Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____Олексій СМІРНОВ
“____” _____2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

за першим (бакалаврським) рівнем вищої освіти

на тему

“Програмне забезпечення системи кібербезпеки клієнтського доступу до бази аналітичних даних грунтів ”

Виконав здобувач вищої освіти
IV курсу, групи _____
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____Ісаченков Е.В.
«____» _____2025р.

Керівник проекту
кандидат технічних наук, доцент
_____Доренський О.П.
«____» _____2025р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Рівень вищої освіти *бакалавр*
Галузь знань ._12 *“Інформаційні технології”*
Спеціальність *125 “Кібербезпека”*
Освітньо-професійна (освітньо-наукова) програма _*“ Кібербезпека ”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____Олексій СМІРНОВ
“____” _____2025 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Ісаченкова Едуарда Віталійовича
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки клієнтського доступу до бази аналітичних даних ґрунтів*
2. Керівник роботи *Доренський Олександр Павлович, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від “*1 7*” *01* 20*2 5* року № *57-02*
3. Строк подання роботи до захисту *22.05.2025 р.*
4. Мета та завдання випускної кваліфікаційної роботи *Метою роботи є розробка програмного забезпечення системи кібербезпеки клієнтського доступу до бази аналітичних даних ґрунтів.*
5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Реалізація роботи.
5. Впровадження системи в експлуатацію
6. Висновки.
6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*3 аркуша*

7. Дата видачі завдання «17» 01 2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем керування	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.05.2025 р.	
8.	Попередній захист роботи	22.05.2025 р.	

Дата видачі завдання
«__»_____20 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
«__»_____20 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Ісаченков Е. В. Програмне забезпечення системи кібербезпеки клієнтського доступу до бази аналітичних даних ґрунтів. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній кваліфікаційній бакалаврській розроблено програмне забезпечення системи, яке призначення для збільшення рівня кібербезпеки клієнтського доступу до бази аналітичних даних ґрунтів.

Метою розробки є ПЗ системи кібербезпеки клієнтського доступу до бази аналітичних даних ґрунтів.

В процесі роботи було ознайомлено з існуючими системи, технологіями, архітектурами та програмними рішеннями. Після аналізу обрано використовувати технологію Dropbox API та шифрування/дешифрування файлів користувача при відправленні/отриманні на хмарне середовище, симетричним шифром Kalyna стандарту 7624:2014 з використанням в режимі CBC.

Система використовує захищений протокол HTTPS для передачі даних через мережу та токен доступу для аутентифікації клієнта. Для реалізації ПЗ використано мови програмування: C++, C, Python.

Для реалізації інтерфейсу та самого клієнтського додатку обрано застосунок QT. Для зберігання даних обрано формат JSON, а при шифруванні формат BIN.

Результат роботи – створено програмне забезпечення системи кібербезпеки клієнтського доступу до бази аналітичних даних ґрунтів.

Ключові слова: ПЗ, система, кібербезпека, API, шифрування, дешифрування, Kalyna, CBC.

ABSTRACT

Isachenkov E.V. Software of the cybersecurity system of client access to the database of soil analytical data. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this bachelor's thesis, the system software is developed to increase the level of cybersecurity of client access to the soil analytical database.

The purpose of the development is the software of the cybersecurity system for client access to the soil analytical database.

In the course of the work, got acquainted with existing systems, technologies, architectures and software solutions. After the analysis, it was chosen to use Dropbox API technology and encrypt/decrypt user files when sending/receiving them to the cloud environment, using the Kalyna symmetric cipher of the 7624:2014 standard in CBC mode.

The system uses the secure HTTPS protocol for data transmission over the network and an access token for client authentication. The following programming languages were used to implement the software: C++, C, and Python.

The QT application was chosen to implement the interface and the client application itself. The JSON format was chosen for data storage and the BIN format for encryption.

The result of the work is the software of the cybersecurity system for client access to the soil analytical database.

Keywords: software, system, cybersecurity, API, encryption, decryption, Kalyna, CBC.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

IT - Інформаційні технології

IoT - Інтернет речей

TLS - Transport Layer Security

SSL - Secure Sockets Layer

AWS - Amazon Web Services

Amazon ECC - Amazon Elastic Compute Cloud

Amazon ECS - Amazon Elastic Container Service

Amazon EKS - Amazon Elastic Kubernetes Service

Amazon S3 - Amazon Simple Storage Service

Amazon RDS - Amazon Relational Database Service

Amazon VPC - Amazon Virtual Private Cloud

AWS IAM - Amazon Identity and Access Management

HTTPS - HyperText Transfer Protocol Secure

IAM - Google cloud Identity and Access Management

AES 128/256 - Advanced Encryption Standard

GKE - Google Kubernetes Engine

API - Application Programming Interface

DevOps - Development & operations

SDK - Software Development Kit

VPN - Virtual private network

CBC - Cipher Block Chain

XOR - eXclusive OR

BIN - Binary files

Вектор IV - Initialization vector

ВСТУП

Інформаційні технології (ІТ) стрімко розвиваються і мають величезний вплив на всі сфери життя. Починаючи з етапу комп'ютеризації і закінчуючи новітніми інноваційними рішеннями, такими як штучний інтелект, блокчейн і Інтернет речей (IoT), ІТ дозволяють автоматизувати процеси, оптимізувати управлінські рішення, підвищити продуктивність і знизити витрати. Успіхи у створенні нових програмних продуктів, розвитку хмарних технологій та підвищенні кібербезпеки стали основою для досягнення більш високого рівня комунікації та обробки даних[24].

Цифровізація - важливий етап у розвитку сучасного бізнесу, що включає інтеграцію цифрових технологій в усі аспекти діяльності організації. Це дозволяє автоматизувати операції, знизити вплив людського фактора і поліпшити доступ до інформації. Цифрова трансформація бізнес-процесів виходить за рамки простого впровадження технологій, дозволяючи компаніям адаптуватися до швидко мінливого ринку, поглиблюючи організаційну структуру і корпоративну культуру, зокрема, використання аналізу великих даних, штучного інтелекту, хмарних рішень дозволяє значно підвищити ефективність процесу, поліпшити взаємодію з клієнтів і партнерів, а також створювати нові бізнес-моделі[25].

Одним із ключових аспектів розвитку цифровізації є автоматизація процесів збору та обробки даних. Це стосується не лише обробки великих даних (Big Data), а й забезпечення надійної інфраструктури для зберігання та доступу до них у реальному часі. В умовах глобалізації та інтеграції компаній у міжнародні платформи значною мірою підвищується необхідність інтеграції хмарних технологій, що дозволяє створити гнучкі, масштабовані й доступні рішення для аналізу та обміну даними.

Необхідність аналізу ґрунтів в сільськогосподарській діяльності є важливим аспектом збільшення продуктивності господарства, що включає

збільшення прибутків та розвитку в цілому. Детальний збір процесів та стану ґрунту може бути корисний для подальших дій з обраною ділянкою, для збільшення позитивних якостей ґрунту в майбутньому[1].

Зважаючи на актуальність цієї проблеми, застосування розроблених програмних продуктів у сфері агрономії та сільського господарства дозволяє значно підвищити продуктивність та оптимізувати ресурси. Розробка програмного забезпечення, яке дозволяє інтегрувати технології цифровізації та автоматизації з практичними аспектами агрономії, сприяє більш ефективному використанню природних ресурсів і забезпеченню сталого розвитку сільського господарства в умовах змінного клімату та глобалізації.

Розробка такого програмного застосунку є ключовим фактором для здійснення аналізу ґрунту. Якщо вся інформація про стан ґрунту буде знаходитися в одному конкретному місці, це економить час та збільшить легкість доступу до даної інформації. Але потрібно пам’ятати конфіденційність та належний захист даної інформації. Компрометація таких даних може нанести критичні наслідки для підприємства, тому потрібно реалізувати належний доступ до таких даних. Мета цієї роботи це реалізувати систему кібербезпеки клієнтського доступу до бази аналітичних даних ґрунтів.

Напрацьовані результати даної бакалаврської роботи були представлені на здобуття освітнього ступеня бакалавра апробовані під час науково-технічної міжнародної конференції «Комп’ютерне моделювання у наукоємних технологіях» (м. Харків, 27-29 листопада 2024 р.)[26] та на Міжнародній науково-практичній конференції «Інформаційні технології в культурі, мистецтві, освіті, науці, економіці та бізнесі» (Київ, 24-25 квітня 2025 р.).

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

Розроблена система кібербезпеки спрямована на вирішення проблеми захисту даних у хмарних сховищах. Головна небезпека – неправомірний доступ і можливість компрометації аналітичних відомостей про ґрунти, що здатне спричинити викривлення інформації та фінансові збитки. Запропонована система забезпечує надійне шифрування та захист переданих і збережених даних, що дає змогу гарантувати конфіденційність і цілісність інформації у хмарному середовищі.

1.1 Призначення системи

Система кібербезпеки клієнтського доступу до бази аналітичних даних ґрунтів присвячена проблемі забезпечення кібербезпеки та захисту інформації хмарної бази даних аналітичної інформації про ґрунти. Ця база даних є складовою інформаційної системи бонітування ґрунтів, що реалізується в рамках ІТ-проекту. Вона покликана мінімізувати ризики несанкціонованого доступу та забезпечити цілісність і конфіденційність даних на всіх етапах їх обробки та зберігання.

Бонітування ґрунтів - це система оцінки якості ґрунтів, яка проводиться для визначення продуктивної цінності ґрунту та його придатності для сільськогосподарського використання[1]. Метою бонітування ґрунтів є встановлення природної родючості ґрунту, що дозволяє класифікувати його за ступенем придатності для вирощування різних сільськогосподарських культур[1].

1.2 Область застосування

Система кібербезпеки доступу клієнтів до бази аналітичних даних ґрунтів розроблена спеціально для аграріїв, де захист інформації є визначальним чинником у прийнятті рішень. У теперішньому сільському господарстві аналіз ґрунтів - важливий етап планування посівів, внесення добрив і прогнозування врожайності. Зібрані аналітичні дані містять надзвичайно важливу інформацію про фізико-хімічні властивості ґрунту, рівень його родючості та можливі ризики виснаження.

Застосування хмарних технологій для збереження та обробки цих даних дає фермерам, агрономам та агрохімічним компаніям швидкий та зручний доступ до інформації з будь-якого місця. Однак зростаюча кількість кіберзагроз ставить під сумнів конфіденційність та цілісність таких даних. Запропонована система дозволяє реалізувати надійний механізм захисту, який запобігає несанкціонованому доступу та забезпечує безпечне шифрування інформації при її передачі та зберіганні.

Система може бути корисною для державних організацій, що займаються моніторингом земельних ресурсів, наукових установ, які досліджують зміни у стані ґрунтів, а також великих агрохолдингів, що впроваджують цифрові технології у виробничі процеси. Інтеграція таких рішень сприяє підвищенню ефективності сільськогосподарських робіт, мінімізації екологічних ризиків та оптимізації використання ресурсів.

Використання хмарних технологій та шифрування Kalyna[23] також може бути корисним для інших проектів, де необхідно захистити конфіденційні дані.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

У світі, який стрімко розвивається, рівень кібербезпеки повинен мати відповідний рівень захищеності. Існує безліч систем, технологій, архітектур, які дозволяють мати підключення до мережі інтернет, але потрібно не забувати важливість захисту підключення та конфіденційність інформації. Особлива увага до частини збереження інформації на хмарних середовищах. Наведені існуючі технології реалізації системи кібербезпеки клієнтського доступу до бази даних та дієві аналоги хмарних платформ для збереження даних онлайн, будуть відображати загальний рівень потрібності в кібербезпеці системи.

2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень

Архітектура кібербезпечної інформаційної системи може мати різні форми, але головним критерієм є підтримка багаторівневого захисту даних:

Архітектура клієнт-сервер — використовує протокол TLS/SSL для захисту каналу зв'язку та забезпечення безпечної передачі інформації між клієнтом і сервером[2]. На стороні клієнта можуть використовуватися додаткові засоби шифрування для зберігання файлів перед відправкою на сервер[2].

Мікросервісна архітектура — забезпечує захист окремих сервісів або мікросервісів, де кожен сервіс може мати індивідуальні параметри кібербезпеки[3]. Наприклад, кожен мікросервіс може використовувати власні ключі шифрування та авторизації для підвищення загальної безпеки системи[3].

Хмарні платформи є одними з найважливіших технологічних інновацій останніх років, що мають значний вплив на ефективність і гнучкість ведення бізнесу та управління інформацією. Мета хмарних платформ полягає в наданні доступу до обчислювальних ресурсів, таких як сервери, сховища, бази даних,

програмне забезпечення та інші IT-ресурси, через Інтернет. Це дозволяє компаніям та організаціям знизити витрати на інфраструктуру, забезпечити масштабованість та зручний доступ до даних із будь-якої точки світу.

Amazon Web Services (AWS)

Хмарна платформа Amazon, що пропонує широкий спектр послуг для зберігання, обробки та аналізу даних, а також створення та розгортання додатків. AWS є одним з найпопулярніших у світі хмарних провайдерів та бізнесу. Він пропонує потужні та гнучкі інструменти для стартапів і великих підприємств[4].

AWS пропонує понад 200 сервісів, які можна використовувати для вирішення широкого спектру завдань. Основними послугами є наступні:

1. Обчислювальні сервіси:

Amazon ECC (Elastic Compute Cloud) — надає віртуальні сервери (інстанси), які можна швидко розгорнути та налаштувати відповідно до ваших потреб; EC2 дозволяє обирати різні типи інстансів, виходячи з ваших потреб у процесорній потужності, оперативній пам'яті, графічних ресурсах тощо. AWS Lambda надає специфічний і простий спосіб розгортання та налаштування віртуального сервера (екземпляра).

AWS Lambda — це безсерверний обчислювальний сервіс, який виконує код у відповідь на певні події. Це ідеальний варіант для функцій, які не працюють постійно, оскільки знижує витрати на інфраструктуру[11].

Amazon ECS та EKS — сервіси для управління контейнерами (Docker), як оркестровкою контейнерів на основі ECS (Elastic Container Service), так і Kubernetes (EKS - Elastic Kubernetes Service) підтримуються[12].

2. Сховище та бази даних:

Amazon S3 (Simple Storage Service) — сервіс зберігання об'єктів, що пропонує високу надійність, масштабованість і простоту використання; S3

підтримує API-інтерфейс для шифрування даних, налаштування політик доступу та зручну інтеграцію з іншими сервісами [5].

Amazon RDS (Relational Database Service) — надає можливість створювати та керувати реляційними базами даних, такими як MySQL, PostgreSQL та Oracle [13].

Amazon DynamoDB - NoSQL— сервіс для зберігання документів і ключових значень, що пропонує високу продуктивність і автоматичне масштабування [14].

3. Мережеві сервіси та безпека:

Amazon VPC (Virtual Private Cloud) — дозволяє створювати віртуальні приватні мережі з можливістю налаштування безпечних з'єднань та управління доступом до ресурсів [15].

AWS IAM (Identity and Access Management) — система управління доступом користувачів і груп до різних ресурсів AWS [16].

AWS CloudFront — це мережа доставки контенту (CDN), яка може прискорити доставку даних користувачам за допомогою серверів по всьому світу [17].

4. Аналітика та обробка даних:

Amazon Redshift — хмарне сховище даних, оптимізоване для швидкої аналітики та запитів великих даних [18].

AWS Glue — сервіс інтеграції даних, який може автоматично виконувати ETL-процеси (завантаження, перетворення та вивантаження даних) для обробки та аналізу даних [19].

Amazon Athena — аналітичний сервіс, який дозволяє виконувати SQL-запити безпосередньо до даних, що зберігаються в S3 [6].

5. Інструменти для штучного інтелекту та машинного навчання:

6. Amazon SageMaker — це платформа для навчання та розгортання моделей машинного навчання. Вона спрощує створення, навчання та інтеграцію моделей у додатки [20].

AWS Rekognition — сервіс для розпізнавання зображень і відео, який може виявляти об'єкти, людей і текст [21].

