні ресурси. Тому на деяких пристроях внесення мінімальних змін у проект може займати багато часу. Цю особливість необхідно враховувати перед початком створення рішення в даному середовищі.

Вибір мови програмування це складний процес, оскільки конкуруючих мов високого рівня досить багато і важко обрати найкращу для задоволення власних потреб.

Протягом останніх декількох років найпопулярнішими та найзатребуванішими мовами програмування є:

- Python – це проста в освоєнні і потужна мова програмування. Вона надає ефективні високорівневі структури даних, а також простий, але ефективний підхід до об'єктно-орієнтованого програмування. [26, стр.19] До особливостей даної мови відносяться:

1) простота – мінімалістична мова;

2)легкість в освоєнні;

3)портування;

4)інтерпретація;

5)об’єктна орієнтованість;

6)розширюваність;

7)вбудованість та ін. [26, 21стр.]

Java - це проста, інтерпретована, мережева, незалежна від архітектури, динамічна, високопродуктивна мова програмування. Основними можливостями даної мови є:

1)автоматичне управління пам’яттю;

2)наявність простих засобів створення мережевих додатків;

3)наявність класів, що дозволяють виконувати HTTP-запити та обробляти відповіді;

4)уніфікований доступ до баз даних;

5)паралельне виконання програм та ін [27, 12 стр].

- C++ - компільована, статично типізований мова програмування загального призначення. C ++ містить засоби розробки програм контрольованої ефективності для широкого спектра задач, від низькорівневих утиліт і драйверів до складних програмних комплексів. До переваг даної мови можна віднести:

1) висока сумісність з мовою C;

2) підтримка різних стилів програмування;

3) перевантаження операторів [28, 527стр.];

4) шаблони C++ ;

5) висока обчислювальна продуктивність;

6) доступність та ін.

- Ruby – це потужна, динамічна, чисто об'єктно-орієнтована мова, при розробці якої основна увага була приділена зручності програмування. Багато вдалих ідей, використаних раніше в таких мовах, як Perl, Python, Smalltalk, LISP і деяких

інших, в Ruby вдалося гармонійно об'єднати і втілити [29, 5стр].

Безумовно створюючи подібний список не можливо обійти стороною таку перспективну мову як C Sharp (C#). C# - проста, сучасна, об'єктно-орієнтована і типобезпечна мову програмування. C# відноситься до широко відомого сімейства мов C, і здасться добре знайомою кожному, хто працював з C, C ++, Java або JavaScript.

Основними особливостями даної мови програмування, що вплинули на її обрання для написання програмного модулю серед інших штучних мов є:

- простота, зручність та надійність коду;
- немає ніяких обмежень у спадкуванні, модифікації і універсалізації створеного програмного продукту;
- велика кількість синтаксичного цукру, що представляє собою спеціальні конструкції, розроблені для розуміння і написання коду;
- поріг входження у мови C # низький. Його синтаксис має багато схожого з іншими мовами програмування, завдяки чому полегшується перехід для програмістів. Мова C # вважається найбільш зрозумілою і придатною для новачків;
- C # відноситься до мов компільованого типу, тому він має всі переваги таких мов;
- C # об'єднує кращі ідеї сучасних мов програмування Java, C ++, Visual Basic і т.д [30].

ВИСНОВКИ ДО РОЗДІЛУ 2

Дослідивши та проаналізувавши методи дослідження у програмуванні можна зробити висновок, що найоптимальнішими для створення подібного ПЗ є методи:

- аналізу та синтезу;
- проектування та програмування;

- експерименту;
- управління зручністю використання програмних продуктів;
- експертного оцінювання властивостей повторно використовування компонентів ПЗ.

Ознайомившись з сучасними середовищами програмування зроблено висновок, що найефективнішим та найзручнішим для вирішення наявної задачі є інтегроване середовище розробки Microsoft Visual Studio 2019.

Порівнявши придатні для програмування даного ПЗ мови, найбільш доречною та перспективною обрано об'єктно-орієнтовану мову програмування C#.

РОЗДІЛ 3. Розробка програмного модулю підвищення надійності комп'ютерної мережі митниці

3.1 Проектування програмного забезпечення.

Проекстування є важливим етапом в життєвому циклі будь-якого ПЗ. На етапі проектування постановка задачі повинна бути перетворена в алгоритм. Щоб уникнути помилок в програмах, алгоритмісти повинні використовувати ретельно розроблені процедури конструювання, засновані на правилах логічного висновку. Метою проектування є визначення внутрішніх властивостей системи і деталізації її зовнішніх (видимих) властивостей на основі виданих до ПЗ вимог (вихідні умови задачі).

Залежно від класу створюваного ПЗ, процес проектування може забезпечуватися як «ручним» проектуванням, так і різними засобами його автоматизації. В процесі проектування ПЗ для вираження його характеристик використовуються різні нотації - блок-схеми, ER-діаграми, UML-діаграми, DFD-діаграми, а також макети.

На цьому відрізку життя програмного забезпечення здійснюють:

- функціональну декомпозицію розв'язуваної задачі, на основі якої визначається архітектура системи цього завдання;
- зовнішнє проектування програмного забезпечення, що виражається у формі зовнішнього взаємодії його з користувачем;
- проектування бази даних, якщо це необхідно;
- проектування архітектури програмного забезпечення – визначення об'єктів,

модулів і їх сполучення.

Блок-схеми є одними з найстаріших та найпопулярніших графічних відображень алгоритмів та процесів. Весь доступний інструментарій для створення блок-схем з вихідного коду - це один з видів зворотного проектування, при якому створюються інспектовані артефакти, однак допускається інспектування до програмування, що і є основною ідеєю використання блок-схем [31].

Основними перевагами блок-схем при виборі засобів проектування є те, що вони:

- роблять алгоритм більш прозорим;
- підвищують дисципліну в процесі документування детального проектування;
- допомагають відстежити дефекти до того, як вони з'являються в програмному коді;
- підвищують надійність продукту.

Блок-схеми складають карту процесів за допомогою взаємопов'язаних графічних елементів, завдяки чому процес легко читабельний та зрозумілий. Блок-схеми корисні при необхідності пояснити, як працює складна і / або абстрактна процедура, система, концепція або алгоритм. Створення блок-схеми також допомагає в плануванні та розробці нового процесу або вдосконаленні існуючого (рис.3.1).

Блок-схеми створюються за допомогою спеціальних середовищ. Одним з найпопулярніших та найзручніших ПЗ є Edraw Max. Edraw Max - надзвичайно потужний інструмент для побудов діаграм. Незалежно від того, чи потрібно розробити блок-схеми, діаграми, UML-діаграми або офісні макети, все що потрібно можна знайти в Edraw Max.

До переваг даного програмного продукту відносять:

- масивні шаблони та символи;
- надійна сумісність файлів;
- динамічність та гнучкість наборів інструментів.

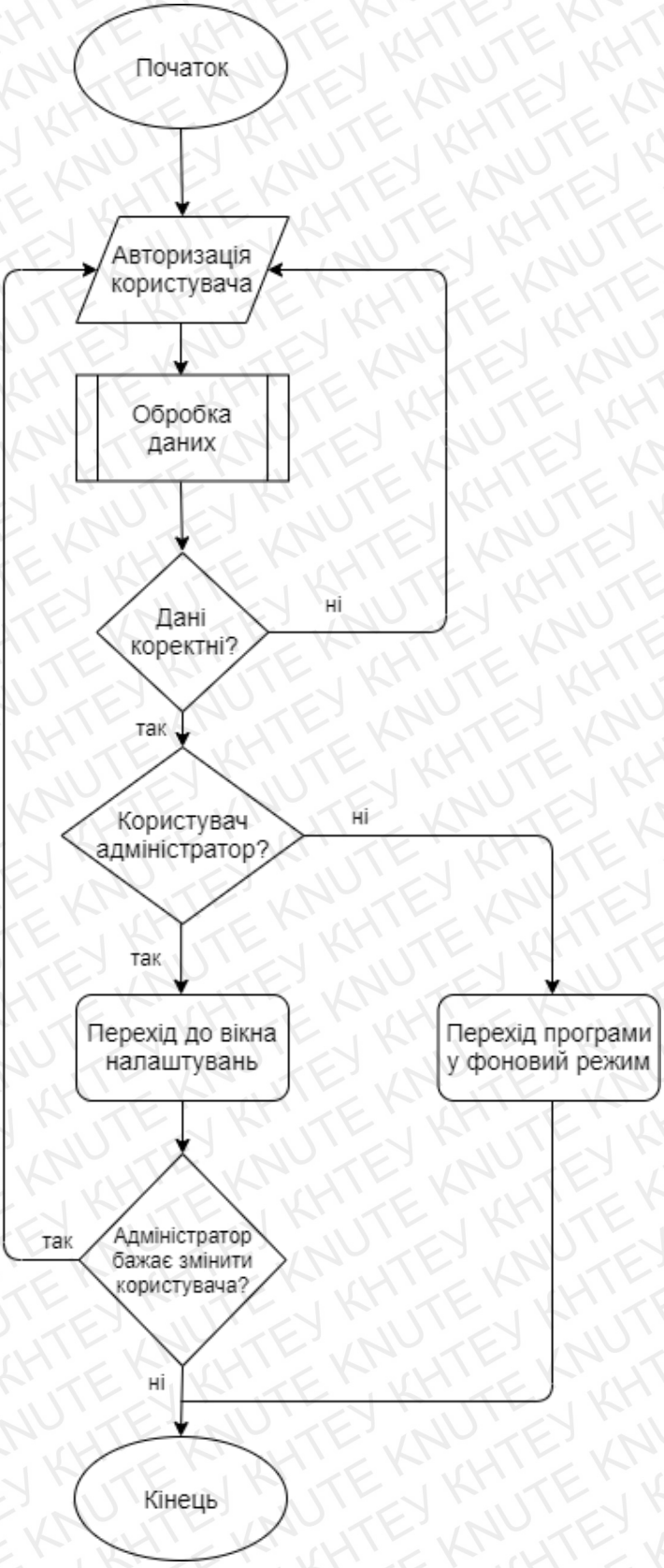


Рис. 3.1 Блок схема програмного модулю ««НКМ»»

3.2 Розробка та реалізація програмного продукту.

Програма складається з декількох файлів вихідного коду для його кращого структурування та розуміння. Файл Program.cs (Додаток А) створюється автоматично та є першим кроком до створення рішення. В даному файлі, як і в наступних, ми підключаємо необхідні простори імен та розпочинаєм розробку ПЗ.

Підключаємо метод для включення візуальних стилів EnableVisualStyles, та посилаємось на відкриття форми авторизації. Проте трапляється, що в ході виконання виникають виключення. Для того щоб зафіксувати їх, заключаємо запуск авторизації в оператор try-catch. При виникненні виключення загальномовне середовище виконання (CLR) шукає оператор catch, який обробляє цей виняток [32]. Якщо виключення знайдеться, то користувачеві відобразиться повідомлення про це та назва файлу у яку запишеться інформація щодо даного виключення(lastError.txt), а також відбудеться зупинка виконання програми. При відсутності виключень блок try із запуском авторизації коректно працюватиме до нашого самостійного завершення роботи (рис.3.2).

Program.cs

ПрограмаЗахисту | ПрограмаЗахисту.Program | Main()

