

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ПРОГРАМИ АНТИВІРУСНОГО СКАНЕРА ДЛЯ
ОПЕРАЦІЙНОЇ СИСТЕМИ WINDOWS. ANTIVIRUS SCANNER
DEVELOPMENT FOR
WINDOWS OS.

Виконав: студент(ка) 4 курсу, групи КНТ-227

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

Ал Омарі А.М.

(прізвище та ініціали)

Керівник Сердюк С.М.

(прізвище та ініціали)

Рецензент Казурова А.Є.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 122 Комп'ютерні науки
(код і найменування)
 Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА(КИ)

Ал Омарі Абдалла Мохаммедовича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка програми антивірусного сканера для операційної системи Windows. Antivirus Scanner Development for Windows OS.
 керівник проєкту (роботи) доц., к.т.н. Сердюк Сергій Микитович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
 затверджені наказом закладу вищої освіти від “30” березня 2021 року
 № 103

2. Строк подання студентом проєкту (роботи) травня 2021 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали та методи. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Сердюк С.М., доцент		
Нормоконтроль	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання 28 лютого 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка архітектури програми.	4 тиждень	Розділ 2
4	Розробка програми.	5–6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми.	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8–9 тижні	Розділ 5, Додатки
7	Захист роботи.	10 тиждень	

Студент(ка)

(підпис)

Ал Омарі А.М.

(прізвище та ініціали)

Керівник проекту (роботи)

(підпис)

Сердюк С.М.

(прізвище та ініціали)

(підпис)

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
80 с., 4 табл., 10 рис., 2 дод., 12 джерел.

ВІРУС, C#, WINDOWS FORMS, СИГНАТУРА, ЗАХИСТ,
ПЛАТФОРМА .NET.

Об'єкт дослідження – процеси антивірусного захисту в операційній системі (ОС) Windows.

Предмет дослідження – методи антивірусного захисту в ОС Windows.

Мета роботи – розробка програми, що дозволяє виявляти вірусні сигнатури у файлах та папках, захищати доступ до папок, видаляти інфіковані файли та фіксувати історію дій користувача з програмою.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування; мова програмування C#; програмна платформа .NET; інтерфейс програмування застосунків Windows Forms; персональний комп'ютер (ПК) з процесором (x64) не менше 1 ГГц під управлінням ОС Windows 10.

Результати. Розроблено застосунок, що виконує антивірусну перевірку файлів/каталогів, запис результатів перевірки, видалення інфікованих файлів/каталогів та захист каталогів шляхом встановлення паролю.

Галузь використання – антивірусний захист в ОС Windows 10.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 80 pages, 10 figures, 4 tables, 2 appendixes, 12 sources.

VIRUS, C #, WINDOWS FORMS, SIGNATURE, PROTECTION, .NET PLATFORM.

Object of study is a antivirus protection processes in the Windows operating system (OS)

Subject of study are methods of antivirus protection in Windows.

The purpose of the work is development of a program that allows you to detect virus signatures in files and folders, protect access to folders, delete virus files and record the history of user actions with the program.

Materials, methods and technical means: object-oriented programming; programming language C#; .NET software platform; Windows Forms application programming interface; a personal computer (PC) with a processor (x64) of at least 1 GHz running Windows 10.

Results. An application has been developed that performs antivirus scanning of files/ directories, recording of scan results, deletion of infected files/directories and protection of directories by setting a password.

The field of use is antivirus protection in OS Windows 10.

ЗМІСТ

	С.
Перелік скорочень та умовних познач 8	8
Вступ 9	9
1 Аналіз предметної області 11	11
1.1 Огляд принципів роботи антивірусних програм.....11	11
1.2 Аналіз програм-аналогів 15	15
1.3 Розробка технічного завдання.....20	20
1.3.1 Підстава для розробки 20	20
1.3.2 Мета розробки та її призначення 20	20
1.3.3 Основні вимоги до програмного продукту.....20	20
1.3.4 Вимоги до технічних засобів, операційної системи і мови програмування 21	21
1.3.5 Обмеження на установку і виконання програми.....22	22
1.3.6 Вимоги до програмної документації.....22	22
2 Матеріали та методи.....23	23
2.1 Аналіз технічного завдання. Алгоритм M5D Hash 23	23
2.2 Обґрунтування вибору операційної системи та мови програмування 27	27
2.3 Визначення компонентів системи.....23	23
3 Основні рішення щодо реалізації компонентів системи 33	33
3.1 Розробка алгоритму роботи програми.....33	33
3.2 Розробка Desktop-застосунку 36	36
3.3 Розробка графічного інтерфейсу користувача 38	38
4 Керівництво програміста.....50	50
4.1 Призначення та умови застосування програми.....50	50
4.2 Характеристики програми.....50	50
4.3 Вхідні та вихідні дані.....50	50
4.4 Атрибути та методи класу Form1 51	51
4.5 Повідомлення програмісту 53	53

5 Керівництво оператора.....	54
5.1 Призначення програми	54
5.2 Умови використання програми.....	54
5.3 Виконання програми.....	54
5.4 Повідомлення оператору	57
Висновки	58
Перелік джерел посилання.....	59
Додаток А. Текст програми.....	61
Додаток Б. Копії слайдів презентації	72

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

АПЗ	– антивірусне програмне забезпечення;
БД	– база даних;
ГГц	– гігагерц;
Мб	– мегабайт;
ОС	– операційна система;
ПК	– персональний комп'ютер;
ПЗ	– програмне забезпечення;
ТЗ	– технічне завдання;
ШПЗ	– шкідливе програмне забезпечення.

ВСТУП

У вік інформаційних технологій, коли у кожної людини є доступ до мережі Internet, майже безмежного джерела інформації, де присутні терабайти різноманітних комп'ютерних файлів якими можуть обмінюватись користувачі на великій кількості різних ресурсах, гостро стоїть питання захисту особистої інформації та комп'ютерних систем від вірусів.

Взагалі поняття «вірус» відноситься тільки до програмного забезпечення (ПЗ) яке здатне інфікувати файли та вносити свої зміни до файлу, роблячи його пошкодженим й непридатним для подальшої роботи. Це поняття прийшло до нас з вже далекого 1970 року де уперше воно було вжито по відношенню до програмного середовища у романі Грегорі Бенфорда «Людина у шрамах».

Цей вид шкідливого програмного забезпечення (ШПЗ) у даний час є найменш поширеним, найчастіше зустрічаються такі види як: хробаки (ШПЗ, що здатне до саморозмноження та самовідновлення на комп'ютері користувача); рекламне ПЗ (автоматично доставляє рекламу на хост комп'ютера); боти (призначені для певного автоматичного виконання певних операцій на комп'ютері користувача без його дозволу); руткити (дозволяють отримати доступ до пристрою третім особам, та керувати їм); трояни (маскуються під лігитимні файли, але після установки вони вносять зміни у систему, й здійснюють шкідливу діяльність без відома користувача) [1].

Для протидії ШПЗ були розроблені програми які виявляють, ліквідують, та захищають пристрої від них – антивіруси. Щоб забезпечити собі безпеку при пошуку інформації в мережі Internet, скачуванні різноманітних файлів, кожен користувач має мати у себе на ПК антивірусне програмне забезпечення (АПЗ), щоби мінімізувати ризик потрапляння ШПЗ до своєї системи.

Отже, розробка АПЗ є актуальною задачею, яка потребує свого вирішення у межах даної дипломної кваліфікаційної роботи бакалавра.

Кожне сучасне АПЗ базується на так званому евристичному аналізаторі коду. Цей модуль за певними алгоритмами аналізує файли в пошуках

підозрілих ділянок коду, які потім порівнюються з інформацією, що зберігається в антивірусній базі [2].

Коли АПЗ стикається з новим ШПЗ, перше, що воно робить, це перевіряє хеш файлу, щоб переконатися, що він був помічений або, можливо, навіть проаналізований раніше і занесений в антивірусну базу. Ці хеші досить часто [3] генеруються з використанням алгоритму MD5, який також передбачається використовувати і в програмі, що розробляється в даній роботі.

Результатом використання АПЗ, що розробляється є захист персональної інформації на етапі експлуатації комп'ютерних систем.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд принципів роботи антивірусних програм

Історія антивірусного програмного забезпечення, звичайно ж, пов'язана з історією комп'ютерних вірусів або «шкідливих програм». До 1980-х років були проведені деякі експерименти з програмами, здатними автоматично поширюватися між комп'ютерами, але саме в це десятиліття були створені і запуснені перші віруси, здатні видаляти дані або відключати системи. Незабаром після цього в магазинах з'явилися перші комерційні антивірусні програми. Однією з компаній-піонерів була G Data, німецька компанія, яка в 1987 році розробила продукт для захисту комп'ютерів Atari. У тому ж році McAfee випустила першу версію свого популярного антивірусного пакета VirusScan [4].

Основна мета антивірусного програмного забезпечення – виявляти, нейтралізувати або знищувати шкідливі програми з комп'ютерів і пристроїв навіть до того, як система стане зараженою.

Щоб ідентифікувати шкідливу атаку, АПЗ буде запускати сканування, порівнюючи файли, що зберігаються в операційній системі його головного комп'ютера, з базою даних, що містить відмінні риси (або «сигнатури») раніше виявлених примірників «шкідливого ПЗ». Ця база даних повинна регулярно оновлюватися, щоб включати в себе сигнатури нових типів ШПЗ, яке постійно з'являється. Крім того, деякі антивірусні програми можуть виявляти загрози, шляхом виявлення шаблонів у файлах, або виявляючи зміни в системі і відстежуючи поведінку компонентів комп'ютера.

Таким чином, сучасні пакети антивірусного програмного забезпечення виконують подвійну роль: аналізують один за іншим файли, що зберігаються на комп'ютері, відповідно до бази даних сигнатур і відстежують роботу пристрою для виявлення будь-якої незвичайної активності.

Слід зазначити що, установка більше одного АПЗ на одному ПК шкідлива. Це пов'язано з тим, що антивірусні пакети можуть помилково

ідентифікувати як «шкідливі програми» і, отже, намагатися заблокувати одне одного, залишаючи комп'ютер незахищеним.

Існує кілька основних підходів, на основі яких працюють антивірусні програми [5], [6]:

- словниковий (сигнатурний) підхід;
- підхід «підозріла поведінка»;
- підхід «емуляції»;
- підхід «пісочниця»;
- виявлення хмарного антивірусу;
- повне сканування системи.

Розглянемо ці підходи більш детально.

Словник – це книга, в якій зазвичай шукають значення і визначення понять та речей. По аналогії антивірусний словник – це файл, який містить визначення вірусів. Під час сканування системи антивірусною програмою коди системних файлів порівнюються з визначеннями вірусів в словнику, щоб визначити, чи є вони вірусом. Якщо файл містить код вірусу, користувач отримує повідомлення про це. Потім користувач повинен вирішити, чи слід видалити файл, помістити в карантин або вилікувати.

Це досить швидкий підхід до пошуку вірусів, оскільки його можна застосувати, тоді коли операційна система відкриває, запускає і закриває будь-який файл. Але першочерговою необхідністю при такому підході є його щоденне оновлення. Щодня з'являються тисячі нових вірусів, які застарілі словники не зможуть виявити або усунути при виявленні, якщо в словнику немає їх визначення. Для підвищення ефективності захисту ПК користувачі повинні регулярно оновлювати антивірусне програмне забезпечення.

Хоча цей підхід вважається дуже ефективним, він все ж не працює в разі олігоморфних, метаморфних і поліморфних вірусів, які можуть змінюватися при кожному новому зараженні за допомогою шифрування. Тобто, кожна модифікація, фактично, являє собою новий екземпляр вірусу. Тому, антивірусний захист на основі сигнатурних баз безсилий проти просунутого

поліморфного вірусу. Однак, у найпростіших поліморфних вірусів – олігоморфних, зустрічаються однакові ділянки коду, за якими їх можна ідентифікувати за допомогою сигнатурних баз.

Підхід «підозріла поведінка» – це як випливає з назви, метод заснований на тому, що поведінка процесу може викликати підозру. Наприклад, невідомий процес, запущений в системі користувача, намагається змінити таблицю FAT або записати дані в будь-якої виконуваний файл. Це безперечно викликає підозри. Таким чином, цей метод може забезпечити захист від нових вірусів.

Виявлення на основі поведінки є корисним методом пошуку вірусів або іншого шкідливого програмного забезпечення, яке намагається вкрати або записати інформацію. Однак сьогодні багатьом програмам потрібно звітувати на Internet-сервері або реєструвати натискання клавіш, щоб запобігти обману в мережі Internet, що призводить до створення помилкових попереджень АПЗ. Таким чином, отримуючи все більше і більше попереджень, користувач схильний ігнорувати їх і тим самим час від часу дозволяє вірусам руйнувати систему.

У сучасному антивірусному програмному забезпеченні використовується поєднання словника і підозрілої поведінки для виявлення, видалення і запобігання ШПЗ.

Наступний підхід – підхід «емуляції» (евристичне сканування). Евристичне сканування є найвідомішим методом виявлення нових вірусів. При цьому підході АПЗ імітує початок коду кожного нового виявленого файлу, який буде виконуватися, перш ніж передати весь контроль самому цьому виконуваному файлу. Якщо програма має код, що саомодифікується або самовідтворюємий або намагається знайти інші виконувані файли, можемо сказати, що виконуваний файл був заражений. Але навіть у цього методу дуже багато помилкових спрацьовувань.

Далі розглянемо підхід «пісочниця». У комп'ютерній безпеці пісочниця – це захисний механізм для ізоляції програм, що були запущені на ПК. Такий підхід використовується для тестування неперевіреного фрагмента коду, який

може бути шкідливою програмою. Це досягається за рахунок жорсткого контролю ресурсів, що надаються цій ненадійній програмі для запуску. Іншими словами, механізм «пісочниця» діє як реальна операційна система і запускає ненадійні гостьові програми. Після запуску, програма із «пісочниці» перевіряється на предмет змін, що носять вірусний характер. Цей метод має високі накладні витрати, що обмежує їх використання.

Визначення хмарного антивірусу використовує комп'ютерну програму, яка збирає інформацію, яка потім завантажується та обробляється сервером у хмарі. Запустивши всі функції виявлення на сервері, комп'ютер користувача позбавлений можливостей додаткової обробки. Хмарний антивірус вимагає підключення до мережі Internet.

Нарешті, повне сканування системи або сканування окремих файлів - це ручна (контрольована) дія, яку вживає користувач для сканування всіх комп'ютерних файлів за допомогою АПЗ. Зазвичай повне сканування не повинно бути необхідним, якщо на комп'ютері встановлена антивірусна програма, яка активно контролює зміни. Однак, якщо ПК діє підозріло або встановлений новий антивірусний сканер, рекомендується провести повне сканування. Під час повного сканування системи переглядаються майже всі файли, тому завершення цього процесу може зайняти багато часу.

Крім вищенаведених підходів, останніми роками активно розвивається стратегія інтелектуального аналізу даних – це нова модель розпізнавання шкідливого програмного забезпечення. З урахуванням особливостей програми, аналіз даних виявляє, чи є файл або програма шкідливим програмним забезпеченням.

Таким чином, не існує методу, який був би точним і захищав ПК на 100%. Антивіруси як такі використовують комбінації вищезазначених методів для захисту системи. Чим більше виправлень безпеки внесено в будь-яку систему або ОС, тим менше буде вірогідність вторгнення в неї. Експерти кажуть, що LINUX є безпечною ОС через складність файлових систем і складних механізмів доступу. Якщо Microsoft спробує зробити такі

удосконалення безпеки і в ОС Windows, тоді ця система отримає більший захист.

1.2 Аналіз програм-аналогів

Перед початком безпосереднього проєктування АПЗ, було проведено аналіз вже існуючих програмних рішень, виділено їх основні переваги та недоліки, сформовані певні підсумки по кожному з програмних продуктів та підсумки аналізу програм-аналогів.

Вибір найкращого антивірусу для захисту комп'ютера може бути складним завданням через необхідність врахування користувачем множини критеріїв.

При порівнянні показників надійності більшість антивірусних пакетів повинна мати свої власні показники в двох основних областях антивірусного захисту: виявлення вірусів за допомогою традиційного сканування вірусів і захисту в режимі реального часу, коли файли зберігаються в системі [7].

