НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО **Декан факультету**	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ** **Завідувач кафедри комп'ютерних наук**
_____Ігор БОЛБОТ (підпис)	**_____Белла ГОЛУБ** (підпис)
«___» _____2026 р.	«___» _____2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему

«Підсистема захисту інформації на мобільних пристроях на базі ОС Android»

Спеціальність 122 – «Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми
д.е.н., професор _____ Руденський Р.А.

Керівник бакалаврської кваліфікаційної роботи
Ст. викладач _____ Касаткін Д.Ю.

Виконав _____ Тетерук М.С.

КИЇВ – 2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Комп'ютерних наук_____
(назва кафедри)

___к.т.н.,доцент_____. Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)

"_06" _січня___ 2026 р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студенту
Тетеруку Максиму Сергійовичу

Спеціальність 122 – «Комп'ютерні науки»

ОП «Комп'ютерні науки»

1. Тема бакалаврської кваліфікаційної роботи «Підсистема захисту інформації на мобільних пристроях на базі ОС Android » затверджена наказом ректора НУБіП України від 06.01.2026 №46 «С»
2. Термін подання завершеної роботи на кафедру_____2026.05.25_____
(рік, місяць, число)
3. Вихідні дані до роботи: механізм функціонування інформаційної системи криптографічного захисту даних, включаючи процеси реєстрації та безпечної авторизації користувачів, шифрування та дешифрування електронних файлів за допомогою криптографічних алгоритмів (AES, DES тощо), формування та управління захищеною локальною картотекою, надійне зберігання облікових записів, шляхів до файлів і паролів у базі даних, виконання процедур резервного копіювання (експорт та імпорт), а також відображення статистичної інформації щодо стану об'єктів у мобільному застосунку.
4. Перелік питань, що розглядаються:
   - Аналіз процесів взаємодії між користувачами та системою локального шифрування і зберігання файлів.
   - Проєктування і розробка інформаційного забезпечення системи безпечного зберігання облікових даних (паролів) та управління доступом.
   - Проєктування і розробка програмного забезпечення мобільного застосунку FileLocker Control.
   - Впровадження та експлуатація мобільного застосунку для криптографічного захисту файлів.

Дата видачі завдання "_06"_січня_____ 2026 р. (дата наказу)

Керівник бакалаврської кваліфікаційної роботи _____/ Касаткін Д.Ю./
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання:_____/ Тетерук М.С. /

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД – база даних

ГБ – гігабайт

ГГц – гігагерци

ОЗУ - оперативно запам'ятовуючий пристрій

ОС – операційна система

СУБД – система управління базою даних

AES (Advanced Encryption Standard) - симетричний алгоритм блочного шифрування

API (ApplicationProgramming Interface) - програмний інтерфейс додатку

IDE (Integrated Development Environment) - інтегроване середовище розробки

PHP (Hypertext Preprocessor) - препроцесор гіпертексту

RAW - формат цифрових файлів зображення

SDK (Software Development Kit) - комплект засобів розробки

SHA (Secure Hash Algorithm) - алгоритм криптографічного хешування.

SSL (Secure Sockets Layer) - рівень захищених сокетів

SQL (Structured query language) – мова структурованих запитів

UML (Unified Modeling Language) – уніфікована мова моделювання

XML (Extensible Markup Language) – розширювана мова розмітки

ВСТУП

Тема захисту особистих даних на мобільних пристрях актуальна як ніколи, оскільки смартфон став мультимедійним центром і своєрідною робочою станцією. Мобільні пристрої, які ми використовуємо щодня, є носіями значного обсягу особистої інформації. Історія дзвінків, веб-переглядів, текстові та голосові повідомлення, адресні книги, календарі, фотографії та інші дані можуть стати причиною серйозних ускладнень, якщо пристрій буде втрачено або викрадено. Для запобігання негативним наслідкам необхідно чітко розуміти розташування конфіденційної інформації на мобільному телефоні, а також доступ до онлайн-даних. Ці дані можуть становити загрозу не лише для власника пристрою, але й для всіх осіб, чиї контакти, повідомлення або фотографії зберігаються на ньому.