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Threading.Tasks;
5   using System.Windows.Forms;
6
7   namespace ПрограмаЗахисту
8   {
        0 references
9       static class Program
10      {
11          /// <summary>
12          /// Головна точка входу для додатка
13          /// </summary>
14          [STAThread]
            0 references
15          static void Main()
16          {
17              Application.EnableVisualStyles();
18              Application.SetCompatibleTextRenderingDefault(false);
19              try
20              {
21                  Application.Run(new Authorization_form());//Запускаємо в першу чергу авторизацію
22
23              }
24              catch (Exception ex)
25              {
26                  using(System.IO.StreamWriter sw=new System.IO.StreamWriter("lastError.txt"))
27                  {
28                      sw.Write(ex.ToString());
29                  }
30                  MessageBox.Show($"Виключення записане у файл lastError.txt \n {ex.ToString()}");
31                  Worker.AppClose();
32              }
33          }
34      }
```

Рис.3.2 Фрагмент файлу Program.cs

Далі переходимо у файл Authorization_form.cs (Додаток Б) в якому описується процес авторизації користувачів. На початку ми ініціалізуємо Firewall, який в подальшому використовуватимемо для заборони відкриття заданих нами ресурсів та додаємо об’єкт для зберігання профілю користувача.

Створюємо вбудований акаунт адміністратора. Тобто, якщо користувачем є адміністратор жодна перевірка не запускається, а також немає жодних обмежень. Форма аутентифікації приховується та відбувається відкриття панелі адміністратора, а також примусове закриття проксі, якщо він запущений (рис.3.3).

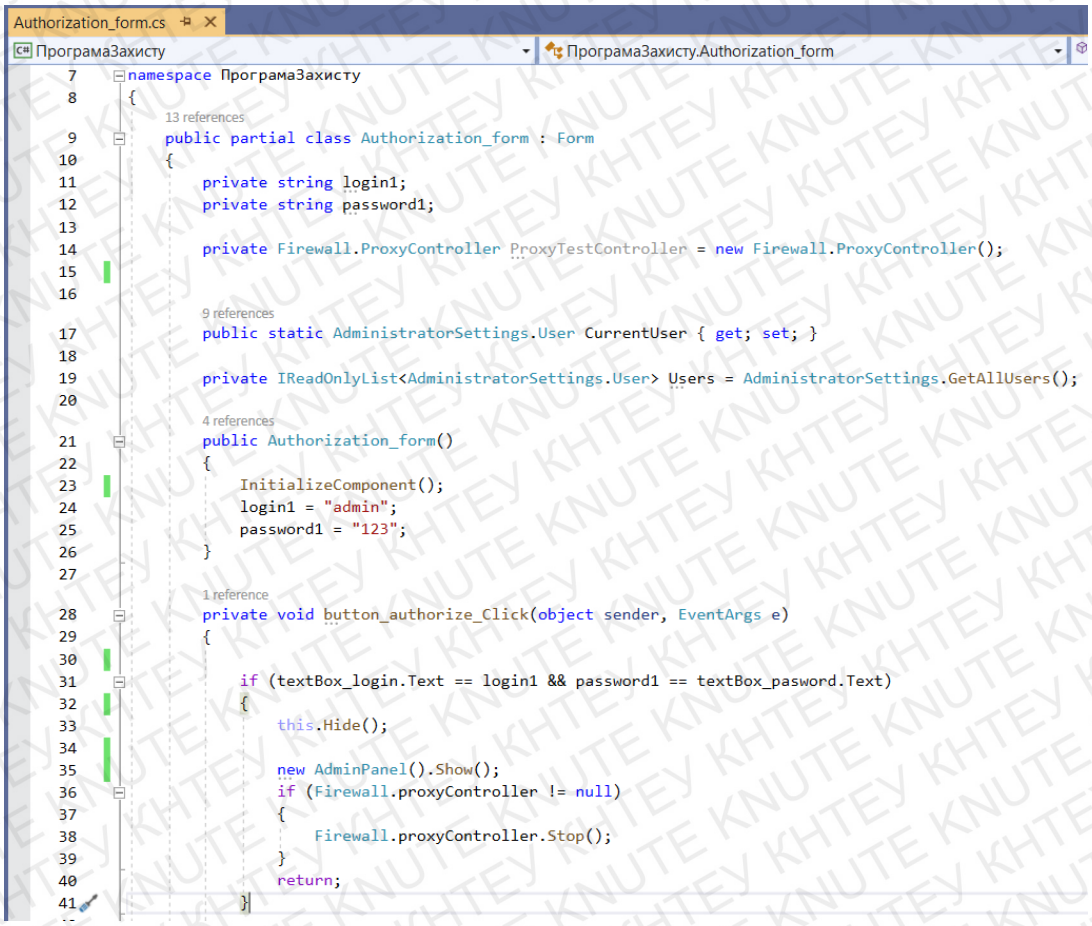
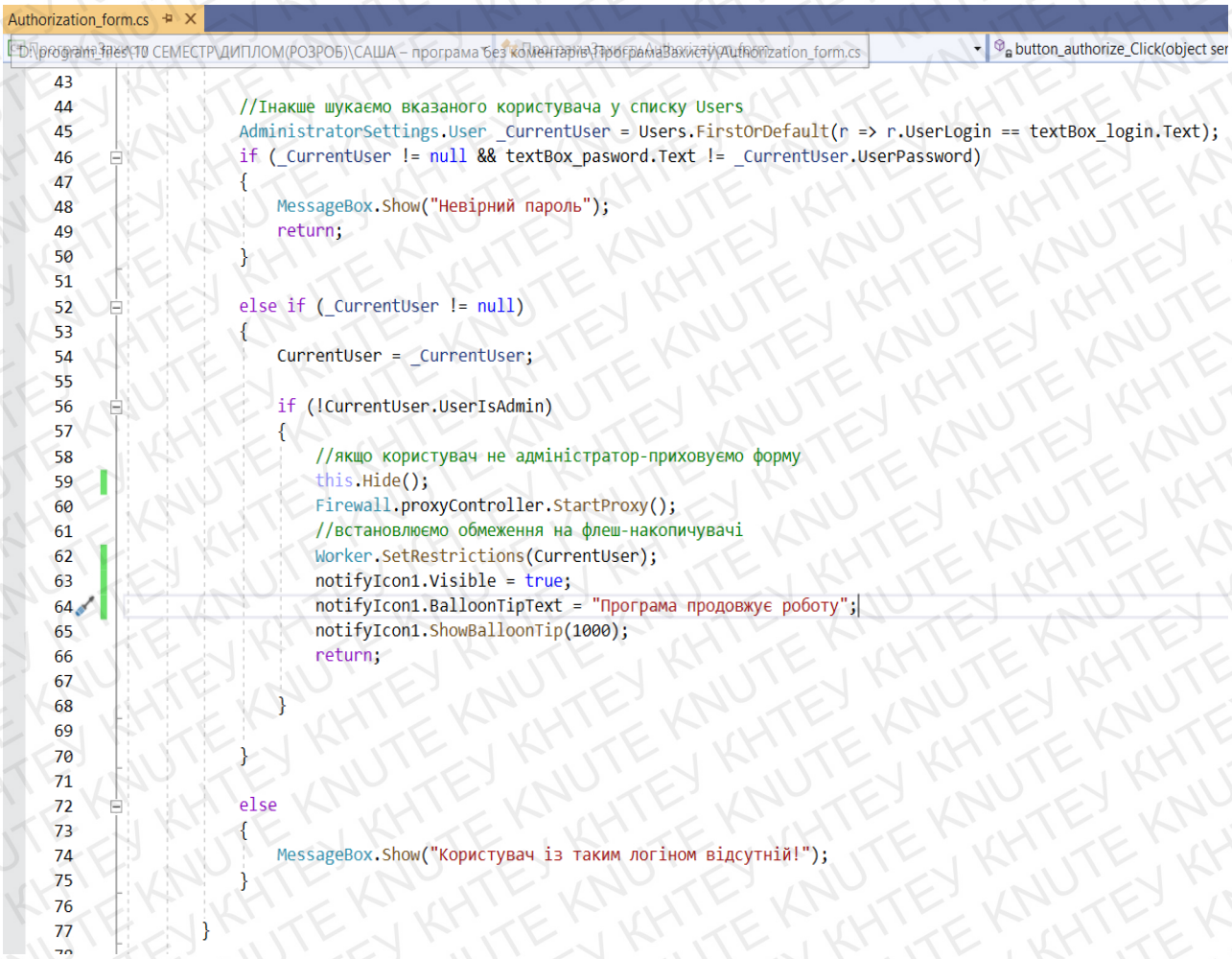


Рис. 3.3 Фрагмент файлу Authorization_form.cs

Якщо користувач не адміністратор, шукаємо даного користувача у списку Users, та в разі некоректно вказаного паролю попереджаємо його про це. Також попереджаємо про відсутність користувача з таким логіном, якщо при введені сталася помилка. В разі коректної авторизації ініціалізуємо проксі-сервер та

встановлюємо обмеження на використання зовнішніх накопичувальних пристроїв відповідно до налаштувань адміністратора. Програма переходить у фоновий режим в якому за допомогою системного трея(по-іншому область сповіщень) є змога змінити користувача та вийти з програми. Реалізовано показ повідомлення, яке інформує перехід програми у фоновий режим, а також час показу даного повідомлення (рис.3.4).




```
43
44              //Інакше шукаємо вказаного користувача у списку Users
45              AdministratorSettings.User _CurrentUser = Users.FirstOrDefault(r => r.UserLogin == textBox_login.Text);
46              if (_CurrentUser != null && textBox_pasword.Text != _CurrentUser.UserPassword)
47              {
48                  MessageBox.Show("Невірний пароль");
49                  return;
50              }
51
52              else if (_CurrentUser != null)
53              {
54                  CurrentUser = _CurrentUser;
55
56                  if (!CurrentUser.UserIsAdmin)
57                  {
58                      //якщо користувач не адміністратор-приховуємо форму
59                      this.Hide();
60                      Firewall.proxyController.StartProxy();
61                      //встановлюємо обмеження на флеш-накопичувачі
62                      Worker.SetRestrictions(CurrentUser);
63                      notifyIcon1.Visible = true;
64                      notifyIcon1.BalloonTipText = "Програма продовжує роботу";
65                      notifyIcon1.ShowBalloonTip(1000);
66                      return;
67
68                  }
69
70              }
71
72              else
73              {
74                  MessageBox.Show("Користувач із таким логіном відсутній!");
75              }
76
77          }
```

Рис.3.4 Фрагмент файлу Authorization_form.cs

Реалізуємо події, які виникають при закритті вікна авторизації та при завершенні роботи програми. При зміні користувача викликаємо форму авторизації. Для підвищення захисту даних, що вводяться при аутентифікації користувача замінюємо символи паролю на зірочки. При даблкліку на іконку треї відображаємо форму авторизації (рис.3.5).

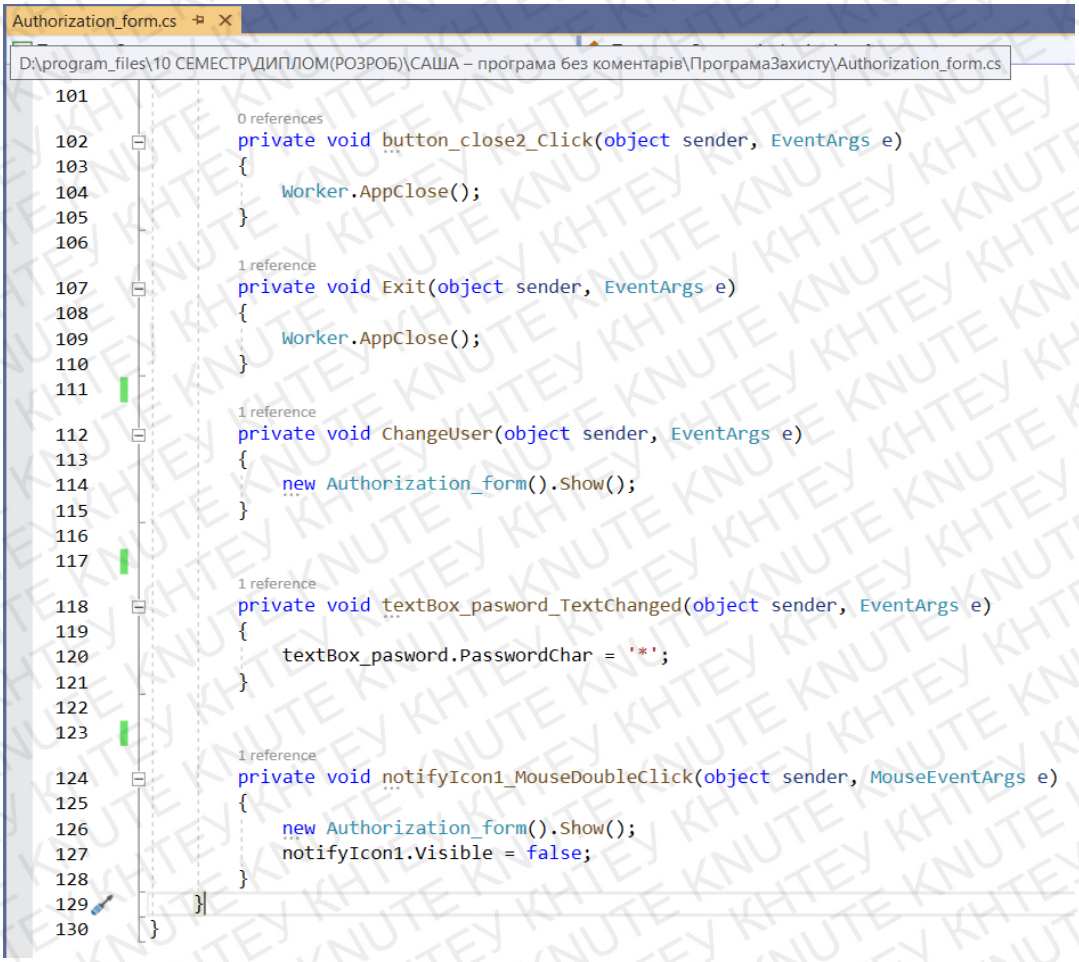


Рис.3.5 Фрагмент файлу Authorization_form.cs

Далі редагуємо файл AdminPanel.cs (Додаток В) у якому вказуються налаштування які може виконати адміністратор та будується таблиця з переліком цих налаштувань. Для початку додаємо метод для вибірки із полей при додаванні користувачів та створюємо новий екземпляр класу із настройками користувача. Ініціалізуємо метод SetTableData (рис. 3.6) за допомогою якого будуємо рядки та колонки для інтерфейсу користувацьких налаштувань. Метод реалізується наступним чином:

- ініціалізуємо нову тимчасову таблицю за допомогою створення екземпляра об’єкту DataTable;
- присвоюємо робочій таблиці значення тимчасової таблиці;
- отримуємо дані (зберігаємо у t)про користувачів та вставляємо їх у таблицю;

- створюємо новий рядок і вставляємо туди дані зі змінної t;
- перетворюємо масиви у яких зберігаються дані щодо заборонених процесів та ресурсів у рядок;

AdminPanel.cs

ПрограмаЗахисту — ПрограмаЗахисту.AdminPanel