7. Функції безпеки AWS

AWS пропонує ряд інструментів безпеки, зокрема:

Шифрування даних — AWS підтримує шифрування як транзитних даних (через HTTPS/TLS), так і даних у сховищі (S3, RDS та інші рівні зберігання).

Контроль доступу — AWS IAM дозволяє користувачам створювати детальні політики доступу для забезпечення захисту від несанкціонованого доступу.

Моніторинг та аудит — такі сервіси, як Amazon CloudTrail та AWS CloudWatch, дозволяють користувачам відстежувати активність користувачів і стан системи в режимі реального часу та вести журнали для аудиту [4].

8. Переваги AWS

AWS пропонує ряд переваг, серед яких:

Масштабованість — ресурси можна швидко масштабувати відповідно до ваших потреб.

Гнучкість — широкий спектр послуг для задоволення різних потреб бізнесу.

Надійність — висока доступність даних завдяки послугам резервного копіювання та реплікації.

Глобальна інфраструктура — доступ до ресурсів у різних регіонах і зонах доступності, що підвищує швидкість і надійність послуг. AWS продовжує розвиватися, впроваджуючи нові технології та покращуючи безпеку, що робить його ефективним рішенням для організацій, яким потрібна масштабована та безпечна хмарна інфраструктура [5].

Google Cloud Storage

Масштабований хмарний сервіс, який забезпечує високу надійність, доступність і безпеку для зберігання великих обсягів даних. Він підходить для зберігання всіх типів даних (структурованих і неструктурованих) і може бути легко інтегрований з іншими сервісами Google Cloud для обробки та аналізу [7].

1. Масштабованість і гнучкість:

Хмарне сховище Google забезпечує масштабованість від невеликих обсягів даних до ексабайт даних. Інфраструктура дозволяє масштабувати його за потреби, що особливо важливо для великих підприємств та проектів з інтенсивним використанням даних.

Різні класи сховищ дозволяють оптимізувати витрати на зберігання залежно від частоти доступу до даних [7].

2. Типи класів сховищ

Стандартне сховище - для часто використовуваних даних. Призначене для зберігання даних, що вимагають високої швидкості доступу та низьких затримок. Рекомендується для активних даних, до яких часто звертаються.

Сховище Nearline — для рідко використовуваних даних (наприклад, доступ до даних здійснюється лише раз на місяць). Ідеально підходить для резервного копіювання та архівування.

Сховище холодної лінії — для даних, до яких рідко звертаються (рідше одного разу на рік). Економічно вигідний варіант для довгострокового зберігання архівних даних.

Архівне зберігання — для дуже рідко використовуваних даних (довгострокове зберігання). Ідеальне рішення для архівів, які потрібно зберігати десятиліттями [7].

3. Висока надійність і доступність:

Реплікація та геореплікація — Google Cloud Storage автоматично реплікує дані між кількома регіонами або зонами, забезпечуючи доступність у разі виходу з ладу окремого дата-центру.

Управління життєвим циклом — користувачі можуть автоматично переміщати дані між класами сховища на основі політик життєвого циклу (на основі віку даних і частоти використання), знижуючи витрати [7].

4. Безпека:

Шифрування даних — всі дані шифруються під час передачі та зберігання; Google Cloud Storage автоматично шифрує дані за допомогою AES-256 або AES-128. Користувачі також можуть використовувати власні ключі шифрування.

Контроль доступу — Cloud Identity and Access Management (IAM) використовується для встановлення детальних прав доступу. Google Cloud також підтримує політики на рівні об'єктів, що дозволяє точно контролювати доступ до даних [22].

5. Інтеграція з іншими сервісами Google Cloud:

Дані зберігаються у форматах, сумісних з іншими сервісами, такими як BigQuery (аналіз великих масивів даних), Dataflow (потокова обробка даних) і Cloud Pub/Sub (управління потоком подій).

Google Kubernetes Engine (GKE) та інші інструменти DevOps у Google Cloud можна легко інтегрувати з Хмарним сховищем для побудови безперервних конвеєрів доставки додатків та обробки даних [8].

6. Простота використання та доступ до API:

Google Cloud Storage пропонує простий користувацький інтерфейс за допомогою Google Cloud Console і потужний API для інтеграції сховища в зовнішні додатки. Крім того, підтримуються SDK для різних мов програмування (наприклад, Python, Java, Node.js).

7. Типові випадки використання Google Cloud Storage:

Резервне копіювання та аварійне відновлення - створення резервних копій даних, які можна легко відновити в екстрених випадках.

Архівування даних — зберігайте дані, які рідко використовуються протягом тривалого періоду часу, щоб отримати до них доступ за потреби.

Хостинг статичного контенту - завантаження та керування веб-сайтами, що містять статичний контент (наприклад, зображення, відео) для швидкої доставки користувачам.

Озеро даних — зберігання великих наборів даних для обробки та аналізу іншими хмарними сервісами, такими як BigQuery та інструменти Google Cloud AI.[7]

8. Переваги та можливості Google Cloud Storage:

Швидкість і продуктивність — глобальна інфраструктура Google забезпечує швидкий доступ до даних навіть для віддалених користувачів. Низька вартість архівування - доступні найдешевші класи зберігання для довгострокового зберігання, що робить їх привабливими для зберігання великих архівів.

Гнучка цінова політика — підтримує різні варіанти ціноутворення залежно від частоти доступу, що дозволяє вибрати найбільш економічно вигідне рішення.

Google Cloud Storage — універсальне та надійне рішення для організацій, яким потрібна гнучка хмарна інфраструктура для зберігання великих обсягів даних, інтеграції з іншими сервісами обробки даних та інструментами безпеки [7].

Технологія Dropbox API

Інтерфейс прикладного програмування, який дозволяє інтегрувати функціонал хмарного сховища Dropbox у сторонні додатки. За допомогою цього інтерфейсу ви можете створювати облікові записи, керувати папками та налаштовувати параметри облікових записів користувачів.

1. Доступ до файлів і папок

Завантаження та вивантаження файлів — API дозволяє завантажувати файли в Dropbox і отримувати файли з нього.

Перегляд, створення, видалення та переміщення папок - API дає вам повний контроль над структурою каталогів.

Отримуйте метадані - такі як розмір файлів і папок, час останньої зміни, тип файлу тощо [9].

2. Синхронізація та керування даними

Синхронізація — автоматичне оновлення даних між Dropbox і вашою локальною прикладною системою.

Версії файлів — відстежувати зміни у файлах, отримайте доступ до попередніх версій і відновлюйте їх за потреби.

3. Контроль доступу та безпека

Токени доступу — API Dropbox забезпечує безпечний доступ до вашого облікового запису Dropbox за допомогою токенів доступу. Кожен токен надає доступ до певного облікового запису або папки і додає рівень захисту.

Функціонал обмеженого доступу — API дозволяє обмежити доступ до певних файлів і папок, встановивши права доступу на рівні програми [9].

4. Інтеграція з іншими системами

Вебхуки — API Dropbox підтримує веб-хуки для отримання повідомлень про зміни в облікових записах користувачів. Це дозволяє додаткам реагувати на зміни у файлах і папках в режимі реального часу.

Масштабована інтеграція — API дозволяє інтегрувати Dropbox у великі інформаційні системи для підтримки ефективного управління даними в масштабі.

5. Клієнтський SDK:

API Dropbox підтримує SDK для різних мов програмування, включаючи Python, Java, JavaScript і Swift, щоб спростити розробку інтегрованих з Dropbox додатків.

SDK надають готові бібліотеки для роботи з API, зменшуючи кількість коду, який потрібно написати для реалізації взаємодії з Dropbox.

Типові випадки використання Dropbox API:

1. Резервне копіювання даних — додатки можуть автоматично створювати резервні копії важливих файлів у Dropbox, щоб забезпечити доступність і захист даних.

2. Спільна робота — додатки, яким потрібен спільний доступ до файлів і папок, можуть використовувати API Dropbox для легкої синхронізації та обміну даними між користувачами.

3. Медіаконтент — додатки, які зберігають та надсилають медіа (зображення, відео), можуть використовувати API для зберігання та легкого доступу до великої кількості файлів.

4. Аналітичні програми — доступ до журналів змін файлів і метаданих дозволяє обробляти і аналізувати файли безпосередньо в Dropbox.

Технічні аспекти використання Dropbox API

1. RESTful API — API Dropbox використовує REST для обміну даними, що робить його сумісним з будь-яким клієнтським додатком, який працює через HTTP.

2. OAuth 2.0 — для аутентифікації використовується стандартний протокол OAuth 2.0, що забезпечує безпеку при аутентифікації користувачів і надає безпечний доступ до даних.

3. Підтримка великих файлів — API дозволяє розбивати великі файли на частини і завантажувати їх окремо.

Технологія Dropbox API, має безліч переваг, такі як:

1. Легка інтеграція з додатками завдяки зрозумілим методам та різноманітним SDK.

2. Безпека завдяки шифруванню та токенам доступу.

3. Гнучка маніпуляція файлами - маніпулювати файлами безпосередньо з хмарного середовища.

Водночас, ця технологія, має свої обмеження:

1. Обмеження на розмір облікового запису - API Dropbox має обмеження на обсяг доступного сховища, залежно від тарифного плану користувача.

2. Залежність від інтернет-з'єднання - доступ до даних і синхронізація вимагають стабільного інтернет-з'єднання. Dropbox API - це потужний інструмент для додатків, які потребують хмарного сховища, доступу до файлів у реальному часі та контролю доступу. Рішення підходить для проектів від простих додатків до великих корпоративних рішень, які потребують надійного та безпечного зберігання даних.

Дослідження методів захисту у хмарних системах

Хоча хмарні обчислення відкривають значні можливості для користувачів, вони також створюють нові виклики для інформаційної безпеки. Оскільки використання хмарних середовищ у комерційних та особистих цілях зростає, інформаційна безпека стає все більш важливим питанням. З одного боку, хмарні провайдери пропонують різні рівні захисту, з іншого боку, користувачі повинні розробляти власні політики безпеки та контролювати доступ до своїх даних.

Основними ризиками в хмарних обчисленнях є:

1. Несанкціонований доступ - можливість несанкціонованого доступу до даних.
2. Втрата даних - ризик видалення або пошкодження інформації через зловмисні дії або технічні збої.
3. Загрози мережевій безпеці - ризик атак, що ставлять під загрозу конфіденційність або цілісність даних.
4. Відповідність - труднощі з дотриманням стандартів і правил.
5. Цілісність даних - ризик фальсифікації даних під час транспортування або зберігання.
6. Слабка безпека - недостатній захист від шкідливих програм.
7. Доступність послуг - ризик переривання доступності послуг через технічні або зловмисні дії.

Для захисту даних у хмарі використовуються такі основні методи:

1. Шифрування - для забезпечення конфіденційності даних на всіх етапах роботи.

2. Аутентифікація та авторизація - обмеження доступу до ресурсів лише авторизованим користувачам.

3. Мережеві технології - VPN та брандмауери захищають мережу.

4. Моніторинг загроз - регулярний моніторинг та аналіз діяльності забезпечує швидке реагування на загрози.

5. Резервне копіювання - забезпечує безпеку даних у разі їх втрати.

6. Відповідність стандартам безпеки - відповідність міжнародним стандартам безпеки підвищує рівень захисту.

7. Фізична безпека - захищає центри обробки даних від фізичного доступу та стихійних лих.

Хоча ці методи і технології можуть допомогти зменшити ризики в хмарному середовищі, важливо пам'ятати, що відповідальність за безпеку даних лежить як на постачальниках, так і на користувачах [10].

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для розробки програмного забезпечення для системи кібербезпеки доступу клієнтів до бази даних аналізу ґрунту були обрані спеціальні програмні засоби та мови програмування. Для зберігання та обробки даних було використано хмарний сервіс Dropbox, інтегрований через Dropbox API. Такий вибір пояснюється наступними факторами:

1. Безпечна передача даних - API Dropbox підтримує захищений протокол HTTPS з протоколом шифрування TLS, що гарантує високий ступінь безпеки під час передачі даних.

2. Простота інтеграції - Dropbox надає простий у використанні API з розширеними функціями управління файлами і папками, які можна швидко інтегрувати в сторонні додатки.

3. Наявність токенів доступу - аутентифікація за допомогою токенів доступу забезпечує безпечний і контрольований доступ до хмарного сховища, а також може бути реалізована аутентифікація клієнтів.

Для реалізації системи було обрано мови Python, C++ та C, які найкраще відповідають вимогам проекту:

1. Висока продуктивність - мови C++ та C забезпечують ефективне управління ресурсами та оптимальну продуктивність.

2. Підтримка криптографічних алгоритмів - ці мови дозволяють гнучко реалізовувати шифрування, наприклад, блоковий шифр «Калина», який вимагає низькорівневих бітових операцій.

3. Об'єктно-орієнтована архітектура - C++ з її об'єктно-орієнтованим підходом спрощує управління складними структурами даних і компонентами, забезпечує чистоту коду і підтримує масштабованість системи.

4. Легка інтеграція з API - Python має потужну бібліотеку для обробки мережевих запитів та взаємодії з API, включаючи Dropbox API. Це дозволяє легко реалізовувати функції для завантаження та отримання даних з хмари та спрощує синхронізацію файлів.

5. Швидкість розробки - завдяки своєму синтаксису Python дозволяє швидко створювати і тестувати інтегровані рішення, що особливо важливо для прискорення розробки проектів, пов'язаних з кібербезпекою та захистом інформації.

6. Системна сумісність - Python легко інтегрується з модулями, написаними на C та C++. Це дозволяє поєднувати продуктивність і криптографічні можливості низькорівневих мов з гнучкістю високорівневих функцій, які пропонує Python.

Для захисту даних клієнтських додатків використовується симетричний блоковий шифр Kalyna в режимі CBC (Cipher Block Chain). Причини вибору:

1. Висока криптографічна стійкість - шифр Kalyna забезпечує високий рівень захисту.

2. Гнучка конфігурація - режим CBC обирається на основі унікальності зашифрованих блоків, що мінімізує ризик повторення шаблонів у даних.

3. Національні стандарти - шифри Kalyna відповідають українським стандартам безпеки.

Для створення клієнтського додатку на C++ було обрано Qt через наступні переваги:

1. Крос-платформна сумісність - Qt дозволяє запускати додатки на різних платформах, що спрощує подальшу підтримку.

2. Підтримка інтеграції API - Qt надає інструменти для роботи з API Dropbox, що значно спрощує реалізацію функціоналу управління файлами.

3. Розширені можливості проектування інтерфейсів - Qt дозволяє створювати зручні інтерфейси, які є інтуїтивно зрозумілими та легкими для навігації.

Формат JSON є найкращим форматом для зберігання даних аналізу і пропонує наступні переваги:

1. Легка серіалізація - JSON можна легко обробити у C++ для швидкого зберігання та пошуку структурованих даних.

2. Розширюваність - JSON зручний для зберігання розширених даних.

Використання Python, C++ та C з Qt, інтеграція з Dropbox через API та використання шифрування Kalyna гарантують високу продуктивність, легку обробку великих обсягів даних та надійний захист даних.

2.3 Розгорнута постановка завдання

Отримавши всі деталі реалізації цієї системи, потрібно виконати розгорнуту постановку завдання. Розробка системи передбачає створення програмного забезпечення на мовах C, C++ та Python із реалізацією алгоритму симетричного шифрування Kalyna в режимі CBC (захищає від повторюваних шаблонів) для надійного захисту даних. Важливим етапом є розробка механізму генерації ключів шифрування та векторів ініціалізації, їх збереження у двійковому форматі, а також реалізація функцій шифрування та дешифрування JSON файлів для забезпечення конфіденційності інформації.

Система інтегрується з API Dropbox, що включає налаштування передачі зашифрованих файлів до хмарного середовища, аутентифікацію клієнта за допомогою Access Token для безпечного та контрольованого доступу, а також автоматичне оновлення та синхронізацію даних між локальним клієнтом і хмарним середовищем.