Відома стороння компанія з тестування антивірусних програм – AV-Comparatives, для визначення щорічних переможців АПЗ використовує наступну серію базових тестів [8]:

- тестування захисту в реальних умовах (Real-World Protection Test). Ці тести оцінюють «реальні» можливості захисту з налаштуваннями за замовчуванням (включаючи функції захисту при виконанні). Зазвичай такі випробування вимагають великих витрат часу і ресурсів;

- тест продуктивності (Performance Test). Програми, що працюють у фоновому режимі, такі як АПЗ для захисту в реальному часі, використовують певний відсоток системних ресурсів. Ці тести допомагають користувачам оцінити свій антивірусний захист з точки зору швидкості системи (продуктивності системи);

- тест захисту від шкідливих програм (Malware Protection Test). Це вдосконалений тест File Detection Test, який компанія AV-Comparatives

проводили в попередні роки. Будь-які зразки, які не були виявлені, наприклад при доступі виконуються в тестовій системі з доступом до Internet/хмара, що дозволяє задіяти такі функції, як захист від «підозрілої поведінки»;

- тест на помилкові сигнали тривоги (False-Alarm Test). Помилкові тривоги іноді можуть доставити стільки ж неприємностей, скільки і справжній вірус. При тестуванні антивіруса важливо вимірювати не тільки можливості виявлення, а й надійність ПЗ – одним з аспектів надійності, безумовно, є тенденція продукту відзначати чисті файли як заражені. Тест на помилкові сигнали тривоги є додатковим до тестів захисту від шкідливих програм;

- тест розширеного захисту від загроз (Advanced Threat Protection Test). Цей тест також відомий як Enhanced Real-World Test, використовує методи злому і проникнення, які дозволяють зловмисникам атакувати певні зовнішні комп'ютерні системи, і оцінює, наскільки добре АПЗ захищає від таких атак. Цей тест перевіряє ступінь захисту від цільових розширених загроз, таких як експлойти (комп'ютерні програми, що використовують вразливості в ПЗ) і безфайлові атаки (атаки, при яких виконуваний файл як шкідливе навантаження запускається безпосередньо в пам'яті, не зберігаючись на диску).

Хоча більшість антивірусних програм можуть запропонувати багато різних функцій, є кілька основних критеріїв, які необхідно враховувати при їх оцінці. Крім того необхідно прийняти до уваги індивідуальні потреби користувача. Можливо є потреба у швидкому АПЗ, що не призведе до затримки роботи ПК, а для іншого користувача можуть бути більш важливі спеціальні функції захисту конфіденційних документів. Тому треба уважно порівнювати продукти і їх функції, щоб знайти оптимальний баланс.

Перше, на що потрібно звернути увагу при аналізі антивірусних програм, – це наскільки добре програма може виявляти віруси, вторгнення або інші небезпеки. Кращі антивірусні програми на ринку пройшли ретельну оцінку, проведену незалежними тестувальниками, тому ці оцінки досить добре відображають можливості виявлення і видалення вірусів.

Наступний важливий для оцінки параметр – продуктивність. Більшість користувачів віддадуть перевагу програмі, яка не буде мати великого впливу на продуктивність ПК. Це буде означати, що не доведеться мати справу з повільною передачею файлів, нечутливим управлінням або нестабільною частотою кадрів в процесі гри на ПК.

Після оцінки можливостей виявлення ШПЗ та продуктивності АПЗ, необхідно провести оцінку додаткових функцій, що надаються кожним антивірусом. Це може бути, наприклад, функція "Батьківський контроль" якщо є діти, які хочуть пограти на ПК батьків. Крім того, в наші дні все більш важливою стає захист онлайн-банкінгу.

Сучасні антивірусні програми не обмежуються тільки захистом операційної системи і файлів від вірусних атак. Тому, необхідний також порівняльний аналіз пропонованих ними функцій, орієнтованих на конфіденційність. Це можуть бути функції захисту вебкамери або інших пристроїв, таких як ноутбуки, смартфони або планшети.

З урахуванням вищенаведених рекомендацій, проведемо порівняльний аналіз п'яти кращих антивірусних програм 2021 року за версією сайту AntivirusGuide за наступними критеріями: рейтинг (згідно AntivirusGuide); можливості виявлення ШПЗ; продуктивність системи; співвідношення ціна/якість; захист (в режимі реального часу, видалення ШПЗ, захист від програм-вимагачів; рятувальний режим (при виявленні загрози, яку неможливо видалити, пропонує перезавантажити комп'ютер в режимі відновлення)); функціональність (підтримка VPN, безпечний онлайн-банкінг, захист вебкамери, захист соціальних мереж).

У зв'язку з тим, що відповідно до Указу № 549 від 16 вересня 2015 року, програмні продукти російських виробників ЗАТ «Лабораторія Касперського», дочірнього підприємства в Україні ТОВ «Лабораторія Касперського» і ТОВ «Доктор Веб» заборонені до використання державними підприємствами, вони не були включені до переліку тих, що аналізуються.

Результати порівняльного аналізу антивірусних програм наведені у табл.

1.1.

Таблиця 1.1 – Порівняльний аналіз антивірусних програм

	Bitdefender	Norton	McAfee	BullGuard	Panda
Рейтинг	9,8	9,3	9,2	9,2	9,0
Виявлення ШПЗ	Відмінно	Відмінно	Відмінно	Добре	Задовільно
Продуктивність	Відмінно	Добре	Відмінно	Добре	Добре
Ціна/якість	Відмінно	Добре	Добре	Задовільно	Добре
Захист					
В режимі реального часу	+	+	+	+	+
Видалення ШПЗ	+	+	+	+	+
Захист від програм-вимагачів	+	+	—	—	+
Рятувальний режим	+	—	—	—	+
Функціональність					
VPN	+	+	—	—	+
Безпечний онлайн-банкінг	+	+	—	—	—
Захист вебкамери	+	+	—	—	—
Захист соціальних мереж	+	+	—	—	—
Ціна, \$ за рік	20	35	35	24	24

За даними табл. 1.1 безперечним лідером серед розглянутих програм є Bitdefender – антивірусний бренд, якому довіряють понад 500 мільйонів користувачів у 150 країнах. Бренд виграв численні антивірусні нагороди від провідних онлайн-лабораторій тестування, включаючи AV Comparatives, AV Test, PcMag та The Anti-Malware Testing Standard Organization [9].

На рис. 1.1 наведено вікно з функціональними можливостями Bitdefender.

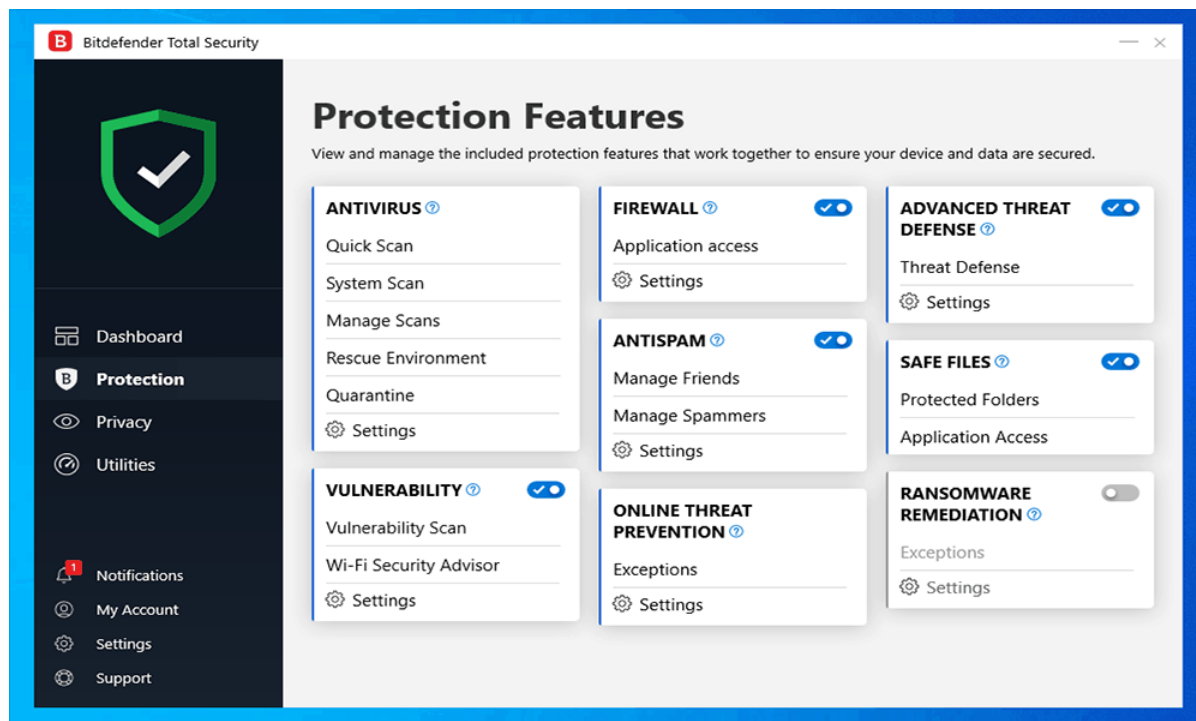


Рисунок 1.1 – Функціональні можливості програми Bitdefender

Проаналізувавши усі аналоги, було виявлено що більшість АПЗ мають високі показники захисту, кросплатформність, простоту інтерфейсу та зручність у використанні.

Серед недоліків, можливо виділити відносно велике споживання ресурсів системи, що може привести до зниження швидкості роботи ПК, а також відсутність україномовного інтерфейсу.

Отже, програмний продукт, що розроблюється має реалізувати позитивні сторони розглянутого АПЗ, а також усунути їх недоліки. Взявши до уваги усі перераховані переваги та недоліки існуючого програмного забезпечення необхідно розробити технічне завдання на свій застосунок.

1.3 Розробка технічного завдання

1.3.1 Підстава для розробки

Підставою для розробки служить завдання на дипломну кваліфікаційну роботу бакалавра на тему "Розробка програми антивірусного сканера для операційної системи Windows", затверджене наказом № 103 від 30 березня 2021 р. по НУ «Запорізька політехніка».

1.3.2 Мета розробки та її призначення

Метою розробки є створення програмного забезпечення, яке призначене для виявлення ШПЗ на пристрої користувача, видалення ШПЗ, та захисту доступу до особистих даних у теках.

1.3.3 Основні вимоги до програмного продукту

Програмне забезпечення, що розроблюється має надавати користувачеві наступні можливості:

- сканування комп'ютерних файлів сигнатурним методом з використанням алгоритму хешування MD5;
- виявлення інфікованого файлу;
- обрання режиму сканування окремого файлу;
- обрання режиму сканування папки;
- можливість запису програмою результатів сканування у файл з розширенням .txt;
- можливість користувачу подивитися результати сканування записані у .txt файл;
- заборона та відкриття доступу до папки;
- видалення інфікованих файлів;

- запис логу сканування програми до .txt-файлу, з фіксацією часу та дати сканування;
- оптимізація дискового простору шляхом виявлення однакових файлів (з однаковими хеш-даними файлу).

Вхідними даними для ПЗ є обрані користувачем для сканування файли або папки на ПК та шлях до них, а також пароль при доступі до блокування та розблокування папки.

Вихідними даними є хеш файлу, назва та шлях до інфікованих файлів у папці, інформація про блокування папки, інформація про розблокування папки, історія сканування, архів історії сканування, результати антивірусного тестування, інформація про результати видалення файлу.

1.3.4 Вимоги до технічних засобів, операційної системи і мови програмування

Застосунок, що розробляється має працювати під керуванням ОС Windows 10. Для забезпечення нормальної роботи застосунку та ОС рекомендована наступна конфігурація комп'ютера [10]:

- 64-х бітний процесор з тактовою частотою ядра понад 1 ГГц;
- 4 Гб оперативної пам'яті;
- відеокарта з 128 Мб RAM, підтримкою DX9 і WDDM;
- більше 128 Гб місця на системному розділі;
- підтримка завантаження з USB або наявність DVD-приводу;
- маніпулятори вводу (клавіатура, миша).

Для розробки програми необхідно використовувати мову C#, програмну платформу .NET з інтерфейсом програмування застосунків Windows Forms та середовище розробки Microsoft Visual Studio 17.

1.3.5 Вимоги до інформаційної та програмної сумісності

ПЗ, що розроблюється повинно підтримувати спільний режим роботи з ОС Windows 10.

1.3.6 Вимоги до програмної документації

Програмна документація має включати: керівництво програміста, керівництво оператора, опис програми та тексти основних програмних модулів.

2 МАТЕРІАЛИ ТА МЕТОДИ

2.1 Аналіз технічного завдання. Алгоритм M5D Hash

Відповідно до пред'явлених у технічному завданні вимог до АПЗ, програмний продукт, що розроблюється має надавати користувачеві можливість виявляти та видаляти інфіковані файли, а також знаходити такі файли у окремих теках. На теперішній час більшість мов програмування (наприклад, C, C++, Java, Python, Ruby і т. д.) мають інструментарій для роботи с файлами. Але у цих мовах програмування відсутні вбудовані функції хешування даних, що є основою роботи антивірусу, що розробляється.

Проводячи аналіз вже існуючих алгоритмів хешування даних, вибір було зроблено на користь алгоритму M5D Hash. Суть алгоритму полягає в тому, що програма на вхід приймає масив будь-якого розміру (тобто файл користувача), а на виході отримуємо строку с хешем який дорівнює 128 бітам. Тобто програма ніби створює цифровий відбиток («дайджест») файлу, який є набором символів довжиною 128 біт, і цей набір є унікальним ідентифікаційним номером файлу, який потім буде порівнюватись з бібліотекою вірусних сигнатур. Якщо буде знайдений збіг, то файл вважається інфікованим. Видалення файлу буде відбуватися при виконанні звичайної вбудованої функції у мові C#.

У технічному завданні також було зазначено, що програма має надавати можливість блокувати доступ до теки. Рішенням поставленої задачі, є надання програмно доступу до теки тільки адміністратору. У цьому разі, тільки адміністратор системи, може отримати доступ до цієї папки. Будь-яка інша людина отримає дозвіл тільки якщо блокування буде знято програмно. Для цього в інтерфейсі застосунку буде реалізовано дві кнопки «Надати доступ» та «Закрити доступ». Можливість користуватися цим функціоналом буде у людини, яка знає пароль. Пароль буде зберігатися у окремому файлі, у якому його можна буде змінити.

Для того, щоб реалізувати можливість передивитись історію сканування та історію дії з програмою, пропонується програмно записувати кожну дію користувача, та відображати її у відповідному вікні. Поруч з вікном будуть розташовані дві кнопки «Записати» та «Передивитись». Якщо користувач хоче записати всі свої дії у архів, то йому потрібно бути натиснути кнопку «Записати», інакше в архів історія не запишеться. Там же у програмі він зможе цю історію передивитися. Запис буде відбуватися у звичайний .txt файл.

Розглянемо алгоритм M5D Hash який є основою для пошуку вірусів у ПЗ, що розробляється.

Почнемо з припущення, що у нас є b -бітове повідомлення в якості вхідних даних, і що необхідно знайти дайджест цього повідомлення. Зазначимо, що b – довільне невід'ємне ціле число. Крім того, b може бути нулем (тобто довжина повідомлення дорівнює нулю), воно не обов'язково має бути кратним восьми, а також воно може бути яким завгодно великим. Нехай частини повідомлення представлені наступним чином:

$$m_0 \ m_1 \ \dots \ m_{\{b-1\}} \quad (2.1)$$

Далі, для обчислення дайджесту повідомлення виконуються наступні п'ять кроків [11].

Крок 1. Вирівнювання потоку. Повідомлення "доповнюється" (вирівнюється), так що його довжина (в бітах) була порівнянна з 448 по модулю 512. Тобто повідомлення розширюється так, що його довжина буде всього лише 64 біта і не буде кратною 512 біт. Заповнення виконується наступним чином: спочатку дописують одиничний біт в кінець повідомлення, а потім додаються біти "0", щоб довжина в бітах доповненого повідомлення стала конгруентним 448 по модулю 512. У цілому, при вирівнюванні додається мінімум один біт і максимум 512 біт. Вирівнювання виконується завжди, навіть якщо довжина повідомлення вже відповідає 448 по модулю 512.

Крок 2. Додавання довжини повідомлення. 64-бітове представлення b (довжина повідомлення до додавання бітів вирівнювання) додається до результату попереднього кроку. Тільки у тому малоймовірному випадку, якщо b більше 2^{64} , використовуються молодші 64 біта b . Ці біти додаються як два 32-бітові слова. Молодше слово додається першим. На цьому етапі результуюче повідомлення (після заповнення бітами і b) має довжину, кратну 512 бітам. Послідовність, що отримали назвемо S .