```
44      private void SetTableData()
45      {
46          DataTable dtTemp = new DataTable();
47
48          dtTemp.Columns.Add("Логін", typeof(string));
49          dtTemp.Columns.Add("Пароль", typeof(string));
50          dtTemp.Columns.Add("Адміністратор", typeof(bool));
51          dtTemp.Columns.Add("Дозволити завантаження", typeof(bool));
52          dtTemp.Columns.Add("Читання із USB", typeof(bool));
53          dtTemp.Columns.Add("Запис на USB", typeof(bool));
54          dtTemp.Columns.Add("Заборонені процеси", typeof(string));
55          dtTemp.Columns.Add("Заборонені ресурси", typeof(string));
56          Users.AllowUserToAddRows = false;
57
58          Users.DataSource = dtTemp;
59
60          foreach (AdministratorSettings.User t in AdministratorSettings.GetAllUsers())
61          {
62              DataTable table = (DataTable)Users.DataSource;
63
64              DataRow _row = dtTemp.NewRow();
65
66              _row["Логін"] = t.UserLogin;
67              _row["Пароль"] = t.UserPassword;
68              _row["Адміністратор"] = t.UserIsAdmin;
69              _row["Читання із USB"] = t.AllowRead;
70              _row["Дозволити завантаження"] = t.AllowDownloads;
71              _row["Запис на USB"] = t.AllowWrite;
72
73              _row["Заборонені процеси"] = string.Join(",", t.RestrictedProcesses.ToArray());
74              _row["Заборонені ресурси"] = string.Join(",", t.RestrictedWebPages.ToArray());
75              table.Rows.Add(_row);
76              table.AcceptChanges();
77          }
78      }
```

Рис.3.6 Фрагмент файлу AdminPanel.cs

Також в даному файлі відбувається ініціалізація таких методів, що відповідають за:

- зупинку проксі та перезавантаження ПК для того щоб налаштування Firewall для нового користувача увійшли в дію;
- показ історії веб-ресурсів які переглядав користувач;
- зміну користувача;
- завершення роботи ПЗ.

Наступним етапом є створення файлу AdministratorSettings.cs (Додаток Г) (рис. 3.7). В даному файлі описується структура із налаштуваннями

адміністратора для кожного окремо взятого користувача. В блоці перевіряємо чи ініціалізований клас AllUsers.

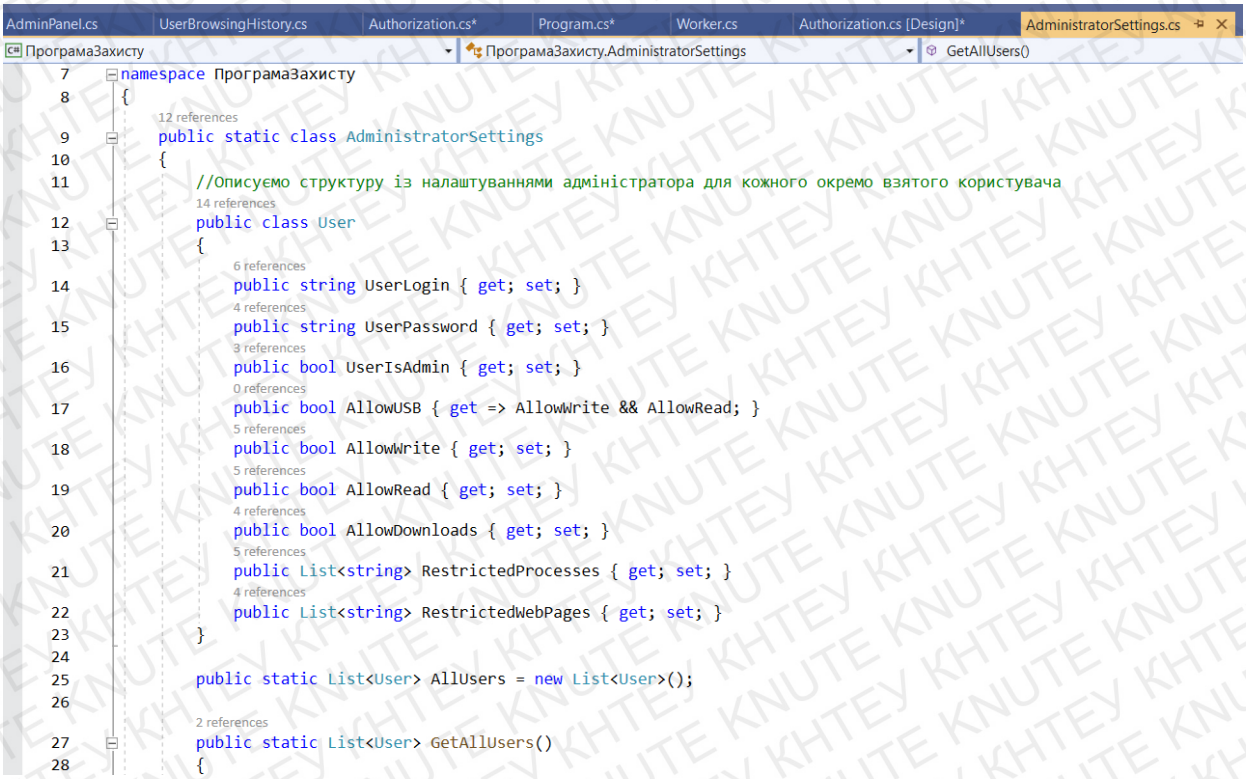


Рис.3.7 Фрагмент файлу AdministratorSettings.cs

Для передачі структурованої інформації між функціональними частинами програми використовуємо JASON – спеціальний текстовий формат для обміну даними, що оснований на JavaScript. Проте не зважаючи на його походження, формат вважається незалежним від JS і може використовувати практично з будь-якою мовою програмування. Формат дає змогу описувати об'єкти та інші структури даних [33].

Дані щодо користувачів, вказані адміністратором, записуються у файл adminsettings у вигляді JSON-об’єктів. Далі при додаванні користувачів метод SetUserSetting перевірятиме їх наявність у даній колекції і в разі відсутності записуватиме їх у файл adminsettings (рис.3.8).

Також у класі AdministratorSettings реалізуємо метод для видалення користувачів. За допомогою класу JsonConvert, що забезпечує конвертацію між .NET типами та JSON типами. переписуємо та оновлюємо JSON -файл.

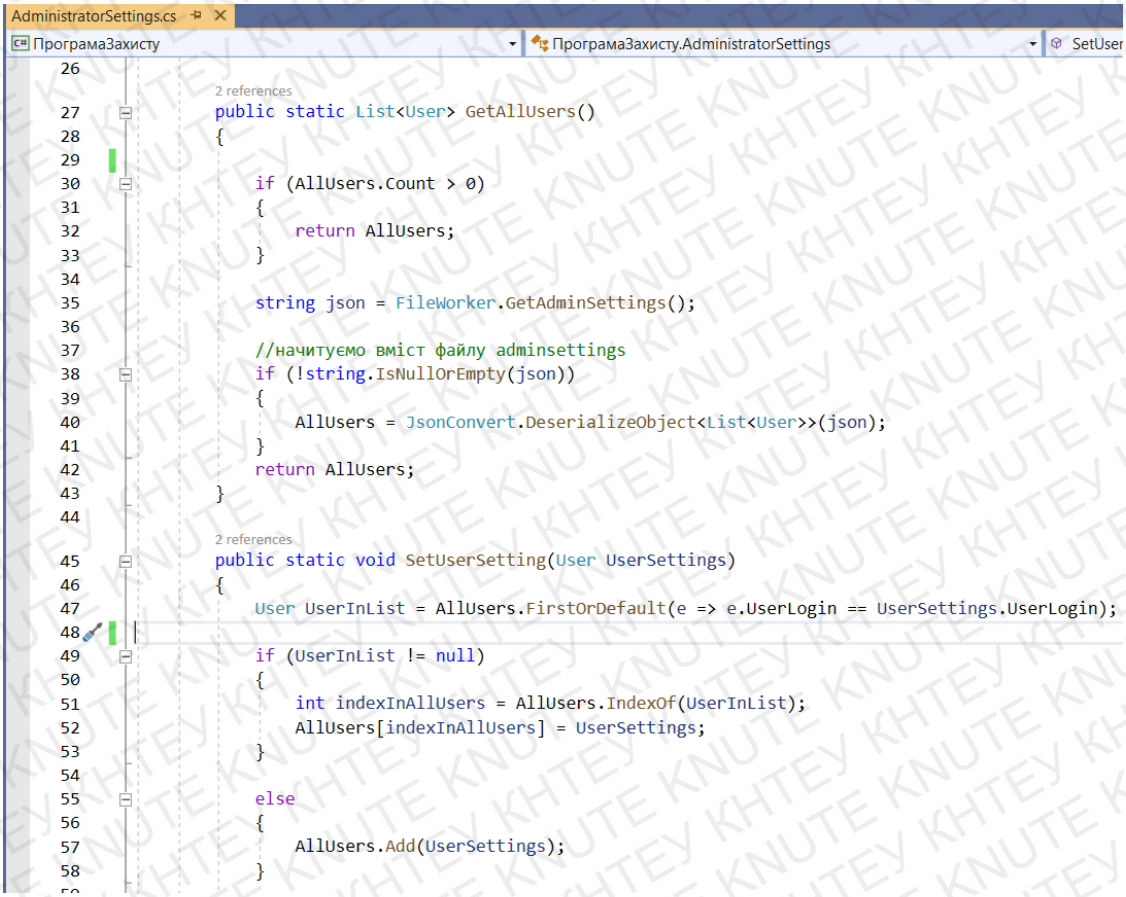


Рис.3.8 Фрагмент файлу AdministratorSettings.cs

Наступним варто розглянути файл під назвою Worker.cs в якому реалізовано більшість функціональних методів, які згодом адміністратор налаштовує конкретно під кожного користувача. В даній ділянці коду ініціалізовано 4 методи, що дозволяють або ж забороняють читання/запис даних на зовнішні накопичувальні пристрої з метою зменшення імовірності зараження локальної мережі шкідливим програмним забезпеченням. Методи працюють за допомогою створення файлів реєстру для кожного з 4-ьох методів. Один з методів представлений на рис. 3.9. Файли реєстру зберігаються на диску С та редагуються автоматично щоразу при зміні користувача або при вході в програму. Це пов'язано з унікальними налаштуваннями безпеки для кожного абонента.

Дані методи зменшують імовірність проникнення на ПК троянських програм, вірусів, бекдорів та іншого шкідливого ПЗ, що перешкоджає коректній роботі

комп'ютера, збирає конфіденційну інформацію, отримує доступ до приватних інформаційних систем та використовує продуктивні ресурси ПК.

Рис.3.9 Фрагмент файлу Worker.cs

В даному файлі також реалізовано метод видалення заборонених процесів (рис. 3.10). Його реалізовано наступним чином:

- повертаємо список процесів, які необхідно завершити з класу Authorization;
- отримуємо список усіх запущених процесів за допомогою системного класу Process з простору імен System.Diagnostics. Цей клас дозволяє управляти вже запущеними процесами, а також запускати нові. В даному класі визначено ряд властивостей і методів, що дозволяють отримувати інформацію про процеси і керувати ними;
- за допомогою знайомого операторів if та try/catch порівнюємо ім'я забороненого процесу із загальним списком запущених процесів та якщо знаходимо збіг завершуємо виконання процесу за допомогою методу kill.

Якщо виникає виключення та з певної причини не вдається завершити процес – показуємо користувачу відповідне повідомлення.

Рис.3.10 Фрагмент файлу Worker.cs

Наступним є метод для перезавантаження ПК який реалізується за допомогою створення нового екземпляру класу ProcessStartInfo. Даний клас визначає набір значень, які використовуються при запуску процесу. У нашому випадку клас викликає запуск командного рядка Windows та передає у нього перелік команд виконання яких необхідне для перезавантаження ПК.

Останнім реалізованим у даному файлі є метод AppClose, створений для правильного та безпечного закриття додатку. За допомогою методу EndAll відбувається очищення пам'яті, яку використовує об'єкт. За допомогою методу Stop зупиняється робота проксі-контролера, задля наступних коректних налаштувань. В разі вимкнення живлення ПК без виходу з даної програми метод Exit повідомить користувача про незавершений процес та порекомендує безпечний вихід з програми.

Вартим уваги є розгляд файлу FileWorker.cs. У класі FileWorker реалізовуються методи начитки та запизу даних із файлів після чого вони повертатимуться у контекст класу із якого вони були викликані (рис.3.11). Контекст класу – це область видимості, і, отже, поза нею ці методи існувати не можуть. StreamWriter і StreamReader-класи, призначені для запису і начитки контенту із файлів десеріалізуватимуться, тобто відновлять початковий стан структури даних із бітової послідовності із json-рядка.

Рис.3.11 Фрагмент файлу FileWorker.cs.

Найоб'ємнішим та найважчим у реалізації є файл Firewall.cs. В даному файлі ми підключаємо та налаштовуємо бібліотеку Titanium Web Proxy, яка є зручним інструментом для реалізації Firewall.

В даному файлі ініціалізуємо колекцію із дозволеними для завантаження типами веб-файлів та реалізуємо структури для запису історії браузера. Методом ProxyController начитуємо історію браузера, запускаємо запис історії браузера по розкладу (раз у 10 секунд), ініціалізуємо проксі-сервер, через який проходитимуть

усі з'єднання(new ProxyServer) та перехват виключень, що можуть з`явитись під час його роботи (рис.3.12).

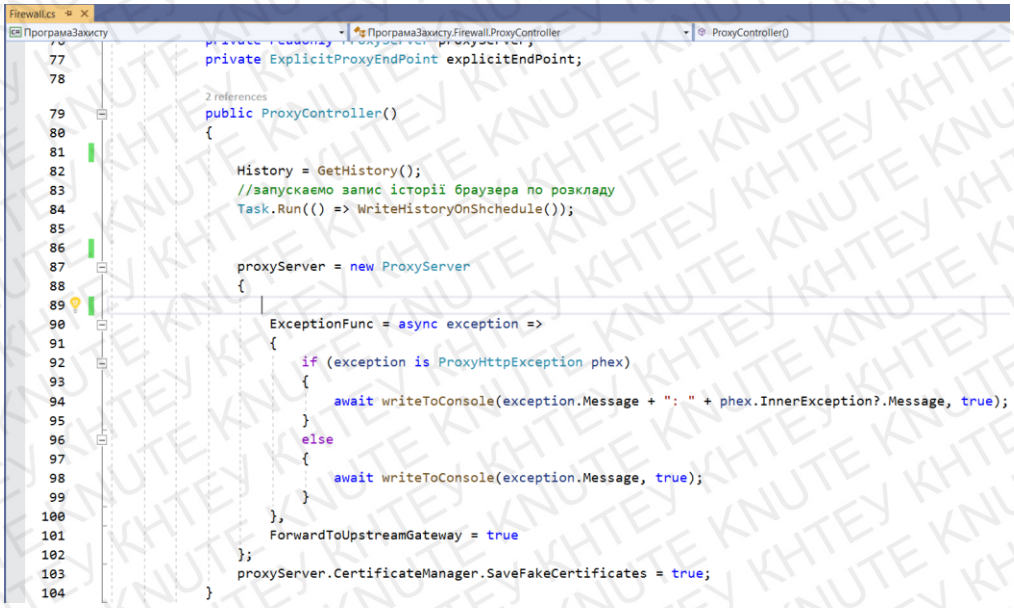


Рис.3.12 Фрагмент файлу Firewall.cs

Використовуючи метод StartProxy створюємо та встановлюємо проксі за вказаною адресою (ExplicitProxyEndPoint(IPAddress.Parse("127.0.0.2"), 8000)) (рис.3.13). Метод Stop слугує для зупинки проксі, а також для неопціонального видалення збережених сертифікатів, які були надані ОС Windows для того, щоб ваш веб-переглядач міг використовувати їх для спілкування з веб-сайтами [34].

Рис.3.13 Фрагмент файлу Firewall.cs

Метод Check створений для перевірки завантажень веб-ресурсів, які виконує користувач. Реалізуємо подію (onResponse), яка виникає при відповіді клієнту. Визначаємо розширення файлу яке намагається завантажити абонент. У деяких випадках, коли файл формується сервером пробуємо дістати його із заголовку Content-Disposition. В інших випадках завантажуємо html-код сторінки (рис. 3.14). Шукаємо у нього тег title, та якщо він наявний і statusCode==200 тоді за допомогою арифметичних операцій дістаємо value тега title. Даний тег служить заголовком для історії перегляду веб-ресурсів.