Для забезпечення зручності роботи передбачено розробку графічного інтерфейсу за допомогою Qt, що надає користувачам можливість переглядати, шифрувати та завантажувати дані у хмару. Інтерфейс також відображатиме стан шифрування, передачі та зберігання інформації.

На завершальному етапі проводиться тестування та налагодження системи. Перевіряється відповідність алгоритмів шифрування вимогам безпеки, коректність процесів шифрування та розшифрування даних. Також здійснюється тестування інтеграції з API Dropbox для забезпечення стабільної передачі та синхронізації файлів. Важливим аспектом є оптимізація продуктивності та усунення виявлених дефектів.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

Система кібербезпеки для доступу клієнтів до бази даних аналізу ґрунту повинна забезпечувати безпечне зберігання, передачу та обробку даних за допомогою шифрування, аутентифікації користувачів та інтеграції з хмарним сервісом Dropbox. З використанням сучасних та надійних мов програмування, таких як: C++, C, Python. А також для легким розуміння інтерфейсом користувача, наприклад, за допомогою інструментарію Qt.

3.1 Опис функціонування системи

Під час запуску система повинна зчитувати конфігураційні дані, необхідні для підключення до Dropbox, та ініціалізує параметри, включаючи зчитування ключа шифрування та вектор ініціалізації.

Користувачі можуть завантажувати локальні файли або створювати нові файли з даними аналізу ґрунту, збереженими у форматі JSON для стандартизації структури.

Перед передачею даних у хмару система шифрує дані за допомогою блочного шифру Kalyna в режимі CBC. Кожен блок об'єднується з попереднім блоком за допомогою операції XOR, що забезпечує додатковий рівень захисту.

Після роботи з даними, користувач автоматично автентифікується в Dropbox за допомогою токена доступу, що забезпечує доступ до хмарного середовища та контрольований доступ до файлового сховища. Після автентифікації система підключається до облікового запису Dropbox і забезпечує встановлення з'єднання для завантаження файлів до Dropbox.

Зашифровані дані передаються до Dropbox у вигляді зашифрованого BIN-файлу для подальшого зберігання через Dropbox API, використовуючи захищений

протокол HTTPS, включаючи TLS. Під час передачі система контролює оновлення файлів на сервері, щоб забезпечити стабільність з'єднання та уникнути втрати даних.

Користувачі можуть отримувати файли з Dropbox для перегляду та подальшої обробки даних. Система виконує зворотне перетворення. Файл розшифровується і застосовується алгоритм Kalyna, який видаляє біти, додані для заповнення останнього блоку під час шифрування. Після розшифрування дані відображаються в інтерфейсі користувача, де їх можна переглянути або оновити.

Система забезпечує безпечну передачу та зберігання даних аналізу ґрунту з використанням шифрування даних у хмарному середовищі Dropbox. Реалізовано всі функції, від автентифікації користувача до шифрування, дешифрування та передачі даних, що забезпечує високий рівень конфіденційності та доступності інформації для автентифікованих користувачів.

3.2 Розробка структурної схеми

Структурна схема є важливою складовою частиною проектування будь-якої інформаційної системи або програмного продукту, що дозволяє зобразити основні компоненти системи, їх взаємозв'язки та потоки даних. Вона служить для організації і наочності структури проекту, допомагаючи зрозуміти основні етапи його функціонування і взаємодії між окремими елементами. Структурна схема є ефективним інструментом для системного аналізу, проектування, а також для подальшої реалізації та управління IT-проектами. Структурна схема забезпечення кібербезпеки клієнтського сервісу бази аналітичних даних ґрунтів представлена на рисунку 3.1.

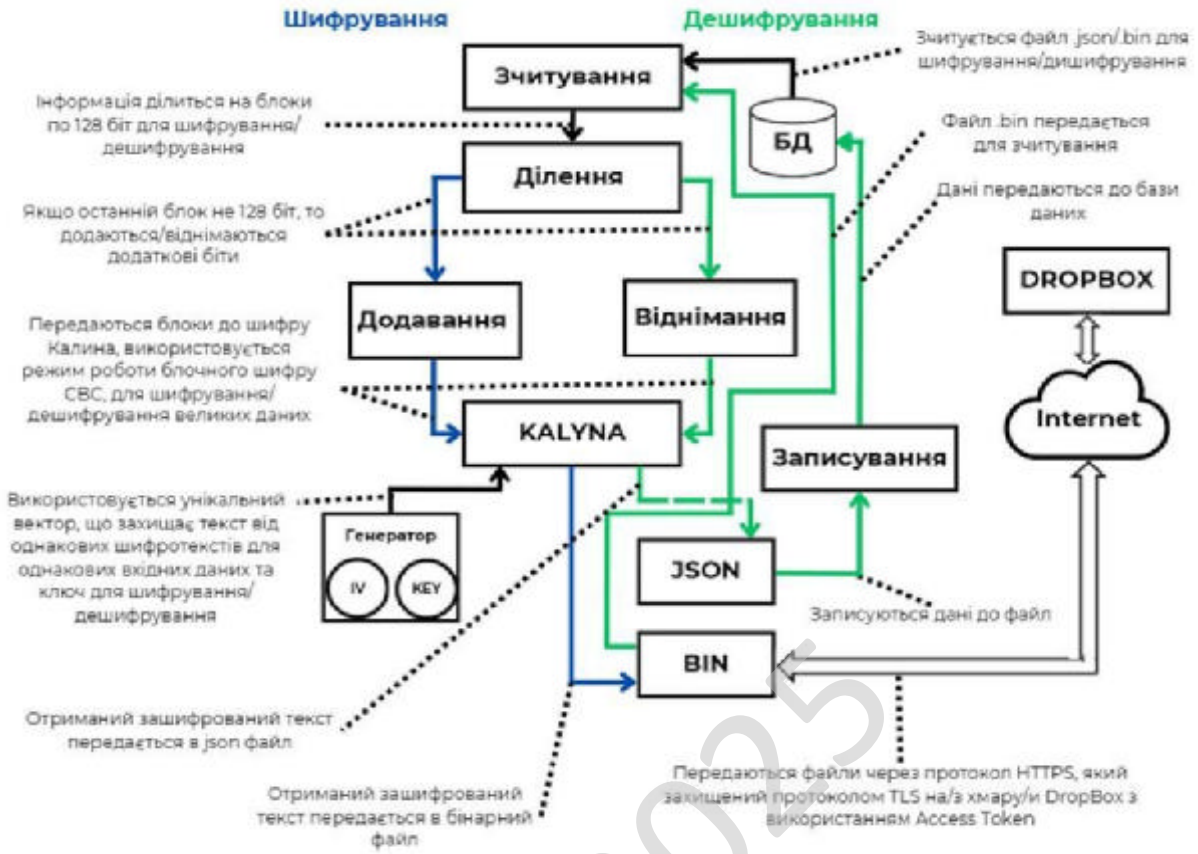

Рисунок 3.1 - Структурна схема забезпечення кібербезпеки клієнтського застосунку бази аналітичних даних ґрунтів

До її складу входять функціональні елементи бази даних, файлів для шифрування і дешифрування, генератора ключів та процесів алгоритму “Калина”: запису й зчитування блоків даних, ділення, віднімання, додавання біт інформації та процесора безпосередньо симетричного блокового перетворення.

При відправці файлів на хмару Dropbox, інформація шифрується. Відбувається зчитування інформації, яку бажаємо шифрувати. Ділення її на блоки по 128 біт, бо шифр «Калина» це блочний шифр. Додавання до останнього блоку додаткових бітів для рівності останнього блоку з іншими. Шифрування блоків інформації, в якому використовується ключ та вектор, які були створені генератором. Записування інформації в бінарний файл та відправлення до хмарного середовища Dropbox використовуючи протокол HTTPS.

При завантаженні файлів з хмари Dropbox, інформація дешифрується. Відбувається завантаження файлів з хмарного середовища Dropbox. Потім інформація зчитується та ділиться на блоки по 128 біт. Відбувається віднімання доставлених бітів. Дані дешифруються з використанням ключа та вектор, та записуються у клієнтський файл, який передається до бази даних. Для подальшого використання клієнтом.

Завдяки інтегруванню функціональних складових, як-от генератора ключів, модулів шифрування та дешифрування, система формує міцний механізм захисту інформації при взаємодії з хмарним сервісом Dropbox. Процедури розбиття даних на блоки, додавання та віднімання бітів, а також застосування ключа та вектора, згенерованих окремим компонентом, гарантують цілісність та конфіденційність даних. Пересилання зашифрованих даних через протокол HTTPS додатково укріплює безпеку транспортування. В результаті, впроваджена система забезпечує ефективний захист даних при зберіганні та обміні через хмарні сервіси, лишаючись зручною та надійною для подальшої роботи клієнта.

3.3 Розробка функціональної схеми

Функціональна схема є важливим елементом проектування інформаційних систем та програмних продуктів, що дозволяє детально описати функції, які повинна виконувати система, а також взаємодію між цими функціями. Вона є графічним відображенням процесів, які відбуваються в межах системи, а також відображає потоки даних між різними її частинами. Функціональна схема слугує для візуалізації та аналізу того, як кожна частина системи виконує свою роль і як це впливає на загальний результат.

Функціональна схема забезпечення кібербезпеки клієнтського сервісу бази аналітичних даних ґрунтів представлена на рисунку 3.2.

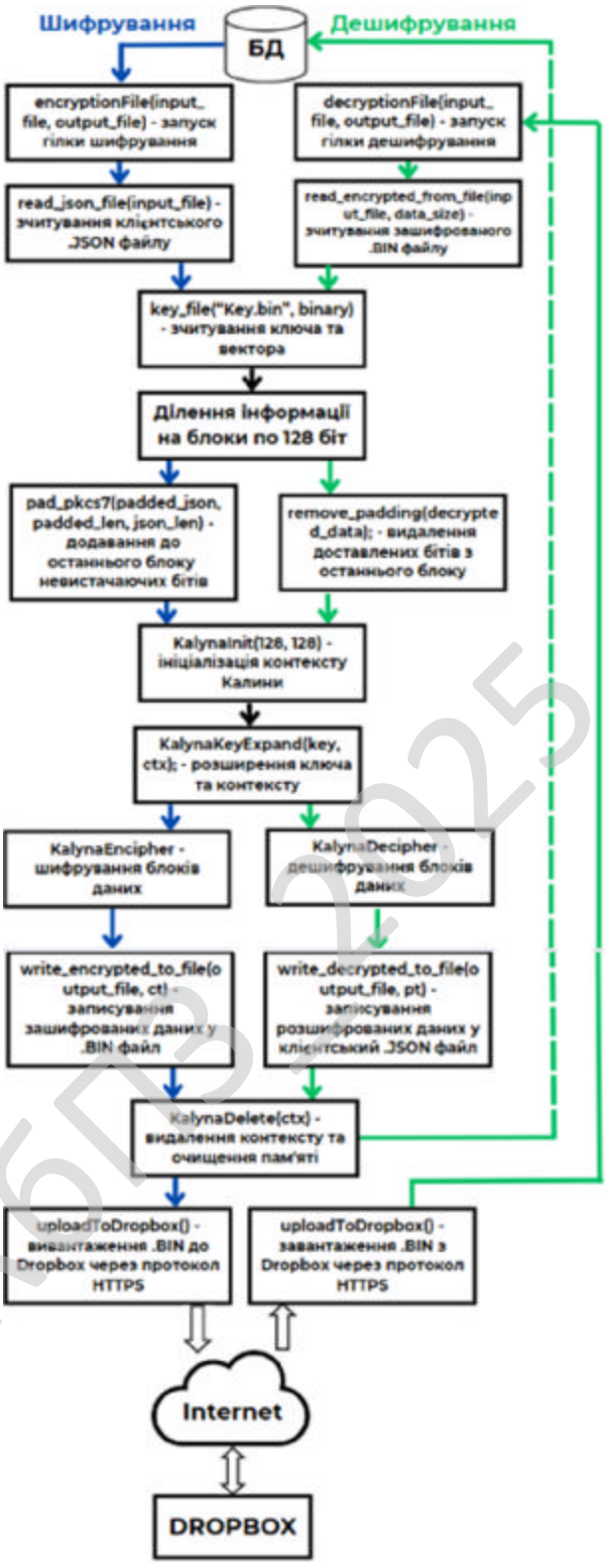

Рисунок 3.2 - Функціональна схема забезпечення кібербезпеки клієнтського застосунку бази аналітичних даних ґрунтів

Функціональна схема, відображає наступну ідею. Коли користувач обирає відправку даних на Dropbox, вмикається функція шифрування «encryptionFile(input_ file, output_file)», вхідні значення це розташування клієнтського файлу та розташування бінарного файлу зашифрований текст. У ній відбувається зчитування .json файлу клієнта функцією «read_json_file(input_file)», вхідне значення функції це розташування клієнтського файлу.

Потім відбувається зчитування файлу ключа функцією «key_file("Key.bin", binary)», вхідне значення це назва файлу та метод зчитування. Даний ключ генерується разом з вектором перед цим процесом користувачем, або ключ та вектор залишається за замовчуванням.

Після відбувається ділення зчитуваної інформації на блоки по 128 біт, тому що шифр Калина – це блочний шифр, який шифрує дані у однакових по розміру блоках. Якщо останній блок менше 128 біт(зазвичай), то вмикається функція «pad_pkcs7(padded_json, padded_len, json_len)», дана функція відповідає за рівномірність останнього блоку. Додає до блоку невистачаючі біти, дописує в кінець нулі.

Відбувається ініціалізація контексту «KalynaInit(128, 128)», це додаткові умови для шифру, в нашому випадку це розмір вхідного блоку та розмір вхідного ключа. Також відбувається розширення ключа та передача контексту «KalynaKeyExpand(key, ctx)», вхідні значення це ключ та згенерований контекст.

Функція «KalynaEncipher» - це функцій самого шифрування. Використовуємо метод CBC, при якому над кожним блоком виконується XOR з попереднім зашифрованим блоком для уникнення шаблонів у вихідних даних. Перший блок XOR-иться з ініціалізаційним вектором (IV), який знаходиться у файлі з ключем.

Потім після завершення шифрування, вся зашифрована інформація записуються у бінарний файл функцією «write_encrypted_to_file(output_file, ct)», вхідні значення це місце розташування бінарного файлу та зашифрований текст.

Записування зашифрованих даних у бінарний файл забезпечує ще більший рівень безпеки файлів, адже людське око даний текст не зможе прочитати.

У кінці функції відбувається видалення згенерованого контексту та очищення пам’яті від виконуваних файлів функцією «KalynaDelete(ctx)», вхідне значення це контекст шифру Калина.

Коли відбулося шифрування файлів та записуваних їх у бінарні файли, тоді вмикається скрипт на Python «uploadToDropbox()». Дана функція зчитує файл «Token.txt», який знаходиться разом з застосунком. Вміст даного файлу це Access Token, який використовується для автентифікації з хмарним середовищем Dropbox. Відбувається автентифікація, якщо вона успішна, то файли передаються через протокол HTTPS, який захищений протоколом TLS на хмарне середовище Dropbox.

HTTPS (Hypertext Transfer Protocol Secure) – це протокол для безпечної комунікації через Інтернет. Він використовує шифрування для захисту даних, переданих між браузером користувача та веб-сервером. HTTPS забезпечує конфіденційність, цілісність і автентифікацію переданих даних [6].

Коли користувач обирає завантажити файли з Dropbox, то вмикається скрипт «uploadToDropbox()». Дана функція завантажує бінарні зашифровані дані для клієнта, використовуючи: автентифікація Access Token та протокол HTTPS, деталі якого було написано вище.

Після успішного завантаження бінарних файлів, вмикається функція «decryptionFile(input_ file, output_file)», функція відповідає за початок дешифрування файлів. Вхідні значення це розташування зашифрованого бінарного файлу та розташування клієнтського .json файлу.

Відбувається зчитування бінарного файлу функцією «read_encrypted_from_file(input_file, data_size)». Вхідні значення це розташування бінарного файлу та змінна, в яку буде додана кількість елементів у відповідному файлі.

