МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп’ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Радецький Богдан Павлович

Програмний модуль для роботи з зашифрованим віртуальним диском / The software module for working with encrypted virtual disk

напрям підготовки: 6.050102 - Комп’ютерна інженерія
фахове спрямування - Комп’ютерні системи та мережі
Бакалаврська робота

Виконав студент групи КСМ-
Богдан Павлович Радецький

Науковий керівник:
к.т.н., С.В. Івасьєв

Тернопіль - 2018

РЕЗЮМЕ

Дипломний проект містить 61 сторінку пояснюючої записки, 8 рисунків, 7 таблиць, 2 додатків, 3 діаграми та 2 – графічні схеми формату А3.

Метою дипломної роботи є дослідження існуючих засобів віртуалізації та розробка додатку для створення зашифрованого віртуального диску.

Методи досліджень базуються на теорії алгоритмів (для аналізу розроблених методів та алгоритмів), методах захисту інформації, технологіях структурного та об’єктно-орієнтованого програмування, сучасних методах дослідження віртуалізації.

Проведено аналіз існуючих програмних та апаратних засобів віртуалізації даних та виявлено їхні слабкі сторони. Досліджено та проаналізовано основні системні WinAPI функці для роботи з жорсткими дисками. Проведено аналіз можливостей середовища розробки програмного продукту Delphi 6. Досліджено основні алгоритми шифрування даних.

Розроблено програмний додаток, що захист інформації шляхом створення зашифрованого віртуального диска в ОС Windows. Проведено тестування та налагодження додатку для захисту інформації.

Ключові слова: ВІРТУАЛІЗАЦІЯ, ШИФРУВАННЯ, ЖОРСТКИЙ ДИСК, ЗАХИСТ ІНФОРМАЦІЇ, AES.

RESUME

The degree project comprises 61 pages of explanatory notes, 8 figures, 7 tables,3 diagrams and 2 apendixes. Volume of graphic materail 2 pages of format A3.

The purpose of the thesis is to study the existing virtualization tools and develop an application to create an encrypted virtual disk.

Research methods are based on the theory of algorithms (for analysis of developed methods and algorithms), information security methods, technologies of structural and object-oriented programming, modern methods of virtualization research.

An analysis of existing software and hardware virtualization data was performed and their weaknesses identified. The basic system functions of WinAPI for working with hard disks are investigated and analyzed. The analysis of the possibilities of the Delphi 6 software development environment has been carried out. The main algorithms of data encryption have been investigated.

A software application is developed that protects information by creating an encrypted virtual disk on Windows OS. The testing and debugging of the application for information security was conducted.

Key words: VIRTUALIZATION, ENCRYPTING, HARD DISK, INFORMATION PROTECTION, AES.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ОД	–	Образ диска
ОС	–	Операційна система
VHD	–	Віртуальний жорсткий диск
ПЗ	–	Програмний засіб
ООП	–	Об`єктно-орієнтоване програмування
UML	–	Уніфікована мова моделювання

ВСТУП

Збільшення числа робочих місць, серверів, програмних та інформаційних ресурсів тягне за собою ріст витрат на управління корпоративною інформаційною мережею і висуває проблему її спрощення. Один із способів такого спрощення – автономізація складових компонентів і додатків, іншими словами, їх незалежність від інформаційного середовища[1]. Великі можливості в цьому напрямку надає ідея віртуалізації, тобто створення штучного об’єкта або середовища.

Контролювати невеликі потоки інформації неважко. Однак, коли йдеться про великі компанії, то уникнути втрати даних при передачі не вдасться. Але віртуалізація дозволяє вирішити цю проблему, адже створення єдиної віртуальної файлової системи на основі декількох фізичних дає змогу кожному працівнику максимально швидко отримати доступ до потрібної інформації[2]. Додатковий захист і контроль доступу для працівників дають змогу уникнути небажаних витіків інформації. На даний момент найпоширенішими реалізаціями даної ідеї стали віртуальні: обладнання, операційні системи, програмне забезпечення, пам`ять, бази даних та мережі.

Метою дипломного проектування є розробка програмного продукту для роботи із віртуальними дисками, що забезпечить можливості створення захищеного сховища інформації та надасть змогу оперувати з файлами, що містяться на ньому.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Віртуальні диски та сфера їх застосування

Віртуалізація – надання набору обчислювальних ресурсів або їх логічного об’єднання, відокремлене від апаратної реалізації, яке забезпечує логічну ізоляцію один від одного обчислювальних процесів, що виконуються на одному фізичному ресурсі. Яскравим прикладом віртуалізації є можливість запуску декількох операційних систем на одному комп’ютері, при чому кожна працює з власним набором обчислювальних ресурсів.

Найбільш перспективною стала віртуалізація дисків, тобто створення образів дисків[3]. Віртуальний диск(ВД) може бути частиною одного реального або складатись із декількох повних реальних дисків або їх частин, які розташовані як в одному, так і в декількох вузлах інформаційного середовища.

Образ диска – комп'ютерний файл, який містить в собі повну копію вмісту та структури файлової системи й даних, що знаходяться на диску - такому як компакт-диск, дискета або розділ жорсткого диска. Образ зазвичай створюється шляхом створення копії сектору вихідного джерела, тим самим ідеально копіюючи структуру і вміст засобу зберігання, що не залежить від файлової системи. В залежності від формату диску, образ може охоплювати один або декілька файлів.

Формат створеного файлу може бути відкритим стандартом, таким як наприклад формат ISO для оптичних дисків, VHD для жорстких дисків, або унікальним для конкретного програмного додатку. Розмір може бути доволі великим оскільки він містить вміст всього диску[4]. Для зменшення вимог до сховища, якщо утиліта обробки образів є файловою системою, вона може не копіювати невикористаний дисковий простір і стиснути безпосередньо зайнятий.

Образи дисків використовуються для: резервного копіювання даних, поширення великих програмних пакетів, як пристрої зберігання для емуляторів

та віртуальних машин та масового встановлення програмного забезпечення на комп’ютери з однаковою конфігурацією (тиражування однотипних систем).

Деякі операційні системи, як наприклад Linux та macOS містять вбудовані функції створення та роботи з ВД, у той час як іншим(старим версіям Microsoft Windows), необхідне додаткове програмне забезпечення. Лише починаючи з Windows 8, Windows має вбудовані функції для роботи з ВД.

Віртуальні диски зазвичай доступні лише для читання і не можуть бути змінені. Для роботи з ними необхідні спеціальні програми які дають можливість виконувати операції стиснення, шифрування, перетворення форматів та ін..

Віртуальні жорсткі диски часто використовуються в програмах шифрування мультимедійного диска ("OTFE"), наприклад FreeOTFE та TrueCrypt , де зберігається зашифрований "образ" диска на комп'ютері[5]. Коли вводиться пароль, образ диска "встановлюється" і стає доступним як новий том на комп'ютері. Файли, написані на цьому віртуальному диску, записуються в зашифрований образ, і ніколи не зберігаються в прозорому вигляді .

Процес створення доступного для використання комп'ютером диска називається "монтаж", процес його видалення називається "демонтаж"; ті ж терміни використовуються для того, щоб зробити зашифрований диск доступним або недоступним.

Образ жорсткого диску інтерпретується монітором віртуальної машини як системний жорсткий диск. IT-адміністратори та розробники програмного забезпечення адмініструють їх через автономні операції за допомогою вбудованих або сторонніх інструментів. З точки зору іменування, образ жорсткого диска для певного монітора віртуальних машин має певне розширення типу файлу, наприклад, .vmdk для VMware VMDK, .vhd для Xen і Microsoft Hyper-V, .vdi для Oracle VM VirtualBox, і т.д.

Віртуальні жорсткі диски зазвичай використовуються для клонування та резервного відновлення даних .

Великим підприємствам часто доводиться купувати або замінювати нові комп'ютерні системи у великій кількості. Встановлення операційної системи та програм в кожному з них один за іншим вимагає багато часу та зусиль і має значну можливість людської помилки. Тому системні адміністратори використовують образ диска для швидкого клонування повністю підготовленого програмного середовища системи відліку[6]. Цей метод економить час та зусилля, і дозволяє адміністраторам зосередитися на унікальних відмінностях, які повинна мати кожна система. Клонування, як правило, може виконуватись лише програмними засобами, оскільки зазвичай вимагає тільки повторення файлової структури та самих файлів.

Резервне відновлення даних – це процес відображення кожного окремого сектора з початкового диску на інший носій, з якого необхідні файли можна буде відновити при потребі[7]. У ситуаціях відновлення даних не можна покладатися на цілісність структури файлів, тому повна копія сектора є обов'язковою.

Образ для відновлення даних повинен мати можливість попередньо налаштувати диски, відключивши певні атрибути (як наприклад SMART та G-List) та можливість працювати з нестабільними дисками(нестабільність читання може бути спричинена зокрема механічним зносом або зовнішніми пошкодженнями). Для відновлення даних віртуальний диск повинен мати змогу зчитувати дані із так званих «поганих секторів». Читання нестабільності є основним фактором при роботі з дисками в операційних системах, таких як Windows. Типова операційна система обмежена у здатності справлятись із дисками, які потребують багато часу на читання. З цих причин програмне забезпечення покладається на BIOS[8]. Для досягнення повного спектру відтворення з відновлених даних потрібен окремий контроль апаратного забезпечення вихідного жорсткого диску. Це відбувається тому, що операційна система має певний набір протоколів або правил для зв`язку із приводом, які не можуть бути порушені.

Для комп’ютерів під керуванням macOS образ диска є найпоширенішим типом файлів, який використовується для завантаження програм з веб-

браузерів. ВД зазвичай стискаються за допомогою AppleDiskImage. Їх перевага полягає у тому, що вони не потребують резервного простору дисків для не архівованих даних. Пакети програм для Windows також іноді поширюються як образи дисків, включаючи формат ISO.

Набуває популярності, ще один тип віртуальних дисків – віддалені. Вони розміщуються на певному сервісі(як наприклад Google або Яндекс)[9]. Він зручний тому, що на ньому можна розмістити інформацію, якою можна буде поділитись з іншими, при цьому надавши їм посилання на файли та виставивши на них відповідні права доступу.

1.2 Методи шифрування інформації

Шифрування є процесом кодування повідомлень або інформації таким чином, щоб тільки авторизовані особи можуть отримати до них доступ. Воно саме по собі не запобігає втручанню, але захищає зрозумілий вміст від майбутнього перехоплювача[10]. Для шифрування інформації використовуються спеціальні алгоритми – шифри, що генерують шифротекст який може бути прочитаний тільки якщо розшифрований. Для того, щоб прочитати зашифровану інформацію необхідний ключ. Ключ – це певна кількість символів які формуються вільним чином з символів що доступні у системі шифрування.

Шифрування жорсткого диска або віртуального жорсткого диска зазвичай здійснюється стороннім програмним забезпеченням. Зазвичай вираз «повне шифрування диска» не означає, що все на диску зашифроване, оскільки перший фізичний сектор на жорсткому диску або іншому носії інформації або аналогічна область завантажувального диска з кодом, що запускає послідовність завантаження операційної системи залишається не зашифрованою. Хоча й існують програми які здатні зашифрувати весь диск.