```
207             //пробуємо взяти html-код сторінки, котру намагається завантажити клієнт
208             string html = await e.GetResponseBodyAsString();
209
210             if (!string.IsNullOrEmpty(html))
211             {
212
213                 string startTitleTag = "<title>";
214                 int indexOfTitle = html.IndexOf(startTitleTag);
215
216                 if (indexOfTitle != -1 && e.HttpClient.Response.StatusCode == (int)HttpStatusCode.OK)
217                 {
218                     int endTitleTag = html.IndexOf("</title>", indexOfTitle + startTitleTag.Length);
219                     string titleTagValue = html.Substring(indexOfTitle + startTitleTag.Length,
220                         endTitleTag - indexOfTitle - startTitleTag.Length);
221
222
223
224                     History.Add(item: new BrowserHistory
225                     {
226                         PageTitle = titleTagValue,
227                         Url = e.HttpClient.Request.RequestUri.AbsoluteUri,
228                         VisitTime = DateTime.Now
229                     });
230                 }
231             }
```

Рис.3.14 Фрагмент файлу Firewall.cs

Реалізуємо заборону завантажень для браузерів. Перевіряємо, що бажає завантажити користувач. Якщо розширення файлу не входить у визначений нами перелік дозволених розширень, забороняємо його завантаження та попереджаємо користувача про відмову у завантаженнях. Реалізація даного методу зменшує імовірність зараження ПК шкідливим ПЗ, яке завантажується разом зі звичайним ПЗ або замість нього, а також підвищує надійність локальної мережі установи, що є важливою інформаційною системою державного значення.

За допомогою методу WriteHistoryOnShchedule кожні 10 секунд пишемо

Історію до файлу, а згодом, для зручності читання, сортуємо масив.

UserBrowsingHistory.cs файл – необхідний для побудови історії відвіданих користувачем веб-ресурсів. Ініціалізуємо подію (UserBrowsingHistory_Load) яка виникає при завантаженні форми з історією з налаштувань адміністратора (рис.3.15). Будуємо так звану таблицю, у якій зберігатимемо час візиту, назву сторінки та url - стандартизовану адресу певного ресурсу, що включає в себе назву протоколу доступу і, власне, шлях до ресурсу, формат якого залежить від схеми доступу.

UserBrowsingHistory.cs

ПрограмаЗахисту

ПрограмаЗахисту.UserBrowsingHistory