Нехай $M[0 \dots N-1]$ позначає слова результуючого повідомлення, де N кратне 16.

Крок 3. Ініціалізація MD-буферу. Буфер з чотирьох слів (A, B, C, D) розміром по 32 біта використовується для обчислення дайджеста повідомлення. Кожне з подвійних слів A, B, C, D ініціалізується наступними шістнадцятирічними значеннями, де першим розташовується наймолодший байт:

```
word A: 01 23 45 67;
word B: 89 ab cd ef;
word C: fe dc ba 98;
word D: 76 54 32 10;
```

У цих словах будуть зберігатися результати проміжних обчислень.

Крок 4. Обчислення в циклі. Спочатку визначимо чотири допоміжні функції, кожна з яких приймає на вхід три 32-бітних слова, а на виході виводить одне 32-бітове слово:

$$F(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$

$$G(X, Y, Z) = (X \wedge Y) \vee (Y \wedge \neg Z)$$

$$H(X, Y, Z) = X \oplus Y \oplus Z$$

$$I(X, Y, Z) = Y \oplus (X \vee \neg Z)$$

Ці чотири функції використовуються для чотирьох раундів обчислень.

У кожній біткової позиції F діє як умова: ЯКЩО X , то Y , інакше Z . Функцію F можна було б визначити за допомогою операції «+» замість « $\vee$ »,

оскільки $(X \wedge Y)$ та $(\neg X \wedge Z)$ ніколи не буде мати «1» в одній і тій же бітовій позиції.

Функції G , H та I схожі на функцію F в тому, що вони діють "порозрядно паралельно" для отримання виводу з бітів X , Y і Z таким чином, що якщо відповідні біти X , Y і Z незалежні і незміщені, то кожен біт $G(X, Y, Z)$, $H(X, Y, Z)$ та $I(X, Y, Z)$ буде незалежним і незміщеним.

Для обчислень використовується ще одна функція – W . Вона обробляє дані шляхом виконання серії простих перетворень і повертає результат.

На цьому кроці також використовується 64-елементна таблиця $T[1 \dots 64]$, побудована з синусоїдальної функції. Позначимо через $T[i]$ i -й елемент таблиці, який дорівнює цілій частини 4294967296 , помноженої на $\text{abs}(\sin(i))$, де i – значення в радіанах.

Далі у чотирьох раундах обчислень виконують наступні кроки:

Крок 4.1. Запам'ятовуємо перші 512 біт послідовності S .

Крок 4.2. Видаляємо перші 512 біт послідовності S (можна обійтися і без видалення, але тоді на першому кроці треба брати не перші 512, а наступні 512 біт).

Крок 4.3. Викликаємо функцію W . Параметри A , B , C , D – це поточні значення відповідних подвійних слів. Параметр T – 512 біт, які було запам'ятовано.

Крок 4.4. Додаємо до A $A0$.

Крок 4.5. $B = B + B0$.

Крок 4.6. $C = C + C0$.

Крок 4.7. $D = D + D0$.

Крок 4.8. Якщо довжина послідовності 0, виходимо.

Крок 4.9. Переходимо до Крок 4.1.

Крок 5. Результат обчислень знаходиться в буфері $ABCD$, це і є хеш. Якщо вивести слова в зворотному порядку $DCBA$, то отримаємо MD5 хеш.

Необхідно відзначити, що даний алгоритм також може застосовуватися для перевірки того, що передача даних на сервер відбулася благополучно. Щоб

перевірити цілісність переданих файлів між серверами, необхідно згенерувати хеш MD5 на обох сторонах сервера і перевірити, чи збігаються вони. Якщо значення хеш-функції збігається, то передача файлу була успішною.

Інше використання MD5 – одностороннє шифрування паролів. На будь-якому сайті, з метою захисту, потрібно зберегти пароль у вигляді відмінному від внутрішнього формату його збереження в базі даних. Для цього використовується MD5 або будь-яка інша функція хешування для генерації хеш-значення вихідного пароля.

Зазвичай розробники проєктують свої програми таким чином, щоб після реєстрації на вебсайті пароль користувача зберігався у вигляді хеш-значення всередині бази даних. При вході в систему пароль буде знову хешуватися і зіставлятися з хеш-значенням в базі даних. Якщо вони збігаються, користувач може отримати доступ. Таким чином, якщо хакер зможе зламати базу даних, він зможе побачити тільки хеш-значення пароля, яке не дозволить йому увійти в аккаунт користувача.

2.2 Обґрунтування вибору операційної системи та мови програмування

Для розробки АПЗ буде використовуватись ПК з встановленою Windows 10 та .Net Framework версії 4.7.2, а також мовою програмування C#. Системні характеристики ПК: екран 15.6 "(1366x768) HD LED, глянцева / Intel Core i3-5005U (2.0 ГГц) / RAM 8 ГБ / HDD 1 ТБ / nVidia GeForce 920MX, 2 ГБ / без ОД / LAN / Wi-Fi.

ОС для керування даним застосунком буде слугувати Windows 10. Причиною цього є невеликі системні вимоги для комфортної розробки антивірусного застосунку, підтримання середовища розробки .Net Framework (середовище розроблено корпорацією Microsoft спеціально для мови C#), та слабкий захист цієї ОС від вірусів, що надасть більше можливостей для тестування АПЗ.

Для написання антивірусної програми є важливою не так мова програмування, як можливість постійно оновляти бази вірусних сигнатур, адже протягом дня може з'явитись до 10 нових вірусів. Для розробки АПЗ може підійти будь-яка мова програмування, що компілюється.

Основними ж критеріями для обрання мови програмування під час розробки АПЗ є: наявність інструментів для роботи з файлами, швидкість відпрацювання функцій програми. Не менш важливими критеріями як і при розробці АПЗ так і при розробці будь-якого іншого продукту з графічними інтерфейсом є наступні: великий та зручний інструментарій для створення користувацького інтерфейсу, легкість написання коду. Також для комфорту розробника, повинна бути можливість відслідковування помилок. Порівняння мов програмування за цими критеріями наведено у табл. 2.1.

Таблиця 2.1 – Порівняння мов програмування

Критерії для порівняння	Назви мов програмування		
	C++	C#	Assembler
Наявність інструментів для роботи з файлами	+	+	—
Легкість у відслідковуванні помилок	+	+	—
Легкість написання коду(що прискорює процес розробки)	—	+	—
Швидкість відпрацювання функцій програми	+	—	+

На основі порівняльних даних з таблиці, з погляду зручності у розробці як системних функцій так і розробки програмного інтерфейсу, C# буде найбільш оптимальним для розробки. Програма не буде займати великий обсяг оперативної пам'яті, а також і не буде споживати великих ресурсів комп'ютера, тому потреба у C++, який важче у написанні коду ніж C#, а також у низкорівневій мові програмування Assembler відпадає, також можливості створення графічного інтерфейсу на C++ та Assembler значно менші, ніж на C#.

Графічний інтерфейс застосунку буде виконано з використанням Windows Forms, який на відміну від найближчого аналога WPF (Windows Presentation Foundation) є менш перевантаженим та надає достатній функціонал для реалізації поставлених задач.

Весь функціонал АПЗ (сканування файлу, сканування теки, блокування доступу до теки, видалення файлу, історія сканування) буде розроблений на фреймворку .Net Framework, що надає можливість хешування файлів за допомогою бібліотеки System.Security.Cryptography. Ця бібліотека надає весь необхідний інструментарій для роботи з хешами файлів, що є основою в алгоритмі виявлення інфікованих файлів.

2.3 Визначення компонентів системи

Так як у складі програмного забезпечення, що розробляється не буде ні бази даних, ні серверу, то програмний застосунок буде складатись з одного модуля – Desktop застосунка. Проектуючи структуру ПЗ, можна розділити Desktop застосунок на компоненти та модулі програми. Компоненти програми забезпечують функціональні можливості АПЗ, тим часом як модулі програми це окремі режими які можуть об'єднувати у собі декілька компонентів які були реалізовані згідно з функціональними вимогами, наведеними у п. 1.3.3 технічного завдання.

Модель системи, що проектується у вигляді діаграми компонентів наведена на рис. 2.1.

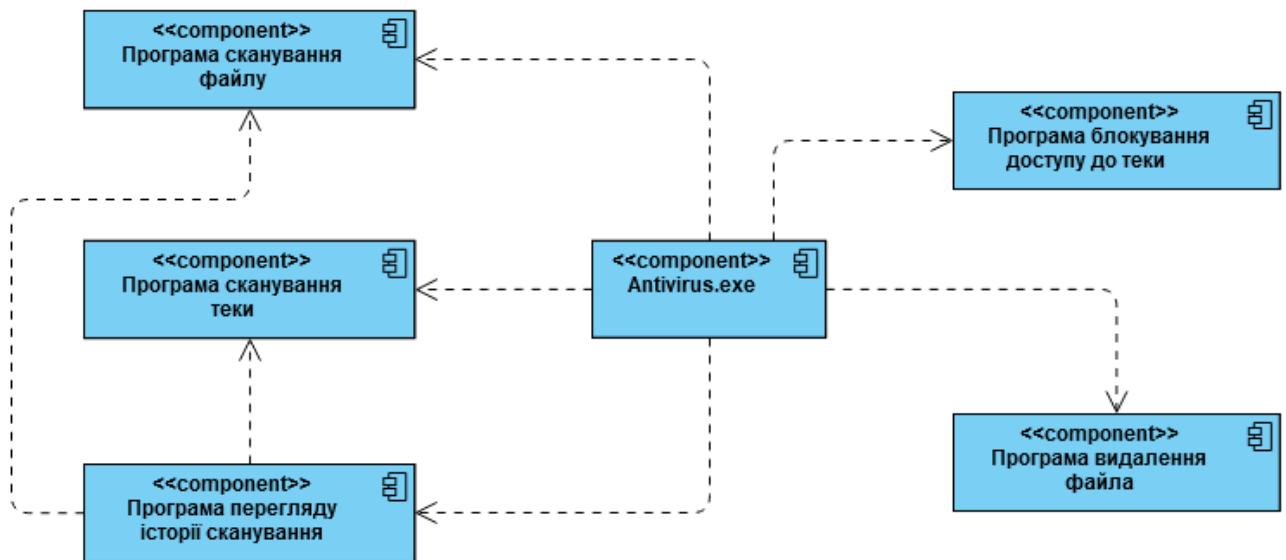


Рисунок 2.1 –Діаграма компонентів застосунку

Як випливає з рис. 2.1, всі компоненти є залежними від компонента «Antivirus.exe», так як в їх методах здійснюється виклик методів цього компонента (незалежного). Крім того, компонент «Програма перегляду історії сканування» є залежним від компонентів «Програма сканування файлу» та «Програма сканування теки» так як цей компонент здійснює читання з них даних.

Перелік функціональних можливостей, що реалізуються компонентами програми:

- сканування комп'ютерних файлів;
- виявлення інфікованого файлу;
- обрання файлу для сканування;
- обрання теки для сканування;
- запис програмою у файл формату .txt результатів сканування та імені файлу;
- перегляд результатів сканування записаних у .txt файлі;
- заборона або відкриття доступу до теки;
- видалення інфікованого файлу;

- запису логу сканування програми до .txt-файлу, та фіксація часу та дати сканування;
- оптимізація дискового простору шляхом виявлення однакових файлів (файлів з однаковими хеш-даними).

Вхідною точкою доступу до компонентів програми та її інтрефейсу, яка є зрозумілою та доступною користувачу є desktop-застосунок. Розглянемо перелік модулів, що входять до складу desktop-застосунку, що проєктується:

- програма сканування файлу реалізує функціональність сканування певного файлу на предмет того, чи є він інфікованим (шкідливим для системи). Програма буде проводити хешування файлу, після цього отриманий хеш порівнює з хешами інфікованих файлів, які у наявності у .txt файлі, який розташовується у кореневій теці проєкту. Вона містить у собі такі компоненти: сканування комп'ютерних файлів, можливість виявлення інфікованого файлу, можливість обрати файл для сканування, оптимізація дискового простору шляхом виявлення однакових файлів;
- програма сканування теки реалізує функціональність сканування певної директорії, на предмет того, чи є в теці інфікований файл, або декілька інфікованих файлів. Відпрацьовує програма аналогічно програмі сканування файлу, тільки для всіх файлів в директорії. Вона містить у собі такі компоненти: можливість обрати теку для сканування, можливість виявлення інфікованого файлу;
- програма блокування доступу до теки реалізує функціональність обмеження доступу до певної теки. Доступ до цієї функціональності може отримати користувач який знає пароль. Він може як ставити блок на теку, так і знімати блок з неї. Містить у собі такі компоненти: заборонити та відкрити доступ до теки або файлу.
- програма видалення файлу реалізує функціональність видалення певного файлу з пристрою користувача. Містить у собі такі компоненти: можливість видаляти інфікований файл;

– програма перегляду історії сканування реалізує функціональність перегляду історії дій користувача у програмі. Містить у собі такі компоненти: запис програмою у .txt файл результатів сканування та ім'я файлу, можливість продивитись результати сканування, які записані у .txt файл, запис логу сканування до txt файлу, та фіксація дати та часу сканування.

Подібна структура не є шаблонною, але дуже скоротить час на розробку, особливо у випадку коли не планується подальше розширення функціональності програмного продукту (або планується не дуже часто). Але, якщо планується розробка продукту який буде постійне підтримуватись та оновлюватись, більш оптимальною буде модульна структура коду.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Розробка алгоритму роботи програми

Визначення основних вимог до ПЗ користувача має велике значення під час проєктування графічного інтерфейсу програмного забезпечення. Це дає змогу визначити найбільш важливі для нього функції та розташувати елементи інтерфейсу які наділені функціональністю програми, не перевантажуючи інтерфейс. Логічно припустити, що найважливіше в АПЗ це виявлення ШПЗ, тому по-перше, потрібно найближче розташувати кнопки які відповідають за сканування файлу та сканування теки, а вже потім повинні розташовуватись кнопки з іншою функціональністю (блокування теки, видалення файлу, історія сканування).

Виявлення вірусу за допомогою програми відбувається шляхом порівняння хешу відсканованого файлу з тими хеш даними, які містяться у txt файлі, що знаходиться у кореневій папці проєкту. З початку програма приймає файл який необхідно відсканувати. Після того як користувач обрав файл для сканування, програма за допомогою алгоритму Md5 Hash, який міститься у окремій функції програми, розбиває структуру вмісту файлу на рядок довжиною у 128 біт, що і є хешем нашого файлу. Після цього, відпрацьовує функція порівняння хешу відсканованого файлу, з хешами які містяться у txt файлі з хешами ШПЗ. Після порівняння, програма генерує відповідне повідомлення напроти надпису «статус»: чи є файл інфікованим, або ні. У випадку зі скануванням теки, виводиться шлях та назва інфікованого файлу. Блок-схема алгоритму роботи програми наведена на рис. 3.1.

Блокування доступу до теки відбувається завдяки блокуванням доступу до системи самою ОС. При вводі паролю у відповідному рядку, користувачу надається можливість блокувати та відкривати доступ до теки, якщо пароль не вірний, користуватися цими функціями користувач не зможе.