В загальному ж методи шифрування дисків або ВД спрямовані на забезпеченняпевними властивостями[11]. Перш за все, дані на диску повинні залишатися конфіденційними. Їх відновлення та збереження мають бути швидкими операціями незалежно від того, де на диску зберігаються дані.Метод шифрування не повинен втрачати дисковий простір, тобто кількість пам`яті, що використовується для шифрування інформації, не повинна бути значно більшою за розмір звичайного тексту.

Загалом виділяють два методи шифрування: симетричне та асиметричне. У симетричному шифруванні ключ шифрування та розшифрування одинаків. Симетричні алгоритми шифрування у свою чергу поділяються на блочні та потокові: блочні співпрацюють з блоками зафіксованого розміру, потокові поетапно опрацьовують текст повідомлення[12].

Симетричні алгоритми не завжди використовуються самостійно. У наш час в криптосистемах використовуються комбінації симетричних та асиметричних алгоритмів, аби отримати переваги обох системах. Так наприклад асиметричні алгоритми можуть використовуватись для розповсюдження ключів симетричних алгоритмів. До найбільш відомими симетричних алгоритмів належать : AES, Twofish, Serpent, TDES та IDEA.

Найпоширенішим симетричним алгоритмом є Advanced Encryption Standard (AES). Це блочний шифр, який підтримує широкий діапазон блоку та ключа[13]. Він має фіксовану довжину ключа у 128 біт, а розмір ключа може приймає значення у 128, 192 чи 256 біт. Оскільки розмір блоку фіксований, то AES оперує із масивом 4x4 байт і носить назву «стан».

Процес шифрування даних складається з декількох етапів:

- обрахування усіх ключів раундів;
- підстановка байтів за допомогою таблиці S-Box;
- зсув у форму з врахуванням різних величин;
- змішування даних всередині кожного стовпця матриці;
- додання форми і ключа раунда.

Для виконання розшифрування операції здійснюються в зворотному порядку, однак замість таблиці S-Box застосовується таблиця зворотних підстановок.

При виконанні шифрування алгоритм має певну кількість раундів(для довжини ключа у 128 біт – 10, 192 біта – 12, 256 біт -14)у яких послідовно виконуються операції :

- subBytes();
- shiftRows();
- mixColumns();
- addRoundKey().

Процедура subBytes проводить нелінійну заміну байтів з використанням таблиці замін (S-Box). Вона забезпечує не лінійність алгоритму шифрування. shiftRows працює із рядками таблиці, циклічно зсуваючи кожен на r байтів, в залежності від номера рядка (для нульового $r = 0$, першого $r = 1$ і т.д.). При виконанні процедури MixColumns, кожна колонка стану перемножується з певним сталим многочленом $c(x)$. У процедурі AddRoundKey, кожен байт стану, з використанням операції XOR, об`єднається з ключем раунду.

Велика поширеність AES, насамперед, зумовлена високим рівнем захисту[14]. Незважаючи на те, що сама природа шифрування має простий алгебраїчний опис зламати зашифровану інформацію надзвичайно важко, а для підбору ключа знадобиться надто багато часу.

Основна перевага симетричних алгоритмів полягає у їх швидкості. При шифруванні вони вимагають менше обчислень. Тим не менш необхідність мати секретний ключ з обох сторін передачі інформації є важливим недоліком, оскільки потенційно ключ може бути перехоплений. Тому ключі необхідно часто змінювати і передавати по безпечних каналах передачі інформації.

При асиметричному шифруванні використовуються два відмінні ключі: відкритий та секретний. Перший використовується для шифрування інформації і може розповсюджуватись відкрито, інший використовується для розшифрування і відомий лише тій стороні якій передається зашифровані дані. Ключ шифрування та ключ дешифрування існують парами, що робить

неможливим використання відкритого ключа з однієї пари та секретного з іншої. Крім того, розшифрування даних за допомогою відкритого ключа неможливе. Всі пари асиметричних ключів пов`язані математичними залежностями. Прикладами асиметричних криптосистем є RSA та DSA.

Асиметричне шифрування вперше було реалізовано в алгоритмі RSA[15]. Він базується на обчислювальній складності задачі факторизації великих цілих чисел. Даний алгоритм складається з чотирьох етапів: генерації ключів, шифрування, дешифрування та розповсюдження ключів.

На першому етапі створюються відкритий та секретний ключі. Для алгоритма RSA етап створення ключів складається з наступних операцій:

1) обираються два прості числа p і q;
2) обчислюється добуток $n(n=p*q)$;
3) вибирається довільне число e ($e<n$), таке, що НСД(e,(p-1)(q-1))=1, тобто не повинне бути взаємно простим із числом (p-1)(q-1);
4) методом Евкліда вирішується рівняння $e*d+(p-1)(q-1)*y=1$,при невідомих змінних d і y, та знаходиться множину пар (d,y), кожна з яких є розв'язком рівняння в цілих числах;
5) числа (e,n) – публікуються як відкритий ключ, у той же час пара чисел (n, d) є секретним ключем, а числа p і q після генерації зазвичай знищуються.

Процес шифрування характеризується тим, що інформація, яка повинна бути зашифрована розбивається на блоки, рівні $k=[\log_2(n)]$ біт. Кожен блок, може бути інтерпретований як число з діапазону $(0; 2^{k-1})$. Для кожного такого числа (m_i) обчислюється вираз $c_i=((m_i)^e) \bmod n$. Блоки c_i це і є зашифровані дані. Їх можна передавати по відкритому каналу, оскільки. операція піднесення в ступінь по модулю простого числа, є незворотною математичною задачею.

Для розшифрування в алгоритмі RSA використовується доведена теорема Ейлера і зокрема окремий її випадок: якщо число n представлене у вигляді двох простих чисел p і q, то для будь-якого x має місце рівність $(x^{(p-1)(q-1)}) \bmod n = 1$. Далі при проведені певних математичних операцій рівняння зводиться до вигляду: $(xe^{*d}) \bmod n = x$. Таким чином щоб прочитати дані

c_i=((m_i)^e)mod n досить піднести c_i (зашифровані дані) до степеня d по модулю m : ((c_i)^d)mod n = ((m_i)^{e*d})mod n = m_i.

Зазвичай весь об'єм даних кодується звичайним блоковим шифром (набагато більш швидким), однак з використанням ключа сеансу. Сам ключ сеансу шифрують саме асиметричним алгоритмом за допомогою відкритого ключа.

1.3 Аналіз існуючих програмних рішень та постановка завдання

Головне завдання дипломної роботи полягає у розробці програмного модуля для роботи із зашифрованими віртуальними дисками.

Для визначення основних вимог яким повинен відповідати проект, розглянемо найпопулярніші програми аналоги: Daemon Tools та Macrium Reflect.

Daemon Tools призначена для створення та запису образів дисків. На даний момент існує п`ять версій програми, які відрізняються функціоналом: Lite, Lite Personal, Pro Advanced, Net та Ultra[16]. Безкоштовно, для некомерційного використання, поширюється лише версія Lite. Хоча для звичайних користувачів може і вистачити набору функцій, що містяться в безкоштовній версій, все ж не всіх вона може задовільнити. Перш за все вона дозволяє створювати до 4 віртуальних CD/DVD приводів, що не надто актуально в наш час, а можливість створювати образи жорстких дисків доступна лише у версії Pro. Враховуючи вартість цієї версії, можна сказати, що це не є економічно вигідно.

До переваг Daemon Tools можна віднести:

- наявність захисту паролем;
- шифрування;
- конвертація форматів образів;
- розбиття образів на декілька файлів.

Однак, не зважаючи на всі переваги, головним недоліком залишається ціна. Користуватись безкоштовною версією незручно, а її функціонал не настільки актуальний як того б хотілось. У той же час повна версія (Pro) коштує надто дорого(55€), що не вигідно для простих користувачів.

Macrium Reflect спеціалізується лише на роботі з віртуальними жорсткими дисками. Основним призначенням програми є створення резервних копій усіх розділів або окремих частин жорсткого диску. Скопійовані файли шифрують ся за допомогою алгоритму AES. Хоч програма і доволі популярна її функціонал доволі обмежений. Фактично вона виконує лише одну просту функцію, а тому їх не можна назвати універсальною.

Якщо в цілому оглянути програми аналоги можна побачити, що більшість з них функціонально застаріли[17]. Одні спеціалізуються на роботі з оптичними дисками, які вже давно застаріли, інші ж навпаки актуальні, однак їх функціонал обмежений(зазвичай реалізують лише декілька функцій). З однієї сторони це добре, адже чим менше функцій містить програма, тим якісніше вони реалізовані, але з іншої це змушує використовувати не одну а декілька програма. Також, на мою думку, варто врахувати, що ціни на такі програми зовсім не відповідають можливостям які ті на дають.

З вище описаного слідує, що головна мета розробки даного програмного забезпечення полягає у створенні програми яка би поєднувала у собі переваги аналогів, однак була б зручнішою у користуванні та економічно вигіднішою. Для виконання завдання буде використовуватись інтегроване середовище програмування Embarcadero Delphi та мова програмування Delphi.

Оскільки популярність використання оптичних дисків з кожним роком падає, то перш за все програма повинна реалізовувати функцію створення віртуальних жорстких дисків(VHD). Потреба у VHD з кожним роком зростає, адже операційні системи з кожним роком модифікуються і стає неможливим використання деяких популярних старих програм. Окрім того не всі програми є багатоплатформними, тобто реалізовані тільки під якусь певну операційну систему. Цю проблему вирішує створення віртуального жорсткого диска і встановлення на нього віртуальної ОС.

Також програма повинна містити функцію шифрування та захисту паролем. Так як наше життя усе тісніше пов`язане з інформаційними технологіями, потреба у захисті інформації як ніколи актуальна, а тому наявність даних функцій обов`язковар[18].

Так як продукт реалізується для широкого використання, то важливою складовою є простий, інтуїтивно зрозумілий інтерфейс. Хоч віртуалізація і актуальна, все ж не всі користувачі персональних комп`ютерів використовують її переваги. Тому враховуючи таких «чайників» усе повинно бути просто та зручно.

Ще одним важлим завданням до створення програми є низькі системні вимоги. Не всі володіють потужними і сучасними комп`ютерами, а тому для того, щоб досягти значного поширення вона не повинна вимагати значних ресурсів, тим самим не створюючи зайвих клопотів користувачу[19].

Програмний засіб повинен бути реалізований під операційну систему Microsoft Windows. Якщо реакція користувачів буде позитивною, то в майбутньому програма може бути реалізована і під інші популярні платформи.

2 ПРОЕКТУВАННЯ ОСНОВНИХ ФУНКЦІЙ

2.1 Аналіз можливостей середовища розробки

При розробці програми використовується програмне середовище Delphi. Це інтегроване середовище розробки, що використовується для створення й підтримки додатків. Спершу воно було призначене виключно для розробки додатків для Microsoft Windows, однак з часом з`явились версії для більшості популярних ОС, зокрема MacOS, iOS, Google Android та Linux, а також, розробники не забувають регулярно оновлювати продукт аби не відставати від конкурентів та не створювати зайвих незручностей для розробників ПЗ.