```
3 references
13  public partial class UserBrowsingHistory : Form
14  {
        1 reference
15      public UserBrowsingHistory()
16      {
17          InitializeComponent();
18      }
19      //Подія, яка виникає при завантаженні форми із історією
        1 reference
20      private void UserBrowsingHistory_Load(object sender, EventArgs e)
21      {
22          DataGridViewColumn VisitTime = new DataGridViewColumn()
23          {
24              HeaderText = "Час візиту",
25              Name = "VisitTime",
26              CellTemplate = new DataGridViewTextBoxCell(),
27              AutoSizeMode= DataGridViewAutoSizeColumnMode.Fill
28          };
29
30          DataGridViewColumn PageTitle = new DataGridViewColumn()
31          {
32              HeaderText = "Сторінка",
33              Name = "PageTitle",
34              CellTemplate = new DataGridViewTextBoxCell(),
35              AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill
36          };
37
38          DataGridViewLinkColumn Url = new DataGridViewLinkColumn()
39          {
40              HeaderText = "Url",
41              Name = "Url",
42              DataPropertyName="Url",
43              AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill
44          };
```

Рис. 3.15 Фрагмент файлу UserBrowsingHistory.cs

Для створення інтерфейсу модулю використовуємо Windows Forms API, що відповідає за графічний інтерфейс користувача і є частиною Microsoft .NET Framework. Даний API вбудований у Visual Studio 2019, що полегшує та оптимізує

створення дизайну. Так як форми є основною частиною програми, необхідно приділити особливу увагу їх зовнішньому вигляду і функцій. В кінцевому рахунку форма являє собою чистий аркуш, який розробники оснащують елементами управління, формуючи призначений для користувача інтерфейс, і кодом для роботи з даними [35].

Користувацький інтерфейс складається з двох вікон – авторизації та вікно налаштувань. Апробація даного модулю на ПК установи одного з відділів пройшла успішно. Перше вікно (рис.3.16), розроблено за допомогою застосунків Visual Studio 2019, виглядає наступним чином:

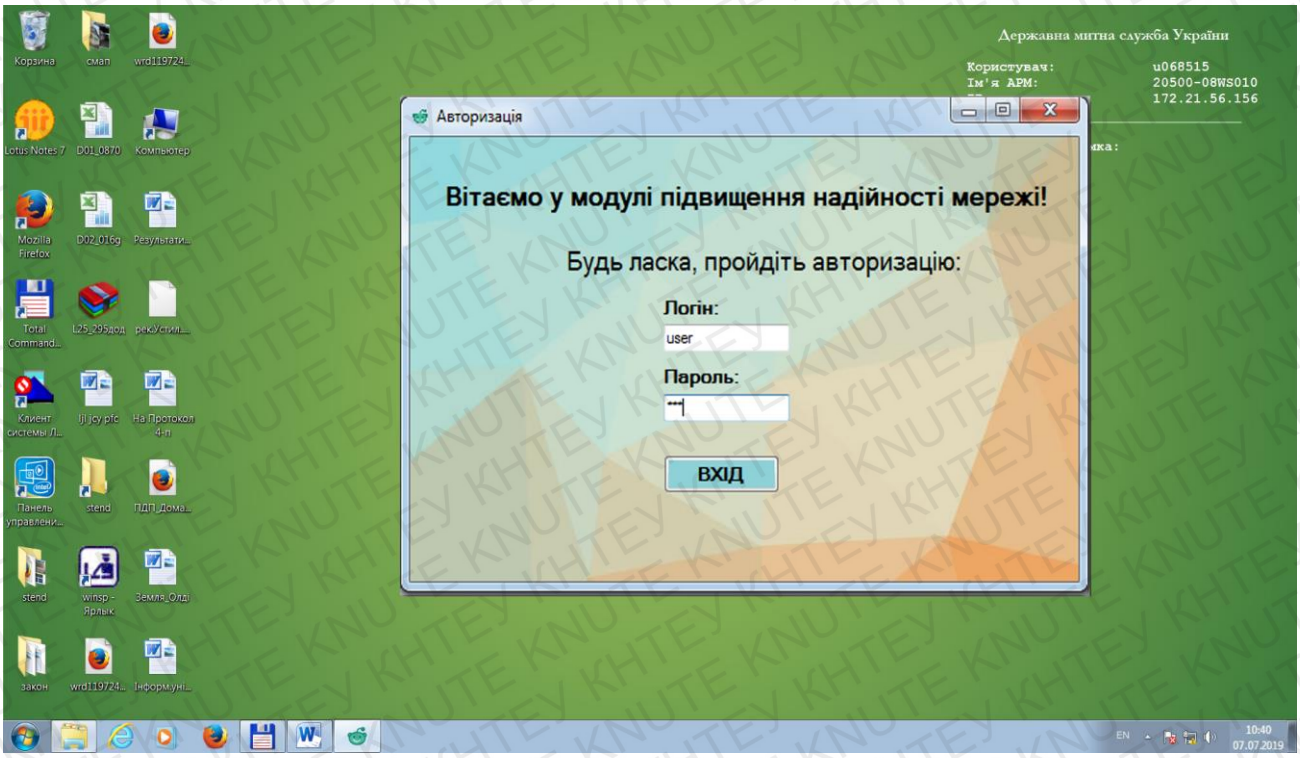


Рис. 3.16. Вікно авторизації програмного модулю

Вікно налаштувань адміністратора (рис.3.17) реалізовано аналогічним до вікна авторизаціїї засобом та включає у себе перелік усіх функціональних можливостей, вказаних у технічному завданні. Для того щоб налаштування щодо зовнішніх накопичувальних пристроїв вступили в дію необхідно перезавантажити ПК для коректного внесення змін в реєстр. При вході у програму користувач повинен погодитись з даними змінами.

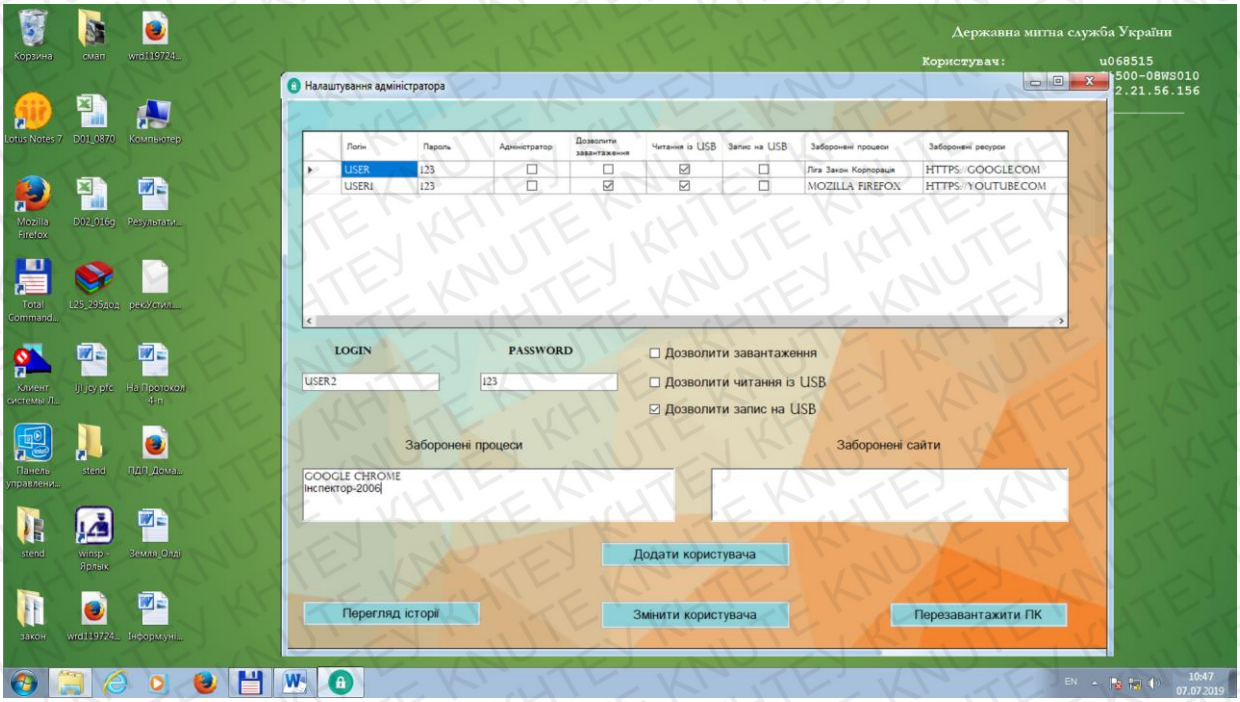


Рис. 3.17 Вікно налаштувань адміністратора

У адміністратора є можливість переглядати історію з фалу history, що знаходиться за адресою \ПрограмаЗахисту \bin \Debug або ж відкривати ці дані через програму (рис):

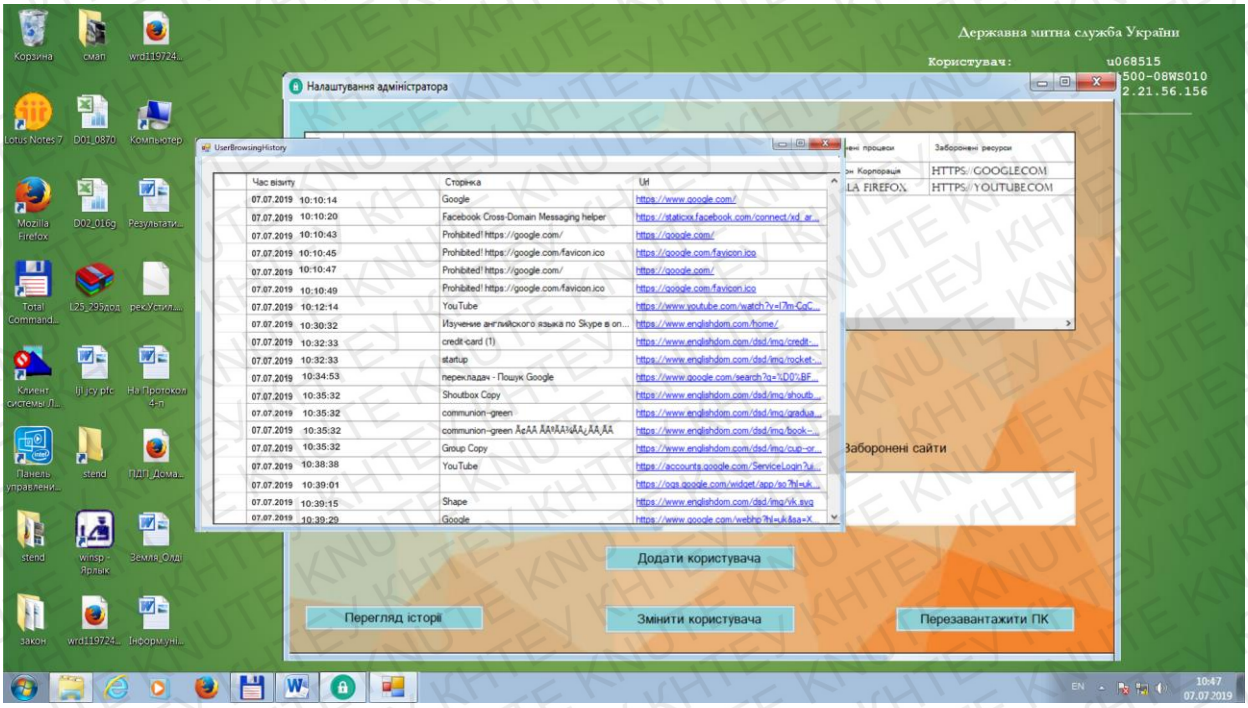


Рис.3. 18 Вікно історії відвіданих веб-ресурсів

Наступним чином виглядає відображення трея даного ПЗ в області сповіщень:

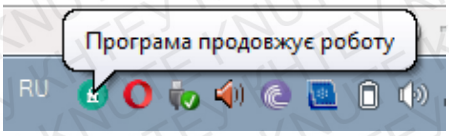


Рис.3.19

3.3 Тестування та відлагодження програмного забезпечення

Як показує досвід програмування, незважаючи на ретельне проведення етапів проектування та використання сучасних технологій програмування, не вдається розробити повністю безпомилкову програму. Основними активними методами виявлення і усунення несправностей є тестування і налагодження. Під тестуванням розуміється факт наявності помилок, а під відлагодженням – інформація де саме допущено помилку [36].

Тестування - процес виявлення наявних в програмі помилок, а налагодження - процес їх усунення.Засоби тестування Visual Studio допомагають в розробці і підтримці високих стандартів якості коду. Вікно «Test Explorer» допомагає розробникам створювати і запускати модульні тести, а також керувати ними. Можна використовувати платформу для виконання модульних тестів Microsoft або одну з декількох сторонніх платформ, в тому числі платформи з відкритим вихідним кодом.

Visual Studio є розширюваною системою і дозволяє використовувати сторонні адаптери модульного тестування, такі як NUnit і xUnit.net. Крім того, функція клонів коду забезпечує високу якість програмного забезпечення, допомагаючи визначити блоки семантично подібного коду, які є засобами для спільного виправлення помилок або рефакторінгу.

Виконання модульних тестів необхідне для забезпечення працездатності коду, визначення потрібного обсягу протестованого коду, а також для виявлення помилок і збоїв, перш ніж з ними зіткнуться користувачі. Модульні тести варто виконувати регулярно, щоб забезпечити правильну роботу коду. Для написання тесту необхідно

у готове рішення додати новий проект. В проекті модульних тестів потрібно додати посилання на проект, який необхідно протестувати. Наступним кроком є додавання коду у тестовий проект. Коли усі підготовчі роботи виконано запускаємо «Test Explorer» та обираємо run all, тобто запускаємо усі наявні тести (рис.3.20). Зелений прапорець означає, що тест пройдено. Червоний хрест вказує на збій у виконання тесту.

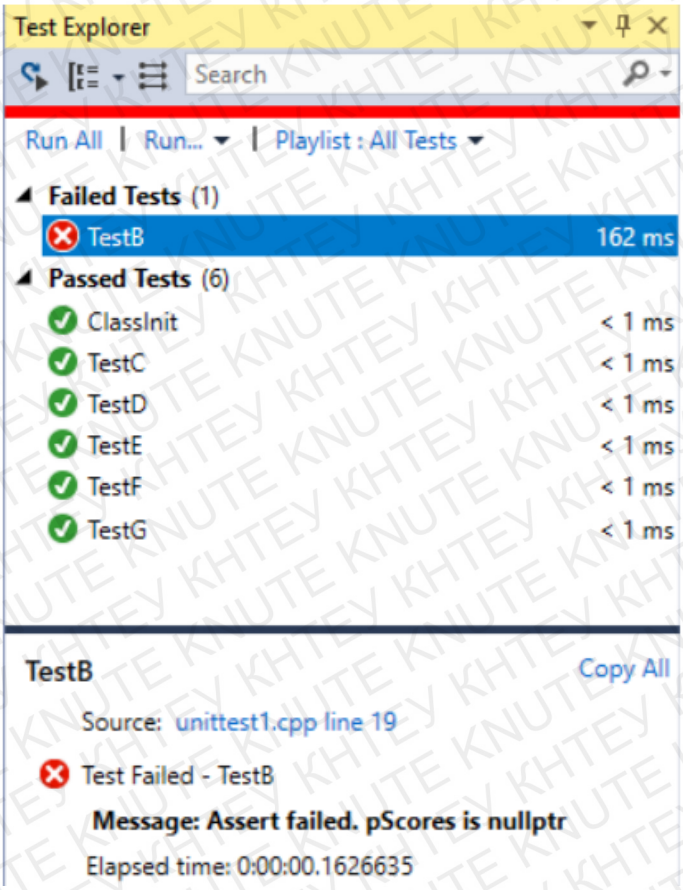


Рис. 3.20 Тестування програмного модулю

Тестування програмних систем не дає тих гарантій їх якості, яких вимагає практика, через сильну недетермінованість таких систем. Шляхом тестування неможливо встановити відсутність помилок на всіх виконаннях недетермінованої системи, що визначаються одним і тим же початковим станом. Математичне обгрунтування надійності програм засновано на формальній верифікації, яка за допомогою формальних доказів дозволяє встановлювати присутність необхідних властивостей програми для всіх допустимих цією програмою виконань. Верифікація

ПЗ є майже єдиним способом аналізу необхідних властивостей ПЗ щодо заданих передумов.

Фундаментом верифікації є логіка, формальна мова логіки, а також формальні моделі та методи. Верифікація програм зазвичай зводить аналіз їх властивостей до доведення істинності умов коректності у вигляді логічних формул. Основним підходом до обґрунтування коректності, є підхід, при якому досліджується (верифікується) не як така програма, а її специфікація (формальна модель). Представницькими з числа широко розповсюджених формальних моделей є мережі Петрі, взаємодіючі послідовні процеси Хоара, тимчасові (темпоральні) логіки [37].

Незважаючи на те, що результати застосування верифікації значно потужніші, вирішальне практичне значення для обґрунтування надійності послідовних програм має тестування. Це пов'язано з тим, що верифікація є дуже трудомістким процесом і вимагає великих витрат, які часто бувають не виправдані.

Відлагодження програми - це спеціальний етап в розробці програми, що складається з виявлення та усунення програмних помилок, факт існування яких вже встановлено. Програмні помилки, як правило, діляться на три види:

- синтаксична помилка - неправильне вживання синтаксичних конструкцій;
- семантична помилка - порушення семантики тієї чи іншої конструкції, наприклад передача функції параметрів, які не відповідають її аргументам;
- логічна помилка - порушення логіки програми, що приводить до невірного результату. Це найбільш важкий для "вилову" тип помилки, бо подібного роду помилки, як правило, криються в алгоритмах і вимагають ретельного аналізу і всебічного тестування.

Відлагодження програми відбувається з допомогою спеціального засобу – налагоджувача. Даний налагоджувач може бути зовнішнім, проте зручно використовувати уже вбудований у Visual Studio 2019 Debugger, який реалізує багато ефективних функцій для відлагоджування проекту. Зручним він є і тому, що дозволяє виправляти помилки та перекомпілювати програму без її перезапуску.

Для налагодження потрібно запустити додаток з Debugger, підключеним до процесу додатка. Для цього найчастіше використовується клавіша F5 (Налагодження > Почати налагодження). Однак перед цим обов'язково потрібно встановити точку зупинки (рис.3.21). Точки зупинки - це один з найпростіших і важливих компонентів надійного налагодження. Точка зупину вказує, де Visual Studio слід призупинити виконання коду, щоб можна було перевірити значення змінних або поведінку пам'яті чи виконання гілки коду.

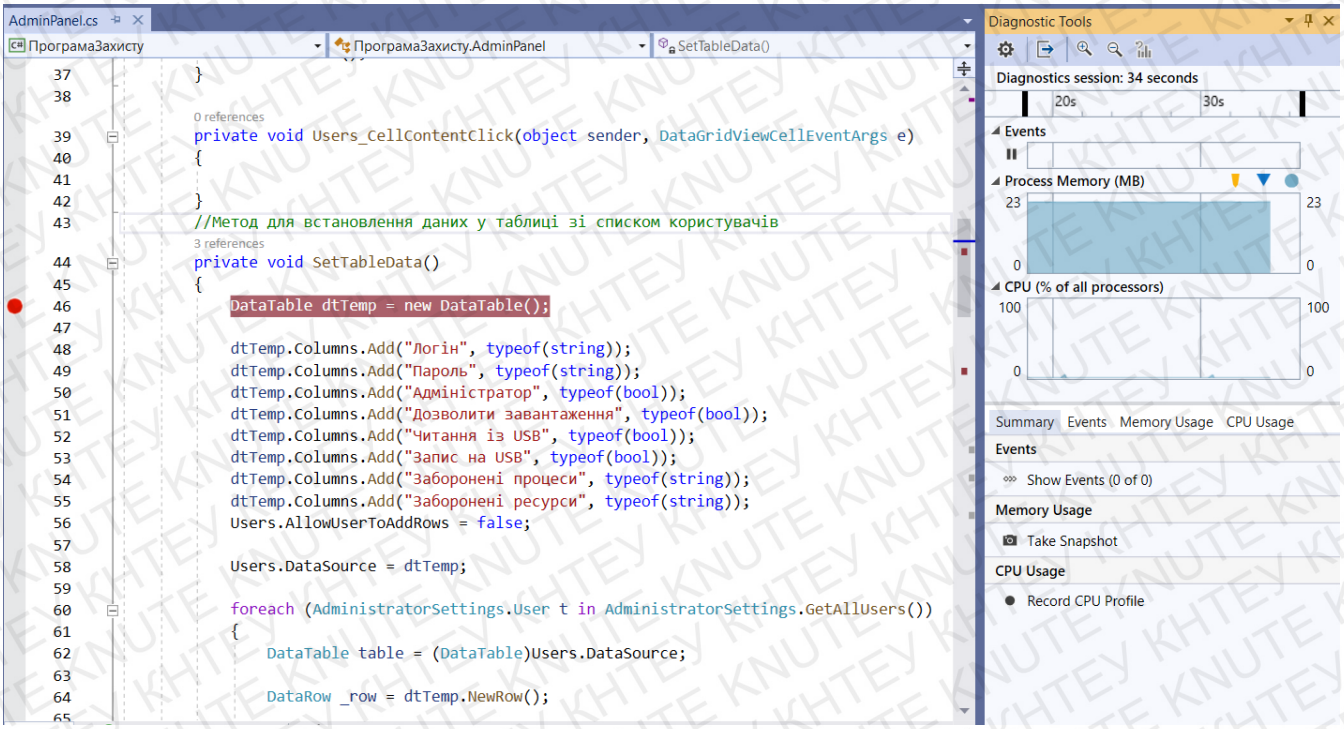


Рис. 3.21 Встановлення точки зупинки

За допомогою налагоджувача є можливість перевірити стан змінних. Засіб налагодження VS рівня вихідного коду дозволяє відстежувати, встановлювати або змінювати значення змінних в процесі виконання коду, встановлювати і видаляти контрольні точки або умови зупинки і т.д. Функції, що дозволяють перевіряти змінні або ж об’єкти, є одними з найбільш корисних в налагоджувачі (рис. 3.22). Реалізовувати цю задачу можна різними способами. Часто при спробі виконати налагодження проблеми користувачу необхідно з'ясувати, чи зберігаються в змінних значення, які потрібні в певному стані додатку.

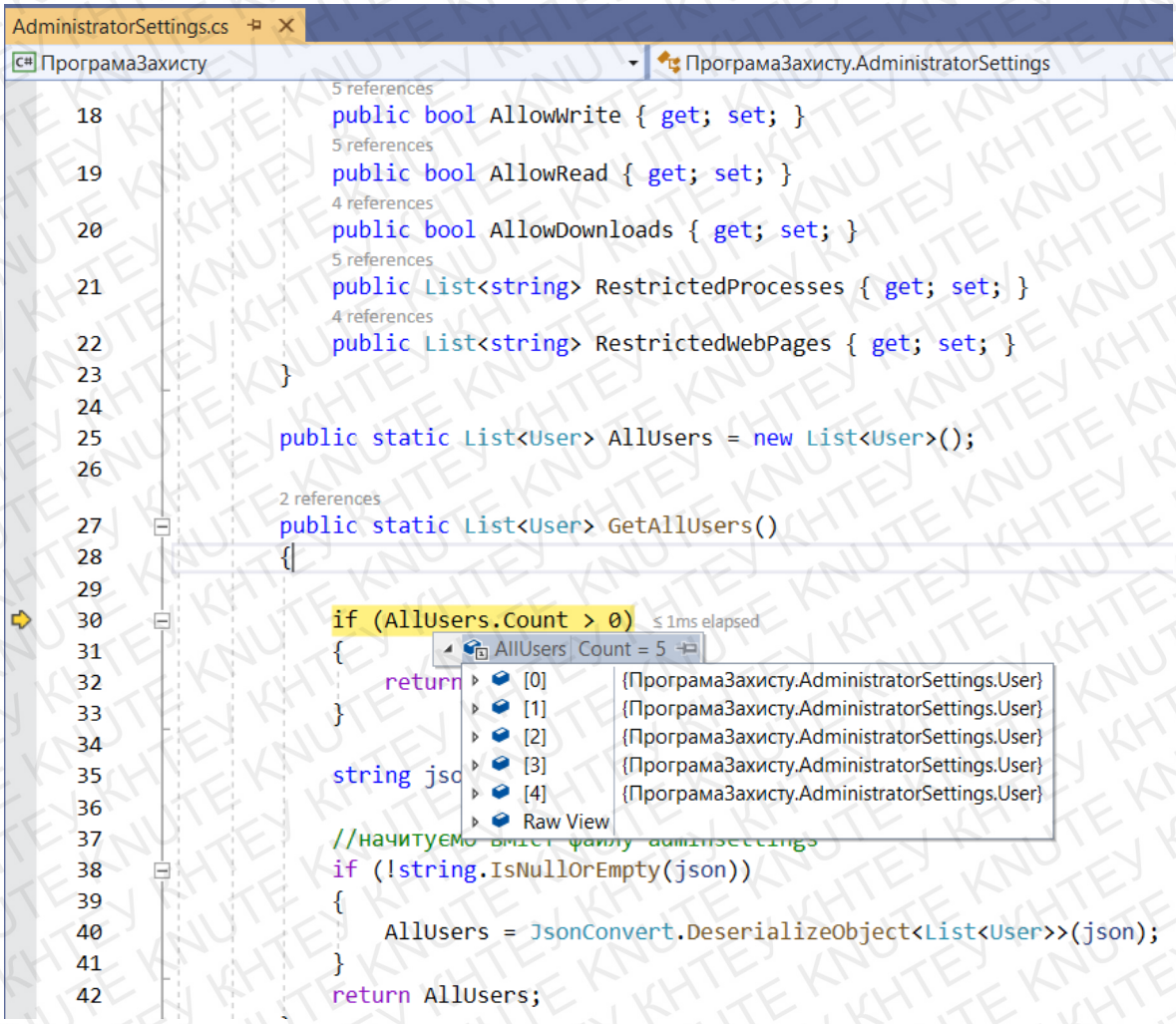


Рис. 3.22 Реалізація функції, що показує к-сть об’єктів в масиві

ВИСНОВКИ ДО РОЗДІЛУ 3

В даному розділі після проектування, програмування, тестування та налагодження програмного рішення підвищення надійності комп’ютерної мережі, а також вровадження даного модулю на декількох ПК мережі установи можна зробити такі висновки:

- доцільним є проектування алгоритму ПЗ, що значно оптимізує розробку додатка та зменшує кількість помилок, що виникають під час його реалізації;
- використання ICP Microsoft Visual Studio 2019 є зручним та комплексним рішенням при створенні подібного ПЗ, ICP адже містить у собі такі необхідні

засоби як: спеціальний редактор коду, компілятор, Test Explorer, Debugger, Windows Form та інші.

- впровадження даного модулю на обчислювальних машинах локальної мережі установи можливе, враховуючи технічні характеристики пристроїв та доцільне беручи до уваги особливості роботи, яку виконують користувачі мережі та інформації, яка передається даною мережею.