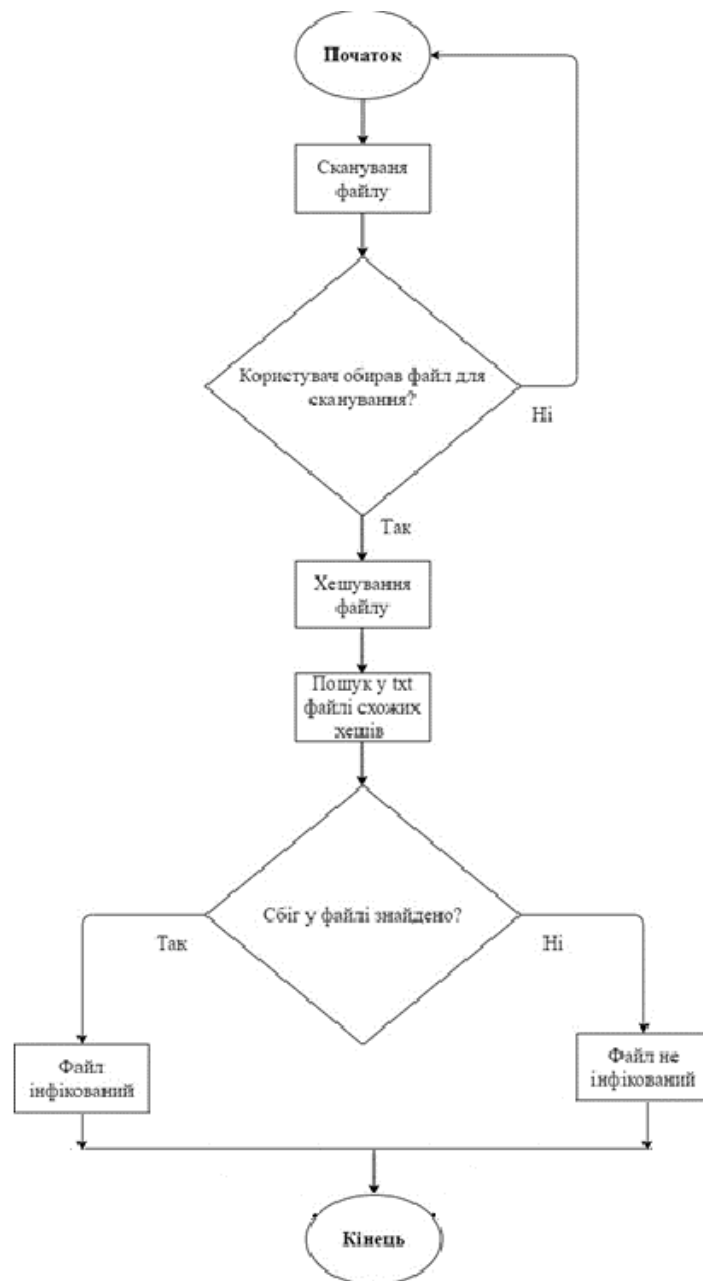


Рисунок 3.1 – Блок-схема алгоритму роботи програми

Спочатку користувач обирає теку до якої він хоче заблокувати доступ. Потім, після обрання теки, користувач повинен у відповідному рядку ввести пароль. Програма перевіряє пароль який був введений, з паролем який зберігається у .txt файлі. Довжина пароля не повинна перевищувати 12 символів. Якщо пароль правильний, програма надає доступ до функції блокування та відкриття доступу. Блокування відбувається на рівні системи, сама ОС програмно надає доступ до файлу лише адміністратору, і тому ніхто інший крім нього не може скористатися цією текою. Розблокування проходить

так само, на рівні ОС знімається доступ тільки адміністратором, й текою можуть користуватись усі користувачі. При закінченні роботи програми блокування не знімається, його можливо буде зняти тільки через посередництво програми. Блок-схема алгоритму блокування теки наведена на рис. 3.2.

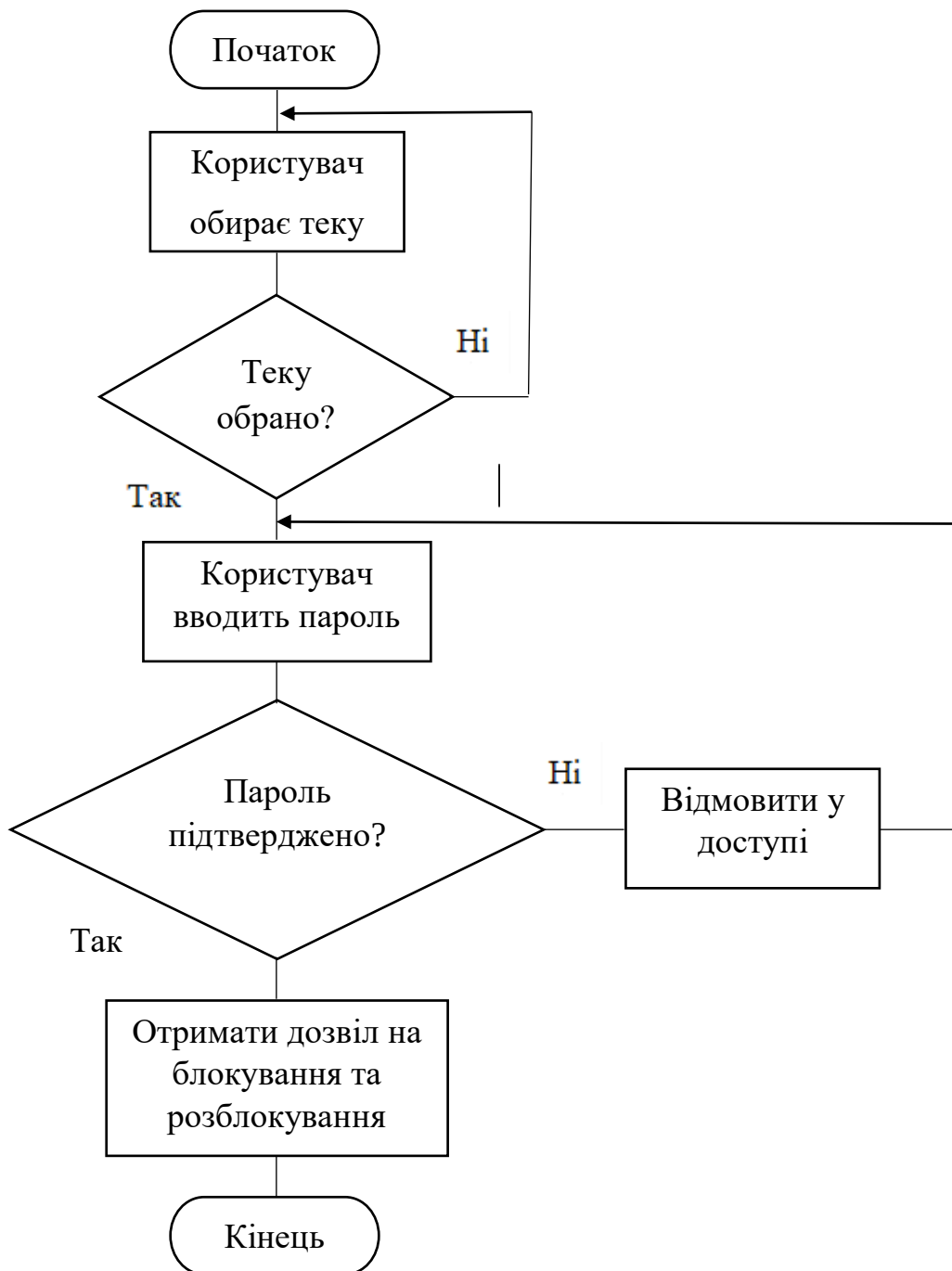


Рисунок 3.2 – Блок-схема алгоритму блокування теки

Функція видалення файлу програми (вона базується на вбудованій в бібліотеці File функції Delete) дозволяє видалити будь-який файл з пристрою. Кожна дія, яку користувач виконує під час роботи з АПЗ, записується у історію сканування, яку користувач може передивитися та яку він може за своїм бажанням записати. Історія яка знаходиться у рядку програми, буде відображати тільки ті дії які користувач зробив протягом однієї сесії. У випадку, коли користувач хоче записати історію в архів для подальшого перегляду, йому потрібно натиснути кнопку «Записати». Після натискання, всі дані з рядка записуються у txt-файл з назвою scanlog, який знаходиться у кореневій теці проєкту.

3.2 Розробка Desktop-застосунку

Оскільки обрана мова програмування C# є виключно об'єктно-орієнтованою, то все функціонування програмного забезпечення будується на взаємодії окремих класів та сутностей предметної області.

Під час виконання дипломної кваліфікаційної роботи, було вирішено обійтись без участі користувацьких класів, так як усі потрібні функції були вже реалізовані у певних бібліотеках. І хоча з точки зору чистоти коду правильніше було би реалізацію деяких алгоритмів винести в окремі класи, а потім посилатися на них, як вже було відзначено у попередніх розділах, такий підхід до написання коду зекономить програмісту досить багато часу.

Будь-який програмний застосунок, що розроблюється за допомогою фреймворку Windows Forms обов'язково складається з представлення (форми) та контролера (оброблювачів подій форми, наприклад, натиснення певної кнопки).

Усередині оброблювачів подій використовуються методи та об'єкти певних системних класів які містяться у бібліотеках C#, та відповідають за реалізацію та логіку роботи програмного застосунку, та передачі даних до системи, де вже на основі отриманих даних, користувач отримає певний

результат (результати антивірусного сканування у нашому випадку). Також, щоб уникнути великої кількості форм, та передачі даних між ними, було прийняте рішення зробити контекстні меню у вигляді панелей, які розташовані на формі. Такий підхід прискорює роботу програми, та робить перехід між контекстними меню більш плавним, так як все знаходиться на одній формі.

У програмі наявний один клас форми Form1, в якому зберігається реалізація усього функціоналу АПЗ. Уся обробка вхідних та вихідних даних програми буде проводиться у цьому класі форми. Опис вищезазначеного класу в нотатії мови UML наведений на рис. 3.3.

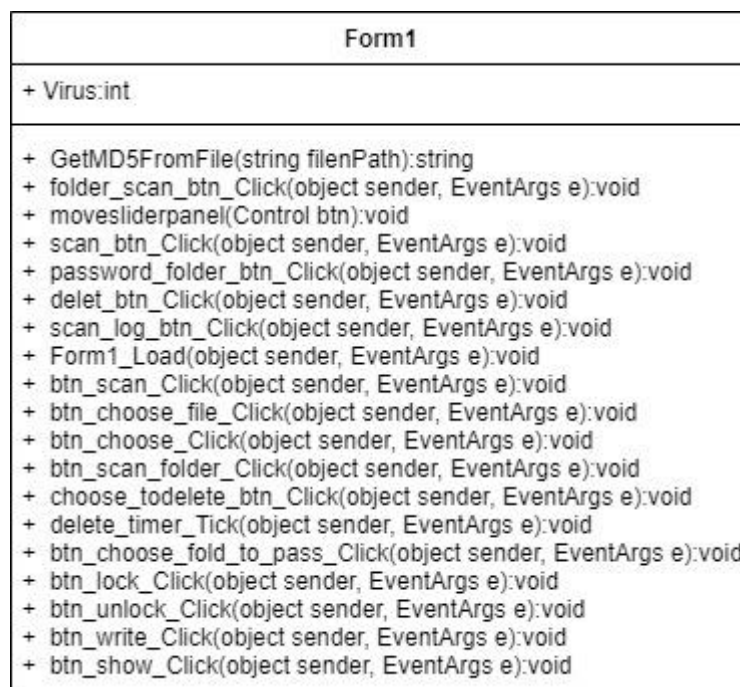


Рисунок 3.3 – Опис класу Form1

Основні методи класу Form1 наступні:

- GetMD5FromFile – метод, який містить у собі алгоритм хешування файлів;
- btn_scan_Click – метод, який реалізує сканування файлу на предмет знаходження у ньому ШПЗ;
- btn_scan_folder_Click – метод, який реалізує сканування теки на предмет знаходження в ній ШПЗ;

- choose_todelete_btn_Click – метод, який реалізує видалення файлу з системи;
- btn_lock_Click – метод, який реалізує блокування доступу до теки;
- btn_unlock_Click – метод, який реалізує розблокування доступу до теки.

3.3 Розробка графічного інтерфейсу користувача

Як вже було зазначено вище, весь інтерфейс програми буде знаходитись в одній формі класу Form1, а вибір режиму програми буде здійснюватися при натисканні на відповідну кнопку, з наступною появою контекстного меню того режиму, який обрав користувач.

При запуску програми, користувачу за замовченням надається режим сканування папки та головне меню з можливістю вибору режимів програми (рис. 3.4).

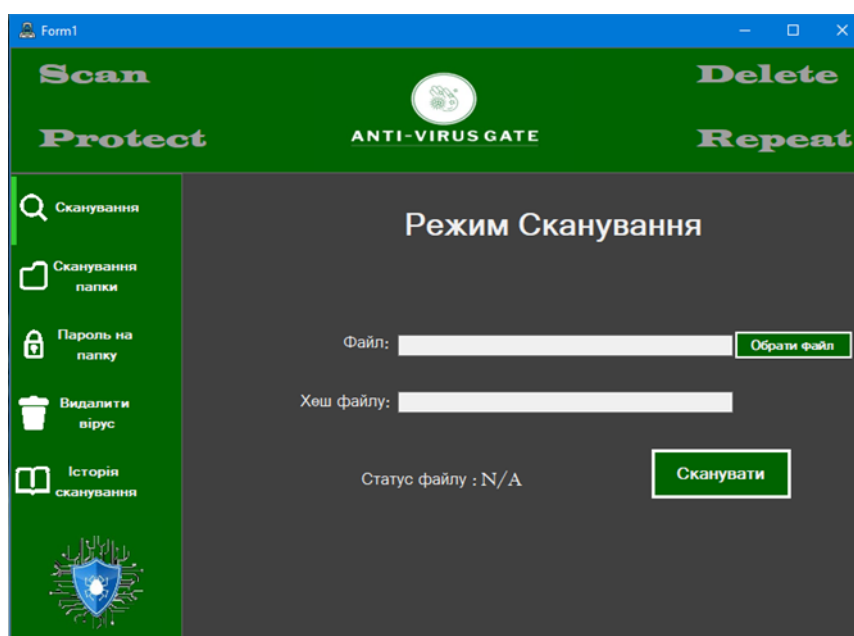


Рисунок 3.4 – Головне меню програми

Для відображення форми при запуску програми, використовується метод `Form1_Load` який задає стартову позицію цього режиму. Реалізація методу:

```
panel_scan_btn.Location = new Point(161, 118);
```

За допомогою об'єкту класу `Control(Location)` створюється екземпляр класу `Point`, який містить у собі координати положення меню режиму, та ініціалізує початкове положення меню при запуску програми.

Для сканування файл на предмет того чи є він інфікований, використовується обробник події натискання на кнопку `btn_scan_Click`. Реалізація методу:

```
var md5signature = File.ReadAllLines("MD5Base.txt");
var coincidences = File.ReadAllLines("coincidences.txt");
string line = "Хеш файлу: " + Environment.NewLine+tbmd5.Text ;
System.IO.File.WriteAllText("coincidences.txt", line);
if (coincidences.Contains(tbmd5.Text))
{
    MessageBox.Show("Знайдений схожий за своєю структурою файл.Радимо видалити його для оптимізації місця.");
}
if (md5signature.Contains(tbmd5.Text))
{
    lb_status.Text = "Файл інфікований";
    lb_status.ForeColor = Color.Red;
    tb_scanlog.Text += " " + lb_status.Text + "Дата та час: " + DateTime.Now.ToString("dd.MM.yyyy HH:mm:ss") + Environment.NewLine;
}
else
{
    lb_status.Text = "Файл чистий";
    lb_status.ForeColor = Color.LimeGreen;
```

```

        tb_scanlog.Text += " " + lb_status.Text +
        "Дата та час: " + DateTime.Now.ToString("dd.MM.yyyy HH:mm:ss") +
        Environment.NewLine;
    }

```

Спочатку іде перевірка на те, чи не збігаються з попереднім файлом їх хеші, якщо збіг програмою знайдений, то програма виводить на екран повідомлення "Знайдений схожий за своєю структурою файл. Радимо видалити його для оптимізації місця.».

Далі програма перевіряє, чи є у бібліотеці ШПЗ збіг за хешем файлу, якщо збіг відбувається, то програма виводить у рядок зі статусом певне повідомлення про те що файл інфікований, якщо збігів не знайдено, то виводить повідомлення про те що файл чистий.

У верхній частині головного меню, розміщені надписи з метою покращення дизайну тобто вони не є елементами керування, як і лого посередині та внизу. Також праворуч розміщені елементи керування які відповідають за виклик контекстного меню режиму, який хоче застосувати користувач. Було вирішено не перевантажувати інтерфейс програми зайвими елементами інтерфейсу, щоб спростити користування програмою та зробити інтерфейс більш лаконічним.

Всі елементи керування які не є декоративними підсвічується при їх наведенні та зліва з краю виділяються зеленим лейблом, для кодування таким чином у якому режимі знаходиться користувач, що ще більше полегшує користування програмою, та робить її інтерфейс інтуїтивно зрозумілим.

Як було зазначено вище, перший режим програми та його контекстне меню при запуску програмного застосунку називається «Сканування» (інтерфейс цього режиму зображено на рис. 3.4). При розробці меню сканування було використано наступні елементи керування:

- кнопки (для виконання операцій сканування та вибору файлу);
- поля з інформацією (шлях та назва файлу, та його хеш);
- інформаційні надписи (режим сканування, файл, хеш, статус);

– надпис що змінює свій надпис у залежності від статусу файлу (інфікований чи чистий).