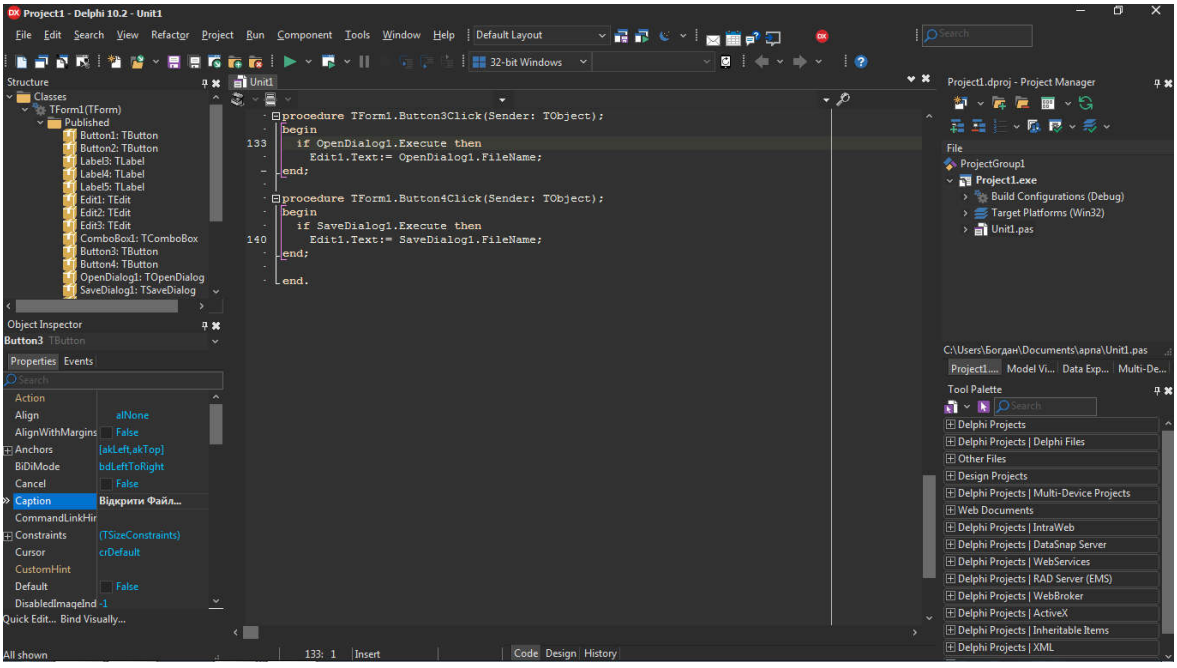

Рисунок 2.1 – Інтерфейс Embarcadero Delphi 10.2

Delphi – це мова програмування високого рівня, що базується на мові Pascal і призначена для швидкої розробки прикладного програмного забезпечення[20]. Створенні програми незалежні від стороннього ПЗ, як наприклад Microsoft .NET Framework Java Virtual або Machine. Виділення і звільнення пам`яті контролюється кодом користувача, що з однієї сторони збільшує вимоги до коду, а з іншої – робить можливим створення складних додатків з високими вимогами до роботи у реальному часі.

Delphi слідує наступним парадигмам об`єктно-орієнтованого програмування: інкапсуляції, наслідуванні, поліморфізмі[21]. Термін інкапсуляція позначає спостережуване в об’єктах об’єднання даних і операцій в одне ціле. Наслідування означає процес створення нових класів на основі вже існуючих. Новий клас містить успадковані ознаки від батьківського класу, а також нові необхідні для робот із конкретним фрагментом коду. Поліморфізм означає, що у похідних класах можна змінювати роботу методів, що вже існують в базовому класі, при чому програмний код, який керує об’єктами батьківського класу , придатний для керування об’єктами дочірнього класу без жодної модифікації.

Для підтримки ООП у Delphi існують об’єктні типи даних, які одночасно описують і дані, і операції над ними[22]. Об’єктні типи даних називаються класами, а їх екземпляри – об’єктами. Класи зазвичай описують суть, що моделюється у програмі. Так, наприклад клас Tdiskgauge описує вимірювач дискового простору.

Для роботи з файлами у Delphi існує декілька можливостей. Перш за все можна використовувати традиційний набір функцій з Turbo Pascal, який працює через файлові змінні.

Також, при організації операції файлового введення або виведення для додатку має значення, яка інформація міститься в файлі. Зазвичай це рядки, проте бувають і двійкові дані чи структурована інформація. Тому відомості про тип даних потрібно задати спочатку. У цьому випадку використовуються спеціальні файлові змінні для визначення типу файлу. Їх поділяють на ті, що не типізуються та ті, що типізуються[23]. Саме тому, на початку роботи з файлом, перш за все, потрібно описати файлову змінну, яка відповідає типу даних цього файлу. У майбутньому вона використовуватиметься при зверненні до файлу.

Крім того, Delphi дає змогу створювати файли, що не типізуються. Для їхнього позначення використовується ключове слово file: var Untypedfile: file; Такі змінні використовують для того, щоб організувати швидке та ефективне введення/виведення в незалежності від типу даних. Дані читаються чи записуються у вигляді двійкового масиву. Для цього застосовують спеціальні

процедури блокового читання і запису. В свою чергу, типізовані файли забезпечують операції введення та виведення з врахуванням конкретного типу даних. Для їх оголошення використовують ключове слово file of, після якого вказується конкретний тип даних. До того ж, можна використовувати будь-які типи даних фіксованого розміру, окрім вказівників. Можна застосовувати й структурні типи, якщо їх складові частини задовольняють вищезгадані умови[24]. Для роботи з текстовими файлами застосовується файлова змінна Textfile.

Найпоширенішими операціями при роботі з файлами є читання та запис. Щоб їх здійснити використовуються спеціальні функції вводу/виводу.

Для їх виконання необхідно здійснити наступні дії:

- оголосити файлову зміну потрібного типу;
- використовуючи функції Assignfile зв`язати її з необхідним файлом;
- відкрити файл використавши функції Append, Reset, Rewrite;
- провести операції читання та запису, при чому в залежності від складності завдання й структури даних, може використовуватись цілий ряд допоміжних функцій;
- закрити файл використавши функцію Closefile.

Як приклад варто розглянути фрагмент коду.

```
…
var F: Textfile;
S: string;
begin
if Opendlg.Execute
then Assignfiie(F, Opendlg.FileName)
else Exit; Reset(F);
while Not EOF(F) do
begin
Readln(F, S);
Memo.Lines.Add(S);
end;
```

```
Closefile(F);
end;
...
```

Уданому циклі виконується читання з файлу текстови хрядків і здійснюється їх запис в компонент Tmemo. Процедура Readln зчитує поточний рядок файлу і переходить на наступний рядок. Цикл виконуватиметься, поки EOF не повідомить, що досягнено кінець файлу. Після завершення операції читання файл закривається. Схожий код використовується і для запису даних у файл.

Відкрити файл можна трьома різними процедурами, в залежності від того, як його планується використовувати.Reset – відкриває існуючий файл для читаннята запису, поточна позиція встановлюється на першому рядку файлу. Append – відкриття файлу для запису в кінці останнього рядка, поточна позиція в кінці файлу. Rewrite – створення нового файлу й його відкриття, поточна позиція на початку файлу. Якщо однойменний файл існує, то він перезаписується.

Читання даних текстових та типізованих файлів виконується за допомогою процедур Read та Readln. За один виклик процедури можна зчитати дані у довільне число змінних. Схожим чином працюють процедури запису у файл Write та Writeln. Кожен параметр запису може мати форму: Pn [: Minwidth [: Decplaces]], де Pn – змінна, що виводиться, або вираз, а Minwidth – мінімальна ширина поля в символах(вона не може бути більшою 0). Decplaces вміщує кількість десяткових ми волів після коми при відображенні дійсних чисел з фіксованою крапкою.

Також існує режим блокового введення/виводу даних між певним фалом та областю дискового простору[25]. Він відрізняється значною швидкістю передачі даних, причому існує пропорційна залежність розміру передаваного блоку та швидкості (чим більший блок, тим більша швидкість його передачі).

Для реалізації даного режиму потрібно використовувати лише не типізовані файлові зміні. Для виконання операцій використовуються процедури Blockread і Blockwrite. Використовуючи блокове читання чи запис розмір блоку

потрібно обирати так, щоб він був кратний розміру одного значення, що зберігається у файлі. Наприклад, якщо у файлі зберігаються значення типу Double, то розмір блоку може бути рівний 8, 16,24,32 і т.д.

2.2 Розробка алгоритмів роботи системи

Архітектура програмного забезпечення включає сукупність всіх рішень про організацію програмної системи. Вона включає вибір структурних елементів та їх інтерфейсів, за допомогою яких була складена система; з'єднання вибраних елементів структури та архітектурний стиль який направляє всю систему. Для графічного представлення архітектури використовуються UML-діаграми. Для їх створення використовується програма VisualParadigm.

UML застосовують на всіх етапах розробки розробки прикладних програм. Завдяки різним видам діаграм, та широкому набору можливостей представлення певних аспектів систем, UMLє універсальним засобом опису програмних та ділових систем.

Найчастіше UML діаграми створюються з використанням додатку Visual Paradigm(Рисунок 2.2). Visual Paradigm - це UML інструмент, що підтримує UML 2, SysML та позначення моделювання бізнес-процесів в групі керування об'єктами. Окрім підтримки моделювання, він забезпечує створення звітів та можливості кодування, включаючи створення коду[26]. Він також може ревертувати інженерні схеми з коду написаного на мовах Java або C++, а також забезпечити інтерактивну обробку для різних мов програмування. Даний програмний засіб підтримує всі 13 типів UML діаграм і чимало інших. Таким чином даний програмний засіб дає змогу максимально детально описати проект розробки не змушуючи користуватись зайвими сторонніми засобами і тому саме його й буде використано для опису розробленого додатку.

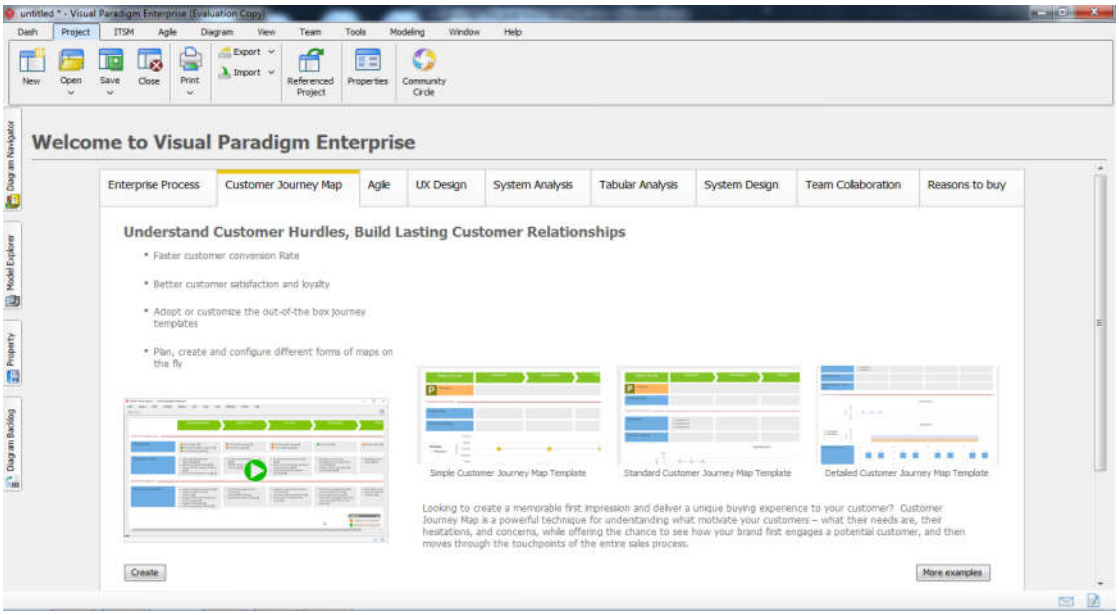

Рисунок 2.2 – Середовище Visual Paradigm

На рисунку 2.3 зображена діаграма прецедентів, що зображає відношення між акторами та прецедентами системи.