ВИСНОВКИ ТА РЕЗУЛЬТАТИ

В ході виконання даної дипломної роботи, аналізу програмного та технічного забезпечення локальної мережі установи, пошуку необхідних методів дослідження, визначення функціоналу ПЗ, проектування, розробки, тестування та апробації програмного модулю підвищення надійності мережі отримано висновки:

1. Програмне та технічне забезпечення інформаційної системи митниці є застарілим та потребує модернізації для оптимізації роботи працівників установи, а також для підвищення безпеки інформації, що оброблюється даною системою.

2. Вплив людського фактору на коректну, налагоджену та безвідмовну роботу апаратних пристрої є значним та потребує зменшення у зв'язку з небезпекою, яку може у собі нести, шляхом контролю та обмеження дій користуваців ПК.

3. Розробка програмного модулю, що підвищить надійність локальної мережі установи є актуальною у час розвитку інформаційного суспільства та інтеграції комп'ютерних технологій у все більшу кількість сфер життя людини.

4. Правильний вибір методів дослідження спрощує та скорочує час його виконання.

5. Проектування ПЗ та сворення алгоритму зменшує кількість помилок, що виникатимуть під час реалізації та тестування продукту.

7.Реалізації подібного продукту вимагає використання якісного інтегрованого середовища розбробки, що містить у собі взаємозалежні засоби програмування, що використовуються на різних стадіях життєвого циклу ПЗ.

8. Тестування та налагодження програми є одним з основних етапів розробки модулю, що дозволяють виявити помилки до масового використання програмного

продукту.

9. Апробація є важливою передексплуатаційною перевіркою, що дозволяє оцінити ефективність ПЗ та перевірити сумісність модулю з апаратними засобами з різними технічними складовими.

Результатами даного дослідження є програмний модулю, що призначений для контролю та обмеження дій користувачів ПК. Адміністратор має змогу додавати користувачів, присвоювати на налаштовувати їх ролі. У ПЗ наявна можливість заборони завантажень з всесвітньої павутини задля зменшення імовірності зараження ПК шкідливим ПЗ, а отже і втрати або розголошення важливої та цінної інформації, незаконного втручання до окремих ланок спільного доступу, таких як бази даних, директорії з конфіденційною інформацією, мережевих файлових систем.

Адміністратор має можливість створювати список заборонених інтернет-ресурсів, а також наявних на ПК процесів для кожного окремого користувача. Існує можливість налаштування безпеки щодо зовнішніх накопичувальних пристроїв – дозвіл/заборона запису або читання. Для контролю відвіданих веб-ресурсів реалізована можливість перегляду історії через програму.

У майбутньому є можливість збільшення методів, що обмежуватимуть доступ до КМ. Окрім цього, можна розширити фукції налаштування безпеки зовнішніх пристроїв. Наприклад, встановлення паролів на важливі службові пристрої і т.д.

В цілому створення подібного продукту для державних установ країни є досить перспективним, адже в еру ІТ та всебічного розповсюдження комп’ютерної техніки людство зіштовхнулось з новими загрозами для інформаційної безпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Олещенко Л.М. Організація комп'ютерних мереж: конспект лекцій [Електронний ресурс]: навч. посіб. — Київ: КПІ ім. Ігоря Сікорського, 2018. — 225с.

2.Лучкова А.В. Особливості побудови і використання сучасних корпоративних комп'ютерних мереж - Вінниця: ВНТУ, 2016. — 4с.

3. Абрамов В.О., Клименко С.Ю. Базові технології комп'ютерних мереж: навч. посіб. — Київ: Видавнича група «АТОПОЛ», 2014. — 262 с.

4. Наказ України "Про затвердження Інструкції з організації митного контролю та митного оформлення міжнародних експрес-відправлень, що переміщуються (пересилаються) через митний кордон України" [Електронний ресурс]: Державна фіскальна служба, 2012. – Режим доступу: https://zakon.rada.gov.ua/laws/show/z1081-07 .

5. Шорошев В. В. Теоретичні і практичні аспекти організації і побудови архітектури захищених комп'ютерних систем: монографія – Київ: ДУІКТ, 2011. – с.257.

6.Офіційний сайт розробника DELL. Режим доступу: https://www.dell.com/ua/business/p/enterprise-products?cat=enterprise-products&view=standard&isredir=true/ Дата доступу: 24.08.19.

7.Шляхи зараження ПК шкідливими програмами[Електронний ресурс], – 2015, Режим доступу: https://zillya.ua/shlyakhi-zarazhennya-pk-shkidlivimi-programami Дата доступу: 24.08.19.

8. Ознаки зараження комп'ютерними вірусами та як цьому запобігти[Електронний ресурс], – 2016, Режим доступу: http://safe-city.com.ua/oznaky-zarazhennya-komp-yuternymy-virusamy-ta-yak-tsomu-zapobigty/ Дата доступу: 24.08.19.

9. Проксі-сервер [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BA%D1%81%D1%96-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80 Дата доступу: 24.08.19

10. Микитишин А.Г., Митник М.М., Стухляк П.Д., Пасічник В.В. Комп'ютерні мережі : Навч. посіб. – Львів: «Магнолія 2006», 2013. – 256 с.

11. Манохін О. Г., Манохіна Л. В., Соловко Г. Ф., Максюта О. В. "Експерт – ДГЦУ – Митниця" // Коштовне та декоративне каміння, 2011. – № 2. – 35-37 с. – Режим доступу: http://nbuv.gov.ua/UJRN/Ktdk_2011_2_11

12. Сидоров М.О. Методи і засоби створення і супроводження програмного забезпечення великих розподілених комп'ютеризованих інформаційних систем //Вісник НАУ. – Київ: НАУ. – № 4, 2014. – 30-37 с.

13. Хоменко В. А. Метод оцінювання повторно використовуваних компонентів програмного забезпечення // Наукові вісті НТУ «КПІ». – Київ: НТУ «КПІ». – № 2, 2012. – 102-107 с.

14. Гученко І.В. Засіб управління зручністю використання програмних продуктів // Інженерія програмного забезпечення . – Київ: НАУ. – № 4 – 2011. – 35-46 с.

15. Гнатовська Г.А. Технологія створення програмних продуктів: конспект лекцій. – Одеса: ОДЕУ, 2015 – 97с.

16. Integrated Development Environment [Електронний ресурс], Режим доступу: https://press.rebus.community/programmingfundamentals/chapter/integrated-development-environment/#footnote-154-1 Дата доступу: 24.08.19.

17. Гудлиф П. Ремесло программиста. Практика написания хорошего кода. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 704 с.

18. David Salter, Rhawi Dantas NetBeans IDE 8 Cookbook. – Birmingham: Packt Publishing, 2014. – 386 pg.

19. NetBeans Platform Learning Trail [Електронний ресурс]. — Режим доступу: https://netbeans.org/kb/trails/platform.html Дата доступу: 24.08.19.

20. Mark Dexter Eclipse And Java For Total Beginners : Companion Tutorial Document,

2007. — 45 pg. [Електронний ресурс]. — Режим доступу: http://eclipsetutorial.sourceforge.net/Total_Beginner_Companion_Document.pdf Дата доступу: 24.08.19.

21. Desktop IDEs [Електронний ресурс]. — Режим доступу: https://www.eclipse.org/ide/ Дата доступу: 24.08.19.

22. Biplab Kumar Modak C++ Application Development with Code::Blocks. – Birmingham: Packt Publishing, 2013. – 128 pg.

23. Тракимус Ю.В. Разработка консольных приложений с помощью Microsoft Visual Studio 2017: учебное пособие – Новосибирск: Изд-во НГТУ, 2018. – 64 с.

24. Добро пожаловать в интегрированную среду разработки Visual Studio [Електронний ресурс]. – 2019, Режим доступу: https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019 Дата доступу: 24.08.19.

25. Майо Д. Самоучитель Microsoft Visual Studio 2010. — Санкт-Петербург: БХВ-Петербург, 2011. — 464 с.

26. Swaroop C. H. A Byte of Python (Translated by Vladimir Smolyar) [Електронний ресурс]. – 2013, Режим доступу: http://wombat.org.ua/AByteOfPython/AByteofPythonRussian-2.01.pdf – 151 pg.

27. Сеттер Р. В. Изучаем JAVA на примерах и задачах: учебное пособие — Санкт-Петербург: Наука и Техника, 2016. — 240 с.

28. Bjarne Stroustrup The C++ Programming Language [Електронний ресурс]. – 2013, Режим доступу: https://anekihou.se/programming/2.%20intermediete.pdf – 1347 pg.

29. Роганов Е. А., Роганова Н. А. Программирование на языке Ruby: учебное пособие. — Москва:МГИУ, 2008.—56 с.

30. A Tour of the C# Language [Електронний ресурс]. – 2019, Режим доступу: https://docs.microsoft.com/uk-ua/dotnet/csharp/tour-of-csharp/

31.Технология разработки программного обеспечения [Електронний ресурс]. – 2019, Режим доступу: https://project.dovidnyk.info/index.php/home/tehnologiyarazrabotkiprogrammnogoobespecheniya/24-detal_noe_proektirovanie

32. Try-catch (C# Reference) [Електронний ресурс]. – 2015, Режим доступу: https://docs.microsoft.com/uk-ua/dotnet/csharp/language-reference/keywords/try-catch

33. JSON [Електронний ресурс]. – 2019, Режим доступу: https://uk.wikipedia.org/wiki/JSON

34. What are Root Certificates for Windows? [Електронний ресурс]. – 2015, Режим доступу: https://www.thewindowsclub.com/what-are-root-certificates-windows

35. Windows Forms [Електронний ресурс]. – 2017, Режим доступу: https://docs.microsoft.com/uk-ua/dotnet/framework/winforms/

36. Плаксин М. А. Тестирование и отладка программ для профессионалов будущих и настоящих — Москва: БИНОМ. Лаборатория знаний, 2015 – 170с.

37. Цыганенко В. Н. Технология программирования:конспект лекций. - Омск: ОГТУ, 2015 – 32 стр.

ДОДАТКИ

Додаток А

Код програмного модулю вихідного файлу Program.cs

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

namespace ПрограмаЗахисту
{
    static class Program
    {
        /// <summary>
        /// Головна точка входу для додатка
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            try
            {
                Application.Run(new Authorization_form());//Запускаємо в першу чергу авторизацію

            }
            catch (Exception ex)
            {
                using(System.IO.StreamWriter sw=new System.IO.StreamWriter("lastError.txt"))
                {
                    sw.Write(ex.ToString());
                }
                MessageBox.Show($"Виключення записане у файл lastError.txt \n {ex.ToString()}");
                Worker.AppClose();
            }
        }
    }
}
```