Відповідно до загальних правил побудови ергономічних інтерфейсів були виконані наступні вимоги:

- всі елементи розташовані симетрично;
- в залежності від статусу файлу, надпис біля статусу позначається певним кольором (якщо файл інфікований то колір червоний, якщо файл чистий то зелений);
- елементи керування зі схожою функціональністю згруповані між собою;
- всі назви біля елементів керування повністю описують функціонал який вони виконують.

Другий режим можна викликати натиснувши кнопку «Сканування папки», що дозволяє перейти до контекстного меню з назвою «Режим сканування папки»(інтерфейс цього режиму наведено на рис. 3.5).

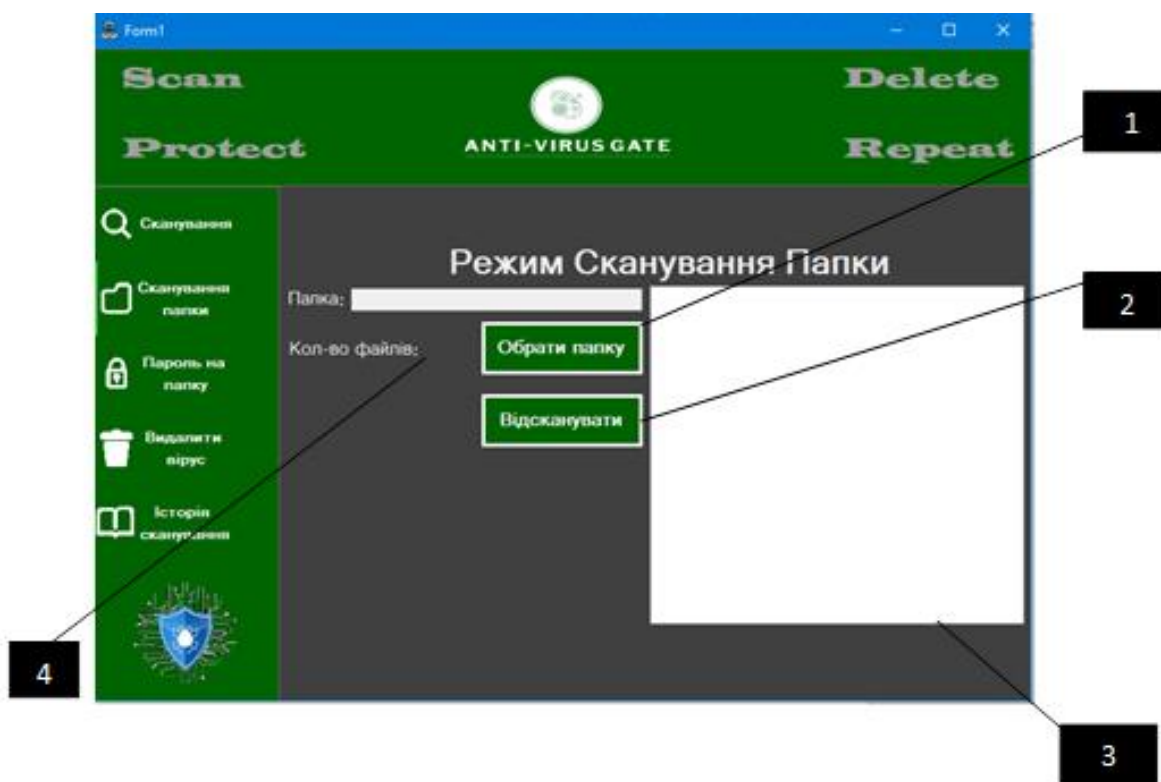


Рисунок 3.5 – Режим сканування папки

Для сканування папки на предмет того чи є в ній інфікований файл, використовується обробник події натискання на кнопку btn_scan_folder_Click. Реалізація методу:

```

        List<string> search =
Directory.GetFiles (@folderBrowserDialog.SelectedPath,
"*.*").ToList();
        var md5signature1 =
File.ReadAllLines ("MD5Base.txt");
        foreach (string item in search)
        {
            var s = GetMD5FromFile (item);
            if (md5signature1.Contains (s))
            {
                lb_count.Text = null;
                virus += 1;
                lb_count.Text = virus.ToString ();
                listBox.Items.Add (item);
                tb_scanlog.Text += "\n Кількість вірусів:
" + lb_count.Text + "Вірусні файли: " + item + "Дата та час: " +
DateTime.Now.ToString ("dd.MM.yyyy HH:mm:ss") +
Environment.NewLine;
            }
        }
    }

```

Спочатку програма у циклі хешує усі файли які знаходяться у папці, потім вона перевіряє кожен файл по черзі у папці на наявність збігів з бібліотекою ШПЗ. Якщо збіг знайдено, програма виводить у рядок назву та шлях до файлу який був інфікований, якщо збіг не знайдено, пропускає цей файл.

При розробці меню сканування папки були використані наступні елементи керування:

- кнопки (для виконання операцій сканування та вибору папки);
- поля з інформацією (назва та шлях до папки, кількість інфікованих файлів у папці);
- інформаційні надписи (режим сканування папки, папка, кількість файлів);

Відповідно до загальних правил побудови ергономічних інтерфейсів були виконані наступні вимоги:

- всі елементи розташовані симетрично;
- елементи керування зі схожою функціональністю згруповані між собою;
- всі назви біля елементів керування повністю описують функціонал який вони виконують.

Третій режим можна викликати натиснувши кнопку «Пароль на папку», що дозволяє перейти до контекстного меню з назвою «Поставити пароль на папку» (інтерфейс цього режиму наведено на рис. 3.6).

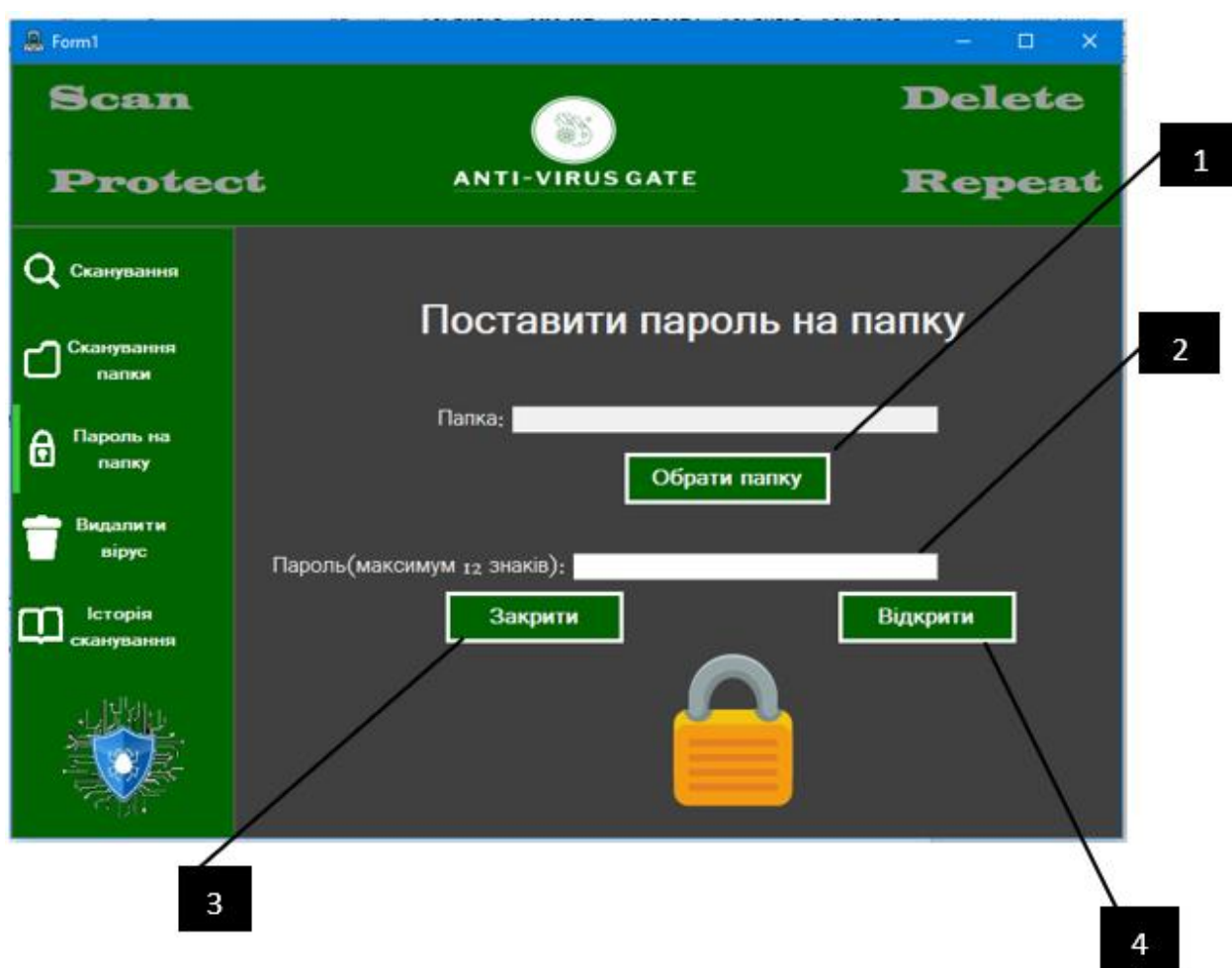


Рисунок 3.6 – Режим встановлення паролю на папку

Для блокування доступу до папки та її розблокування використовуються два обробника подій натискання на кнопку `btn_lock_Click` для блокування та `btn_unlock_Click` для розблокування. Реалізація `btn_lock_Click`:

```

var pass1 = File.ReadAllLines("pass.txt");
    if (pass1.Contains(tb_pass.Text))
    {
        try
        {
            string folderPath = tb_folder_sear.Text;
            string adminUserName = Environment.UserName;
            DirectorySecurity ds =
Directory.GetAccessControl(folderPath);
            FileSystemAccessRule fsa = new
FileSystemAccessRule(adminUserName, FileSystemRights.FullControl,
AccessControlType.Deny);
            ds.AddAccessRule(fsa);
            Directory.SetAccessControl(folderPath, ds);
            MessageBox.Show("Locked");
            tb_scanlog.Text += "\n ||Режим доступу: " +
folderBrowserDialog_pass.SelectedPath + "Закрито " + "Дата та час:
" + DateTime.Now.ToString("dd.MM.yyyy HH:mm:ss") +
Environment.NewLine;
            pb_pass.Image =
Properties.Resources.unlock_fold1;
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
    else
    {
        MessageBox.Show("Не вірний пароль");
    }
}

```

Реалізація btn_unlock_Click:

```

var pass2 = File.ReadAllLines("pass.txt");
    if (pass2.Contains(tb_pass.Text))
    {
        try
        {
            string folderPath = tb_folder_sear.Text;
            string adminUserName = Environment.UserName;//
getting your adminUserName
            DirectorySecurity ds =
Directory.GetAccessControl(folderPath);
            FileSystemAccessRule fsa = new
FileSystemAccessRule(adminUserName, FileSystemRights.FullControl,
AccessControlType.Deny);
            ds.RemoveAccessRule(fsa);
            Directory.SetAccessControl(folderPath, ds);
            MessageBox.Show("UnLocked");
        }
    }
}

```

```

        tb_scanlog.Text += "\n ||Режим доступу: " +
folderBrowserDialog_pass.SelectedPath + "Відкрито " + "Дата та
час: " + DateTime.Now.ToString("dd.MM.yyyy HH:mm:ss") +
Environment.NewLine;
        pb_pass.Image =
Properties.Resources.lock_fold1;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }

```

Як можна побачити, реалізація обробників подій майже однакова окрім того, що в методі `btn_lock_Click` за допомогою методу `ds.AddAccessRule(fsa)` додається доступ тільки адміністратору, а у `btn_unlock_Click` за допомогою методу `ds.RemoveAccessRule(fsa)` видаляється цей доступ.

Спочатку програма за допомогою методу класу `Directory.GetAccessControl` інкапсулює методи доступу до папки, потім створюється екземпляр класу `FileSystemAccessRule`, який приймає параметри `adminUserName`, `FileSystemRights.FullControl`, `AccessControlType.Deny`, ці параметри являють собою ім'я адміністратора, рівень контролю доступу та чи використовується екземпляр для блокування доступу чи ні. Потім програма додає доступ тільки адміністратору і за допомогою класу `Directory` та його методу `SetAccessControl` додає обрану папку до список контролю доступу ACL [12]. Після цього програма додає дію блокування чи розблокування до історії сканування за певну сесію, виводить повідомлення про стан файлу (заблоковуваний чи розблоковуваний), та міняє іконку замку в залежності від того закрита папка чи відкрита (якщо закрита то іконка замку закрита, якщо відкрита, то замок відкритий).

При розробці меню блокування доступу до папки були використані наступні елементи керування:

- кнопки (для виконання операцій блокування доступу, відкриття доступу, та вибору папки);
- поля з інформацією (назва та шлях до папки);
- поле для вводу даних (рядок за назвою пароль);

- інформаційні надписи (поставити пароль на папку, папка, пароль (максимум 12 знаків));
- зображення замку, що відображає, чи закритий користувач доступ до папки, чи відкрив його.

При розробці меню блокування доступу до папки було реалізовано вимоги до ергономічності інтерфейсу аналогічні попередньому меню.

Четвертий режим можна викликати натиснувши кнопку «Видалити вірус», що дозволяє перейти до контекстного меню з назвою «Видалити файл». Інтерфейс цього режиму наведено на рис. 3.7.

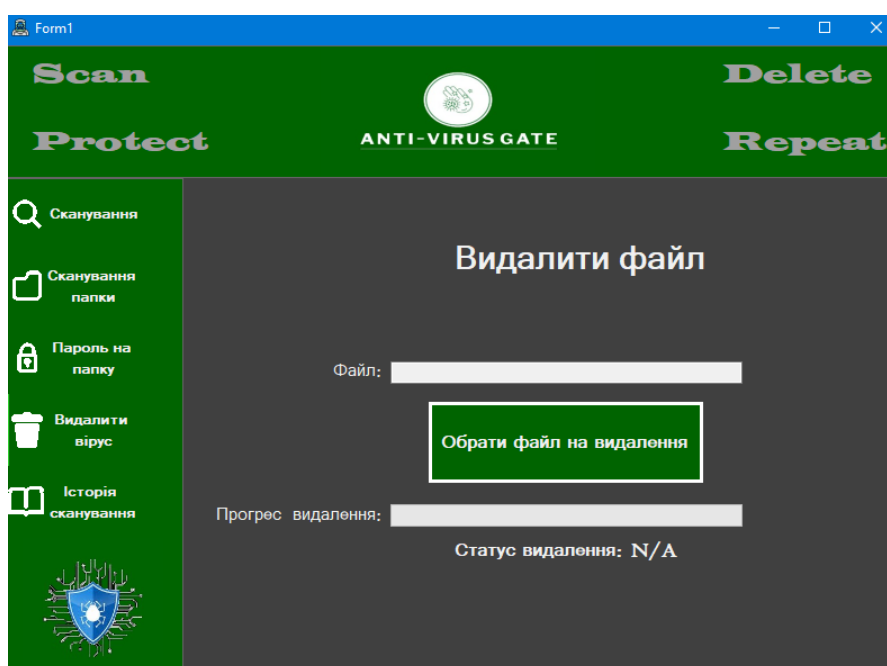


Рисунок 3.7 – Режим видалення файлу

У цьому режимі реалізовано шкалу прогресу, яка заповнюється згідно з тіками таймера. Метод який реалізує заповнення згідно з тіками таймера, а також реалізація таймера, називається `delete_timer_Tick`. Реалізація методу:

```
delete_timer.Interval = 10;
if (deleteprogres.Value >= deleteprogres.Maximum)
{
    deleteLbl.Text = "Успішно видалено";
    deleteLbl.ForeColor = Color.Green;
    delete_timer.Stop();
}
```

```

else
{
    deleteprogres.Value += 10;
}

```

Спочатку для таймеру задається інтервал, потім програма перевіряє, чи досягла шкала прогресу максимуму, якщо так, то виводиться повідомлення про успішне видалення, повідомлення міняє колір на зелений та таймер зупиняється. Якщо шкала прогресу не досягла максимуму, то значення шкали збільшується на 10.

При розробці меню видалення файлу були використані наступні елементи керування:

- кнопки (для виконання операцій обрання та видалення файлу);
- поля з інформацією (файл та шлях до нього, прогрес видалення);
- інформаційні надписи (видалити файл, статус видалення, прогрес видалення, файл);
- індикатор прогресу видалення.

Відповідно до загальних правил побудови ергономічних інтерфейсів були виконані вимоги аналогічні інтерфейсу «Сканування папки». Крім того, у залежності від статусу видалення файлу, надпис біля статусу позначається певним кольором (якщо файл видалений то з'являється зелений надпис успішного видалення).