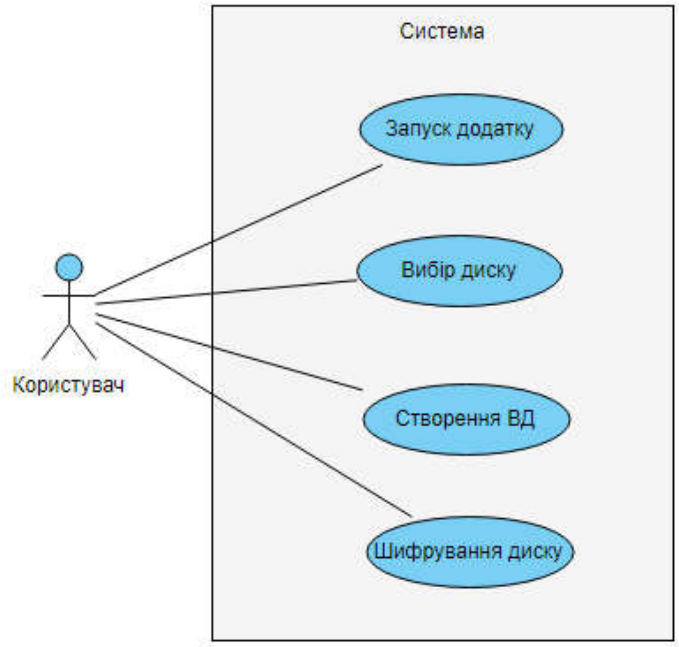

Рисунок 2.2 – Діаграма прецедентів

Фактично вона показує які дії здатен виконати користувач використовуючи розроблений програмний засіб.Для того, щоб показати, яким чином відбувається взаємодія об’єктів програмного засобу за часом,

використовується діаграма послідовності. Вона дає змогу побачити які процеси виконуються в першу чергу, а виконання яких потребує попередніх дій.

На рисунку 2.3 зображена діаграма прецедентів розробленого програмного додатку. Вона показує яким чином користувач взаємодіє з програмою, та які дії доступні для виконання на кожному етапі роботи.

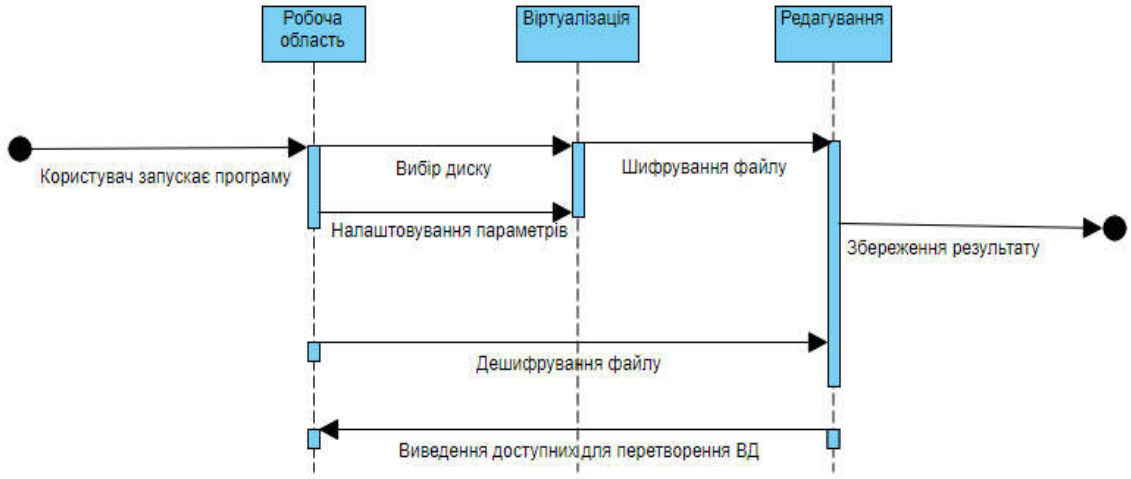

Рисунок 2.3 – Діаграма послідовності

Для представлення структури програми необхідно використовувати діаграму класів[27]. Вона дає змогу побачити ієрархію класів, що використовуються в програмі, які функції й змінні якому класу належать. Крім того, даний тип діаграм також дозволяє побачити залежність класів, тобто як результат роботи одного, впливатиме на дії іншого та інше.

Існують різні точки зору на побудову діаграм класів в залежності від їх застосування:

– концептуальна(діаграма описує модель предметної області, в ній присутні тільки класи прикладних об’єктів);

– специфікації(діаграма застосовується при проектуванні інформаційних систем);

– реалізації(діаграма містить тільки класи які використовуються в програмному коді).

2.3 Проектування інтерфейсу додатку

Проектування інтерфейсу ставить на меті розробку «зовнішнього вигляду» програми, який буде з`являтись перед користувачем після її запуску. Перш за все він повинен бути інтуїтивно зрозумілим, щоб користувачу не потрібно було б пояснювати як ним користуватись. Окрім того, у нього повинен бути швидкий доступ до всіх основних функцій програмного додатку.

Інтерфейс – це свого роду «обличчя» програми. Він створює перше враження і по ньому звичайні користувачі будуть оцінювати зручність користування додатком. Щоб не викликати у користувача відрази він повинен мати наступні властивості: природність, узгодженість, гнучкість, дружність, простота, принцип зворотного зв`язку та естетична привабливість.

Природній інтерфейс – такий, що не змушує користувача змінювати звичні методи вирішення задачі. Це означає, що повідомлення та результати, які видає програма, не повинні вимагати додаткових пояснень.

Узгодженість дозволяє користувачам переносити знання на нові завдання, освоювати нові аспекти швидше і завдяки цьому фокусувати свою увагу на вирішенні конкретної задачі[28]. При цьому користувач не буде витрачати свій час на те, щоб зрозуміти яку операцію виконувати та чи інша команда. Узгодженість важлива для всіх аспектів інтерфейсу включаючи імена команд, візуальне представлення інформації і поведінка інтерактивних елементів. Важливо, щоб одна і таж команд виконувала одну і ту ж функцію, незалежно від того де вона знаходиться.

Користувачі зазвичай вивчають особливості роботи з новим програмний продуктом методом проб і помилок. Тому на кожному етапі роботи інтерфейс повинен приймати до уваги таку поведінку і дозволяти виконувати тільки відповідний набір дій. Тобто, якщо користувач виконує недопустиму дію програма повинна попереджати його, що це може нашкодити системі та по можливості давати змогу відмінити останні дії.

Принцип зворотного зв’язку означає, що кожна дія користувача повинна отримувати візуальне або звукове підтвердження того, що програма сприйняла введену команду, при чому вид реакції повинен враховувати природу виконаної дії[29]. Якщо на виконання певних команд потрібен деякий час, то користувач повинен бачити прогрес виконання у вигляді невеликого діалогового вікна.

Інтерфейс повинен бути максимально простим. Імена команд повинні бути максимально короткими, однак достатньо інформативними, щоб у користувача не виникало зайвих запитання. На екрані повинна бути представлена лише та інформація, та ті команди які необхідні при виконанні конкретної операції. Зайва нагромадженість може заплутати користувача, що може привести до неправильних дій. Головне вікно програми має бути представлене простим меню, розташованим з лівого боку, панеллю інструментів яка повинна включати лише найчастіше використовуванні команди та «Справка», в якій міститиметься вся інформація про програму, та те як нею користуватись. Усі команди мають бути представлені відповідними значками, які характеризуватимуть ту дію, яку виконує команда. Доступ до додаткових функцій надаватиметься через меню. Окрім того, усі основні команди повинні мати можливість виконуватись через комбінацію клавіш. Це спрощує користування програмою для професійних користувачів та тих людей яких швидкість виконання завдання грає важливу роль.

Гнучкість інтерфейсу – це здатність враховувати рівень підготовки та працездатності користувача. Дана властивість означає можливість зміни структури діалогу або вхідних даних[30]. При потребі програма може бути доповнена додатковими функціями, а інтерфейс модифікований під нові потреби. Окрім того, самі технології щорічно змінюються. Це все не повинно створювати додаткових проблема користувачеві при експлуатації програмного додатку.

Коректне візуальне представлення використовуваних об’єктів забезпечує передачу важливої додаткової інформації про поведінку та взаємодію різних об’єктів. У той же час варто пам’ятати, що кожен візуальний елемент, який

з’являється на екрані потенційно вимагатиме уваги користувача, яка тим не менш не безмежна.

Варто враховувати, що при створенні інтерфейсу важливо роль грає кольорова гамма та шрифт. Гармонічне поєднання кольорів спрощує сприйняття, а м’які кольори не викликатимуть втоми при довгій роботі. Для цих цілей ідеально підійде поєднання білого та світло-голубий колір. Розмір шрифту повинен підбиратись в залежності від величини елемента вікна у якому розташована команда.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Засоби розробки програмного засобу

Оскільки на даний момент найпоширенішою операційною системою залишається Microsoft Windows, то зосередимось на тому аби реалізувати продукт саме для її користувачів. Головними перевагами Microsoft Windows перед конкурентами є:

– багатозадачність, що дозволяє оптимально організувати процесорний час, тобто синхронність потоків та процесів;

– надійні протоколи кодування та шифрування, які дозволяють захищати персональні дані користувачів та іншу інформації від несанкціонованого доступу;

– зручна організація файлової системи, що дозволяє швидко отримати доступ до необхідного каталогу;

– підтримка об`єктів нижчого рівня та DOS додатків;

– доступність користувацького інтерфейсу, який дозволяє максимально швидко адаптуватись до робити з даною операційною системою.

Оскільки в ОС Microsoft Windows існують 32- та 64-розрядні системи, то при розробці програмного додатку використовуватимо відповідний драйвер та бібліотеку VirtualDisk.dll яка дозволить створювати віртуальні диски. Також, для покращення безпеки реалізуємо програму таким чином, що доступ до створеного зашифрованого диску можна було здійснити лише з використанням прав адміністратора.

Починати розробку програмного додатку необхідно із розробки сценарія діалогу користувача із системою. Це важливо оскільки важливо розуміти, що програмний продукт направлений на широку аудиторію може викликати складнощі у непідготовлених користувачів, а тому аби уникнути виникнення потенційних помилок потрібно одразу передбачити всі можливі варіанти розвитку подій.