Потім відбувається зчитування ключа та вектор, ділення інформації на блоки, ініціалізація контексту та розширення ключа, ці всі процеси ідентичні як для шифрування, вони всі були описані вище.

Після відбувається дешифрування інформації за допомогою шифру Калина, а саме функції «KalynaDecipher». Використовуємо знову блочний режим CBC для дешифрування великої кількості інформації.

Після успішного дешифрування, інформація записується у клієнтський файл .json функцією «write_decrypted_to_file(output_file, pt)». Вхідні значення відповідно це розташування клієнтського файлу та змінна в якій знаходиться дешифрований текст.

Далі відбувається видалення контексту та очищення пам’яті від виконуваних файлів, після чого відбувається видалення доставлених бітів з останнього блоку функцією «remove_padding(decrypted_data)». Вхідні значення це дешифрований текст. Дана функція є обов’язковою, тому що синтаксис .json не буде працювати, тому що в кінці файлу є невідповідні дані, які були доставлення задля створення блоків по 128 біт.

3.4 Розробка діаграми процесів

Діаграма процесів є важливим інструментом для візуалізації, аналізу та документування всіх процесів, що відбуваються в рамках системи або організації. Вона дозволяє наочно представити зв'язки між різними етапами роботи, а також зрозуміти, як кожен процес впливає на загальний результат. Зазвичай діаграма процесів використовується для детального опису алгоритмів, потоків даних та взаємодії між різними компонентами системи. Діаграма процесів забезпечення кібербезпеки клієнтського сервісу бази аналітичних даних ґрунтів представлена на рисунку 3.3.

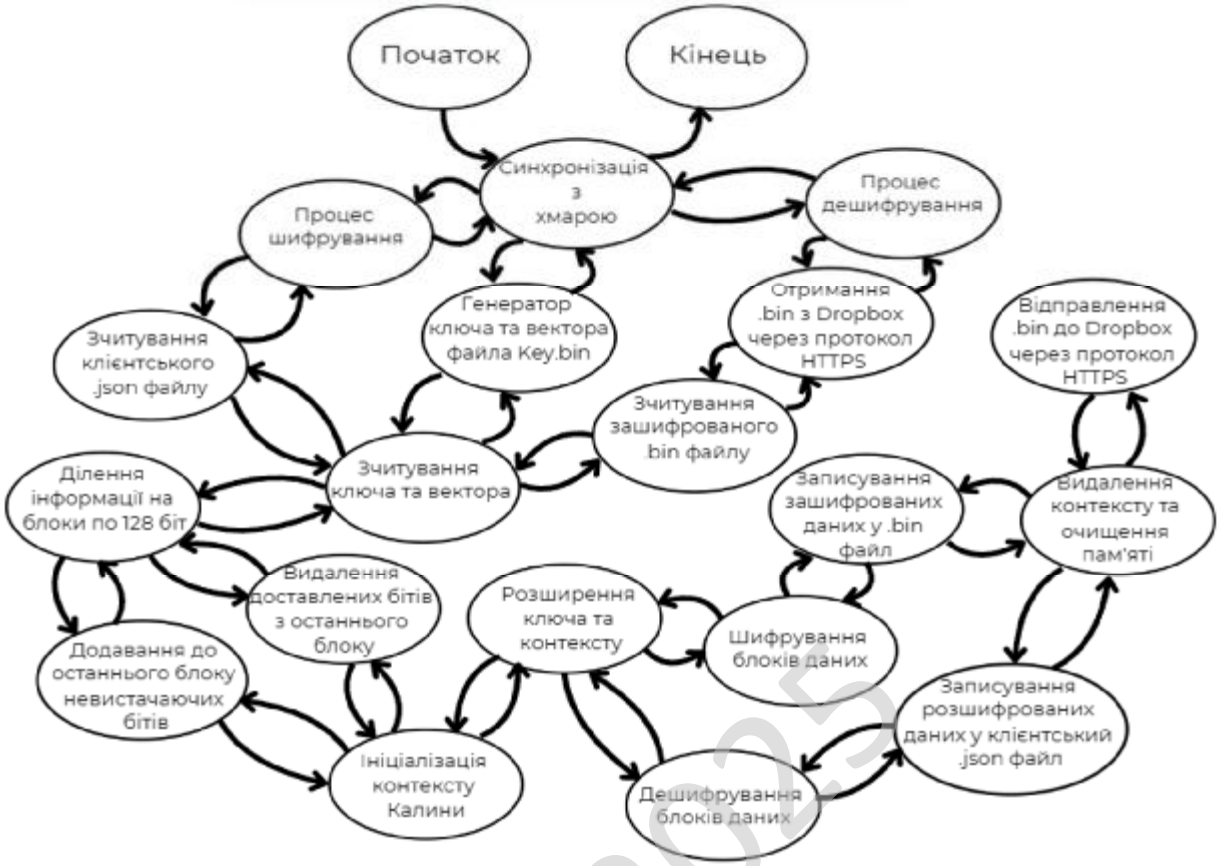

Рисунок 3.3 - Діаграма процесів забезпечення кібербезпеки клієнтського застосунку бази аналітичних даних ґрунтів

Після початку діаграми відбувається процес синхронізації з хмарою, є три варіанти подій.

Варіант один, це генератор ключа та вектор, результатом роботи є бінарний файл, який містить унікальний ключ та вектор. Даний процес потрібен для подальшого шифрування та дешифрування інформації. По замовчування в клієнті є файл ключа та вектор, дана функція його оновлює.

Варіант два, це процес шифрування, зчитується клієнтський .json файл. Для шифрування за допомогою шифру «Калина» потрібен унікальний ключ, який генерується в першому варіанті подій. Зчитується бінарний файл ключа та ділить інформацію на блоки по 128 біт, тому що «Калина» це блочний шифр, шифрування йде по блоках. Але блоки повинні всі мати розмір в 128 біт, тому до останнього блоку додається невистачаючі біти. Ініціалізується контекст(параметри),

розширюється ключ та додається контекст вже до шифру «Калина». Після чого відбувається шифрування блоків, та записування зашифрованого тексту у бінарний файл. Видаляється створений контекст та очищається пам’ять від виконуваних файлів, задля зменшення навантаження на систему. Зашифрований текст відправляється до Dropbox з використанням протоколу HTTPS, який включає в себе безпечну передачу даних.

Варіант три, це процес дешифрування. Система отримує шифрований бінарний файл з хмарного середовище Dropbox через мережевий протокол HTTPS. Зчитується зашифрований файл та ключ разом з вектором, який був згенерований в першому варіанті подій. Зашифрована інформація знову ділиться на блоки по 128 біт та відбувається зворотна дія, а саме віднімання доставлених бітів при процесі додавання в другій події. Відбувається ініціалізація контексту та розширення ключа з додавання цього контексту до шифру «Калина». Після чого відбувається дешифрування інформації, але при умові, що ключ введений той, який використовувався при шифрування даної інформації. Потім дешифрований текст записується в клієнтський додаток, тим самим оновлює його та передається клієнту.

Завдяки втіленню трьох складових - генеруванню унікального ключа та вектора ініціалізації, шифруванню та подальшому розшифровуванню – створюється цілісний захист даних у клієнтському оточенні. Згенерований ключ забезпечує неповторність кожної сесії шифрування, що зменшує ймовірність злому. Процес шифрування, з використанням блочного шифру «Калина» та розділенням даних на частини, гарантує міцність та опірність до атак. Безпечна передача зашифрованих файлів через HTTPS надає захищене транспортування до хмарного сховища Dropbox. Розшифровування, в свою чергу, дає можливість відновити дані тільки при наявності вірного ключа, що забезпечує секретність інформації. Отже, представлені етапи утворюють надійну схему захисту даних у хмарному середовищі, гарантуючи цілісність, конфіденційність та доступність інформації.

4 РЕАЛІЗАЦІЯ РОБОТИ

Для наочного представлення використаних функцій їх реалізації буде продемонстровано у вигляді блок-схем для детального відображення.

4.1 Розробка блок-схем та опис алгоритмів функціонування системи

Основна блок-схема зображена на рисунку 4.1.

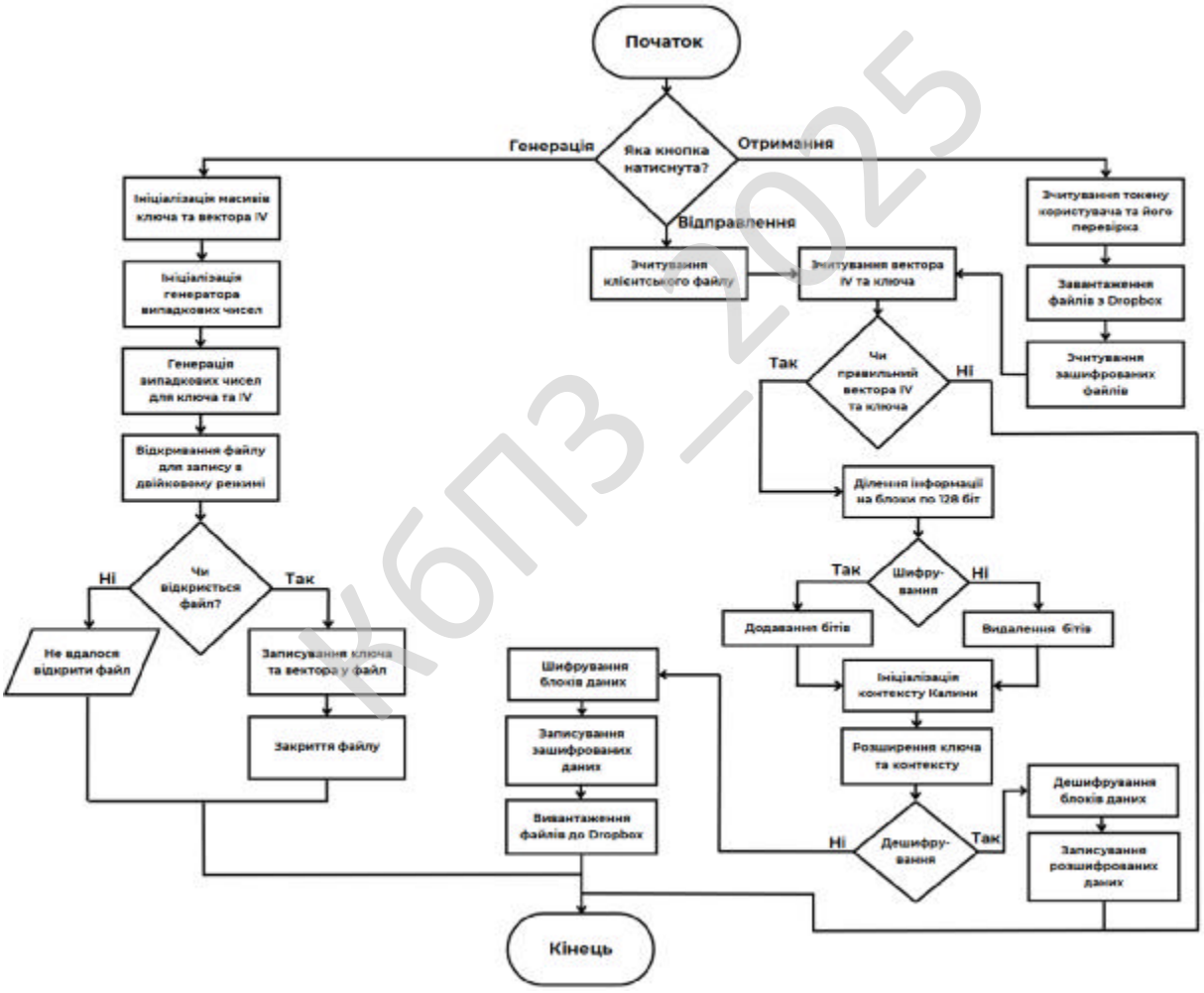

Рисунок 4.1 – Основна блок-схема програмного забезпечення системи

Після натискання кнопки з’являється три різні варіанти подій: генерація, отримання, відправлення.

Після натискання кнопки генерація відбуваються такі дії. Спочатку відбувається ініціалізація масивів для ключа та вектора. Потім відбувається ініціалізація генератора, який буде генерувати даний ключ та вектор. Відбувається генерація ключа та вектор генератором. Відкриття бінарного файлу, в який буде записуватися згенерований ключ та вектор. Відбувається перевірка, якщо файл не відкрився, то виводиться помилка та завершується алгоритм, якщо файл відкрився, то згенеровані дані записуються у бінарний файл, закривається відкритий файл та алгоритм успішно завершується.

Після натискання кнопки отримання розпочинається процес отримання файлів з хмарного середовища Dropbox. Зчитується токен доступу користувача та його перевірка. Далі завантаження зашифрованих файлів з Dropbox через протокол HTTPS та їхнє зчитування. Зчитування та перевірка правильності вектора та ключа, якщо негативна, то алгоритм завершується. Якщо перевірка пройшла успішно, тоді відбувається ділення даних на блоки по 128 біт. При дешифруванні здійснюється видалення не потрібних бітів. Виконується ініціалізація контексту Калина та відбувається розширення ключа та контексту. Здійснюється дешифрування блоків даних та записування розшифрованих даних у файл, після чого алгоритм успішно завершується.

Після натискання кнопки відправлення здійснюється процес відправлення файлів на хмару. Відбувається процес зчитування клієнтського файлу. Зчитування та перевірка файлу ключа з вектором для подальшого шифрування, якщо негативна, то алгоритм завершується. При позитивній перевірці здійснюється ділення блоків по 128 біт та при шифруванні додавання невистачаючих бітів. Здійснюється ініціалізація та розширення контексту та ключа. Потім шифрування блоків даних та записування їх до бінарного файлу. Відправлення бінарних зашифрованих даних до хмарного середовище Dropbox та успішне завершення алгоритму.

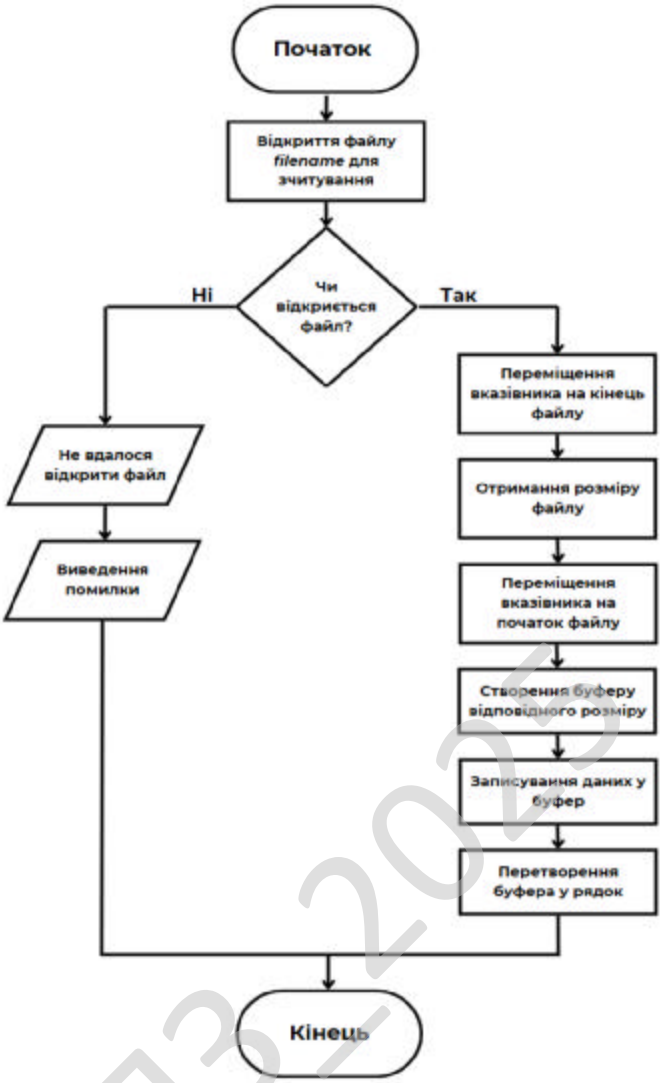

Рисунок 4.2 – Алгоритм зчитування клієнтського файлу для шифрування

Опис алгоритму зчитування клієнтського файлу для шифрування, це відображено на рисунку 4.2. Спочатку відбувається відкриття клієнтського файлу. Також відбувається перевірка, якщо файл не відкрився, то виводиться помилка та завершується алгоритм, якщо файл відкрився, то пересувається вказівник в кінець файлу, для того, щоб в наступному блоці дізнатися розмір файлу. Вказівник повертається на початок файлу та створюється буфер відповідного розміру. Потім дані записуються у буфер та перетворюються у рядок, після чого відбувається успішне завершення алгоритму.