Останній режим можна викликати натиснувши кнопку «Історія сканування», що дозволяє перейти до контекстного меню з назвою «Історія сканування». Інтерфейс цього режиму наведено на рис. 3.8).

При натисканні на кнопку «Продивитись» з'являється додаткове контекстне меню, а точніше файл з історією сканування сеансів, яку користувач захотів зберегти (рис. 3.9).

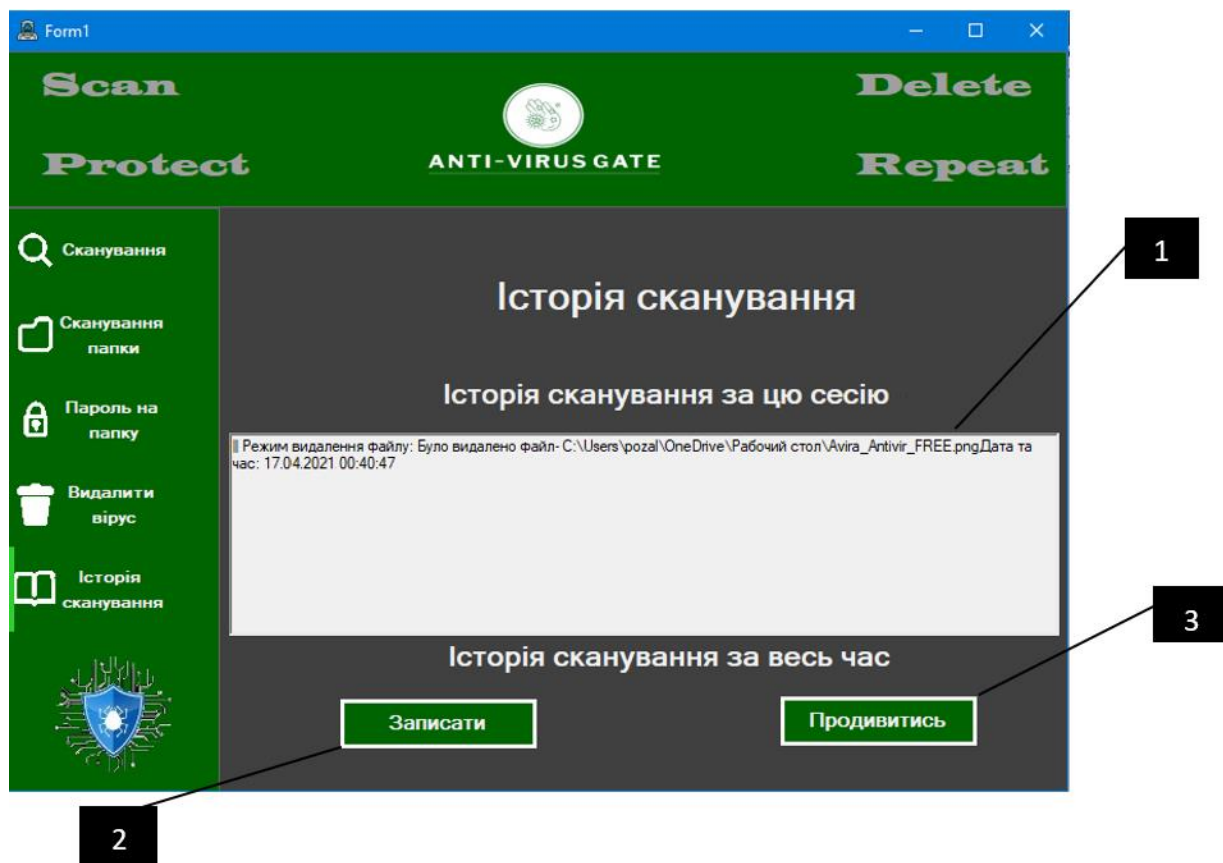


Рисунок 3.8 – Історія сканування

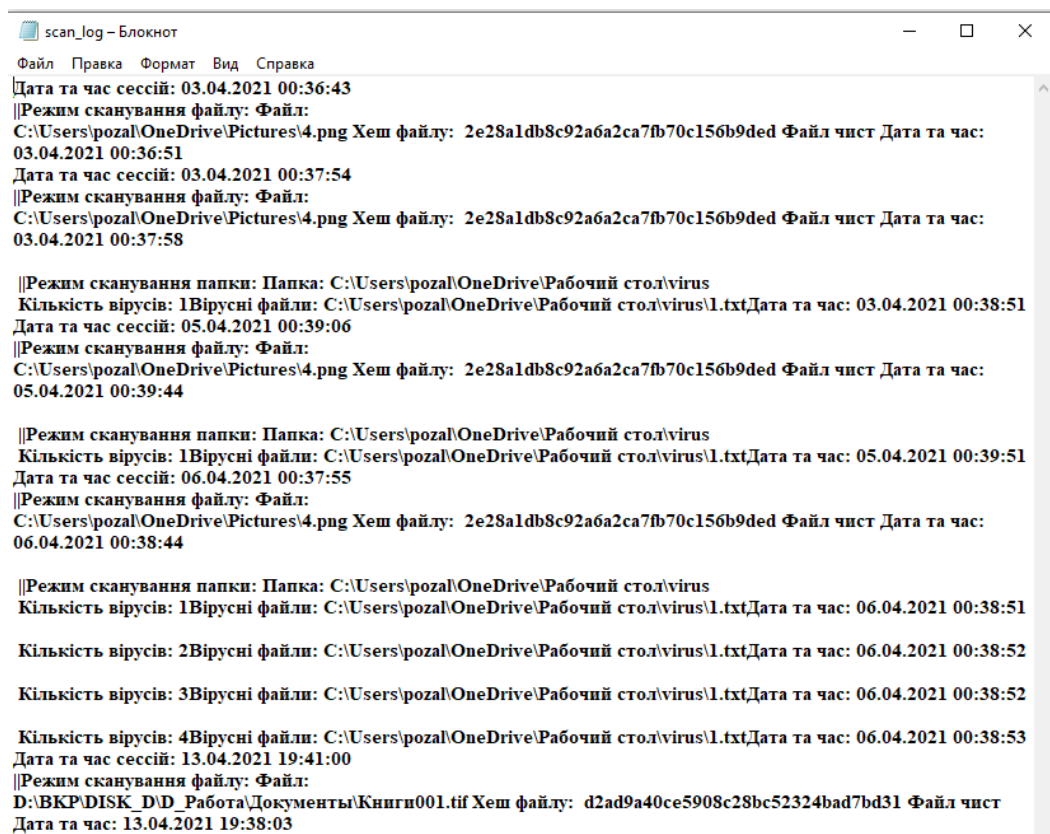


Рисунок 3.9 – Файл збережених історій сканування

При розробці меню історії сканування були використані наступні елементи керування:

- кнопки (для виконання операцій записати і продивитись);
- поля з інформацією (історія сканування даного сеансу);
- інформаційні надписи (історія сканування, історія сканування за цю сесію, історія сканування за весь час).

Відповідно до загальних правил побудови ергономічних інтерфейсів, при розробці меню історії сканування було дотримано вимог аналогічно інтерфейсу «Сканування папки».

Файли програми розробленого застосунку наведені у Додатку А.

Позначки на рис. 3.5, 3.6 та 3.8 щодо елементів інтерфейсу програми будуть пояснені у розділі «Керівництво оператора».

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення та умови застосування програми

Розроблене програмне забезпечення призначене для виявлення ШПЗ, захисту доступу до папок файлової системи та захисту комп'ютерної системи від ШПЗ.

Для коректного функціонування Desktop-застосунку необхідна версія ОС Windows 10.

4.2 Характеристики програми

Для виявлення ШПЗ у програмі застосовується сканування комп'ютерних файлів сигнатурним методом з використанням алгоритму хешування MD5.

Для запуску програмного забезпечення необхідно виконати файл Antivirus.exe. Програма не потребує та не підтримує виклик з додатковими аргументами у командному рядку.

Програмне забезпечення «відловлює» можливі помилки в роботі оператора, а також охороняє його від можливих помилок введення інформації або порушення логіки роботи. Наприклад, у режимі блокування доступу до папки, при спробі натиснути кнопку заблокувати без наявної папки, генерується системна помилка неприпустимого імені.

4.3 Вхідні та вихідні дані

Вхідними даними для ПЗ є обрані користувачем для сканування файли або папки на ПК та шлях до них, а також пароль при доступі до блокування та розблокування папки.

Вихідними даними є хеш файлу, назва та шлях до інфікованих файлів у папці, інформація про блокування папки, інформація про розблокування папки,

історія сканування, архів історії сканування, результати антивірусного тестування, інформація про результати видалення файлу.

4.4 Атрибути та методи класу Form1

Desktop-застосунок реалізовано у вигляді одного класу форми Form1. Перелік атрибутів та методів класу Form1 наведено у табл. 4.1.

Таблиця 4.1 – Атрибути та методи класу Form1

Форма	Атрибут/Метод	Тип даних	Короткий опис
1	2	3	4
Form1	GetMD5FromFile	string	Знаходить хеш обраного файлу
	folder_scan_btn_Click	void	Викликає режим сканування папки
	movesliderpanel	void	Логіка переміщення зеленої панелі при натисканні на кнопки
	scan_btn_Click	void	Викликає режим сканування файлу
	password_folder_btn_Click	void	Викликає режим блокування папки
	delet_btn_Click	void	Викликає режим видалення файлу
	scan_log_btn_Click	void	Викликає режим історії сканування
	Form1_Load	void	Визначає початкову позицію режиму сканування файлу при запуску програми
	btn_scan_Click	void	Оброблювач подій натискання на кнопку сканування файлу

Продовження таблиці 4.1

1	2	3	4
Form1	btn_choose_file_Click	void	Оброблювач подій натискання на кнопку вибору файлу для сканування
	btn_choose_Click	void	Оброблювач подій натискання на кнопку вибору папки для сканування
	btn_scan_folder_Click	void	Оброблювач подій натискання на кнопку сканування папки
	choose_todelete_btn_Click	void	Оброблювач подій натискання на кнопку видалення файлу
	delete_timer_Tick	void	Заповнення шкали прогресу видалення за певним інтервалом
	btn_choose_fold_to_pass_Click	void	Оброблювач подій натискання на кнопку вибору папки для блокування доступу
	btn_lock_Click	void	Оброблювач подій натискання на кнопку блокування папки
	btn_unlock_Click	void	Оброблювач подій натискання на кнопку розблокування папки
	btn_write_Click	void	Оброблювач подій кнопки запису історії сканування в архів
	btn_show_Click	void	Оброблювач подій кнопки відкриття архіву сканування

Назви методів-оброблювачів подій сконструйовані наступним чином: Назва об'єкта форми_Подія(об'єкт, аргументи події). Наприклад, `folder_scan_btn_Click(object sender, EventArgs e)`. Тип, що повертається – `void`. Зокрема, програмою обробляються події натискання кнопки (Click), відображення (Load) та проходження певного часу таймера (Tick).

4.5 Повідомлення програмісту

Повідомлень програмісту у програмі не передбачено.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення програми

Розроблене програмне забезпечення призначене для виявлення вірусних сигнатур у файлах та папках, захисту доступу до папок, видалення інфікованих файлів та фіксації історії дій користувача з програмою.

5.2 Умови використання програми

Застосунок «Antivirus» працює під управлінням операційної системи Windows 10. Саме версія ОС обумовлює склад та системні вимоги до технічних засобів (див. п. 1.3.4).

5.3 Виконання програми

Для запуску програмного забезпечення необхідно виконати файл Antivirus.exe. Програма не потребує та не підтримує виклик з додатковими аргументами у командному рядку.

Після запуску програми оператору відкривається її головна форма (рис. 3.4). Знаходячись на головній формі оператор може виконати наступні дії:

- перейти до режиму «Сканування папки»;
- перейти до режиму «Пароль на папку»;
- перейти до режиму «Видалити вірус»;
- перейти до режиму «Історія сканування»;
- відсканувати файл на віруси;
- обрати файл для сканування.

Перейшовши до режиму сканування папки (рис. 3.5). (шляхом натискання на кнопку «Сканування папки»), оператор може відсканувати папку на наявність у ній ШПЗ. Перед тим як відсканувати папку, оператору доведеться її вибрати. Зробити це можна натиснувши кнопку «Обрати папку»

(елемент «1» на рис.3.5). Після того, як папка була обрана, натиснувши на кнопку «Відсканувати» (елемент «2» на рис.3.5). у полі праворуч (елемент «3» на рис.3.5), у випадку знаходження ШПЗ, будуть виведено на екран відповідне повідомлення. Також оператор може побачити кількість файлів у папці біля надпису «Кол-во файлів» (елемент «4» на рис.3.5). Якщо користувач спробує натиснути кнопку сканування не обравши папку, на екрані з'явиться повідомлення про помилку.

Перейшовши до режиму сканування папки, оператор може:

- перейти до режиму «Сканування файлу»;
- перейти до режиму «Пароль на папку»;
- перейти до режиму «Видалити вірус»;
- перейти до режиму «Історія сканування»;
- обрати папку;
- відсканувати папку на віруси.

Перейшовши до режиму пароль на папку (рис.3.6), оператор може:

- перейти до режиму «Сканування файлу»;
- перейти до режиму «Сканування папки»;
- перейти до режиму «Видалити вірус»;
- перейти до режиму «Історія сканування»;
- обрати папку для блокування;
- ввести пароль для доступу до функціоналу блокування;
- заблокувати доступ до папки;
- розблокувати доступ.

Перейшовши до режиму «Пароль на папку» (шляхом натискання на кнопку «Пароль на папку»), оператор може обрати папку для блокування доступу (елемент «1» на рис. 3.6), та заблокувати доступ до папки (елемент «3» на рис. 3.6), або розблокувати її (елемент «4» на рис. 3.6). Спочатку оператору потрібно обрати папку для блокування натиснувши кнопку «Обрати папку». Після того як папку було обрано, оператор повинен ввести пароль у рядку

«пароль» (елемент «2» на рис. 3.6). Якщо пароль вірний, оператор отримує доступ до функціоналу програми, і може як заблокувати доступ до папки, так і розблокувати її. Якщо оператор спробує натиснути кнопку заблокувати без наявності папки для блокування на формі, на екрані з'явиться повідомлення про помилку.

Перейшовши до режиму видалення файлу (рис. 3.7), оператор може:

- перейти до режиму «Сканування файлу»;
- перейти до режиму «Сканування папки»;
- перейти до режиму «Пароль на папку»;
- перейти до режиму «Історія сканування»;
- обрати файл для видалення;
- видалити файл.

Перейшовши до режиму «Історія сканування» (рис. 3.8), оператор може:

- перейти до режиму «Сканування файлу»;
- перейти до режиму «Сканування папки»;
- перейти до режиму «Пароль на папку»;
- перейти до режиму видалити вірус;
- записати історію сесії;
- продивитись архів історії сканування.

В цей режим оператор може перейти шляхом натискання на кнопку «Історія сканування». Оператор може продивитися історію своїх дій у програмі протягом даної сесії (елемент «1» рис.3.8), записати поточну сесію за своїм бажанням (елемент «2» рис.3.8), та продивитися архів історії сканування за весь час (рис. 3.9) шляхом натискання кнопки «Продивитись» (елемент «3» рис.3.8). У випадку якщо оператор захоче записати сесію у якій він не виконував ніяких дій, то програма просто запише дату та час сесії.

5.4 Повідомлення оператору

Всі повідомлення, що може отримати оператор під час роботи з програмним забезпеченням описані у табл. 5.1.

Таблиця 5.1 – Повідомлення оператору

Повідомлення	Причина появи	Подальші дії
Шлях має неприпустиму форму	Відсутність папки для сканування	Обрати папку для сканування
Не обрано вірусу для перевірки при скануванні	Спроба зробити сканування для вірусу, що не був обраний з БД	Оператор має обрати вірус, відкривши вікно роботи з антивірусною БД. Прапорець успішного вибору вірусу - поява його імені у відповідному текстовому полі на головній формі.
Неприпустиме ім'я Ім'я параметра: name	Оператор не обрав папку для блокування доступу	Обрати папку для блокування доступу

ВИСНОВКИ

У дипломній кваліфікаційній роботі бакалавра було вирішено актуальну проблему антивірусного захисту інформації користувача на ПК.