Для опису діалогу між користувачем та додатком використовується діаграма взаємодії(Рисунок 3.1).

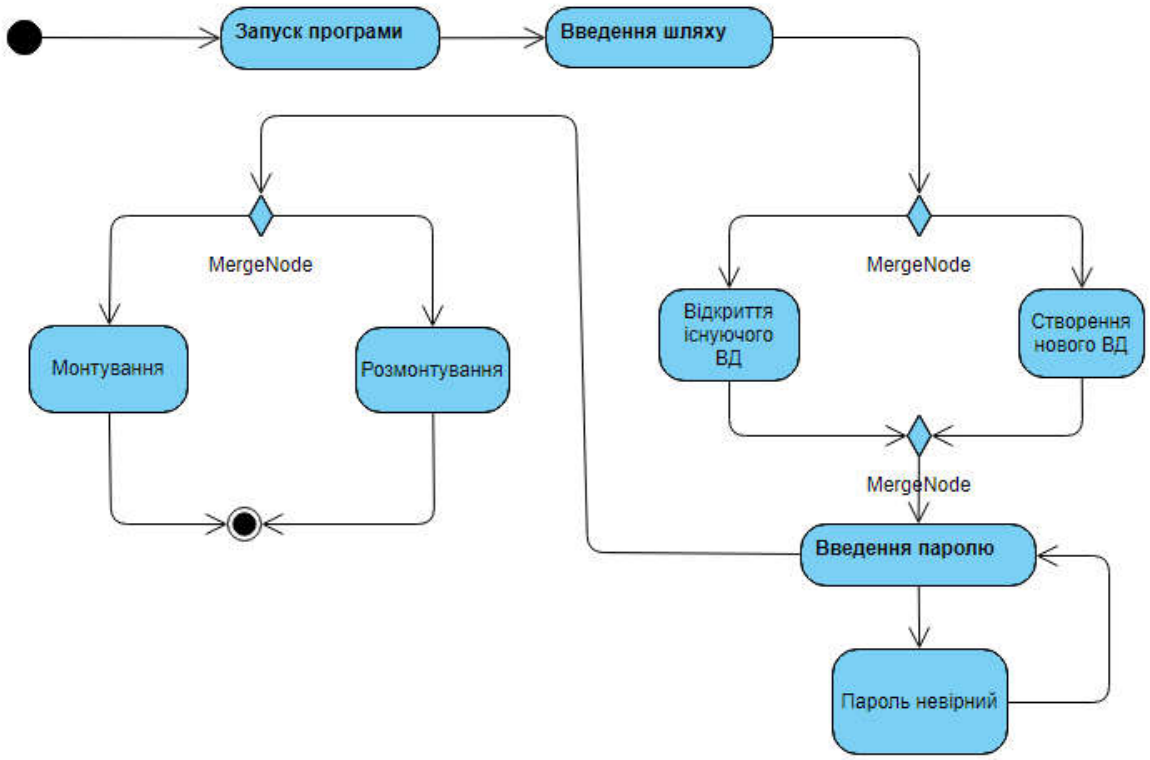

Рисунок 3.1 – Діаграма взаємодії

З представленої діаграми можна побачити, як програма поведе себе в залежності від того які дії виконує користувач. Створивши даний сценарій ми маємо змогу перейти до безпосередньої розробки самого програмного засобу.

Для реалізації поставленої задачі використовувались наступні функції:

– function VD_Initialize(): DWORD; stdcall; external 'VirtualDisk.dll';

– function VD_MountDisk(DriveLetter: Char; FileName: WideString; FileSize: TLargeInteger; Key: WideString; ReadOnly: Boolean): DWORD; stdcall; external 'VirtualDisk.dll';

– function VD_UnMountDisk(DriveLetter: Char; ForceUnMount: Boolean): DWORD; stdcall; external 'VirtualDisk.dll';

– function VD_QueryDiskFileName(DriveLetter: Char; var FileName: WideString): DWORD; stdcall; external 'VirtualDisk.dll';

– function VD_InstallDriver(DriverPath: WideString; StartType: DWORD

= $00000003): DWORD; stdcall; external 'VirtualDisk.dll';

– function VD_UnInstallDriver: DWORD; stdcall; external 'VirtualDisk.dll';

– function VD_StartDriver: DWORD; stdcall; external 'VirtualDisk.dll';

– function VD_StopDriver: DWORD; stdcall; external 'VirtualDisk.dll';

– function IsWow64Process(hProcess: THandle; var Wow64Process: BOOL): BOOL; stdcall; external kernel32 name 'IsWow64Process';

Дані функції реалізують всі основні задачі які виконуються програмою:

– ініціалізація системи користувача;

– встановлення відповідного драйвера;

– монтування диска;

– розмонтування диска;

– шифрування диска.

Для шифрування диска використовується алгоритм AES 128, який на даний момент вважається найпоширенішим алгоритмом шифрування для захисту персональних даних.

3.2 Програмні модулі системи

В основі розробки додатку в середовищі Delphi лежить проект. Його основною частиною є форма, в якій містяться компоненти необхідні для вирішення задачі. Крім того, до складу проекту ще входять програмні модулі, зовнішні бібліотеки, піктограми, картинки. Кожний такий елемент розміщується в окремому файлі, адже має строго певне призначення. При компілюванні компілятор обробляє кожен файл проекту після чого будує виконуваний файл.

Створення додатку розпочнемо із створення форми. Форма – це текстовий файл із розширенням .drm, а по суті – «обличчя» нашого додатку, адже це те, що бачить користувач при роботі з програмою. У даному файлі на

спеціальній мові задаються початкові значення для властивостей форми та її компонентів, зокрема кнопок, області вводу та ін. Форма, яка використовуватиметься у розроблюваному додатку зображена на рисунку 3.2.

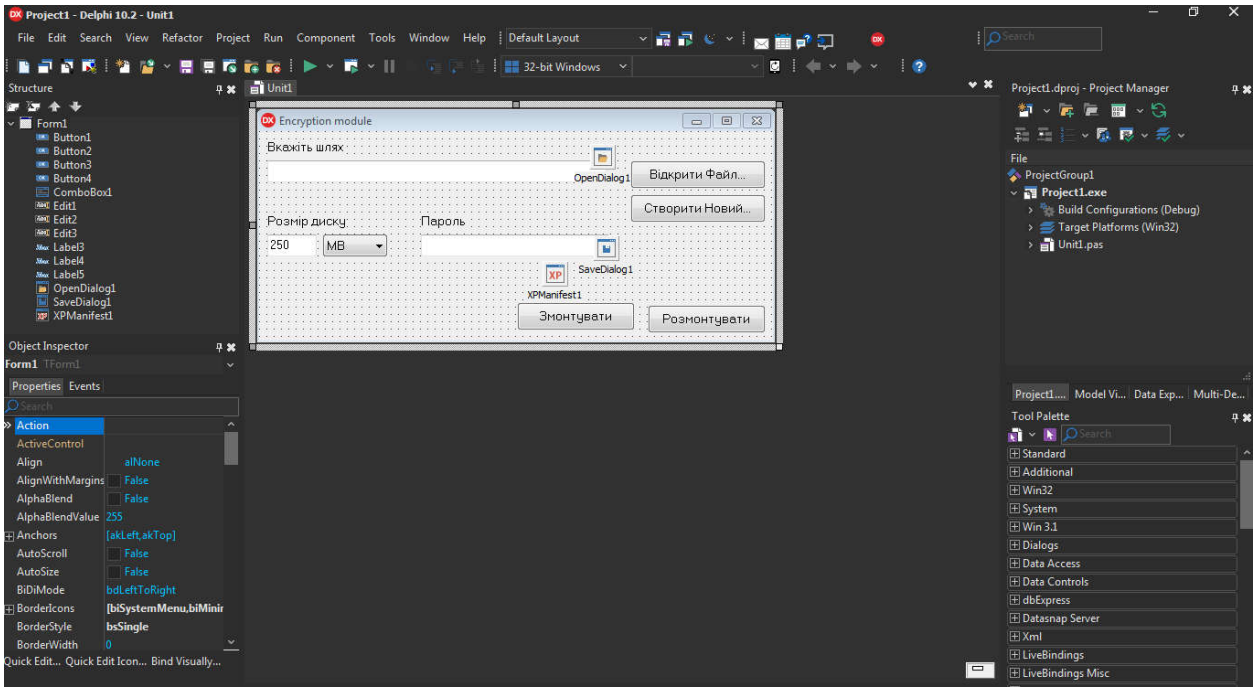

Рисунок 3.2 – Форма проекту

Форма має чимало властивостей в яких доволі легко заплутатись адже всі властивості подані в алфавітному порядку, тому схожі за значенням можуть бути розкидані по списку. Взагалі, при створені нового вікна середовище самостійно задає початкові значення, однак їх можна легко змінити відповідно до вимог.

Для спрощення роботи з формами використовується програмний ідентифікатор(Рисунок 3.3). У ньому містяться всі властивості які користувач використав при розробці. Якщо виникає потреба щось змінити, можна скористатись саме ним. Це дуже зручно коли доводиться працювати з масивним описом. Достатньо просто натиснути на певний об’єкт і в ідентифікаторі виділяться ті властивості які відповідають за його опис.

Кожній формі відповідає свій програмний модуль в якому містяться всі зв`язані з формою оголошення та методи обробки подій. Вони розміщуються в окремих файлах та мають розширення .pas. Крім того, в них можна тримати

допоміжні процедури, класи, функції. В нашому випадку для опису роботи програми необхідний лише один програмний модуль, який і міститиме всю інформацію.

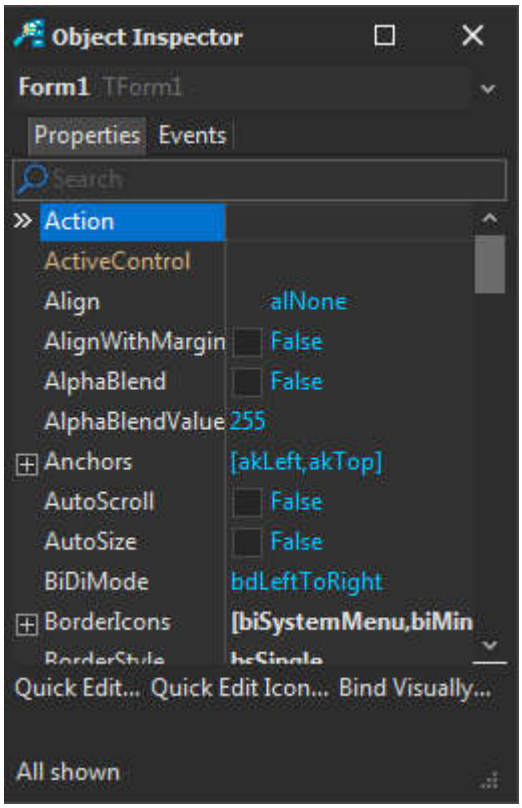

Рисунок 3.3 – Програмний ідентифікатор форми

Розберемо детально створення програмного модуля. Починається він із назви модуля та змісту інтерфейсної секції. В ній містяться усі стандарті модулі бібліотеки VCL та класи компонентів, які були використані при створені форми:

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, ExtDlgs, XPMan;

Модулі можна завдавати і вручну, проте середовище може самостійно поповнювати список, коли на форму додаються нові компоненти.