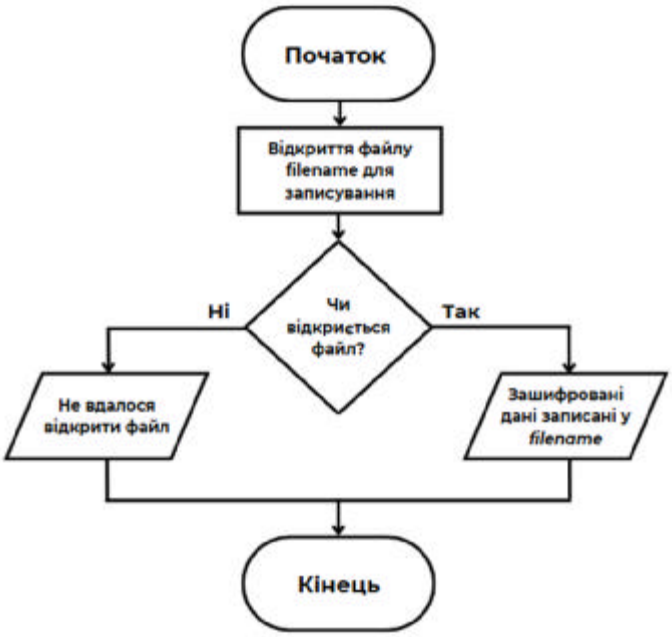

Рисунок 4.3 – Алгоритм записування зашифрованих даних у бінарний файл

Опис алгоритму записування зашифрованих даних у бінарний файл рисунок 4.3. Спочатку відбувається відкриття бінарного файлу, в який очікуємо записати готовий текст. Відбувається перевірка, якщо файл не відкрився, то виводиться помилка та завершується алгоритм, якщо файл відкрився, то зашифровані дані записуються у бінарний файл та алгоритм успішно завершується.

4.2 Захист розробленого програмного забезпечення

Основним методом захистом є шифрування, а саме блочним симетричним шифром “Калина” в режимі CBC для шифрування\дешифрування великих обсягів інформації. У режимі кожен блок відкритих даних (plain text) перед шифруванням комбінується з попереднім зашифрованим блоком (або з початковим вектором ініціалізації — IV), що забезпечує додаткову безпеку шифрування. Для використання даного алгоритму потрібно мати унікальний ключ, який має 128 біт випадкових даних та вектор, який ускладнює передбачення результату. Перед

відправкою файлів на хмарне середовище відбувається шифрування, а перед завантаженням файлів відбувається дешифрування, але ключ та вектор повинен співпадати з цими процесами. Такий метод запобігає уникнення доступу до даних, бо дані будуть зашифровані.

Для забезпечення безпеки доступу до системи використовується авторизація через Access Token, це унікальний ідентифікатор, який надається хмарним сервісом або системою для авторизації та доступу до певних ресурсів. Він використовується для аутентифікації клієнта, тобто перевірки того, що він має право доступу до конкретних даних чи функцій у системі. Access Token є частиною механізму безпеки, який дозволяє захистити дані, обмежуючи доступ до них лише авторизованим користувачам., який генерується самим хмарним середовищем Dropbox. Його ціль це швидка авторизації клієнта до хмари, унеможливлює доступ іншого користувача до хмари без необхідного токена. Передача даних відбувається через захищений протокол TLS, який шифрує канали передачі даних і забезпечує захищене з'єднання між клієнтом і сервером хмарного сервісу.

Використання цих методів значно підвищує рівень кібербезпеки, створюючи надійний захист даних від несанкціонованого доступу. Застосування шифрування в поєднанні з авторизацією через Access Token та захищеними каналами передачі через TLS забезпечує не лише захист інформації під час зберігання, але й при її передачі. Це дозволяє мінімізувати ризики компрометації даних, зберігаючи їх цілісність та конфіденційність у хмарному середовищі.

5 ВПРОВАДЖЕННЯ СИСТЕМИ В ЕКСПЛУАТАЦІЮ

Впровадження системи кібербезпеки для доступу клієнтів до баз даних аналізу ґрунту передбачає кілька важливих кроків для забезпечення безперешкодної інтеграції розробленого рішення в операційне середовище та гарантування належного рівня захисту даних при використанні системи в реальному середовищі.

Перед впровадженням системи було виконано низку підготовчих дій:

1. Проведено перевірку всіх функціональних модулів на тестовому середовищі з метою визначення можливих помилок та вразливостей.

2. Здійснено налаштування хмарного середовища Dropbox, включно з генерацією Access Token для забезпечення контрольованого доступу до бази даних і гарантування безпеки інформації.

3. Проведено підготовку конфігураційних файлів, що включають параметри шифрування, ключі та вектори ініціалізації, які зберігаються у захищених бінарних файлах.

Налаштування середовища:

1. Встановлення клієнтського програмного забезпечення на робочі станції та підключення до мережі підприємства для стабільного доступу до хмарного сервісу.

2. Налаштування Dropbox API для інтеграції з Python-скриптами, що відповідають за передачу даних у хмару, і забезпечення авторизації користувача для кожної сесії.

3. Оптимізація параметрів безпеки — конфігурація дозволів доступу, моніторинг активності клієнтського застосунку, а також налаштування журналювання дій для подальшого аналізу й контролю доступу до файлів.

На початковому етапі експлуатації система розгортається на обмеженій кількості робочих станцій для тестування в реальних умовах. На цьому етапі проводиться:

1. Навчання персоналу щодо використання клієнтського застосунку, який забезпечує зручність взаємодії з даними.

2. Інструктаж з безпечного використання доступу до хмарного середовища, включаючи захист Access Token і дотримання процедур аутентифікації.

3. Ознайомлення з алгоритмом шифрування даних симетричним шифром «Калина» та принципами роботи режиму CBC, що забезпечує захист файлів під час передачі до Dropbox.

Після початкового запуску система працює в умовах реальних даних та активного використання. Проводяться:

1. Функціональне тестування — перевірка всіх функцій системи, включаючи шифрування, передачу й дешифрування файлів, а також забезпечення безперебійної синхронізації з Dropbox.

2. Тестування безпеки — моделювання можливих атак (наприклад, спроби несанкціонованого доступу до файлів) з метою оцінки стійкості системи та рівня захищеності даних.

3. Аналіз продуктивності — оцінка швидкодії системи під навантаженням, щоб визначити оптимальні параметри роботи шифрування та передачі даних у хмарне середовище.

Після впровадження в повну експлуатацію система підлягає регулярному моніторингу для забезпечення стабільності роботи та своєчасного виявлення можливих проблем. До заходів супроводу входить:

1. Періодичне оновлення конфігураційних параметрів, шифрувальних ключів та інших безпекових атрибутів.

2. Підтримка та оновлення Python-скриптів, що інтегруються з Dropbox API, для забезпечення стабільності взаємодії з хмарним середовищем.

3. Здійснення регулярних аудитів безпеки, а також адаптація та налаштування системи відповідно до нових вимог безпеки та технологій.

Етапи впровадження системи дозволили налаштувати й оптимізувати роботу клієнтського застосунку з аналітичними даними ґрунтів в існуючу апаратну систему «IsacMaps» рисунок 5.1, забезпечуючи безпечну передачу й захист даних у хмарному середовищі.

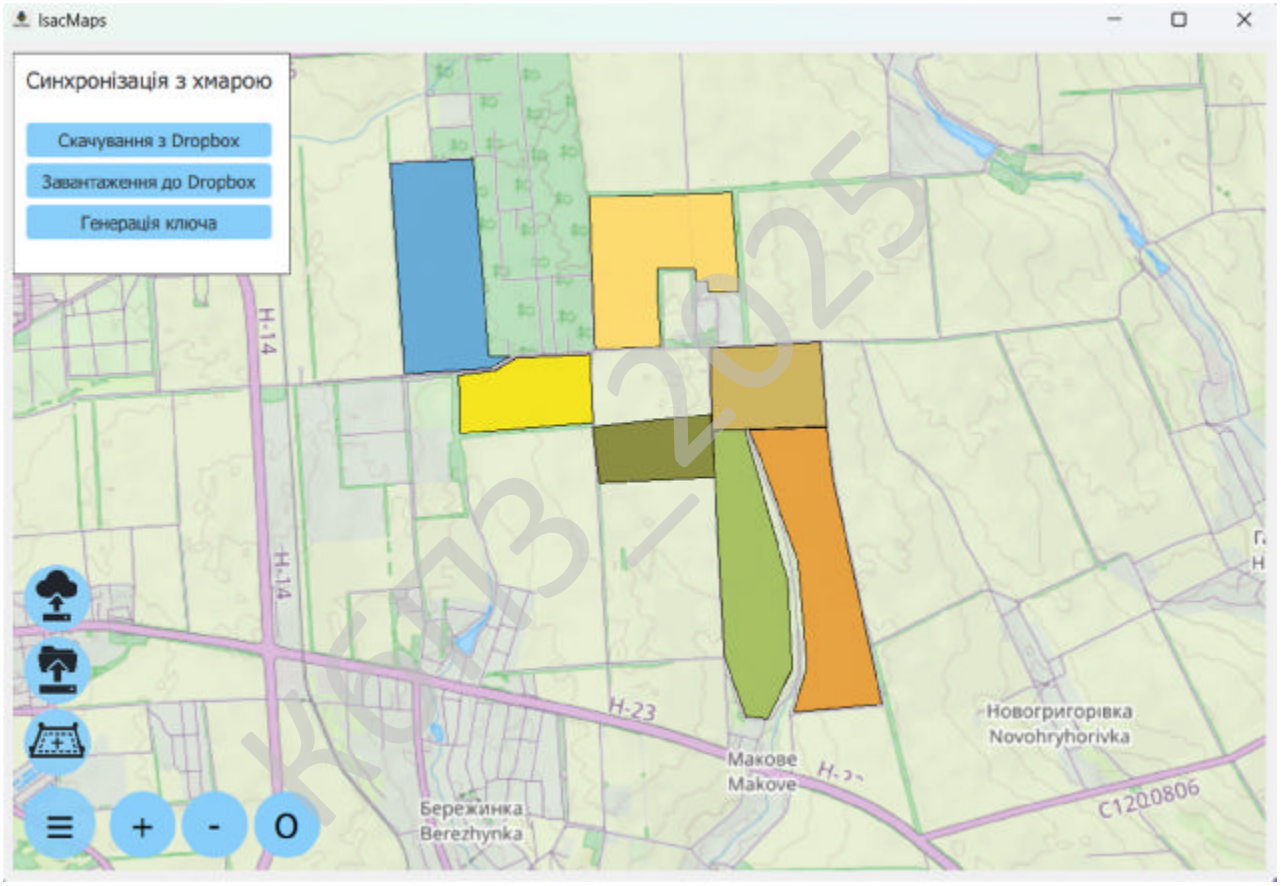

Рисунок 5.1 - Знімок екрану застосунку «IsacMaps», до якого впроваджено технологію

ОСНОВНІ ВИСНОВКИ

Метою роботи була реалізація системи захисту доступу клієнта до бази аналітичних даних ґрунтів. Оскільки інформаційні технології стрімко розвиваються у всіх сферах життя. Необхідність цифровізації аналізу ґрунтів в сільськогосподарській діяльності є ключовим фактором збільшення прибутків та розвитку підприємства. Створення програмного застосунку, який буде в собі зберігати детальний аналіз ґрунту, пришвидшить та полегшить в подальшому вибір діяльності на ґрунті.

Призначення такої системи захистить передані дані на хмарне середовище, чим самим дозволить легко дістатися до даних, але якщо буде наданий дозвіл. Система захисту доступу до бази аналітичних даних ґрунтів має важливе значення для фермерських господарств, оскільки вона дозволяє зберігати та обробляти величезний обсяг даних, які стосуються різних аспектів якості ґрунтів. Аналіз ґрунтів є ключовим елементом в аграрному секторі, оскільки він безпосередньо впливає на ефективність сільськогосподарських процесів. Система, що реалізована в рамках цього проєкту, дає змогу фермерам отримувати точні аналітичні дані про їхні ґрунти, визначати необхідність внесення добрив, а також оптимізувати використання ресурсів, що в свою чергу знижує витрати та збільшує врожайність. Завдяки такій системі фермери можуть не тільки оптимізувати свої виробничі процеси, а й покращити результати свого бізнесу, адже дані, що зберігаються в безпечному хмарному середовищі, завжди доступні для аналізу та прийняття оперативних рішень. У підсумку, фермерське господарство отримує потужний інструмент для підвищення своєї продуктивності, що важливо в умовах сучасного конкурентного ринку.

Існують безліч технологій, архітектур та хмарних ресурсів. Ключові архітектури можна виділити дві: клієнт-серверна та мікросервісна архітектура.

Великі компанії такі як: Google, Amazon та інші, створюють нові програмні рішення даної проблеми.

Наприклад, Amazon Web Services(AWS) це хмарна платформа, яка пропонує велику кількість сервіс близько 200, для роботи з даними на хмарі, їхній аналіз, обчислення, управління і багато іншого. Google Cloud Storage це ще одна велика хмарна платформа зі своїми сховищами: Nearline, сховищем холодної лінії та архівним зберіганням. Перевагами таких хмарних платформ є масштабованість, гнучкість та звісно велика надійність. Також існує хмарне середовище Dropbox API, який виражається своєї легкістю у використанні та простотою підключення до стороннього додатку. Є інтерфейс прикладного програмування, з якого можна створювати облікові записи, керувати теками та багато інших функції для розробника. Саме цей сервіс буде використаний в даній роботі, такий вибір пояснюється: безпечною передачею даних через протокол HTTPS, простотою інтеграції з використанням API, та наявністю токена доступу для аутентифікації клієнта, що збільшує рівень захисту даних. Для реалізації ПЗ використано мови програмування: C++, C, Python. Для збільшення захисту при передачі та можливого компрометації даних використовувати симетричний шифр Kalyna стандарту 7624:2014 з використанням в режимі CBC для шифрування/дешифрування великих даних. Для реалізації інтерфейсу та самого клієнтського додатку обрано застосунок QT. Для зберігання даних обрано формат JSON, а при шифруванні формат BIN.

Система працює таким чином, під час запуску система зчитує конфігураційні дані, необхідні для зв'язку з хмарою Dropbox та зчитування ключа шифрування/дешифрування з вектором ініціалізації. Після редагування чи додавання даних аналізу ґрунту, збереженими у форматі JSON дані шифруються з використанням блочного шифру Kalyna в режимі CBC. Клієнт автоматично аутентифікувати через Access Token та дані успішно передаються до хмари. Завантаження даних з хмари відбувається навпаки, спочатку завантажуються дані, потім вони дешифруються використовуючи той самий ключ та вектор при

шифруванні та записуються у клієнтський файл для подальшого використання. Детальніше функціонування системи відображено на структурних, функціональних схемах та діаграмі процесів. Також реалізовані алгоритми в блок-схемах, для детального відображення окремих функції ПЗ.