Код програмного модулю вихідного файлу Authorization_form.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;


namespace ПрограмаЗахисту
{
   public partial class Authorization_form : Form
   {
      private string login1;
      private string password1;

      private Firewall.ProxyController ProxyTestController = new Firewall.ProxyController();

      public static AdministratorSettings.User CurrentUser { get; set; }

      private IReadOnlyList<AdministratorSettings.User> Users =
AdministratorSettings.GetAllUsers();

      public Authorization_form()
      {
         InitializeComponent();
         login1 = "admin";
         password1 = "123";
      }

      private void button_authorize_Click(object sender, EventArgs e)
      {

         if (textBox_login.Text == login1 && password1 == textBox_pasword.Text)
         {
            this.Hide();

            new AdminPanel().Show();
            if (Firewall.proxyController != null)
            {
               Firewall.proxyController.Stop();
            }
            return;
         }

         //Інакше шукаємо вказаного користувача у списку Users
         AdministratorSettings.User _CurrentUser = Users.FirstOrDefault(r => r.UserLogin ==
textBox_login.Text);
         if (_CurrentUser != null && textBox_pasword.Text != _CurrentUser.UserPassword)
         {
            MessageBox.Show("Невірний пароль");
```

```
            return;
        }

        else if (_CurrentUser != null)
        {
            CurrentUser = _CurrentUser;

            if (!CurrentUser.UserIsAdmin)
            {
                //якщо користувач не адміністратор-приховуємо форму
                this.Hide();
                Firewall.proxyController.StartProxy();
                //встановлюємо обмеження на флеш-накопичувачі
                Worker.SetRestrictions(CurrentUser);
                notifyIcon1.Visible = true;
                notifyIcon1.BalloonTipText = "Програма продовжує роботу";
                notifyIcon1.ShowBalloonTip(1000);
                return;

            }
        }
        else
        {
            MessageBox.Show("Користувач із таким логіном відсутній!");
        }
    }

    //подія, яка виникає при закритті вікна
    private void Form2_FormClosing(object sender, FormClosingEventArgs e)
    {
        Worker.AppClose();
    }
    //подія, що виникає при завершенні роботи програми
    public void EndApplication(object sender, EventArgs e)
    {
        Worker.AppClose();
    }
    private void Form2_Load(object sender, EventArgs e)
    {
        //додаємо контексне меню для іконки у треї
        notifyIcon1.ContextMenuStrip = new ContextMenuStrip();
        //і додаємо туди елементи (1 аргумент назва елементу, 2-картинка, 3-подія, яка
виникає при натисканні на елемент)
        notifyIcon1.ContextMenuStrip.Items.Add("Вихід із програми", null, Exit);
        notifyIcon1.ContextMenuStrip.Items.Add("Змінити користувача", null, ChangeUser);
    }

    private void button_close2_Click(object sender, EventArgs e)
    {
        Worker.AppClose();
    }
```

Продовження додатку Б

```
        private void Exit(object sender, EventArgs e)
        {
            Worker.AppClose();
        }
        private void ChangeUser(object sender, EventArgs e)
        {
            new Authorization_form().Show();
        }
        private void textBox_pasword_TextChanged(object sender, EventArgs e)
        {
            textBox_pasword.PasswordChar = '*';
        }
        private void notifyIcon1_MouseDoubleClick(object sender, MouseEventArgs e)
        {
            new Authorization_form().Show();
            notifyIcon1.Visible = false;
        }
    }
}
```

Продовження додатку Б

Код програмного модулю вихідного файлу AdminPanel.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Windows.Forms;

namespace ПрограмаЗахисту
{
    public partial class AdminPanel : Form
    {
        public AdminPanel()
        {
            InitializeComponent();
        }

        //Метод для вибірки даних із полей при додаванні нового користувача
        private void UpdateOrAddUser_Click(object sender, EventArgs e)
        {

            AdministratorSettings.User User = new AdministratorSettings.User()
            {
                UserLogin = UserLogin_.Text,
                AllowDownloads = AllowDownloads.Checked,
                AllowRead = USBAllowRead.Checked,
                AllowWrite = USBAllowWrite.Checked,
                RestrictedProcesses = RestrictedProc.Text.Split(',').ToList(),
                RestrictedWebPages = RestrictedSites.Text.Split(',').ToList(),
                UserPassword = UserPassword_.Text
            };
            AdministratorSettings.SetUserSetting(User);
            SetTableData();
        }

        private void UserList_Load(object sender, EventArgs e)
        {
            SetTableData();
        }

        private void Users_CellContentClick(object sender, DataGridViewCellEventArgs e)
        {

        }
        //Метод для встановлення даних у таблиці зі списком користувачів
        private void SetTableData()
        {
            DataTable dtTemp = new DataTable();

            dtTemp.Columns.Add("Логін", typeof(string));
            dtTemp.Columns.Add("Пароль", typeof(string));
```

```
            dtTemp.Columns.Add("Адміністратор", typeof(bool));
            dtTemp.Columns.Add("Дозволити завантаження", typeof(bool));
            dtTemp.Columns.Add("Читання із USB", typeof(bool));
            dtTemp.Columns.Add("Запис на USB", typeof(bool));
            dtTemp.Columns.Add("Заборонені процеси", typeof(string));
            dtTemp.Columns.Add("Заборонені ресурси", typeof(string));
            Users.AllowUserToAddRows = false;

            Users.DataSource = dtTemp;

            foreach (AdministratorSettings.User t in AdministratorSettings.GetAllUsers())
            {
                DataTable table = (DataTable)Users.DataSource;
                DataRow _row = dtTemp.NewRow();

                _row["Логін"] = t.UserLogin;
                _row["Пароль"] = t.UserPassword;
                _row["Адміністратор"] = t.UserIsAdmin;
                _row["Читання із USB"] = t.AllowRead;
                _row["Дозволити завантаження"] = t.AllowDownloads;
                _row["Запис на USB"] = t.AllowWrite;

                _row["Заборонені процеси"] = string.Join(",", t.RestrictedProcesses.ToArray());
                _row["Заборонені ресурси"] = string.Join(",", t.RestrictedWebPages.ToArray());
                table.Rows.Add(_row);
                table.AcceptChanges();
            }
        }
        private void Users_CellEndEdit(object sender, DataGridViewCellEventArgs e)
        {
            if (e.RowIndex >= 0)
            {
                DataGridViewRow row = this.Users.Rows[e.RowIndex];
                AdministratorSettings.User _user = new AdministratorSettings.User()
                {
                    UserLogin = row.Cells[0].Value != null ? row.Cells[0].Value.ToString() : "error",
                    UserPassword = row.Cells[1].Value != null ? row.Cells[1].Value.ToString() : "error",
                    UserIsAdmin = row.Cells[2].Value != null ? Convert.ToBoolean(row.Cells[2].Value) :
false,
                    AllowRead = row.Cells[4].Value != null ? Convert.ToBoolean(row.Cells[4].Value) :
false,
                    AllowDownloads = row.Cells[3].Value != null ?
Convert.ToBoolean(row.Cells[3].Value) : false,
                    AllowWrite = row.Cells[5].Value != null ? Convert.ToBoolean(row.Cells[5].Value) :
false,
                    RestrictedProcesses = row.Cells[6].Value != null ?
row.Cells[6].Value.ToString().Split(',').ToList() : new List<string>(),
                    RestrictedWebPages = row.Cells[7].Value != null ?
```

Продовження додатку В

```
row.Cells[7].Value.ToString().Split(',').ToList() : new List<string>()
            };
            AdministratorSettings.SetUserSetting(_user);
            SetTableData();

        }
    }
    //Кнопочка для перезавантаження ПК
    private void button2_Click(object sender, EventArgs e)
    {
        Firewall.proxyController.Stop();
        Worker.RebootPC();
    }

    private void ShowBrowsingHistory_Click(object sender, EventArgs e)
    {
        new UserBrowsingHistory().Show();
    }

    private void AdminPanel_FormClosed(object sender, FormClosedEventArgs e)
    {
        Worker.AppClose();
    }

    private void UserChange_Click(object sender, EventArgs e)
    {
        this.Hide();
        new Authorization_form().Show();
    }
    private void Users_UserDeletingRow(object sender, DataGridViewRowCancelEventArgs e)
    {
        if (e.Row.Cells[0].Value != null)
        {
            string user = e.Row.Cells[0].Value.ToString();
            if (!string.IsNullOrEmpty(user))
            {
                string result = AdministratorSettings.DeleteUser(user);
                if (!string.IsNullOrEmpty(result))
                {
                    MessageBox.Show(result);
                }
            }
        }
        else
        {
            MessageBox.Show("Помилка! Ім'я користувача порожнє!");
        }
    }
  }
}
```

Продовження додатку В

Код програмного модулю вихідного файлу AdministratorSettings.cs

```
using System;
using System.IO;
using System.Linq;
using Newtonsoft.Json;
using System.Collections.Generic;

namespace ПрограмаЗахисту
{
    public static class AdministratorSettings
    {
        //Описуємо структуру із налаштуваннями адміністратора для кожного окремо взятого користувача
        public class User
        {
            public string UserLogin { get; set; }
            public string UserPassword { get; set; }
            public bool UserIsAdmin { get; set; }
            public bool AllowUSB { get => AllowWrite && AllowRead; }
            public bool AllowWrite { get; set; }
            public bool AllowRead { get; set; }
            public bool AllowDownloads { get; set; }
            public List<string> RestrictedProcesses { get; set; }
            public List<string> RestrictedWebPages { get; set; }
        }

        public static List<User> AllUsers = new List<User>();

        public static List<User> GetAllUsers()
        {

            if (AllUsers.Count > 0)
            {
                return AllUsers;
            }

            string json = FileWorker.GetAdminSettings();

            //начитуємо вміст файлу adminsettings
            if (!string.IsNullOrEmpty(json))
            {
                AllUsers = JsonConvert.DeserializeObject<List<User>>(json);
            }
            return AllUsers;
        }

        public static void SetUserSetting(User UserSettings)
        {
```

```
            User UserInList = AllUsers.FirstOrDefault(e => e.UserLogin == UserSettings.UserLogin);

            if (UserInList != null)
            {
                int indexInAllUsers = AllUsers.IndexOf(UserInList);
                AllUsers[indexInAllUsers] = UserSettings;
            }

            else
            {
                AllUsers.Add(UserSettings);
            }

            string adminsettings = JsonConvert.SerializeObject(AllUsers);
            FileWorker.WriteAdminSettings(adminsettings);

        }

        //Метод для видалення користувача
        public static string DeleteUser(string userName)
        {
            string error = string.Empty;
            User user = AllUsers.FirstOrDefault(e => e.UserLogin.Equals(userName,
StringComparison.OrdinalIgnoreCase));
            if (user == null)
            {
                //Такого бути не повинно, але краще перестрахуватись
                error = "Такого користувача немає";
                return error;
            }
            AllUsers.Remove(user);
            string adminsettings = JsonConvert.SerializeObject(AllUsers);
            FileWorker.WriteAdminSettings(adminsettings);
            return error;
        }
    }
}
```

Код програмного модулю вихідного файлу Worker.cs

```
using System;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
using Microsoft.Win32;

namespace ПрограмаЗахисту
{
   internal class Worker
   {
      private static readonly int regOKReadStat = 3;
      private static readonly string regOKReadKey =
"SYSTEM\\CurrentControlSet\\services\\USBSTOR";
      private static readonly string regOKReadVal = "start";

      private static readonly int regOKWriteStat = 0;
      private static readonly string regOKWriteKey =
"SYSTEM\\CurrentControlSet\\services\\USBSTOR";
      private static readonly string regOKWriteVal = "WriteProtect";

      private static CancellationTokenSource ts = new CancellationTokenSource();
      private static CancellationToken ct = ts.Token;

      private static Task KillProcOnSchedule { get; set; }

      public static void SetRestrictions(AdministratorSettings.User userSettings)
      {
         KillProcOnSchedule = Task.Run(() => Kill(), ct);

         if (userSettings.AllowRead)
         {
            AllowRead();
         }
         else
         {
            RestrictRead();
         }

         if (userSettings.AllowWrite)
         {
            AllowWrite();
         }

         else
         {
            RestrictWrite();
```

```
        }
    }
    //Метод для отримання значень із реєстру.
    //1 аргумент-ключ реєстру, 2-значення ключа, 3-потрібне значення ключа
    public static bool GetRegValue(string regKey, string regValue, int requiredValue)
    {
        try
        {
            using (RegistryKey key = Registry.LocalMachine.OpenSubKey(regKey))
            {
                if (key != null)
                {
                    object o = key.GetValue(regValue);
                    return o != null && (o as int?) == requiredValue;
                }
            }
        }

        catch (Exception)
        {
            return false;
        }

        return false;
    }

    public static void EndAll()
    {
        AllowRead();
        AllowWrite();
        ts.Cancel();
    }
    private static void AllowRead()
    {
        if (!GetRegValue(regOKReadKey, regOKReadVal, regOKReadStat))
        {
            string pathToFile = @"C:\";
            string filePath = Path.Combine(pathToFile, "Дозвіл читання зовнішніх пристроїв.reg");
            string text_acess_write = "Windows Registry Editor Version 5.00 \r\n" + "\r\n" +
             ";Дозволити читання зовнішніх накопичувачів\r\n" +
             "[HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\services\\USBSTOR]
\r\n" +
             @"""Start""" +
             "= dword:00000003";
            File.WriteAllText(filePath, text_acess_write);
            string path = filePath;
            if (File.Exists(path))
            {
                Process.Start(path);
            }
```

Продовження додатку Д

```
            else
            {
                MessageBox.Show("Не вдалося створити файл");
            }
        }
    }

    private static void RestrictRead()
    {
        string pathToFile = @"C:\";
        string filePath = Path.Combine(pathToFile, "Заборона читання зовнішніх
пристроїв.reg");
        string text_acess_write = "Windows Registry Editor Version 5.00 \r\n" + "\r\n" +
          ";Заборона читання зовнішніх накопичувачів \r\n" +
          "[HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\services\\USBSTOR] \r\n"
+
          @"""Start""" +
          "= dword:00000004";
        File.WriteAllText(filePath, text_acess_write);

        string path = filePath;
        if (File.Exists(path))
        {
            Process.Start(path);
        }

        else
        {
            MessageBox.Show("Не вдалося створити файл");
        }
    }

    private static void AllowWrite()
    {
        if (!GetRegValue(regOKWriteKey, regOKWriteVal, regOKWriteStat))
        {
            string pathToFile = @"C:\";
            string filePath = Path.Combine(pathToFile, "Дозвіл запису на зовнішні пристрої.reg");
            string text_acess_write = "Windows Registry Editor Version 5.00 \r\n" + "\r\n" +
              ";Відключити запис файлів на зовнішні накопичувачі \r\n" +

"[HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Control\\StorageDevicePolicies]
\r\n" +
              @"""WriteProtect""" +
              "= dword:00000000";
            File.WriteAllText(filePath, text_acess_write);

            string path = filePath;
            if (File.Exists(path))
                Process.Start(path);
```