У розділі опису type оголошується клас форми, який за замовчуванням має назву TForm1. Компоненти, що були розміщенні на формі представленні полями. Далі слідують заголовки методів обробки подій. Їх назви формуються

автоматично на основі імені компоненту та події, яку він генерує. За замовчуванням і поля, і методи обробки подій приймають атрибут видимості (published). Тому з ними можна працювати на візуальному рівні, зокрема бачити їх імена у вікні властивостей. Після чого вказуються процедури, які виконуватимуться внаслідок виконання користувачем тих чи інших дій:

- FormCreate(Sender: TObject);
- Button1Click(Sender: TObject);
- Button2Click(Sender: TObject);
- Button3Click(Sender: TObject);
- Button4Click(Sender: TObject);
- Label4Click(Sender: TObject);

Зазвичай даний фрагмент коду генерується автоматично на основі дій користувача, які він виконує із формою, тому якшо необхідно внести зміни, то краще використовувати саме її. Якщо виникає потреба у додатковому описі, то можна використати додаткові секції private та public. З ними можна працювати лише на програмному рівні. Прийнято, що у секції private містяться дані які необхідні виключно для конкретної форми, у той час як в public поміщається інформація яка використовується в інших модулях. Оскільки в нашому випадку працюємо з єдиним модулем то різниця між даними секції не представлятиме важливості. Після того, як ми описали клас, можна перейти до оголошення об’єкту форми.

Після завершення опису інтерфейсної частини, переходимо до безпосередньої реалізації головного завдання нашого проекту, тобто до розділу implementation.

Перш за все потрібно підключити файл опису форми: {$R *.dfm}. Далі слідує опис процедур яка виконують ті чи інші функції. Так спершу, в залежності від того, якою системою користується користувач(32- або 64-розрядною) встановлюватиметься відповідний драйвер.

Далі, описується усі помилки які можуть виникнути якщо неправильно експлуатувати програму. Так наприклад, якщо користувач не вів пароль програма видасть помилку і вимагатиме його вводу або якщо користувач ввів

неправильний шлях до файлу то програма видасть помилку про неправильний шлях. Усе це необхідно для того, щоб користувач чітко розумів чому програма не виконує тих дій, які від неї вимагаються, та що йому необхідно виконати для того, щоб продовжити роботу.

Коли опис всіх необхідних компонентів завершено, здійснюється опис двох основних процедур: монтування та розмонтування. Саме для при їх описі ми і використовуватимо бібліотеку VirtualDisk.dll.

Після написання коду, переходимо до створення головного файлу проекту. Він необхідний, для того, щоб зв`язати всі файли які використовуються при розробці проекту, а також для того, щоб ми нарешті могли відкомпілювати проект. Даний файл має розширення .drp і для кожного проекту існує лише один такий файл.

Нарешті після завершення написання коду здійснюється компіляція та збирання проекту. Компіляція – процес отримання об’єктних модулів із вихідних текстів програмних модулів. Збирання – процес отримання виконуваного файла з об’єктних модулів. Саме виконуваний файл користувач використовуватиме для запуску програмного засобу.

Також, для користувачів створена можливість самостійно обирати розміри створеного віртуального диску. Це дуже зручно, оскільки фіксований формат приносить чимало клопотів. Так наприклад, якщо у користувача виникне потреба у тому, щоб завантажити декілька важливих текстових документів, а розмір диску надто великий, то великий відсоток дискового простору залишиться невикористаним, при чому кількість реально зайнятої пам`яті залишиться незмінною або навпаки, через великий розмір файлу він просто не поміститься на віртуальний диск.

Таким чином, після завершення програмної реалізації проекту можна перейти до процесу тестування його роботи. Якщо виникнуть якісь помилки їх можна буде виправити й таким чином донести до користувача готовий робочий продукт.

3.3 Тестування та верифікація

Перед тим як представляти розроблений продукт необхідно протестувати. Для цього, запустимо програму і перевіримо чи виконуються всі основні функції. Після запуску ми бачимо інтерфейс програми(Рисунок 3.4)

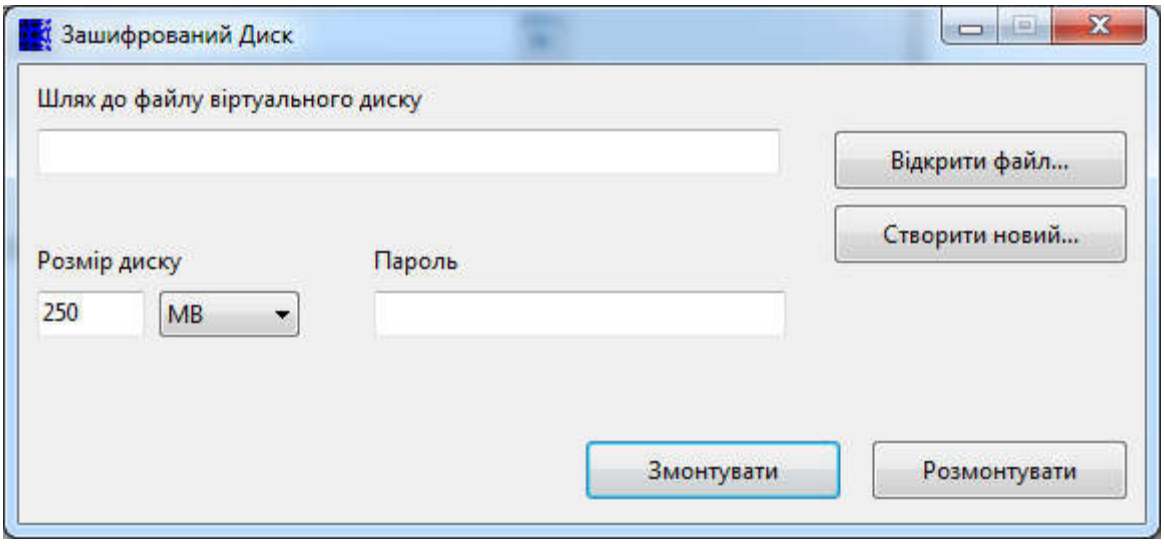

Рисунок 3.4 – Інтерфейс розробленого додатку

З рисунка бачимо, що всі кнопки відображаються правильно. Далі перевіримо виконання головних функцій. Створимо новий зашифрований диск та введемо простий пароль(Рисунок 3.5)

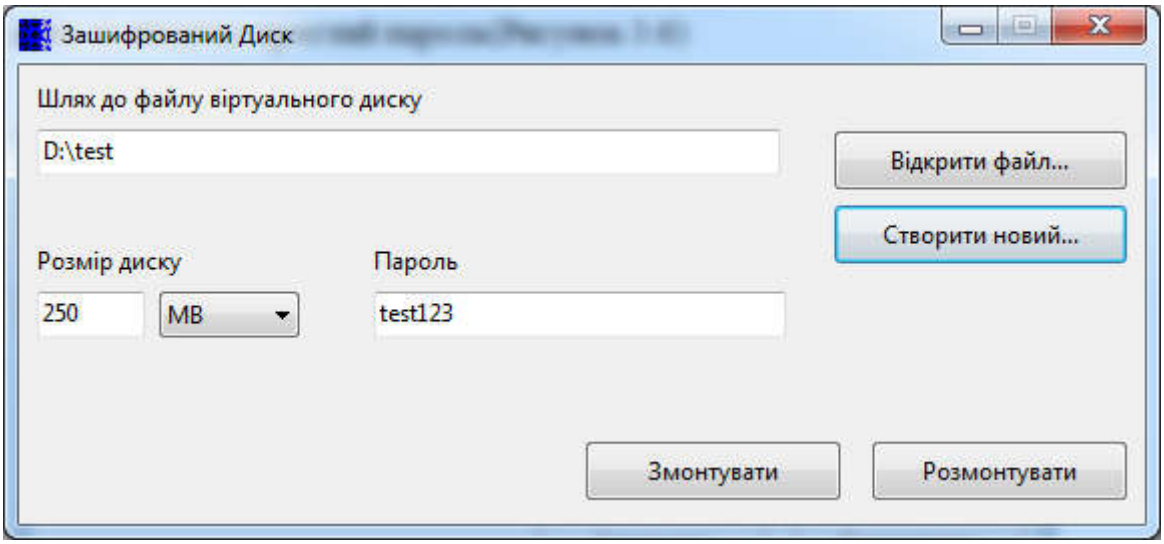

Рисунок 3.5 – Створення віртуального диску

Після того, як натиснемо кнопку змонтувати на комп`ютері з`явиться новий «Локальний диск M:»(Рисунок 3.6). При цьому сам файл диску знаходитиметься на локальному диску D. Перш ніж починати працювати з даним диском необхідного його відформатувати. Після чого з`являється можливість завантажувати на нього необхідну інформацію.

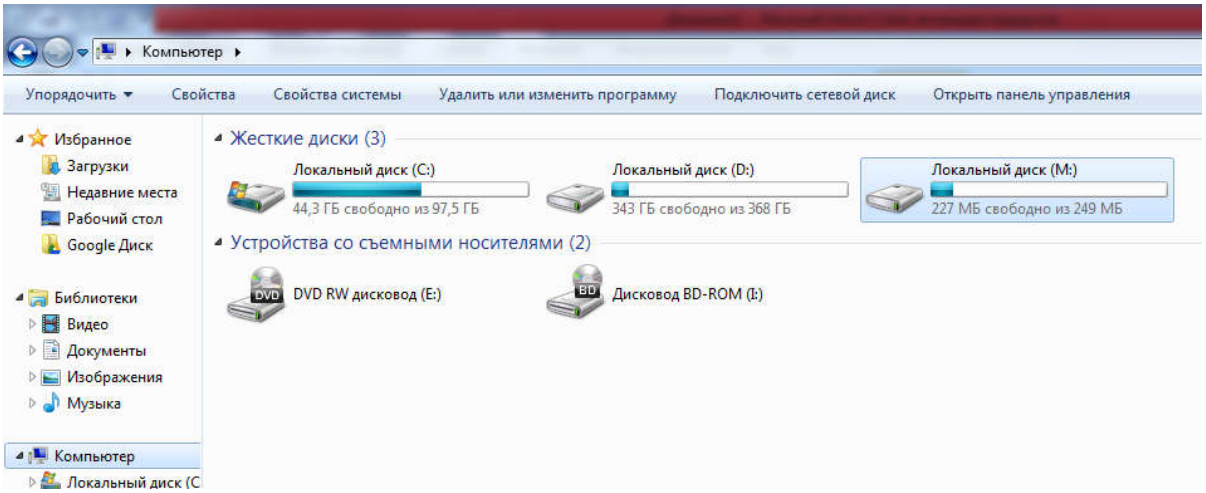

Рисунок 3.6 – Створений віртуальний диск

Далі, перевіримо як диск працює із готовим віртуальним диском. Для цього відкриємо його, використовуючи програму. І спробуємо завантажити який-небудь файл. Як видно із рисунка 3.7 файл успішно завантажено і над можна виконувати ті чи інші дії.