Перед впровадженням даної системи було проведено низку підготовчих дій, а саме: проведено перевірку всіх функціональних модулів, налаштування хмарного середовища Dropbox з генерацією Access Token та генерацію конфігураційних файлів таких як, бінарний файл ключа з вектором. Потім встановлення самого ПЗ на робочі станції, проведено інструктажі для персоналу, таких як: використання клієнтського застосунку, використанням доступу до хмари та ознайомленням з алгоритмом шифрування Kalyna. Також було проведено низку тестувань: функціональне тестування перевірка всіх функцій, тестування безпеки моделювання атак, аналіз продуктивності оцінка швидкодії.

Реалізації такого програмного застосунку дозволяє мінімізувати витрати на збереження аналізу даних ґрунтів. Збільшило продуктивність та легкість з використання даних аналізу. Використанні методи захисту передачі даних, зменшили ризик компрометації інформації та збільшили конфіденційність даних. Результати цієї кваліфікаційної роботи на здобуття освітнього ступеня бакалавра апробовані під час науково-технічної міжнародної конференції «Комп'ютерне моделювання у наукоємних технологіях» (м. Харків, 27-29 листопада 2024 р.) та на Міжнародній науково-практичній конференції «Інформаційні технології в культурі, мистецтві, освіті, науці, економіці та бізнесі» (Київ, 24-25 квітня 2025 р.)

СПИСОК ДЖЕРЕЛ

1. Superagronom. URL: https://superagronom.com/slovnik-agronoma/bonituvannya-gruntu-id20047 (дата звернення: 21.10.2024).
2. Training Qateslab. URL: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/ (дата звернення: 27.10.2024).
3. Globallogic. URL: https://www.globallogic.com/ua/insights/blogs/microservices-architecture-for-beginners-part-one/ (дата звернення: 28.10.2024).
4. Amazon S3: Introduce. URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/GetStartedWithS3.html (дата звернення: 30.10.2024).
5. Learning Amazon Web Services: A Hands On Guide to the Fundamentals of AWS Cloud. Addison Wesley, 2019. 400 с.
6. Amazon Athena: Introduce. URL: https://aws.amazon.com/athena/faqs/?nc1=h_ls (дата звернення: 01.11.2024).
7. Lakshmanan V. Data Science on the Google Cloud Platform. O'Reilly Media, Incorporated, 2022.
8. Google Cloud Kubernetes Engine: Introduce. URL: https://cloud.google.com/kubernetes-engine/docs (дата звернення: 04.11.2024).
9. Dropbox Fundamentals Course. URL: https://learn.dropbox.com/self-guided-learning/dropbox-fundamentals-course/how-to-use-dropbox (дата звернення: 05.11.2024).
10. Назаренко Д.М. Дослідження методів захисту у хмарних системах. Міжнародна науково-технічна конференція. Харків, 2023, С. 170-172. URL: https://ice.nure.ua/wp-content/uploads/2024/01/50_Nazarenko-D.M.-_3_Str.170-172.pdf (дата звернення: 05.11.2024).
11. Amazon Lambda: Resources. URL: https://aws.amazon.com/lambda/resources/?nc1=h_ls&aws-lambda-resources-blog.sort-

by=item.additionalFields.createdDate&aws-lambda-resources-blog.sort-order=desc (дата звернення: 06.11.2024).

12. Amazon ECS: Documentation. URL: https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html (дата звернення: 06.11.2024).

13. Amazon RDS: Documentation. URL: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html (дата звернення: 06.11.2024).

14. Amazon Dynamo Database: Resources. URL: https://aws.amazon.com/dynamodb/getting-started/?nc1=h_ls (дата звернення: 06.11.2024).

15. Amazon VPC: Features. URL: https://aws.amazon.com/vpc/features/?nc1=h_ls (дата звернення: 06.11.2024).

16. Amazon IAM: Documentation. URL: https://aws.amazon.com/iam/resources/?nc1=h_ls&iam-blogs.sort-by=item.additionalFields.createdDate&iam-blogs.sort-order=desc (дата звернення: 06.11.2024).

17. Amazon CloudFront: Documentation. URL: https://docs.aws.amazon.com/cloudfront/#lang/en_us (дата звернення: 06.11.2024).

18. Amazon Redshift: Introduction. URL: https://docs.aws.amazon.com/redshift/latest/dg/welcome.html (дата звернення: 06.11.2024).

19. Amazon Glue: Documentation. URL: https://docs.aws.amazon.com/glue/ (дата звернення: 06.11.2024).

20. Amazon SageMaker: Resources. URL: https://aws.amazon.com/sagemaker/getting-started/?nc1=h_ls (дата звернення: 06.11.2024).

21. Amazon Rekognition: Introduction. URL: https://aws.amazon.com/rekognition/resources/?nc1=h_ls (дата звернення: 06.11.2024).

22. Google Identity: Documentation. URL: https://cloud.google.com/architecture/identity/federating-gcp-with-active-directory-synchronizing-user-accounts (дата звернення: 06.11.2024).

23. ДСТУ 7624:2014. Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного блокового перетворення. [Введ. 01-07-2015]. Вид. офіц. Київ: Мінекономрозвитку України, 2016. 228 с.

24. Сучасний IT-ринок: тенденції, рівень зарплат, поради новачкам і досв ідченим розробникам, Немчинський С. URL: https://dou.ua/forums/topic/46109/ (да та звернення: 21.10.2024).

25. Що буде далі з цифровізацією України, Сабадишина Ю. URL: https://dou.ua/lenta/news/mykhailo-kornyeev-results/

26. Доренський О.П., Ісаченков Е.В. Структурно-функціональна модель забезпечення кібербезпеки клієнтського застосунку бази аналітичних даних ґрунтів. Комп'ютерне моделювання у наукоємних технологіях (КМНТ-2024): зб. наук. праць міжнар. наук.-техн. конф. (Харків, 27-29 лист. 2024 р.). Х.: ХНУ імені В. Н. Каразіна. С. 78–79.

27. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. Springer, Cham. 2021, pp 66-84.

28. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

29. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I.,Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 166-171.

30. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

31. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and

Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

32. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

33. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43.

34. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660.

35. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

36. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

37. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.701-706.

38. Смірнов О.А. Козлов Я.О., Смірнова Т.В. «Дослідження застосування SIEM-систем для забезпечення кібербезпеки та захисту інформації». *II Міжнародна науково-практична Інтернет-конференція «Інновації та*

перспективні шляхи hозвитку інформаційних технологій (ІПШРІТ-2023)» м.Черкаси 6 грудня 2023 року – Черкаси: ЧДТУ.– 2023. – С.251-252.

39. Козлов Я.О., Смірнова Т.В., Смірнов О.А. «Дослідження SIEM-систем для забезпечення кібербезпеки». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 26.

40. Козлов Я.О., Козірова Н.Л., Смірнов О.А. «Дослідження структури та принципу роботи SIEM-системи». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 59.

41. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 2(72), С. 170-178.

42. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

43. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

44. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу

запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

45. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». *CEUR Workshop Proceedings* Volume 2732, 2020, Pages 214-227.

46. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Книшук А.В. «Вступ до кібербезпеки»: навчальний посібник – Кропивницький: ЦНТУ – 2022. – 968 с.

47. Теорія та практика сучасного інформаційно-психологічного протиборства: навчальний посібник / [В.М. Петрик, С.О. Гнатюк, М.М. Присяжнюк та ін.]; за заг. ред. С.О. Гнатюка, В.М. Петрика та О.А Смірнова. – Полтава, 2022. – 334 с.

48. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». *International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019*; Odessa; Ukraine; 9-13 September 2019. P.22-28.

49. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

50. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

51. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

52. Smirnov, O., Kuznetsov, A., Shekhanin, K., Chepurko, I. Detecting Hidden Information in FAT. Монографія: In.: ISCI'2019: Information Security in Critical

Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 412-429. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook)

53. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

54. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

Додаток А

Технічне завдання

ЗМІСТ

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки клієнтського доступу до бази аналітичних даних ґрунтів.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 57-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки клієнтського доступу до бази аналітичних даних ґрунтів.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми літератури та існуючих аналогів.

5 Технічні вимоги

5.1 Вміст проекту

Складовими розробки є:

– огляд та аналіз аналогічних існуючих систем на предмет їхньої відповідності сучасним вимогам.

– вибір і обґрунтування вибору засобів для побудови системи та мови програмування;

– опис і розробка функціональних, структурних схем та діаграм процесів.

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– систему передачі даних через протокол HTTPS з використанням унікального Access Token;

– шифрування користувацького файлу для відправки та дешифруванню під час отримання;

– цілісність даних у процесі роботи та при зберіганні;

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на кількість шифрування та розмір файлу.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати архітектуру клієнт/сервер.

5.5 Вимоги до надійності

Програмні модулі створені відповідно до всіх правил, що регламентують стандартні виклики процедур, функцій, методів і форм, визначених технічною документацією середовища розробки.

5.6 Умови експлуатації

Робочі місця користувачів програмного забезпечення повинні відповідати таким умовам експлуатації:

– температура повітря: 15–22 °С;
– відносна вологість повітря до 70%;
– атмосферний тиск 100 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на персональних електронно-обчислювальних машинах (ПЕОМ) архітектури IBM PC, функціонувати під керуванням операційних систем Windows 10/11 та підтримувати сумісні з цією платформою пристрої й прикладне програмне забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна забезпечуватися шляхом реалізації стандартного інтерфейсу взаємодії з операційними системами Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Core i3/8 GB/1.2 TB/Full HD 15” 2 GB або сумісні з ним.

5.8.2 Мова програмування

Середовище C++, C, Python,

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у вигляді опису структури даних, схем, алгоритмів та діаграм, а також текстів вихідних модулів програмного забезпечення відповідно до ЄСПД.

7 Перелік документів, що розробляються

– Структурна схема системи	– 1 аркуш.
– Функціональна схема системи	– 1 аркуш.
– Діаграма процесів	– 1 аркуш.
– Блок-схема алгоритму роботи програми	– 3 аркуша.
– Пояснювальна записка	– 48 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації за темою випускної кваліфікаційної роботи першого (бакалаврського) рівня вищої освіти. Формулювання задачі для виконання випускної кваліфікаційної роботи першого (бакалаврського) рівня вищої освіти (розробка технічного завдання).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи першого (бакалаврського) рівня вищої освіти.

8.3 Розробка функціональних схем, блок-схем та алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу програмного забезпечення.

8.6 Тестування програмного забезпечення, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки та виконання робіт, пов'язаних з графічною частиною.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 25.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 02.06.2025 р.

ДОДАТОК Б

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Доренський О. П.

Програмне забезпечення системи клієнтського доступу до бази аналітичних даних ґрунтів

Лістинг програми

Код документу 12

Носій: USB-флеш-накопичувач

Загальна кількість аркушів: 24

Літера: РП

Кропивницький - 2025 року

Основна програма

Файл mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"


MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    loadCoordinates();

    QQmlContext* context = ui->quickWidget_MapView->rootContext();
    context->setContextProperty("mainWindow", this);

    ui->quickWidget_MapView->setSource(QUrl(QStringLiteral("qrc:/map.qml")));
    ui->quickWidget_MapView->show();

    auto Obje = ui->quickWidget_MapView->rootObject();

     connect(this, SIGNAL(setCenterPosition(QVariant,QVariant)), Obje,
SLOT(setCenterPosition(QVariant, QVariant)));
     connect(this, SIGNAL(setLocationMarking(QVariant, QVariant)), Obje,
SLOT(setLocationMarking(QVariant, QVariant)));

     updateTimer = new QTimer(this);
     connect(updateTimer, &QTimer::timeout, this, &MainWindow::updateMap);
     updateTimer->start(1000);

     //emit setCenterPosition(48.46308, 32.55626);
     //emit setLocationMarking(48.46297, 32.54959);

}

void MainWindow::saveToCoordFile() {
    QFile file("coordinates.json");
    QJsonArray jsonArray;
    if (file.open(QIODevice::ReadOnly)) {
        QByteArray savedData = file.readAll();
        jsonArray = QJsonDocument::fromJson(savedData).array();
        file.close();
    }
    for (auto it = polygonInfo.begin(); it != polygonInfo.end(); ++it) {
        QJsonObject jsonObj;
        jsonObj.insert("id", it.key());
        QVariantList coordinates = it.value().toList();
        jsonObj.insert("coordinates",
QJsonArray::fromVariantList(coordinates));
        jsonArray.append(jsonObj);
    }
    QJsonDocument jsonDoc(jsonArray);
    if (file.open(QIODevice::WriteOnly | QIODevice::Truncate)) {
        file.write(jsonDoc.toJson());
        file.close();
    }
```

```
}
void MainWindow::saveIdToFile(const QString &id, const QString &text) {
    QFile file("information.json");
    QJsonObject jsonObj;
    if (file.open(QIODevice::ReadOnly)) {
        QByteArray savedData = file.readAll();
        jsonObj = QJsonDocument::fromJson(savedData).object();
        file.close();
    }
    jsonObj.insert(QString("id_%1").arg(id), QJsonValue(text));

    QJsonDocument jsonDoc(jsonObj);
    if (file.open(QIODevice::WriteOnly | QIODevice::Truncate)) {
        file.write(jsonDoc.toJson());
        file.close();
    }
}
void MainWindow::saveTextToFile(const QString &id, const QString &text) {
    QFile file("information.json");
    QJsonObject jsonObj;
    if (file.open(QIODevice::ReadOnly)) {
        QByteArray savedData = file.readAll();
        jsonObj = QJsonDocument::fromJson(savedData).object();
        file.close();
    }
    jsonObj.insert(QString("text_%1").arg(id), QJsonValue(text));

    QJsonDocument jsonDoc(jsonObj);
    if (file.open(QIODevice::WriteOnly | QIODevice::Truncate)) {
        file.write(jsonDoc.toJson());
        file.close();
    }
}
void MainWindow::saveColorToFile(const QString &id, const QString &color) {
    QFile file("information.json");
    QJsonObject jsonObj;
    if (file.open(QIODevice::ReadOnly)) {
        QByteArray savedData = file.readAll();
        jsonObj = QJsonDocument::fromJson(savedData).object();
        file.close();
    }
    jsonObj.insert(QString("color_%1").arg(id), QJsonValue(color));

    QJsonDocument jsonDoc(jsonObj);
    if (file.open(QIODevice::WriteOnly | QIODevice::Truncate)) {
        file.write(jsonDoc.toJson());
        file.close();
    }
}

void MainWindow::updateMap(){
    polygonCoord.clear();
    loadCoordinates();
    emit mapUpdated();
}
void MainWindow::loadCoordinates() {
    QFile file("coordinates.json");
    if (!file.open(QIODevice::ReadOnly)) {
        qWarning("Couldn't open file.");
        return;
    }
    QByteArray data = file.readAll();
    QJsonDocument doc(QJsonDocument::fromJson(data));
    QJsonArray array = doc.array();
```

```
    foreach (const QJsonValue &value, array) {
        QJsonObject obj = value.toObject();
        QString id = obj["id"].toString();
        QJsonArray coordinates = obj["coordinates"].toArray();
        QVariantList coords;
        foreach (const QJsonValue &coord, coordinates) {
            QJsonObject coordObj = coord.toObject();
            double latitude = coordObj["latitude"].toDouble();
            double longitude = coordObj["longitude"].toDouble();
            coords.append(QVariant::fromValue(QGeoCoordinate(latitude,
longitude)));
            //qDebug() << "Latitude: " << latitude << ", Longitude: " <<
longitude;
        }
        polygonCoord[id] = QVariant::fromValue(coords);
    }
    emit mapUpdated();
}

std::string MainWindow::read_json_file(const std::string& filename) {
    std::ifstream file(filename, std::ios::binary);
    if (!file) {
        std::cerr << "Could not open file " << filename << std::endl;
        return "";
    }

    // Знаходимо розмір файлу
    file.seekg(0, std::ios::end);
    size_t file_size = file.tellg();
    file.seekg(0, std::ios::beg);

    // Читаємо файл у пам'ять
    std::vector<char> buffer(file_size);
    file.read(buffer.data(), file_size);
    return std::string(buffer.begin(), buffer.end());
}
uint64_t* MainWindow::read_encrypted_from_file(const char* filename, size_t*
data_size) {
    std::ifstream file(filename, std::ios::binary);
    if (!file) {
        std::cerr << "Could not open file " << filename << std::endl;
        return nullptr;
    }