Для досягнення поставленої мети, в процесі виконання роботи було проведено огляд принципів роботи антивірусних програм. У якості підходу, для виявлення ШПЗ було обрано словниковий (сигнатурний) підхід, який використовується в більшості сучасного АПЗ.

Проведений аналіз вже наявних програм-аналогів дозволив виявити властиві їм переваги і недоліки, а також послужив основою для розробки ТЗ на програму антивірусного сканера. Було прийнято рішення розробляти програму у вигляді десктопного застосунку, що виконується під управлінням ОС Windows 10. Мовою розробки застосунку було обрано C#.

У відповідності з функціональними вимогами, зазначеними у ТЗ, розроблено модель системи у вигляді діаграми компонентів у нотації мови моделювання UML, а також блок-схему алгоритму роботи програми.

Виявлення вірусу за допомогою програми відбувається шляхом порівняння хешу відсканованого файлу з тими хеш даними, які містяться у файлі .txt формату який відіграє роль вірусної БД. Для хешування даних використано відомий алгоритм M5D Hash.

Практичне використання розробленого програмного забезпечення дозволяє користувачу виявляти вірусні сигнатури у файлах та папках, здійснювати захист доступу до папок шляхом встановлення паролів, видаляти інфіковані файли та фіксувати історію дій користувача з програмою.

Галузь використання розробленого застосунку – антивірусний захист інформації в ОС Windows 10.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Компьютерные вирусы и вредоносное ПО: факты и часто задаваемые вопросы [Электрон. ресурс]. – Режим доступа: <https://www.kaspersky.ru/resource-center/threats/computer-viruses-and-malware-facts-and-faqs>.
2. Який антивірус краще для ПК. Які антивіруси хороші? Рейтинг антивірусів, відгуки [Електрон. ресурс]. – Режим доступу: <https://maylohack.ru/uk/programmy/kakoi-antivirus-luchshe-dlya-pk-kakie-antivirusy-horoshie.html>.
3. Grooten, M. It's 2016. Can we stop using MD5 in malware analyses? [Electronic resource] / M. Grooten. – Access mode: <https://www.virusbulletin.com/blog/2016/07/its-2016-can-we-stop-using-md5-malware-analyses/>.
4. Blanco, A. G. Antivirus software guide: what are they for and how to use them [Electronic resource] / A. G. Blanco. – Access mode: <https://www.bbva.com/en/antivirus-software-guide-what-are-they-for-and-how-to-use-them/>.
5. Balasubramanian, S. Working principle of an Antivirus- How an antivirus works? [Electronic resource] / S. Balasubramanian. – Access mode: <https://www.techulator.com/resources/3984-Working-principle-of-an-Antivirus-How-an-antivirus-works-.aspx>.
6. How Antivirus Works [Electronic resource]. – Access mode: <https://antivirus.comodo.com/faq/how-antivirus-works.html>.
7. Обзор антивируса McAfee [Электрон. ресурс]. – Режим доступа: <https://www.10bestantivirus.ru/reviews/mcafee>.
8. Summary Report 2020 [Electronic resource]. – Access mode: <https://www.av-comparatives.org/tests/summary-report-2020/>.
9. Best Antivirus 2021 [Electronic resource]. – Access mode: <https://www.antivirusguide.com/best-antivirus/>.

10. Реальные системные требования для Windows 10 [Электрон. ресурс].
– Режим доступа: <https://g-ek.com/windows-10-1903-s-minimalnye-trebovania>.
11. MD5 [Электрон. ресурс]. – Режим доступа:
<http://kriptografea.narod.ru/MD5.html>.
12. Access control list [Electronic resource]. – Access mode:
https://en.wikipedia.org/wiki/Access-control_list.

ДОДАТОК А
Текст програми

A.1 Код файла Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace Antivirus
{
    static class Program
    {
        /// <summary>
        /// Головна точка входу для застосунку
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

A.2 Код файла FORM1.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;
using System.Security.Cryptography;
using System.Security.AccessControl;
using System.IO;
using System.Diagnostics;
using System.Threading;
namespace Antivirus
{
    public partial class Form1 : Form
    {
        int virus;
        public Form1()
        {
            InitializeComponent();
            panel_scan_btn.Visible = true;
            panel_folder_scan.Visible = false;
            panel_password_btn.Visible = false;
            panel_delete_btn.Visible = false;
            panel_scan_log_btn.Visible = false;
            panel_scan_btn.Height = 445;
            panel_scan_btn.Width = 644;
            panel_folder_scan.Height = 445;
            panel_folder_scan.Width = 644;
            panel_password_btn.Height = 445;
            panel_password_btn.Width = 644;
            panel_delete_btn.Height = 445;
            panel_delete_btn.Width = 644;
            panel_scan_log_btn.Height = 445;
            panel_scan_log_btn.Width = 644;
        }
        public string GetMD5FromFile(string filePath)
        {
            using (var md5 = MD5.Create())

```

```

        {
            using (var stream = File.OpenRead(filenPath))
            {
                return
BitConverter.ToString(md5.ComputeHash(stream)).Replace("-",
string.Empty).ToLower();
            }
        }
    }

private void folder_scan_btn_Click(object sender,
EventArgs e)
    {
        movesliderpanel(folder_scan_btn);
        panel_scan_btn.Visible = false;
        panel_password_btn.Visible = false;
        panel_delete_btn.Visible = false;
        panel_scan_log_btn.Visible = false;
        panel_folder_scan.Visible = true;
        panel_folder_scan.Location = new Point(161, 118);
    }

private void movesliderpanel(Control btn)
    {
        sliderpanel.Top = btn.Top;
        sliderpanel.Height = btn.Height;
    }

private void scan_btn_Click(object sender, EventArgs
e)
    {
        movesliderpanel(scan_btn);
        panel_folder_scan.Visible = false;
        panel_password_btn.Visible = false;
        panel_delete_btn.Visible = false;
        panel_scan_log_btn.Visible = false;
        panel_scan_btn.Visible = true;
    }

```

```

private void password_folder_btn_Click(object sender,
EventArgs e)
{
    movesliderpanel(password_folder_btn);
    panel_scan_btn.Visible = false;
    panel_folder_scan.Visible = false;
    panel_delete_btn.Visible = false;
    panel_scan_log_btn.Visible = false;
    panel_password_btn.Visible = true;
    panel_password_btn.Location = new Point(161,
118);
}
private void delet_btn_Click(object sender, EventArgs
e)
{
    movesliderpanel(delet_btn);
    panel_scan_btn.Visible = false;
    panel_folder_scan.Visible = false;
    panel_password_btn.Visible = false;
    panel_scan_log_btn.Visible = false;
    panel_delete_btn.Visible = true;
    panel_delete_btn.Location = new Point(161, 118);
}
private void scan_log_btn_Click(object sender,
EventArgs e)
{
    movesliderpanel(scan_log_btn);
    panel_scan_btn.Visible = false;
    panel_folder_scan.Visible = false;
    panel_password_btn.Visible = false;
    panel_delete_btn.Visible = false;
    panel_scan_log_btn.Visible = true;
    panel_scan_log_btn.Location = new Point(161,
118);
}

```

```

private void Form1_Load(object sender, EventArgs e)
{
    panel_scan_btn.Location = new Point(161, 118);
}
private void btn_scan_Click(object sender, EventArgs
e)
{
    var md5signature =
File.ReadAllLines("MD5Base.txt");
    if (md5signature.Contains(tbmd5.Text))
    {
        lb_status.Text = "Файл інфікований ";
        lb_status.ForeColor = Color.Red;
        tb_scanlog.Text += " " + lb_status.Text +
"Дата та час: " + DateTime.Now.ToString("dd.MM.yyyy HH:mm:ss") +
Environment.NewLine;
    }
    else
    {
        lb_status.Text = "Файл чистий";
        lb_status.ForeColor = Color.LimeGreen;
        tb_scanlog.Text += " " + lb_status.Text +
"Дата та час: " + DateTime.Now.ToString("dd.MM.yyyy HH:mm:ss") +
Environment.NewLine;
    }
}
private void btn_choose_file_Click(object sender,
EventArgs e)
{
    OpenFileDialog ofd = new OpenFileDialog();
    if (ofd.ShowDialog() == DialogResult.OK)
    {
        tbfilepath.Text = ofd.FileName;
        tbmd5.Text = GetMD5FromFile(ofd.FileName);
    }
}

```

```

        tb_scanlog.Text += "||Режим сканування файлу:
Файл: \n" + tbfilepath.Text + " Хеш файлу:  " + tbmd5.Text;
    }
}

private void btn_choose_Click(object sender,
EventArgs e)
{
    folderBrowserDialog.ShowDialog();
    tb_folder.Text =
folderBrowserDialog.SelectedPath; virus = 0;
    lb_count.Text += virus.ToString();
    tb_scanlog.Text += "\n ||Режим сканування папки:
Папка: " + tb_folder.Text;
    listBox.Items.Clear();
}

private void btn_scan_folder_Click(object sender,
EventArgs e)
{
    List<string> search =
Directory.GetFiles(@folderBrowserDialog.SelectedPath,
"*.*").ToList();
    var md5signature1 =
File.ReadAllLines("MD5Base.txt");
    foreach (string item in search)
    {
        var s = GetMD5FromFile(item);
        if (md5signature1.Contains(s))
        {
            lb_count.Text = null;
            virus += 1;
            lb_count.Text += virus.ToString();
            listBox.Items.Add(item);
            tb_scanlog.Text += "\n Кількість вірусів:
" + lb_count.Text + "Вірусні файли: " + item + "Дата та час: " +

```

```

DateTime.Now.ToString("dd.MM.yyyy          HH:mm:ss")          +
Environment.NewLine;
        }
    }
}

private void choose_todelete_btn_Click(object sender,
EventArgs e)
{
    OpenFileDialog path = new OpenFileDialog();
    if (path.ShowDialog() == DialogResult.OK)
    {
        tb_delete.Text = path.FileName;
        delete_timer.Start();
        tb_scanlog.Text += "|| Режим видалення файлу:
" + "Було видалено файл- " + path.FileName + "Дата та час: " +
DateTime.Now.ToString("dd.MM.yyyy          HH:mm:ss")          +
Environment.NewLine;

        File.Delete(path.FileName);
    }
}

private void delete_timer_Tick(object sender,
EventArgs e)
{
    delete_timer.Interval = 10;
    if (deleteprogres.Value >= deleteprogres.Maximum)
    {
        deleteLbl.Text = "Успішно видалено";
        deleteLbl.ForeColor = Color.Green;
        delete_timer.Stop();
    }
    else
    {
        deleteprogres.Value += 10;
    }
}
}

```

```

        private void btn_choose_fold_to_pass_Click(object
sender, EventArgs e)
        {
            if (folderBrowserDialog_pass.ShowDialog() ==
DialogResult.OK)
            {
                tb_folder_sear.Text =
folderBrowserDialog_pass.SelectedPath;
                tb_scanlog.Text += "\n ||Режим захисту папки:
Папка: " + tb_folder_sear.Text + Environment.NewLine;
            }
        }
        private void btn_lock_Click(object sender, EventArgs
e)
        {
            var pass1 = File.ReadAllLines("pass.txt");
            if (pass1.Contains(tb_pass.Text))
            {
                try
                {
                    string folderPath = tb_folder_sear.Text;
                    string adminUserName =
Environment.UserName;
                    DirectorySecurity ds =
Directory.GetAccessControl(folderPath);
                    FileSystemAccessRule fsa = new
FileSystemAccessRule(adminUserName, FileSystemRights.FullControl,
AccessControlType.Deny);
                    ds.AddAccessRule(fsa);
                    Directory.SetAccessControl(folderPath,
ds);
                    MessageBox.Show("Locked");
                    tb_scanlog.Text += "\n ||Режим доступу: "
+ folderBrowserDialog_pass.SelectedPath + "Закрито " + "Дата та

```

```

час: " + DateTime.Now.ToString("dd.MM.yyyy HH:mm:ss") +
Environment.NewLine;

        pb_pass.Image =
Properties.Resources.unlock_fold1;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
else
{
    MessageBox.Show("Не вірний пароль");
}
}
private void btn_unlock_Click(object sender,
EventArgs e)
{
    var pass2 = File.ReadAllLines("pass.txt");
    if (pass2.Contains(tb_pass.Text))
    {
        try
        {
            string folderPath = tb_folder_sear.Text;
            string adminUserName =
Environment.UserName; // getting your adminUserName
            DirectorySecurity ds =
Directory.GetAccessControl(folderPath);
            FileSystemAccessRule fsa = new
FileSystemAccessRule(adminUserName, FileSystemRights.FullControl,
AccessControlType.Deny);
            ds.RemoveAccessRule(fsa);
            Directory.SetAccessControl(folderPath, ds);
            MessageBox.Show("UnLocked");

```

```

        tb_scanlog.Text += "\n ||Режим доступу: "
+ folderBrowserDialog_pass.SelectedPath + "Відкрито " + "Дата та
час: " + DateTime.Now.ToString("dd.MM.yyyy HH:mm:ss") +
Environment.NewLine;

        pb_pass.Image =
Properties.Resources.lock_fold1;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
else
{
    MessageBox.Show("Не вірний пароль");
}
}

private void btn_write_Click(object sender, EventArgs e)
{
    string lines = "Дата та час сесій: " +
DateTime.Now.ToString("dd.MM.yyyy HH:mm:ss") + Environment.NewLine
+ tb_scanlog.Text;
    System.IO.File.AppendAllText("scan_log.txt",
lines);
}

private void btn_show_Click(object sender, EventArgs
e)
{
    string fpath = "scan_log";
    Process proc = Process.Start("notepad.exe",
fpath);

    proc.WaitForExit(); proc.Close();
}
}
}

```

ДОДАТОК Б
Копії слайдів презентації

ДИПЛОМНА РОБОТА

РОЗРОБКА ПРОГРАМИ АНТИВІРУСНОГО СКАНЕРА ДЛЯ ОПЕРАЦІЙНОЇ СИСТЕМИ WINDOWS

Розробив: ст. групи КНТ-227
Керівник: доцент, к.т.н.

Ал Омарі А.М.
Сердюк С.М.

Рисунок Б.1 – Слайд №1

Мета роботи

розробка програми, що дозволяє виявляти вірусні сигнатури у файлах та папках, захищати доступ до папок, видаляти інфіковані файли та фіксувати історію дій користувача з програмою

Об'єкт дослідження – процеси антивірусного захисту в операційній системі (ОС) Windows

Предмет дослідження – методи антивірусного захисту в ОС Windows



НУ “ЗП”, Кафедра програмних засобів

Рисунок Б.2 – Слайд №2

Задачі, що вирішуються

- огляд принципів роботи антивірусних програм;
- аналіз програм-аналогів;
- розробка технічного завдання;
- визначення компонентів системи;
- розробка алгоритмів роботи програми;
- програмна реалізація застосунку;
- розробка необхідної документації.



НУ “ЗП”, Кафедра програмних засобів

Рисунок Б.3 – Слайд №3

3

Алгоритм M5D Hash

Крок 1. Вирівнювання потоку.

Крок 2. Додавання довжини повідомлення.

Крок 3. Ініціалізація MD-буферу.