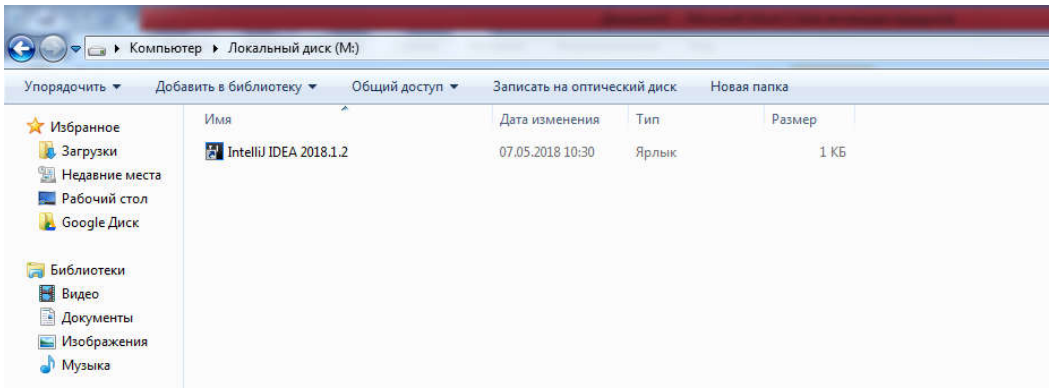

Рисунок 3.7 – Вміст створеного віртуального диску

Таким чином, після проведення тестування можна із впевненістю сказати, що програма працює відмінно, жодних помилок виявлено не було. Для тестування використовувався ноутбук HP-Pavilion g6 із встановленою операційною системою Microsoft Windows 7 (32-bit).

4 ТЕХНІКО-ЕКОНОМІЧНИЙ РОЗДІЛ

У даному розділі дипломної роботи проводиться економічне обґрунтування доцільності розробки програмного продукту спостереженна за мережевою активністю ОС Windows.

Зокрема, здійснюється розрахунок витрат на розробку даного програмного забезпечення, експлуатаційних витрат, ціни на споживання проектного рішення, визначаються показники економічної ефективності нового програмного продукту, обґрунтовуються відповідні висновки.

Розроблений програмний модуль призначений для захисту системи автоматизованого управління сервісними функціями житлового приміщення.

4.1 Розрахунок витрат на розробку програмного забезпечення

Витрати на розробку і впровадження програмних засобів (К) включають:

$$K = K_1, + K_2, \tag{4.1}$$

де K_1 - витрати на розробку програмних засобів, грн.;

K_2 - витрати на відлагодження і дослідну експлуатацію програми рішення задачі на комп'ютері, грн.

Витрати на розробку програмних засобів включають:

- витрати на оплату праці розробників ($B_{ОП}$);
- витрати на відрахування у спеціальні державні фонди ($Вф$);
- витрати на покупні вироби ($Пв$);
- витрати на придбання спецобладнання для проведення експериментальних робіт ($Об$);
- накладні витрати ($Н$);

– інші витрати (*Ів*).

Витрати на оплату праці включають заробітну плату (ЗП) всіх категорій працівників, безпосередньо зайнятих на всіх етапах проектування. Розмір ЗП обчислюється на основі трудоємності відповідних робіт у людино-днях та середньої ЗП відповідних категорій працівників.

У розробці проектного рішення задіяні наступні спеціалісти - розробники, а саме: керівник проекту; студент-дипломант; консультант техніко-економічного розділу (таблиця 4.1).

Таблиця 4.1 - Вихідні дані для розрахунку витрат на оплату праці

Посада виконавців	Місячний оклад, грн.
Керівник ДП, доцент	5470
Консультант техніко-економічного розділу, доцент	5470
Студент	1287

Витрати на оплату праці розробників проекту визначаються за формулою:

$$B_{ОП} = \sum_{i=1}^{N} \sum_{j=1}^{M} n_{ij} \cdot t_{ij} \cdot C_{ij} \ , \qquad (4.2)$$

де n_{ij} – чисельність розробників *i*-ої спеціальності *j*-го тарифного розряду, осіб;

t_{ij} – затрачений час на розробку проекту співробітником *i*-ої спеціальності *j*-го тарифного розряду, год;

C_{ij} – годинна ставка працівника *i*-ої спеціальності *j*-го тарифного розряду, грн.

Середньогодинна ставка працівника може бути розрахована за формулою:

$$C_{ij} = \frac{C_{ij}^{0}(1+h)}{РЧ_{i}}, \qquad (4.3)$$

де C_{ij} – основна місячна заробітна плата розробника *i*-ої спеціальності *j*-го тарифного розряду, грн.;

h – коефіцієнт, що визначає розмір додаткової заробітної плати (при умові наявності доплат);

$РЧ_i$ - місячний фонд робочого часу працівника *i*-ої спеціальності *j*-го тарифного розряду, год. (приймаємо 168 год.).

Результати розрахунку записуються до таблиці 4.2.

Таблиця 4.2 - Розрахунок витрат на оплату праці

Посада виконавців	Час розробки, год	Погодинна заробітна плата, грн/год.	Витрати на розробку, грн
Керівник ДП, доцент	20	80,42	1608,4
Консультант техніко-економічного розділу, доцент	2	80,42	160,84
Студент	120	7,14	856,8
Разом			2636,04

Величну відрахувань у спеціальні державні фонди визначають у відсотковому співвідношенні від суми основної та додаткової заробітних плат. Згідно діючого нормативного законодавства єдиний соціальний внесок складає 16,4% від суми заробітної плати: $B_Ф = 0{,}164 \cdot B_{ОП}$, $B_Ф = \frac{16{,}4}{100} \cdot 2636{,}04 = 430{,}02$ *грн*.

У таблиці 4.3 наведений перелік купованих виробів і розраховані витрати на них.

Таблиця 4.3- Розрахунок витрат на матеріали та комплектуючі

Найменування купованих виробів	Одиниця виміру	Ціна, грн	Кількість купованих виробів	Сума, грн	Транспортні витрати (10% від суми)	Загальна сума, грн
Папір (формат А4)	*уп*	80,0	2	160,00	16,0	176,0
Ручка кулькова	*шт*	4,0	2	8,00	0,8	8,80
Диски CD-R	*шт*	2,0	2	4,00	0,4	4,40

Продовження таблиці 4.3

Зошит, 96 арк	*шт*	9,50	1	9,50	0,95	10,45
Тонер для принтера	*уп*	49	1	49	4,9	53,9
Ліцензія середовища розробки	*шт*	6000	1	6000	0	6000
Разом						6253,55

Витрати на використання комп'ютерної техніки включають витрати на амортизацію комп'ютерної техніки, витрати на користування програмним забезпеченням, витрати на електроенергію, що споживається комп'ютером. За даними обчислювального центру ТНЕУ для комп'ютера типу IBM PC/ATX вартість години роботи становить 5,32 грн. Середній щоденний час роботи на комп'ютері – 2 години. Розрахунок витрат на використання комп'ютерної техніки приведений в таблиці 4.4.

Накладні витрати проектних організацій включають три групи видатків: витрати на управління, загальногосподарські витрати, невиробничі витрати. Вони розраховуються за встановленими відсотками до витрат на оплату праці.

Таблиця 4.4- Розрахунок витрат на використання комп'ютерної техніки

Назва етапів робіт, при виконанні яких використовується комп'ютер	Час використання комп'ютера, год.	Витрати на використання комп'ютера грн.
Проведення досліджень та оформлення їх результатів	60	319,2
Оформлення техніко-економічного розділу	8	42,56
Оформлення ДП	12	63,84
Разом	80	425,6

Середньостатистичний відсоток накладних витрат приймемо 150% від заробітної плати, тому $H = 1{,}5 \cdot 2636{,}04 = 3954{,}06$ (*грн.*).

Інші витрати є витратами, які не враховані в попередніх статтях. Вони

становлять 10% від заробітної плати: $I = 2636,04 \cdot 0,1 = 263,60$ (грн.)

Витрати на розробку програмного забезпечення складають $K_1 = B_{ОП} + B_Ф + B_{ПВ} + H + I = 2636,04 + 430,02 + 6253,55 + 3954,06 + 263,60 = 12537,27$ (грн.).

1.1 Витрати на відлагодження і дослідну експлуатацію програмного продукту визначаємо за формулою:

1.2

$$K_2 = S_{м.г.} \cdot t_{від}, \quad (4.4)$$

де $S_{м.г.}$ - вартість однієї машино-години роботи ПК, грн./год;

$t_{від}$ - комп'ютерний час, витрачений на відлагодження і дослідну експлуатацію створеного програмного продукту, год.

Загальна кількість днів роботи на комп'ютері дорівнює 30 днів. Середній щоденний час роботи на комп'ютері – 2 години. Вартість години роботи комп'ютера дорівнює 5,32 грн. Тому $K_2 = 5,32 \cdot 60 = 319,2$ *грн*.

На основі отриманих даних складаємо кошторис витрат на розробку програмного забезпечення (таблиця 4.5).

Таблиця 4.5 - Кошторис витрат на розробку програмного забезпечення

Найменування витрат	Сума витрат, грн.
Витрати на оплату праці	2636,04
Відрахування у спеціальні державні фонди	430,02
Витрати на куповані вироби	6253,55
Накладні витрати	3954,06
Інші витрати	263,60
Витрати на використання комп'ютерної техніки, відлагодження і дослідну експлуатацію програмного продукту	744,8
Разом	13282,07

4.2 Розрахунок експлуатаційних витрат

Для оцінки економічної ефективності розроблюваного програмного продукту слід порівняти його з аналогом, тобто існуючим програмним забезпеченням ідентичного функціонального призначення.

Експлуатаційні одноразові витрати по програмному забезпеченню і аналогу включають вартість підготовки даних і вартість роботи комп'ютера (за час дії програми):

$$E_{\text{П}} = E_{1\text{П}} + E_{2\text{П}}, \tag{4.5}$$

де $E_{\text{п}}$ - одноразові експлуатаційні витрати на ПЗ, грн.;

$E_{1\text{п}}$ - вартість підготовки даних для експлуатації ПЗ, грн.;

$E_{2\text{п}}$ - вартість роботи комп'ютера для виконання проектного рішення, грн.

Річні експлуатаційні витрати $B_{\text{еп}}$ визначаються за формулою:

$$B_{E\text{П}} = E_{\text{П}} * N_{\text{П}}, \tag{4.6}$$

де $N_{\text{п}}$ - періодичність експлуатації ПЗ, раз/рік.

Вартість підготовки даних для роботи на комп'ютері визначається за формулою:

$$E_{1\text{П}} = \sum_{l=1}^{n} n_i t_i c_i, \tag{4.7}$$

де i - категорії працівників, які приймають участь у підготовці даних (i=1,2,...n);

n_i - кількість працівників і-ої категорії, осіб.;

t_i - трудомісткість роботи співробітників *i*-ої категорії по підготовці даних, год.;

c_i - середнього годинна ставка працівника *i*-ої категорії з врахуванням додаткової заробітної плати, що знаходиться із співвідношення:

$$c_i = \frac{c_i^0 (1+b)}{m}, \tag{4.8}$$

де c_i^0 - основна місячна заробітна плата працівника *i*-ої категорії, грн.;

b - коефіцієнт, який враховує додаткову заробітну плату (приймемо 0,57);

m - кількість робочих годин у місяці, год.