    // Визначаємо розмір файлу
    file.seekg(0, std::ios::end);
    size_t file_size = file.tellg();
    file.seekg(0, std::ios::beg);

    // Виділяємо пам'ять для зашифрованих даних
    auto data = new uint64_t[file_size / sizeof(uint64_t)]; // Кількість
елементів
    if (data == nullptr) {
        std::cerr << "Memory allocation failed" << std::endl;
        return nullptr;
    }

    // Читаємо дані з файлу
    file.read(reinterpret_cast<char*>(data), file_size);
    file.close();

    *data_size = file_size / sizeof(uint64_t);  // Визначаємо кількість
елементів
    return data;
```

```
}

void MainWindow::pad_pkcs7(std::vector<uint8_t>& block, size_t block_size,
size_t data_size) {
    uint8_t pad_value = block_size - data_size;
    block.resize(block_size, pad_value);
}
void MainWindow::remove_padding(std::vector<uint8_t>& data) {
    if (data.empty()) {
        return;
    }
    uint8_t padding_length = data.back();
    if (padding_length > 16 || padding_length > data.size()) {
        throw std::runtime_error("Invalid padding");
    }
    data.resize(data.size() - padding_length);
}

void MainWindow::write_encrypted_to_file(const std::string& filename, const
std::vector<uint64_t>& data) {
    std::ofstream file(filename, std::ios::binary);
    if (!file) {
        std::cerr << "Could not open file for writing: " << filename <<
std::endl;
        return;
    }
    file.write(reinterpret_cast<const char*>(data.data()), data.size() *
sizeof(uint64_t));
    std::cout << "Encrypted data written to " << filename << std::endl;
}
void MainWindow::write_decrypted_to_file(const std::string& filename, const
std::vector<uint64_t>& data) {
    std::vector<uint8_t> decrypted_data(reinterpret_cast<const
uint8_t*>(data.data()),
                                        reinterpret_cast<const
uint8_t*>(data.data() + data.size()));

    remove_padding(decrypted_data);
    std::ofstream output_file(filename, std::ios::binary);
    if (!output_file) {
        throw std::runtime_error("Could not open file for writing");
    }
    output_file.write(reinterpret_cast<const char*>(decrypted_data.data()),
decrypted_data.size());
}

void MainWindow::encryptionFile(const QString& input_file, const QString&
output_file){
    // Зчитування JSON з файлу
    std::string json = read_json_file(input_file.toStdString()); //
Перетворення QString в std::string
    if (json.empty()) {
        return;  // Помилка при зчитуванні файлу
    }

    uint64_t key[2];
    uint64_t iv[2];
    std::ifstream key_file("Key.bin", std::ios::binary);
    if (!key_file) {
        qDebug() << "Could not open key and IV file";
        QMessageBox::information(this, "Key", "Відсутній ключ!");
        return;
    }
    key_file.read(reinterpret_cast<char*>(key), sizeof(key));
```

```
    key_file.read(reinterpret_cast<char*>(iv), sizeof(iv));
    key_file.close();

    // Підготовка тексту
    size_t json_len = json.length();
    size_t padded_len = ((json_len + 15) / 16) * 16;  // Округлення до 16
байт
    std::vector<uint8_t> padded_json(json.begin(), json.end());
    pad_pkcs7(padded_json, padded_len, json_len);  // Застосовуємо доповнення

    // Кількість блоків для шифрування
    size_t num_blocks = padded_len / 16;

    // Масиви для шифрування/дешифрування
    std::vector<uint64_t> pt(reinterpret_cast<uint64_t*>(padded_json.data()),
reinterpret_cast<uint64_t*>(padded_json.data()) + num_blocks * 2);
    std::vector<uint64_t> ct(num_blocks * 2);

    kalyna_t* ctx = KalynaInit(128, 128);
    KalynaKeyExpand(key, ctx);

    // Шифрування
    uint64_t prev_ct[2];
    memcpy(prev_ct, iv, sizeof(prev_ct)); // Ініціалізація вектора
    for (size_t i = 0; i < num_blocks; i++) {
        pt[i * 2] ^= prev_ct[0];  // XOR з попереднім шифрованим блоком
        pt[i * 2 + 1] ^= prev_ct[1];
        KalynaEncipher(pt.data() + i * 2, ctx, ct.data() + i * 2); //
Шифрування блоку
        memcpy(prev_ct, ct.data() + i * 2, sizeof(prev_ct));  // Оновлюємо
попередній шифрований блок
    }

    // Запис зашифрованих даних у файл
    write_encrypted_to_file(output_file.toStdString(), ct); // Перетворення
QString в std::string
    KalynaDelete(ctx);
}
void MainWindow::decryptionFile(const QString& input_file, const QString&
output_file){

    // Зчитування зашифрованих даних з файлу
    size_t data_size;
    uint64_t* ct = read_encrypted_from_file(input_file.toStdString().c_str(),
&data_size);
    if (ct == nullptr) {
        return;  // Помилка при зчитуванні файлу
    }

    uint64_t key[2];
    uint64_t iv[2];
    std::ifstream key_file("Key.bin", std::ios::binary);
    if (!key_file) {
        qDebug() << "Could not open key and IV file";
        QMessageBox::information(this, "Key", "Відсутній ключ!");
        delete[] ct; // Звільняємо пам'ять перед виходом
        return;
    }
    key_file.read(reinterpret_cast<char*>(key), sizeof(key));
    key_file.read(reinterpret_cast<char*>(iv), sizeof(iv));
    key_file.close();

    // Кількість блоків для дешифрування
    size_t num_blocks = data_size / 2;
```

```
    // Масив для збереження розшифрованих даних
    std::vector<uint64_t> pt(num_blocks * 2);

    kalyna_t* ctx = KalynaInit(128, 128);
    KalynaKeyExpand(key, ctx);

    // Дешифрування
    uint64_t prev_ct[2];
    memcpy(prev_ct, iv, sizeof(prev_ct)); // Ініціалізація вектора
    for (size_t i = 0; i < num_blocks; i++) {
        KalynaDecipher(ct + i * 2, ctx, pt.data() + i * 2); // Дешифрування
блоку
        pt[i * 2] ^= prev_ct[0]; // XOR з попереднім шифрованим блоком
        pt[i * 2 + 1] ^= prev_ct[1];
        memcpy(prev_ct, ct + i * 2, sizeof(prev_ct)); // Оновлюємо попередній
шифрований блок
    }

    // Запис розшифрованих даних у файл
    write_decrypted_to_file(output_file.toStdString(), pt); // Перетворення
QString в std::string
    KalynaDelete(ctx);
    delete[] ct;

    // Видалення зашифрованого файлу
    if (remove(input_file.toStdString().c_str()) != 0) {
        qDebug() << "Помилка при видаленні файлу:" << input_file;
    } else {
        qDebug() << "Файл" << input_file << "успішно видалено.";
    }
}

void MainWindow::uploadToDropbox() {
    QString pythonScriptPath = "uploadDropbox.py"; // Вкажи шлях до скрипта

    // Створюємо процес для запуску Python-скрипта
    QProcess process;

    // Вказуємо, що будемо використовувати Python
    process.start("python", QStringList() << pythonScriptPath);

    // Чекаємо на завершення процесу
    process.waitForFinished();

    // Отримуємо стандартний вивід та помилки
    QString output = process.readAllStandardOutput();
    QString error = process.readAllStandardError();

    // Лог для виводу результату
    qDebug() << "Output:" << output;
    qDebug() << "Error:" << error;
}
void MainWindow::downloadFromDropbox() {
    QString pythonScriptPath = "downloadDropbox.py"; // Вкажи шлях до скрипта

    // Створюємо процес для запуску Python-скрипта
    QProcess process;

    // Вказуємо, що будемо використовувати Python
    process.start("python", QStringList() << pythonScriptPath);

    // Чекаємо на завершення процесу
    process.waitForFinished();
```

```
    // Отримуємо стандартний вивід та помилки
    QString output = process.readAllStandardOutput();
    QString error = process.readAllStandardError();

    // Лог для виводу результату
    qDebug() << "Output:" << output;
    qDebug() << "Error:" << error;
}

void MainWindow::gen_key_and_iv(const QString& filename){
    uint64_t key[2];
    uint64_t iv[2];
    std::random_device rd;  // Генератор випадкових чисел
    std::mt19937_64 gen(rd()); // 64-бітний генератор
    std::uniform_int_distribution<uint64_t> dis;

    // Генерація випадкових ключа і IV
    key[0] = dis(gen);
    key[1] = dis(gen);
    iv[0] = dis(gen);
    iv[1] = dis(gen);

    // Перетворюємо QString у std::string
    std::string std_filename = filename.toStdString();
    std::ofstream file(std_filename, std::ios::binary);

    if (!file) {
        std::cerr << "Could not open file for writing: " << std_filename <<
std::endl;
        return;
    }

    qDebug() <<
"||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||||KeyKalyna:" << key[0] << key[1];
    qDebug() <<
"||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||||IVKalyna:" << iv[0] << iv[1];

    // Записуємо ключ і IV у файл
    file.write(reinterpret_cast<const char*>(key), sizeof(uint64_t) * 2);
    file.write(reinterpret_cast<const char*>(iv), sizeof(uint64_t) * 2);

    file.close();
}

MainWindow::~MainWindow()
{
    delete ui;
}
```

Файл mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
```

```
#include <QVariant>
#include <QtCore>
#include <QtGui>
#include <QtQuick>
#include <QIcon>
#include <QFile>
#include <QTextStream>
#include <QJsonObject>
#include <QMap>
#include <QJsonDocument>
#include <QIODevice>
#include <QString>
#include <QJsonArray>
#include <QList>
#include <QQmlEngine>
#include <QGeoCoordinate>
#include <QMessageBox>
#include <QObject>

#include <QProcess>
#include <QTimer>
#include <kalyna.h>
#include "transformations.h"
#include "tables.h"

#include <iostream>
#include <fstream>
#include <cstring>
#include <cstdint>
#include <vector>

QT_BEGIN_NAMESPACE
namespace Ui {
class MainWindow;
}
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

    void writeInformation(QString id);

    void saveToCoordFile() ;
     Q_INVOKABLE void createPolygon(const QString &name, const QVariantList
&coordinates);

     Q_INVOKABLE QStringList getIds() const;
     Q_INVOKABLE QVariantList getCoordinates(const QString &id) const;
     Q_INVOKABLE QStringList getInformation(const QString &id) const;
     Q_INVOKABLE QString getColorFromFile(const QString &id);

     Q_INVOKABLE void saveTextToFile(const QString &id, const QString &text);
     Q_INVOKABLE void saveIdToFile(const QString &id, const QString &text);
     Q_INVOKABLE void saveColorToFile(const QString &id, const QString
&color);

     Q_INVOKABLE void updateMap();
     Q_INVOKABLE void deletePolygon(QString id);
     Q_INVOKABLE void backupFiles();
```

```
    Q_INVOKABLE void downloadFromDropbox();
    Q_INVOKABLE void uploadToDropbox();

    Q_INVOKABLE void encryptionFile(const QString& input_file, const
QString& output_file);
    Q_INVOKABLE void decryptionFile(const QString& input_file, const
QString& output_file);

    std::string read_json_file(const std::string& filename);
    uint64_t* read_encrypted_from_file(const char* filename, size_t*
data_size);

    void pad_pkcs7(std::vector<uint8_t>& block, size_t block_size, size_t
data_size);
    void remove_padding(std::vector<uint8_t>& data);

    void write_encrypted_to_file(const std::string& filename, const
std::vector<uint64_t>& data);
    void write_decrypted_to_file(const std::string& filename, const
std::vector<uint64_t>& data);

    Q_INVOKABLE void gen_key_and_iv(const QString& filename);


private:
    Ui::MainWindow *ui;
    QMap<QString, QVariant> polygonCoord;
    QMap<QString, QVariant> polygonInfo;
    QTimer *updateTimer;

signals:
    void setCenterPosition(QVariant, QVariant);
    void setLocationMarking(QVariant, QVariant);
    Q_INVOKABLE void mapUpdated();

public slots:
    Q_INVOKABLE void loadCoordinates();
};
#endif // MAINWINDOW_H
```


Файл main.cpp

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    a.setWindowIcon(QIcon(":/image/logoico.ico"));
    QPixmap pixmap(":/image/logo.png"); // шлях до зображення

    QSplashScreen splash(pixmap);
    splash.setEnabled(false);
```

```
    splash.show();

    QTimer::singleShot(2000, [&]() {
        MainWindow *w = new MainWindow;
        w->setWindowTitle(QObject::tr("IsacMaps"));
        w->setWindowIcon(QIcon(":/image/logo.png"));


        w->show();
        splash.finish(w); // Приховує splash, коли відкриється головне вікно
    });

    return a.exec();
}
```

Файл map.qml

```
import QtQuick 2.15
import QtLocation 5.15
import QtPositioning 5.11
import QtQuick.Dialogs 1.3
import QtQuick.Controls 2.15
import QtQuick.Layouts 1.12

Rectangle {
    id: window

    property double latitude: 48.467714 // 48.510559227953316
    property double longitude: 32.561239 //  32.37612902122501
    property var marker: null
    property var polygons: []
    property Component locationmarker: locmarker

    function setCenterPosition(lati, longi) {
        mapview.pan(latitude - lati, longitude - longi)
        latitude = lati
        longitude = longi
    }

    function setLocationMarking(lati, longi) {
        if (marker != null) {
            mapview.removeMapItem(marker)
        }

        marker = locationmarker.createObject(mapview, {
                                                 "coordinate":
QtPositioning.coordinate(
                                                                 lati,
longi)
                                             })
        latitudeArea.text = lati
        longitudeArea.text = longi
        mapview.addMapItem(marker)
    }

    function updatePolygons() {
        var ids = mainWindow.getIds()
```

```
        console.log("Current IDs:", ids) // Лог для перевірки отриманих ID

        // Оновлюємо вже існуючі полігони
        for (var i = 0; i < ids.length; i++) {
            var id = ids[i]
            var coords = mainWindow.getCoordinates(id)
            console.log("Updating Polygon ID:", id, "with Coordinates:",
                        coords) // Лог для перевірки координат

            // Якщо полігон існує, оновлюємо його
            if (polygons[id]) {
                polygons[id].path = coords
            } else {
                // Якщо полігон не існує, створюємо новий
                var polygon = polygonComponent.createObject(mapview, {

"objectName": id
                                                                    })
                polygon.path = coords
                polygons[id] = polygon // Зберігаємо в масиві
                mapview.addMapItem(polygon)
                console.log("Created new polygon for ID:",
                            id) // Лог для нових полігонів
            }
        }

        // Видалення полігонів, які більше не потрібні
        for (var existingId in polygons) {
            if (ids.indexOf(existingId) === -1) {
                mapview.removeMapItem(polygons[existingId])
                delete polygons[existingId] // Видаляємо з масиву
                console.log("Removed polygon for ID:",
                            existingId) // Лог для видалених полігонів
            }
        }
    }

    Map {
        id: mapview
        anchors.fill: parent
        plugin: Plugin {
            name: "osm"
            PluginParameter {
                name: "osm.mapping.custom.host"
                value:
"http://tile.thunderforest.com/landscape/%z/%x/%y.png?apikey=0dd0e0f729d34
003832ec8828e00cb7c&fake=.png"
            }
        }
        activeMapType: supportedMapTypes[supportedMapTypes.length - 1]
        center: QtPositioning.coordinate(latitude, longitude)
        zoomLevel: 13.5
        Connections {
            target: mainWindow
            onMapUpdated: {
                updatePolygons()
            }
        }
```

```
        MouseArea {
            id: mouseArea
            anchors.fill: parent
            acceptedButtons: Qt.RightButton
            onClicked: {
                var coord = mapview.toCoordinate(Qt.point(mouse.x,
mouse.y))
                setLocationMarking(coord.latitude, coord.longitude)
            }
        }