```
word A: 01 23 45 67;
word B: 89 ab cd ef;
word C: fe dc ba 98;
word D: 76 54 32 10;
```

Крок 4. Обчислення в циклі (4 раунди).

$$F(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$

$$G(X, Y, Z) = (X \wedge Y) \vee (Y \wedge \neg Z)$$

$$H(X, Y, Z) = X \oplus Y \oplus Z$$

$$I(X, Y, Z) = Y \oplus (X \vee \neg Z)$$



НУ “ЗП”, Кафедра програмних засобів

Рисунок Б.4 – Слайд №4

4

Алгоритм M5D Hash

Крок 4.1. Запам'ятовуємо перші 512 біт послідовності S .

Крок 4.2. Видаляємо перші 512 біт послідовності S (можна обійтися і без видалення, але тоді на першому кроці треба брати не перші 512, а наступні 512 біт).

Крок 4.3. Викликаємо функцію W . Параметри A, B, C, D – це поточні значення відповідних подвійних слів. Параметр T – 512 біт, які було запам'ятовано.

Крок 4.4. Додаємо до A $A0$.

Крок 4.5. $B = B + B0$.

Крок 4.6. $C = C + C0$.

Крок 4.7. $D = D + D0$.

Крок 4.8. Якщо довжина послідовності 0, виходимо.

Крок 4.9. Переходимо до Крок 4.1.

Крок 5. Результат обчислень знаходиться в буфері ABCD, це і є хеш

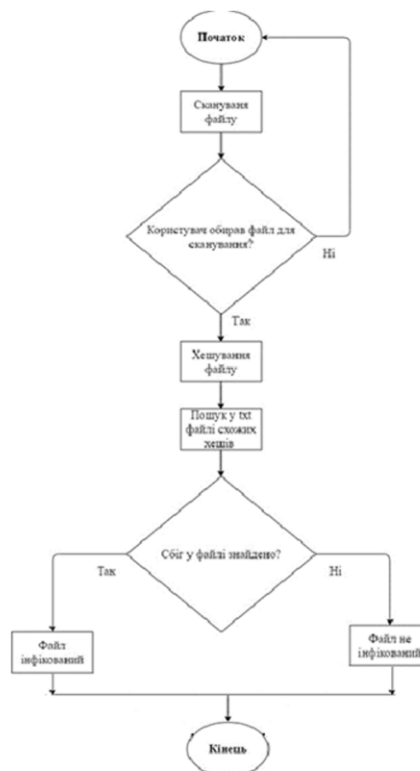


НУ "ЗП", Кафедра програмних засобів

5

Рисунок Б.5 – Слайд №5

Блок-схема алгоритму роботи програми

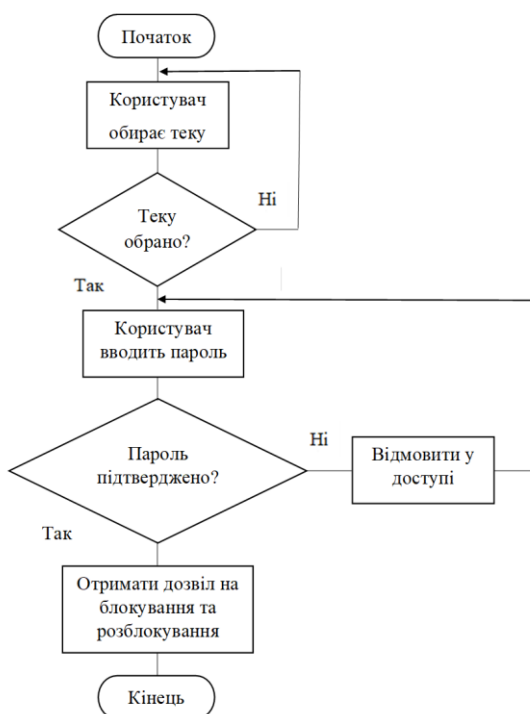


НУ "ЗП", Кафедра програмних засобів

6

Рисунок Б.6 – Слайд №6

Блок-схема алгоритму блокування теки

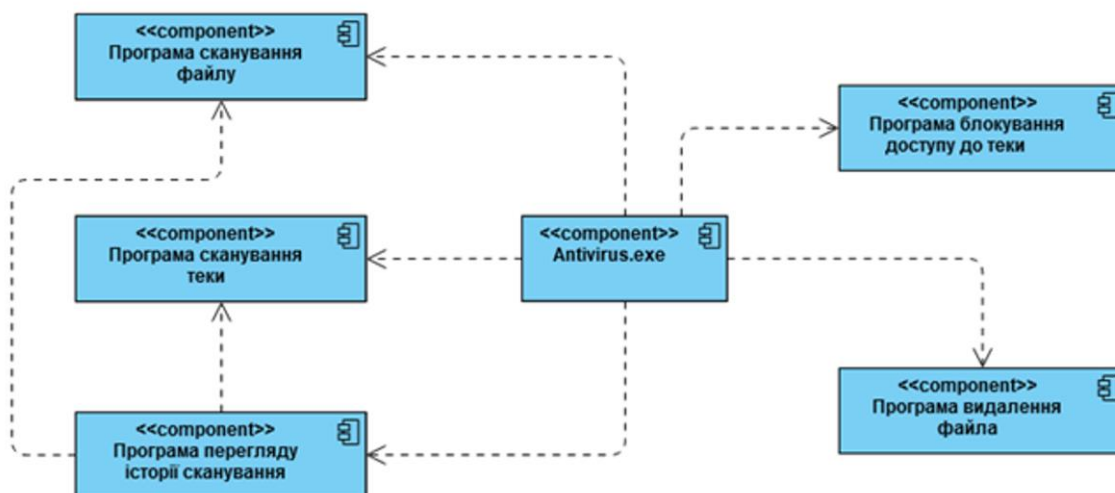


НУ "ЗП", Кафедра програмних засобів

7

Рисунок Б.7 – Слайд №7

Діаграма компонентів застосунку



НУ "ЗП", Кафедра програмних засобів

8

Рисунок Б.8 – Слайд №8

Опис класу Form1

Form1
+ Virus:int
+ GetMD5FromFile(string filePath):string + folder_scan_btn_Click(object sender, EventArgs e):void + movesliderpanel(Control btn):void + scan_btn_Click(object sender, EventArgs e):void + password_folder_btn_Click(object sender, EventArgs e):void + delet_btn_Click(object sender, EventArgs e):void + scan_log_btn_Click(object sender, EventArgs e):void + Form1_Load(object sender, EventArgs e):void + btn_scan_Click(object sender, EventArgs e):void + btn_choose_file_Click(object sender, EventArgs e):void + btn_choose_Click(object sender, EventArgs e):void + btn_scan_folder_Click(object sender, EventArgs e):void + choose_todelete_btn_Click(object sender, EventArgs e):void + delete_timer_Tick(object sender, EventArgs e):void + btn_choose_fold_to_pass_Click(object sender, EventArgs e):void + btn_lock_Click(object sender, EventArgs e):void + btn_unlock_Click(object sender, EventArgs e):void + btn_write_Click(object sender, EventArgs e):void + btn_show_Click(object sender, EventArgs e):void



НУ "ЗП", Кафедра програмних засобів

9

Рисунок Б.9 – Слайд №9

Атрибути та методи класу Form1

Атрибут/Метод	Тип даних	Короткий опис
GetMD5FromFile	string	Знаходить хеш обраного файлу
folder_scan_btn_Click	void	Викликає режим сканування папки
movesliderpanel	void	Логіка переміщення зеленої панелі при натисканні на кнопки
scan_btn_Click	void	Викликає режим сканування файлу
password_folder_btn_Click	void	Викликає режим блокування папки
delet_btn_Click	void	Викликає режим видалення файлу
scan_log_btn_Click	void	Викликає режим історії сканування
Form1_Load	void	Визначає початкову позицію режиму сканування файлу при запуску програми
btn_scan_Click	void	Оброблювач подій натискання на кнопку сканування файлу
btn_choose_file_Click	void	Оброблювач подій натискання на кнопку вибору файлу для сканування
btn_choose_Click	void	Оброблювач подій натискання на кнопку вибору папки для сканування
btn_scan_folder_Click	void	Оброблювач подій натискання на кнопку сканування папки
choose_todelete_btn_Click	void	Оброблювач подій натискання на кнопку видалення файлу
delete_timer_Tick	void	Заповнення шкали прогресу видалення за певним інтервалом
btn_choose_fold_to_pass_Click	void	Оброблювач подій натискання на кнопку вибору папки для блокування доступу
btn_lock_Click	void	Оброблювач подій натискання на кнопку блокування папки
btn_unlock_Click	void	Оброблювач подій натискання на кнопку розблокування папки
btn_write_Click	void	Оброблювач подій кнопки запису історії сканування в архів

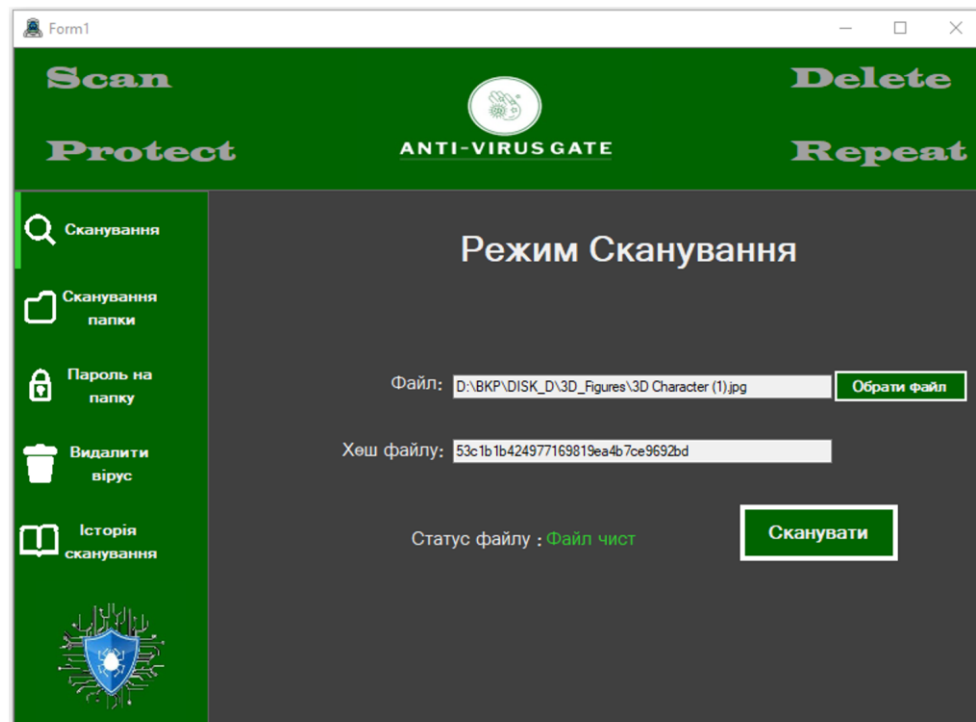


НУ "ЗП", Кафедра програмних засобів

10

Рисунок Б.10 – Слайд №10

Головне меню програми

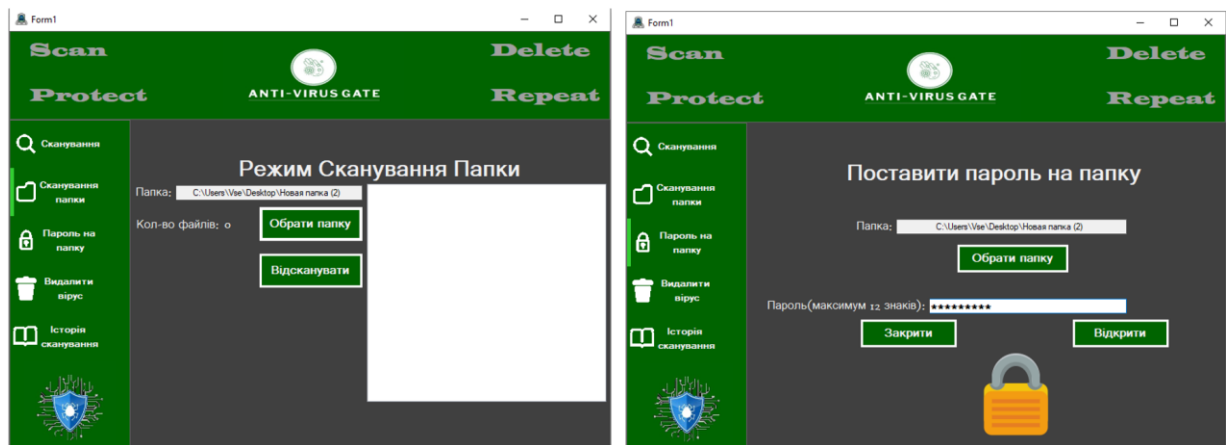


НУ "ЗП", Кафедра програмних засобів

Рисунок Б.11 – Слайд №11

11

Режими програми



Режим сканування папки

Режим встановлення паролю

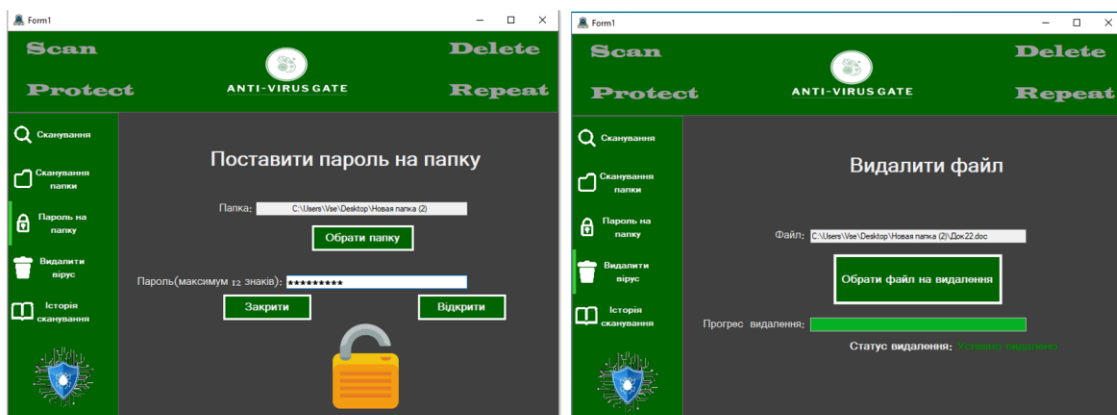


НУ "ЗП", Кафедра програмних засобів

Рисунок Б.12 – Слайд №12

12

Режими програми



Режим розблокування папки

Режим видалення файлу

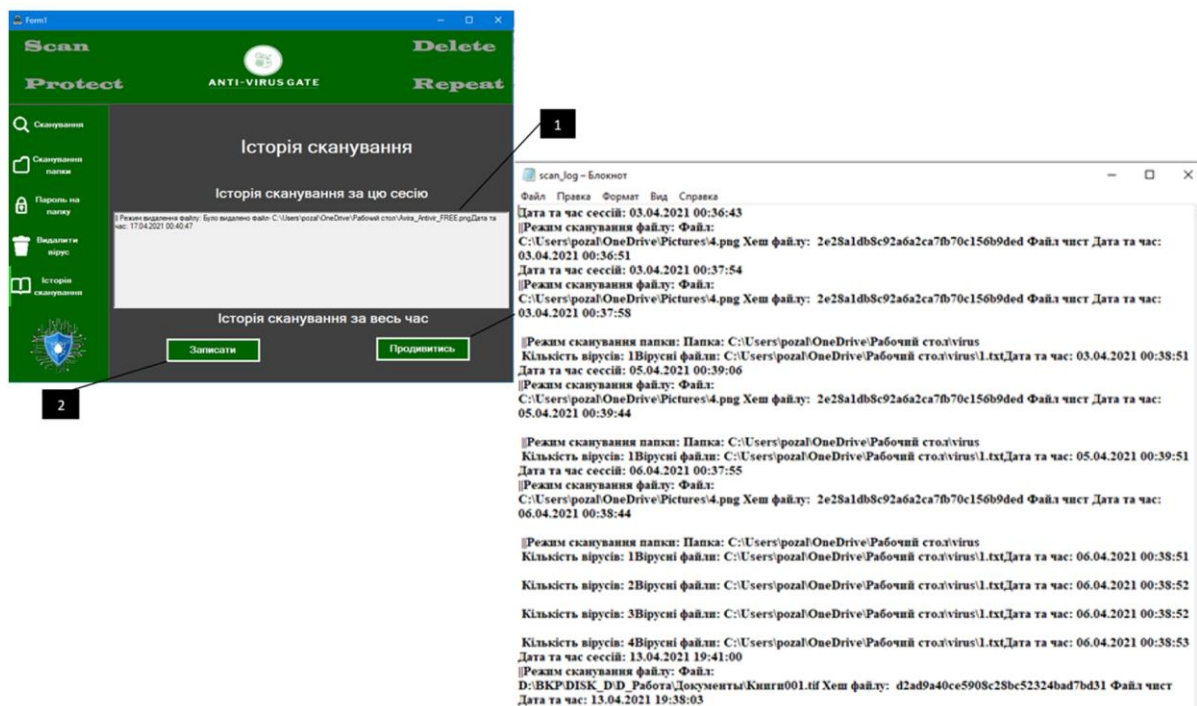


НУ "ЗП", Кафедра програмних засобів

13

Рисунок Б.13 – Слайд №13

Історія сканування



НУ "ЗП", Кафедра програмних засобів

14

Рисунок Б.14 – Слайд №14

Висновки

- проведено огляд принципів роботи антивірусних програм;
- проведено аналіз програм-аналогів;
- розроблено технічне завдання;
- визначено компоненти системи;
- розроблено алгоритми роботи програми;
- програмно реалізовано застосунок;
- розроблено необхідну документацію.



НУ “ЗП”, Кафедра програмних засобів

Рисунок Б.15 – Слайд №15

15

**Доклад закінчено,
дякую за увагу!**



НУ “ЗП”, Кафедра програмних засобів

Рисунок Б.16 – Слайд №16

16