Трудомісткість підготовки даних для даного проектного рішення складає 3 год., відповідно для аналога - 3,5 год. Результати представлені у таблиці 4.6.

Таблиця 4.6 - Розрахунок витрат на підготовку даних та реалізацію проектного рішення на комп'ютері

Час роботи співробітників, год.	Середньогодинна заробітна плата, грн./год.	Витрати , грн.
Проектне рішення		
3	10,7	32,1
Аналог		
3,5	10,7	37,45

Витрати на експлуатацію комп'ютера визначається за формулою:

$$E_{2П} = t * S_{МГ}, \tag{4.9}$$

де *t* - витрати машинного часу для реалізації проектного рішення, год.;

$S_{МГ}$ - вартість однієї години роботи комп'ютера, грн./год.

Отже, $E_{2п}$=3·5,32=15,96 грн.; $E_{2а}$=3,5·5,32 =18,62 грн.;

$E_п$=32,1+15,96=48,06 грн.; $E_а$=37,45+18,62 = 56,07 грн.;

$B_{еп}$= 48,06*252=12111,12 грн.; $B_{еа}$=56,07*252=14129,64 грн.

4.3 Розрахунок ціни споживання проектного рішення

Ціна споживання - це витрати на придбання і експлуатацію проектного рішення за весь строк служби:

$$Ц_{С(П)} = Ц_П + B_{(E)NPV}, \quad (4.10)$$

де $Ц_П = K(1+\frac{П_P}{100}) + K_0 + K_к$ - ціна придбання проектного рішення, грн.;

K - кошторисна вартість;

$П_р$ - прибуток;

$K_о$ - витрати на прив'язку та освоєння рішення на конкретному об'єкті, грн.;

Отже, $Ц_п$=13282,07·(1+0,3) = 17266,69 грн.

Вартість витрат на експлуатацію проектного рішення (за весь час його експлуатації), в грн. обчислюється так:

$$B_{епрv} = \sum_{t=0}^{T} \frac{B_{еП}}{(1+R)^t}, \quad (4.11)$$

де $B_{еп}$ - річні експлуатаційні витрати, грн.;

T - строк служби проектного рішення, років;

R - річна ставка проценту банку.

Отже, $B_{епрv} = \sum_{t=1}^{3} \frac{12111,12}{(1+0,23)^t} = 2436,296$ грн.; $B_{еа} = \sum_{t=1}^{3} \frac{14129,64}{(1+0,23)^t} = 28429,09$ грн.

Тоді ціна споживання проектного рішення: $Ц_{сп}$ = 17266,69 +24362,96= =41629,65 грн.

Аналогічним чином визначається ціна споживання для аналогу: $Ц_{са}$ = =13470,7+28429,09=41899,79 грн.

4.4 Визначення показників економічної ефективності

Економічний ефект в сфері проектування: $E_{ПР} = Ц_{А} - Ц_{ПР}$ = 41899,79 - - 41629,65 =270,14 грн.

Річний економічний ефект в сфері експлуатації: $E_{КС} = B_{ЕА} - B_{ЕП}$ = =28429,09 – 24362,96 = 4066,13грн.

Сумарний ефект: $E = E_{тр} + \Delta E_{екс}$ =270,14 + 4066,13 = 4336,27 грн.

Результати усіх здійснених розрахунків представлені в таблиці 4.7.

Таблиця 4.7 - Показники економічної ефективності проектного рішення

Найменування	Одиниці вимірювання	Значення показників	
		Базовий варіант	Новий варіант
Капітальні вкладення	*грн.*	-	13282,07
Ціна придбання	*грн.*	13470,7	17266,69
Річні експлуатаційні витрати	*грн.*	28429,09	24362,96
Ціна споживання	*грн.*	41899,79	41629,65
Економічний ефект в сфері проектування	*грн.*	-	270,14
Економічний ефект в сфері експлуатації	*грн.*	-	4066,13
Сумарний ефект	*грн.*	4336,27	

Отже, у даному розділі проведений розрахунок витрат на розробку даного програмного забезпечення, експлуатаційних витрат, ціни на споживання проектного рішення, обчислені показники економічної ефективності нового програмного продукту. Згідно проведеного економічного обґрунтування дане

проектне рішення є конкурентноздатним. Отримано економічний ефект у розмірі 4336,27 грн. і тому розробка і впровадження цього проектного рішення є економічно доцільними.

ВИСНОВКИ

1. На основі аналізу існуючих рішень, щодо віртуалізації даних виявлено ряд недоліків, усунення яких визначає актуальність даної роботи.

2. На основі аналізу технічного завдання та недоліків існуючих систем сформовано вимоги до програмного забезпечення та функцій, які ним передбачені.

3. На основі сформованих вимог обґрунтовано вибір середовища програмування та основних інструментів створення програмного продукту.

4. На основі аналізу методів роботи з дисками та шифрування даних в системі Windows створено ряд процедур та функцій, що забезпечують створення захищеного віртуального диску .

5. На основі аналізу досліджених в роботі WinAPI функцій створено та проведено тестування повноцінного додатку для створення зашифрованих віртуальних дисків на базі операційної системи Windows.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Иванов, Д. В. Виртуализация общества. [Текст] / Д.В. Иванов. - М.: Петербургское Востоковедение, 2002. - 224 с.
2. Ленгоун, Д. Віртуалізація настільних комп`ютерів за допомогою VMware View 5: моногр.[Текст] / Д. Ленгоун. - М.: ДМК Пресс, 2013. - 268 с.
3. Віртуальні машины [Електронний ресурс] // Информаційний сайт про високі технології. Режим доступу: http://all-ht.ru/inf/vpc/p_0_0.html
4. Виртуальные машины на платформе Microsoft Virtual PC 2007 [Електронний ресурс] // WindowsFAQ.ru: FAQ, статьи, обзоры программ, операционных систем и серверного программного обеспечения. Режим доступу: http://www.windowsfaq.ru/content/view/566/46/
5. Ежова, Е. Н. Виртуализация как средство деформации и трансформации пространства и времени в медиа-рекламной картине мира [Текст] / Е.Н. Ежова. - Москва: РГГУ, 2010. - 303 с..
6. Диттнер, Р. Виртуализация и Microsoft Virtual Server 2005 [Текст] / Р. Диттнер , К. Мейджорз , М. тен Селдан, Т.Гротениус,Д. Рул мол., Дж. Грин. - М.: Бином-Пресс, 2008. - 432 с..
7. Мельниченко А. Обучение вместо программирования. Электронные компоненты и системы [Текст] / А. Мельниченко. – 2004. – №12. – С.36-40.
8. Введение в виртуализацию. Часть 1 [Електроний ресурс] //. Режим доступу: http://interface31.ru/tech_it/2012/07/vvedenie-v -virtualizaciyuchast -1.html
9. Как работают виртуальные машины – принцип работы [Електронний ресурс] // Режим доступу: http://winsetting.ru/kak-rabotayut-virtualnye-mashinyprincip-raboty.html
10. Баричев С. Г.Основы современной криптографии[Текст]/ С. Г. Баричев, В. В. Гончаров, Р. Е. Серов. — М.: ДИАЛОГ-МИФИ, 2011. — 175 с.
11. Панасенко С.П. Алгоритмы шифрования. Специальный справочник[Текст]/ С.П. Панасенко – СПб.: БХВ-Петербург, 2009 – 576 с

12. Столлингс, В. Криптография и защита сетей: принципы и практика[Текст]/ В. Столлингс — Вильямс, 2001 - 698 с.
13. Advanced Encryption Standart [Електронний ресурс]// Режим доступу: https://uk.wikipedia.org/wiki/Advanced_Encryption_Standard
14. Баричев С. Г. Основы современной криптографии — 3-е изд. Стандарт AES. Алгоритм Rijdael [Текст]/ С. Г. Баричев, В. В. Гончаров, Р.Е. Серов — М.: Диалог-МИФИ, 2011. — С. 30–35. — 176 с.
15. Rives, Shamir and Adelman [Електронний ресурс]// Режим доступу: https://ru.wikipedia.org/wiki/RSA
16. Daemon Tools – найкращі програми для роботи з образами дисків [Електронний ресурс] // Режим доступу: https://www.daemon-tools.cc
17. Віртуальний диск. Яку програму найкраще обрати [Електронний ресурс] // Режим доступу: http://pcpro100.info/virtualnyiy-disk
18. Методичні рекомендації до виконання дипломного проекту з освітньо-кваліфікаційного рівня “Бакалавр” напряму підготовки 6.050102 «Комп’ютерна інженерія» фахового спрямування «Комп’ютерні системи та мережі» / О.М. Березький, Л.О.Дубчак, Р.Б. Трембач, Г.М. Мельник, Ю.М. Батько, С.В. Івасьєв / Під ред. О.М. Березького. - Тернопіль: ТНЕУ, 2016.–65 с.
19. Паздрій І.Р. Методичні вказівки до написання техніко-економічного розділу для дипломних проектів на здобуття освітньо-кваліфікаційного рівня “Бакалавр” напряму підготовки 6.050102 «Комп’ютерна інженерія» / І.Р.Паздрій. - Тернопіль: ТНЕУ, 2015.– 36 с.
20. Климова, Л. М. Delphi 7. Основы программирования. Решение типовых задач. Самоучитель [Текст] / Л.М. Климова – КУДИЦ-Образ, 2017. - 480 с.
21. Культин Н. Основы программирования в Delphi 7 [Текст] /Н. Культин. - М.: СПб: БХВ, 2011. - 608 с.
22. Санников, Е. В. Курс практического программирования в Delphi. Объектно-ориентированное программирование [Текст] / Е.В. Санников. - М.: Солон-Пресс, 2013. - 188 с.

23. Осипов, Д. Delphi. Профессиональное программирование [Текст] / Д.Осипов. - М.: Символ-плюс, 2013. - 820 с.
24. Осипов, Д. Delphi. Программирование для Windows, OS X, iOS и Android [Текст] / Д. Осипов. - М.: "БХВ-Петербург", 2014. - 464 с.
25. Ревич, Ю. Нестандартные приемы программирования на Delphi[Текст] / Ю. Ревич. - М.: БХВ-Петербург, 2016. - 560 с.
26. Санников, Е. В. Курс практического программирования в Delphi. Объектно-ориентированное программирование / Е.В. Санников. - М.: Солон-Пресс, 2013. - 188 с.
27. Фленов, М.Е. DirectX и Delphi. Искусство программирования [Текст] / М.Е. Фленов. - М.: БХВ-Петербург, 2010. - 482 с.
28. Раскин Дж. Интерфейс: новые направления в проектировании компьютерных систем [Текст] /Дж.Раскин — Символ-плюс, 2005.
29. Мандел Т. Дизайн интерфейсов [Текст] /Т.Мандел — ДМК Пресс, 2005.
30. Тидвелл Дж. Разработка пользовательских интерфейсов [Текст]/ Дж.Тидвелл — Питер, 2007.