        Rectangle {
            id: toggleButton
            width: 65
            height: 65
            radius: width * 0.5
            color: mouseAreaTog.pressed ? "deepskyblue" : "lightskyblue"
            opacity: 0.9
            anchors.bottom: parent.bottom
            anchors.left: parent.left
            anchors.margins: 10
            z: 2
            // Властивість для стану меню
            property bool menuExpanded: false

            Text {
                text: "≡"
                anchors.centerIn: parent
                font.pixelSize: 40
                font.pointSize: 18
            }

            MouseArea {
                id: mouseAreaTog
                anchors.fill: parent
                onClicked: {
                    toggleButton.menuExpanded = !toggleButton.menuExpanded

                    // Анімація прозорості
                    zoomInButton.opacity = toggleButton.menuExpanded ? 1 :
0
                    zoomOutButton.opacity = toggleButton.menuExpanded ? 1
: 0
                    resetZoomButton.opacity = toggleButton.menuExpanded ?
1 : 0
                    createPolygonButton.opacity =
toggleButton.menuExpanded ? 1 : 0
                    backupButton.opacity = toggleButton.menuExpanded ? 1 :
0
                    synchronButton.opacity = toggleButton.menuExpanded ? 1
: 0
                }
            }
        }
        Row {
            id: bottomButtonsRow
            spacing: 15
            anchors.bottom: parent.bottom
            anchors.left: toggleButton.left
```

```
            anchors.leftMargin: -8
            anchors.bottomMargin: 11
            // Горизонтальні кнопки (зліва направо)
            Rectangle {
                id: zoomInButton
                width: 60
                height: 60
                radius: width * 0.5
                color: mouseAreaZomIn.pressed ? "deepskyblue" :
"lightskyblue"
                opacity: 0
                anchors.bottom: parent.bottom

                visible: opacity > 0
                z: 1

                Behavior on opacity {
                    NumberAnimation {
                        duration: 250
                    }
                }
                Behavior on x {
                    NumberAnimation {
                        duration: 250
                        easing.type: Easing.OutQuad
                    }
                }

                // Динамічне позиціонування
                Binding on x {
                    when: zoomInButton.visible
                    value: toggleButton.menuExpanded ? toggleButton.x +
toggleButton.width
                                                       + 10 :
toggleButton.x
                }

                Text {
                    text: "+"
                    anchors.centerIn: parent
                    font.pointSize: 18
                }

                MouseArea {
                    id: mouseAreaZomIn
                    anchors.fill: parent
                    onClicked: mapview.zoomLevel += 1
                }
            }

            Rectangle {
                id: zoomOutButton
                width: 60
                height: 60
                radius: width * 0.5
                color: mouseAreaZomOut.pressed ? "deepskyblue" :
"lightskyblue"
                opacity: 0
                anchors.bottom: parent.bottom
```

```
                visible: opacity > 0
                z: 1

                Behavior on opacity {
                    NumberAnimation {
                        duration: 250
                    }
                }
                Behavior on x {
                    NumberAnimation {
                        duration: 250
                        easing.type: Easing.OutQuad
                    }
                }

                Binding on x {
                    when: zoomOutButton.visible
                    value: toggleButton.menuExpanded ? toggleButton.x +
toggleButton.width
                                                       + 75 :
toggleButton.x
                }

                Text {
                    text: "-"
                    anchors.centerIn: parent
                    font.pointSize: 18
                }

                MouseArea {
                    id: mouseAreaZomOut
                    anchors.fill: parent
                    onClicked: mapview.zoomLevel -= 1
                }
            }

            Rectangle {
                id: resetZoomButton
                width: 60
                height: 60
                radius: width * 0.5
                color: mouseAreaZomRes.pressed ? "deepskyblue" :
"lightskyblue"
                opacity: 0
                anchors.bottom: parent.bottom
                visible: opacity > 0
                z: 1

                Behavior on opacity {
                    NumberAnimation {
                        duration: 250
                    }
                }
                Behavior on x {
                    NumberAnimation {
                        duration: 250
                        easing.type: Easing.OutQuad
                    }
                }
```

```
                Binding on x {
                    when: resetZoomButton.visible
                    value: toggleButton.menuExpanded ? toggleButton.x +
toggleButton.width
                                                       + 140 :
toggleButton.x
                }

                Text {
                    text: "O"
                    anchors.centerIn: parent
                    font.pointSize: 18
                }

                MouseArea {
                    id: mouseAreaZomRes
                    anchors.fill: parent
                    onClicked: {
                        mapview.zoomLevel = 13.9
                        mapview.center =
QtPositioning.coordinate(48.467714,

32.561239)
                    }
                }
            }
        }
        // Вертикальні кнопки (знизу догори)
        Rectangle {
            id: createPolygonButton
            width: 59
            height: 59
            radius: width * 0.5
            opacity: 0
            anchors.left: parent.left
            anchors.leftMargin: toggleButton.anchors.leftMargin
            visible: opacity > 0
            z: 1

            Behavior on opacity {
                NumberAnimation {
                    duration: 250
                }
            }
            Behavior on y {
                NumberAnimation {
                    duration: 250
                    easing.type: Easing.OutQuad
                }
            }
            Binding on y {
                when: createPolygonButton.visible
                value: toggleButton.menuExpanded ? toggleButton.y -
toggleButton.height
                                                   - 5 : toggleButton.y
            }

            Image {
```

```
                id: imageCre
                anchors.centerIn: parent
                width: 60
                height: 60
                source: mouseAreaCre.pressed ?
"qrc:/image/bCreateDark.png" : "qrc:/image/bCreate.png"
                fillMode: Image.PreserveAspectFit
                smooth: true
            }
            MouseArea {
                id: mouseAreaCre
                anchors.fill: parent
                onClicked: {
                    synchronDialog.close()
                    if (createPolygonDialog.visible) {
                        createPolygonDialog.close()
                    } else {
                        createPolygonDialog.open()
                    }
                }
            }
        }
        Rectangle {
            id: backupButton
            width: 59
            height: 59
            radius: width * 0.5
            opacity: 0
            anchors.left: parent.left
            anchors.leftMargin: toggleButton.anchors.leftMargin
            visible: opacity > 0
            z: 1

            Behavior on opacity {
                NumberAnimation {
                    duration: 250
                }
            }
            Behavior on y {
                NumberAnimation {
                    duration: 250
                    easing.type: Easing.OutQuad
                }
            }
            Binding on y {
                when: backupButton.visible
                value: toggleButton.menuExpanded ? toggleButton.y -
toggleButton.height
                                                   * 2 - 5 :
toggleButton.y
            }
            Image {
                id: imageBack
                anchors.centerIn: parent
                width: 60
                height: 60
                source: mouseAreaBac.pressed ?
"qrc:/image/bBackupDark.png" : "qrc:/image/bBackup.png"
                fillMode: Image.PreserveAspectFit
```

```
                smooth: true
            }

            MouseArea {
                id: mouseAreaBac
                anchors.fill: parent
                onClicked: mainWindow.backupFiles()
            }
        }
        Rectangle {
            id: synchronButton
            width: 59
            height: 59
            radius: width * 0.5
            opacity: 0
            anchors.left: parent.left
            anchors.leftMargin: toggleButton.anchors.leftMargin
            visible: opacity > 0
            z: 1

            Behavior on opacity {
                NumberAnimation {
                    duration: 250
                }
            }
            Behavior on y {
                NumberAnimation {
                    duration: 250
                    easing.type: Easing.OutQuad
                }
            }

            Binding on y {
                when: synchronButton.visible
                value: toggleButton.menuExpanded ? toggleButton.y -
toggleButton.height
                                                   * 3 - 5 :
toggleButton.y
            }

            Image {
                anchors.centerIn: parent
                width: 60
                height: 60
                source: mouseAreaSyn.pressed ?
"qrc:/image/bSynhroneDark.png" : "qrc:/image/bSynhrone.png"
                fillMode: Image.PreserveAspectFit
                smooth: true
            }
            MouseArea {
                id: mouseAreaSyn
                anchors.fill: parent
                onClicked: {
                    createPolygonDialog.close()
                    if (synchronDialog.visible) {
                        synchronDialog.close()
                    } else {
                        synchronDialog.open()
                    }
```

```
                }
            }
        }
        // Обробник зміни розміру вікна
        Connections {
            target: parent
            function onWidthChanged() {
                // Оновлюємо позиції кнопок при зміні ширини
                zoomInButton.x = Qt.binding(function () {
                    return toggleButton.menuExpanded ? toggleButton.x +
toggleButton.width
                                                     + 10 :
toggleButton.x
                })
                zoomOutButton.x = Qt.binding(function () {
                    return toggleButton.menuExpanded ? toggleButton.x +
toggleButton.width
                                                     + 75 :
toggleButton.x
                })
                resetZoomButton.x = Qt.binding(function () {
                    return toggleButton.menuExpanded ? toggleButton.x +
toggleButton.width
                                                     + 140 :
toggleButton.x
                })
            }
            function onHeightChanged() {
                // Оновлюємо позиції кнопок при зміні висоти
                createPolygonButton.y = Qt.binding(function () {
                    return toggleButton.menuExpanded ? toggleButton.y -
toggleButton.height
                                                     - 5 :
toggleButton.y
                })
                backupButton.y = Qt.binding(function () {
                    return toggleButton.menuExpanded ? toggleButton.y -
toggleButton.height
                                                     * 2 - 5 :
toggleButton.y
                })
                synchronButton.y = Qt.binding(function () {
                    return toggleButton.menuExpanded ? toggleButton.y -
toggleButton.height
                                                     * 3 - 5 :
toggleButton.y
                })
            }
        }
        Dialog {
            id: backupSuccess
            title: "\nУспішне створення резервної копії!"
            font {
                family: "Roboto"
                pixelSize: 20
            }
        }
        Dialog {
            id: synchronDialog
```

```
            closePolicy: Popup.NoAutoClose
            width: 250
            height: 200
            ColumnLayout {
                Label {
                    text: "Синхронізація з хмарою"
                    font {
                        family: "Roboto"
                        pixelSize: 20
                    }
                }
                Item {
                    height: 15
                }
                Button {
                    text: "Скачування з Dropbox"
                    Layout.fillWidth: true
                    font.family: "Roboto"
                    font.pixelSize: 16

                    background: Rectangle {
                        id: downloadBackground
                        color: mouseAreaDow.pressed ? "deepskyblue" :
"lightskyblue"
                        radius: 4
                    }
                    MouseArea {
                        id: mouseAreaDow
                        anchors.fill: parent
                        onClicked: {
                            mainWindow.downloadFromDropbox()
                            mainWindow.decryptionFile("information.bin",
                                                      "information.json")
                            mainWindow.decryptionFile("coordinates.bin",
                                                      "coordinates.json")
                        }
                    }
                }
                Button {
                    text: "Завантаження до Dropbox"
                    Layout.fillWidth: true
                    font.family: "Roboto"
                    font.pixelSize: 16

                    background: Rectangle {
                        id: uploadBackground
                        color: mouseAreaUpl.pressed ? "deepskyblue" :
"lightskyblue"
                        radius: 4
                    }

                    MouseArea {
                        id: mouseAreaUpl
                        anchors.fill: parent
                        onClicked: {
                            mainWindow.encryptionFile("information.json",
                                                      "information.bin")
                            mainWindow.encryptionFile("coordinates.json",
                                                      "coordinates.bin")
```

```
                            mainWindow.uploadToDropbox()
                        }
                    }
                }
                Button {
                    text: "Генерація ключа"
                    Layout.fillWidth: true
                    font.family: "Roboto"
                    font.pixelSize: 16

                    background: Rectangle {
                        id: keygenBackground
                        color: mouseAreaGen.pressed ? "deepskyblue" :
"lightskyblue"
                        radius: 4
                    }

                    MouseArea {
                        id: mouseAreaGen
                        anchors.fill: parent
                        onClicked: {
                            mainWindow.gen_key_and_iv("Key.bin")
                        }
                    }
                }
            }
        }

        Dialog {
            id: createPolygonDialog
            closePolicy: Popup.NoAutoClose
            ColumnLayout {
                TextArea {
                    id: nameArea
                    placeholderText: "Введіть назву              "
                    font {
                        family: "Roboto"
                        pixelSize: 20
                    }
                    width: 200
                    wrapMode: Text.WrapAnywhere
                    background: Rectangle {
                        color: "transparent"
                        border.color: "black"
                        border.width: 1
                    }
                }
                TextArea {
                    id: latitudeArea
                    placeholderText: "Введіть широту          "
                    font {
                        family: "Roboto"
                        pixelSize: 20
                    }
                    width: 200
                    wrapMode: Text.WrapAnywhere
                    background: Rectangle {
                        color: "transparent"
                        border.color: "black"
```

```
                        border.width: 1
                    }
                }
                TextArea {
                    id: longitudeArea
                    placeholderText: "Введіть довготу          "
                    font {
                        family: "Roboto"
                        pixelSize: 20
                    }
                    width: 200
                    wrapMode: Text.WrapAnywhere
                    background: Rectangle {
                        color: "transparent"
                        border.color: "black"
                        border.width: 1
                    }
                }
                ListView {
                    id: coordinatesList
                    width: 200
                    height: 100
                    model: ListModel {
                        id: coordinatesModel
                    }
                    delegate: Text {
                        text: latitude + ", " + longitude
                    }
                    ScrollBar.vertical: ScrollBar {}
                }
                Item {
                    height: 10
                }
                RowLayout {
                    Button {
                        id: buttonCreateAdd
                        text: "Додати"
                        background: Rectangle {
                            color: buttonCreateAdd.pressed ? "deepskyblue"
: "lightskyblue"
                            radius: 5
                        }
                        font {
                            family: "Roboto"
                            pixelSize: 20
                        }
                        Layout.fillWidth: true
                        onClicked: {
                            coordinatesList.model.append({
                                                             "latitude":
Number(
                                                             latitudeArea.text),
                                                             "longitude":
Number(longitudeArea.text)
                                                         })
                            setLocationMarking(Number(latitudeArea.text),
                                               Number(longitudeArea.text))
                            latitudeArea.text = ""
```

```
                            longitudeArea.text = ""
                        }
                    }
                    Button {
                        id: buttonCreateDel
                        text: "Видалити"
                        background: Rectangle {
                            color: buttonCreateDel.pressed ? "deepskyblue"
: "lightskyblue"
                            radius: 5
                        }
                        font {
                            family: "Roboto"
                            pixelSize: 20
                        }
                        Layout.fillWidth: true
                        onClicked: {
                            if (coordinatesList.model.count > 0) {
                                coordinatesList.model.remove(
                                            coordinatesList.model.count -
1)
                            }
                            setLocationMarking(0, 0)
                        }
                    }
                }
                Button {
                    id: buttonCreateCre
                    text: "Створити"
                    background: Rectangle {
                        color: buttonCreateCre.pressed ? "deepskyblue" :
"lightskyblue"
                        radius: 5
                    }
                    font {
                        family: "Roboto"
                        pixelSize: 20
                    }
                    Layout.fillWidth: true
                    onClicked: {

                        var coordinates = []
                        for (var i = 0; i < coordinatesList.model.count;
i++) {
                            coordinates.push({
                                             "latitude":
coordinatesList.model.get(

i).latitude,
                                             "longitude":
coordinatesList.model.get(

i).longitude
                                         })
                        }
                        var ids = mainWindow.getIds()
                        var id = 0
                        for (var j = 0; j < ids.length; j++) {
                            if (nameArea.text === ids[j]) {
```

```
                                console.log(ids[j])
                                id = ids[j]
                            }
                        }
                        if (coordinates.length >= 3 && nameArea.text !==
""
                                && nameArea.text !== id) {
                            mainWindow.createPolygon(nameArea.text,
coordinates)
                            mainWindow.updateMap()
                            updatePolygons()
                            coordinatesModel.clear()
                            nameArea.text = ""
                            createPolygonDialog.close()
                        } else {
                            errorDialog.open()
                        }
                    }
                }
            }
        }
```