Продовження додатку Д

```
            else
                MessageBox.Show("Не вдалося створити файл");
        }
    }

    private static void RestrictWrite()
    {
        string pathToFile = @"C:\";
        string filePath = Path.Combine(pathToFile, "Заборона запису на зовнішні
пристрої.reg");//Створюємо файл реєстру на заборону запису
        string text_deny_write = "Windows Registry Editor Version 5.00 \r\n" + "\r\n" +
          ";Відключити запис файлів на зовнішні накопичувачі \r\n" +

"[HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Control\\StorageDevicePolicies]
\r\n" +
          @"""WriteProtect""" +
          "= dword:00000001";
        File.WriteAllText(filePath, text_deny_write);

        string path = filePath;
        if (File.Exists(path))
            Process.Start(path);
        else
            MessageBox.Show("Не вдалося створити файл");
    }

    private static void Kill()
    {
        while (true)
        {
            if (ct.IsCancellationRequested)
            {
                break;
            }

            if (Authorization_form.CurrentUser.RestrictedProcesses.Count == 0)
            {
                return;
            }

            foreach (Process proc in Process.GetProcesses())//Отримаємо список усіх запущених
процесів
            {
                string prcName = proc.ProcessName;
                if (IsToKill(prcName))
                {
                    try
                    {
                        proc.Kill();
                    }
```

Продовження додатку Д

```
                catch (Exception Ex)
                {

                    MessageBox.Show(Ex.ToString());
                }
            }
        }
        Thread.Sleep(10000);
    }
}

    private static bool IsToKill(string prcName)
    {
        //За допомогою лямбда виразу перевіряємо чи є процес prcName, переданий у
аргументі в колекції Form2.CurrentUser.RestrictedProcesses
        return Authorization_form.CurrentUser.RestrictedProcesses.FirstOrDefault(e =>
e.Equals(prcName, StringComparison.OrdinalIgnoreCase)) != null;
    }
    //метод для перезавантаження ПК
    public static void RebootPC()
    {
        ProcessStartInfo p = new ProcessStartInfo("cmd", "/r shutdown -f -r -t 0")
        { CreateNoWindow = true };
        Process.Start(p);
    }
    //Метод для правильного завершення роботи програми
    public static void AppClose()
    {
        EndAll();
        Firewall.proxyController.Stop();
        Application.Exit();
    }
  }
}
```

Продовження додатку Д

Додаток Е

Код програмного модулю вихідного файлу FileWorker.cs

```
using System.IO;

namespace ПрограмаЗахисту
{
   class FileWorker
   {
      public static string GetHistory()
      {
         string json = string.Empty;

         using (StreamReader sr = new StreamReader("history")) //для начитки контенту
         {
            json = sr.ReadToEnd();
         }

         return json;
      }

      public static void SetHistory(string HistoryJSON)
      {
         using (StreamWriter sw=new StreamWriter("history")) //для запису контенту
         {
            sw.Write(HistoryJSON);
         }
      }

      public static string GetAdminSettings()
      {
         string json = string.Empty;

         using (StreamReader sr = new StreamReader("adminsettings"))
         {
            json = sr.ReadToEnd();
         }
         return json;
      }

      public static void WriteAdminSettings(string adminsettings)
      {
         using (StreamWriter sw = new StreamWriter("adminsettings"))
         {
            sw.Write(adminsettings);
         }
      }

   }
}
```

Додаток К

Код програмного модулю вихідного файлу Firewall.cs

```
using System;
using System.Linq;
using System.Collections.Generic;
using System.Net;
using System.Net.Security;
using System.Text;
using Newtonsoft.Json;
using System.Threading;
using System.Threading.Tasks;
using Titanium.Web.Proxy;
using Titanium.Web.Proxy.EventArguments;
using Titanium.Web.Proxy.Exceptions;
using Titanium.Web.Proxy.Helpers;
using Titanium.Web.Proxy.Http.Responses;
using Titanium.Web.Proxy.Models;


namespace ПрограмаЗахисту
{
    public static class Firewall
    {
        //ініціалізуємо колекцію із дозволеними типами файлів
        private static IReadOnlyList<string> AcceptedFileTypes = new List<string>

{
            ".html",
            ".htm",
            ".php",
            ".aspx",
            ".css",
            ".js",
            ".img",
            ".png",
            ".ico",
            ".gif",
            ".json",
            ".woff"
        };

        //ініціалізуємо структуру для запису історії браузера
        public class BrowserHistory
        {
            public DateTime VisitTime;
            public string Url;
            public string PageTitle;
        }

        private static List<BrowserHistory> History = new List<BrowserHistory>();
```

```
        private static IReadOnlyList<string> AcceptedContentTypes = new List<string>
        {
            "text/html",
            "text/css",
            "text/javascript",
            "text/plain",
            "text/xml",
            "text/cache-manifest",
            "application/x-protobuf",
            "application/octet-stream",
            "application/json",
            "application/xml",
            "application/javascript",
            "image/png",
            "image/gif",
            "image/jpg",
            "image/jpeg",
            "image/x-icon",
            "image/webp",
            "font/woff",
        };

        public static ProxyController proxyController = new ProxyController();

        public class ProxyController
        {
            private readonly SemaphoreSlim @lock = new SemaphoreSlim(1);
            private readonly ProxyServer proxyServer;
            private ExplicitProxyEndPoint explicitEndPoint;

            public ProxyController()
            {

                History = GetHistory();
                //запускаємо запис історії браузера по розкладу
                Task.Run(() => WriteHistoryOnShchedule());


                proxyServer = new ProxyServer
                {

                    ExceptionFunc = async exception =>
                    {
                        if (exception is ProxyHttpException phex)
                        {
                            await writeToConsole(exception.Message + ": " + phex.InnerException?.Message,
true);
                        }
                        else
                        {
```

Продовження додатку К

```
                await writeToConsole(exception.Message, true);
            }


},
            ForwardToUpstreamGateway = true
        };
        proxyServer.CertificateManager.SaveFakeCertificates = true;
    }

    public void StartProxy()
    {
        //ініціалізація проксі
        if (proxyServer != null && !proxyServer.ProxyRunning)
        {
            proxyServer.BeforeRequest += onRequest;
            proxyServer.BeforeResponse += onResponse;

            proxyServer.ServerCertificateValidationCallback += OnCertificateValidation;
            proxyServer.ClientCertificateSelectionCallback += OnCertificateSelection;

            explicitEndPoint = new ExplicitProxyEndPoint(IPAddress.Parse("127.0.0.2"), 8000);


            proxyServer.AddEndPoint(explicitEndPoint);
            proxyServer.Start();

            if (RunTime.IsWindows)
            {
                proxyServer.SetAsSystemProxy(explicitEndPoint, ProxyProtocolType.AllHttp);
            }
        }
    }

    //метод для зупинки проксі
    public void Stop()
    {
        if (proxyServer.ProxyRunning)
        {

            proxyServer.BeforeRequest -= onRequest;
            proxyServer.BeforeResponse -= onResponse;
            proxyServer.ServerCertificateValidationCallback -= OnCertificateValidation;
            proxyServer.ClientCertificateSelectionCallback -= OnCertificateSelection;

            proxyServer.Stop();

            // можна видаляти збережені сертифікати (неопціонально)
        }
```

Продовження додатку К

```
        }

        // подія, яка виникає при запиті
        private async Task onRequest(object sender, SessionEventArgs e)
        {
            await writeToConsole(e.HttpClient.Request.Url);

            //ініціалізуємо налаштування користувача

            if (Authorization_form.CurrentUser != null)
            {
                foreach (string restrictedSite in Authorization_form.CurrentUser.RestrictedWebPages)
                {
                    if (e.HttpClient.Request.RequestUri.AbsoluteUri.Contains(restrictedSite))
                    {
                        History.Add(new BrowserHistory
                        {
                            PageTitle = $"Prohibited! {e.HttpClient.Request.RequestUri.AbsoluteUri}",
                            Url = e.HttpClient.Request.RequestUri.AbsoluteUri
                        });

e.Ok($"<html><head><title>Prohibited!{e.HttpClient.Request.RequestUri.AbsoluteUri}" +
                            $"</title></head>" +
                            "<body>Acces to this site is prohibited</body></html>");
                        break;
                    }
                }
            }
        }

        //метод для перевірки файлу, який намагається завантажити користувач
        private static bool Check(string ext)
        {
            foreach (string allowedExt in AcceptedFileTypes)
            {
                if (ext.Contains(allowedExt))
                {
                    return true;
                }
            }
            return false;
        }

        //подія, яка виникає при відповіді клієнту
        private async Task onResponse(object sender, SessionEventArgs e)
        {
            //визначаємо розширення файлу, який намагається завантажити користувач
            string ext =
System.IO.Path.GetExtension(e.HttpClient.Request.RequestUri.AbsolutePath);
```

Продовження додатку К

```
            if (string.IsNullOrEmpty(ext))
            {
//У деяких випадках, коли файл формується сервером пробуємо дістати його із заголовку Content-Disposition
                if (e.HttpClient.Response.Headers.Headers.TryGetValue("Content-Disposition", out HttpHeader o))
                {

                }
            }
            string html = await e.GetResponseBodyAsString();

            if (!string.IsNullOrEmpty(html))
            {

                string startTitleTag = "<title>";
                int indexOfTitle = html.IndexOf(startTitleTag);

                if (indexOfTitle != -1 && e.HttpClient.Response.StatusCode == (int)HttpStatusCode.OK)
                {
                    int endTitleTag = html.IndexOf("</title>", indexOfTitle + startTitleTag.Length);
                    string titleTagValue = html.Substring(indexOfTitle + startTitleTag.Length,
                        endTitleTag - indexOfTitle - startTitleTag.Length);



                    History.Add(item: new BrowserHistory
                    {
                        PageTitle = titleTagValue,
                        Url = e.HttpClient.Request.RequestUri.AbsoluteUri,
                        VisitTime = DateTime.Now
                    });
                }
            }
            //беремо налаштування адміністратора


            if (Authorization_form.CurrentUser != null && !Authorization_form.CurrentUser.AllowDownloads)
            {

                if (e.HttpClient.Request.Headers.Headers.ContainsKey("User-Agent"))
                {

                    if (!string.IsNullOrEmpty(ext))
                    {
                        if (!Check(ext))
                        {
```

Продовження додатку К

```
                byte[] btBody = Encoding.UTF8.GetBytes("Downloads are prohibited!");

                OkResponse response = new OkResponse(btBody)
                {
                    HttpVersion = e.HttpClient.Request.HttpVersion
                };

                e.Respond(response);
                e.TerminateServerConnection();
            }
        }
    }
}

private static object obj = new object();

//валідація SSL сертифікатів
public Task OnCertificateValidation(object sender, CertificateValidationEventArgs e)
{
    if (e.SslPolicyErrors == SslPolicyErrors.None)
    {
        e.IsValid = true;
    }

    return Task.FromResult(0);
}

public Task OnCertificateSelection(object sender, CertificateSelectionEventArgs e)
{
    return Task.FromResult(0);
}

private async Task writeToConsole(string message, bool useRedColor = false)
{
    return;
    await @lock.WaitAsync();

    if (useRedColor)
    {
        ConsoleColor existing = Console.ForegroundColor;
        Console.ForegroundColor = ConsoleColor.Red;
        Console.WriteLine(message);
        Console.ForegroundColor = existing;
    }
    else
    {
        Console.WriteLine(message);
    }
```

Продовження додатку К

```
            @lock.Release();

        }

        private static void WriteHistoryOnShchedule()
        {
            //Кожні 10 секунд пишемо історію до файлу
            while (true)
            {
                System.Threading.Thread.Sleep(10000);
                lock (obj)
                {
                    string json = JsonConvert.SerializeObject(History);
                    FileWorker.SetHistory(json);
                }
            }
        }

        public static List<BrowserHistory> GetHistory()
        {
            if (History.Count == 0)
            {
                string historyJson = FileWorker.GetHistory();
                if (!string.IsNullOrEmpty(historyJson))
                {
                    return History =
JsonConvert.DeserializeObject<List<BrowserHistory>>(historyJson);
                }
            }
            //Сортуємо масив
            return History.OrderByDescending(i => i.VisitTime).ToList();
        }

    }
  }
}
```

Продовження додатку К

Код програмного модулю вихідного файлу UserBrowsingHistory.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace ПрограмаЗахисту
{
   public partial class UserBrowsingHistory : Form
   {
      public UserBrowsingHistory()
      {
         InitializeComponent();
      }
      //Подія, яка виникає при завантаженні форми із історією
      private void UserBrowsingHistory_Load(object sender, EventArgs e)
      {
         DataGridViewColumn VisitTime = new DataGridViewColumn()
         {
            HeaderText = "Час візиту",
            Name = "VisitTime",
            CellTemplate = new DataGridViewTextBoxCell(),
            AutoSizeMode= DataGridViewAutoSizeColumnMode.Fill
         };
         DataGridViewColumn PageTitle = new DataGridViewColumn()
         {
            HeaderText = "Сторінка",
            Name = "PageTitle",
            CellTemplate = new DataGridViewTextBoxCell(),
            AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill
         };
         DataGridViewLinkColumn Url = new DataGridViewLinkColumn()
         {
            HeaderText = "Url",
            Name = "Url",
            DataPropertyName="Url",
            AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill
         };
         HistoryGridView.Columns.Add(VisitTime);
         HistoryGridView.Columns.Add(PageTitle);
         HistoryGridView.Columns.Add(Url);

         foreach(Firewall.BrowserHistory history in Firewall.ProxyController.GetHistory())
         {
            HistoryGridView.Rows.Add(history.VisitTime,history.PageTitle,history.Url);    } } } }
```