У сучасних смартфонів багато місця для зберігання даних. Тому, у кого є фізичний доступ до пристрою, може бути нескладно отримати цю інформацію.

Шифрування є найкращою наявною технологією, для захисту даних від зловмисників, урядів, постачальників послуг. На поточний момент воно досягло такого рівня розвитку, що при коректному використанні його практично неможливо зламати.

Метою дипломного проекту є забезпечення конфіденційності та підвищення рівня безпеки даних на мобільному пристрої з ОС Android.

Об’єктом дослідження даного проекту є процеси шифрування файлів та процеси взаємодії клієнт-серверного додатку з БД.

Предметом дослідження є підсистема захисту інформації на мобільних пристроях на базі ОС Android.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка задачі

Основною задачею є розробка підсистеми захисту інформації на мобільних пристроях на базі ОС Android. Підвищення безпеки мобільного телефону - обов'язкова процедура для всіх їх власників. Зловмисники мріють про злом будь-якого обладнання, щоб роздобути паролі від банківських рахунків, кредитних карт, електронних гаманців і інших персональних даних. І навіть якщо ви поставили складний пароль при розблокуванні телефону, несподіваний вірус може проникнути при скачуванні сумнівного додатку

Для підвищення конфіденційності та рівня безпеки даних на мобільному пристрої, було розроблено інформаційну систему захисту файлів на ОС Android.

Основні переваги такої інформаційної системи:

- шифрування файлів за вибраним алгоритмом;
- блокування файлів за допомогою паролю;
- можливість шифрування за допомогою синхронних або асинхронних методів;
- перегляд захищених додатком файлів.

Дана підсистема передбачає зберігання особистих даних у захищеному, від несанкціонованого доступу, вигляді.

1.2 Огляд інформаційних джерел та існуючих рішень

1.2.1 Android — це комплексна операційна система та екосистема програмного забезпечення, побудована на базі ядра Linux. Початково вона розроблялася виключно для мобільних пристроїв: смартфонів, планшетних комп'ютерів та електронних книг. Однак з часом сфера її застосування значно розширилася: сьогодні платформа Android є універсальною і успішно використовується не лише в мобільному сегменті, але й у смарт-телевізорах, мультимедійних приставках, автомобільних системах та інших "розумних" пристроях.

Open Handset Alliance створює стандарти для мобільних пристроїв, що працюють на Android. Alliance надає Android SDK (комплект для розробки програмного забезпечення)

Мова Java використовується в основному для написання коду для Android, навіть незважаючи на те, що можна використовувати інші мови.

Переваги операційної системи Android:

- Android є більш налаштованим. Змінити практично все можна;
- в Android будь-яку нову публікацію можна зробити легко і без будь-якого процесу перегляду;
- Android пропонує відкриту платформу;
- легкий доступ до Android Play Market;
- у майбутніх версіях є підтримка для збереження зображень RAW;
- вбудоване бета-тестування та поетапна розробка;
- вбудована інтеграція з хмарним сховищем Google. 15 Гб безкоштовно, 2$ / місяць за 100 ГБ, 1 ТБ за 10$. Доступні програми Amazon Photos, OneDrive та Dropbox.

Недоліки операційної системи Android:

- зазвичай на Java потрібно більше коду, ніж на Objective-C;
- складні макети та анімації, важче кодувати в Android;
- програми з Android Market можуть містити вірус;

- багато “процесів” у фоновому режимі, що призводять до того, що акумулятор швидко розряджається;
- реклама, завжди буде показ реклами, у верхній або нижній частині програми;
- для викрадення вашої інформації з невідомих ресурсів можна встановити додатки із низьким рівнем безпеки та підробленими програмами;
- висока фрагментація пристрою.[1]

За даними інформаційного порталу StatCounter (рис. 1.1), основними опереаційними системами на ринку мобільних девайсів в 2025 - 2026 роках залишаються програмні продукти від Apple і Google. За підрахунками StatCounter, частка мобільних платформ Android за результатами першого кварталу 2026 року становить 70% усього ринку, а програмного продукту від Apple – 29%[2].

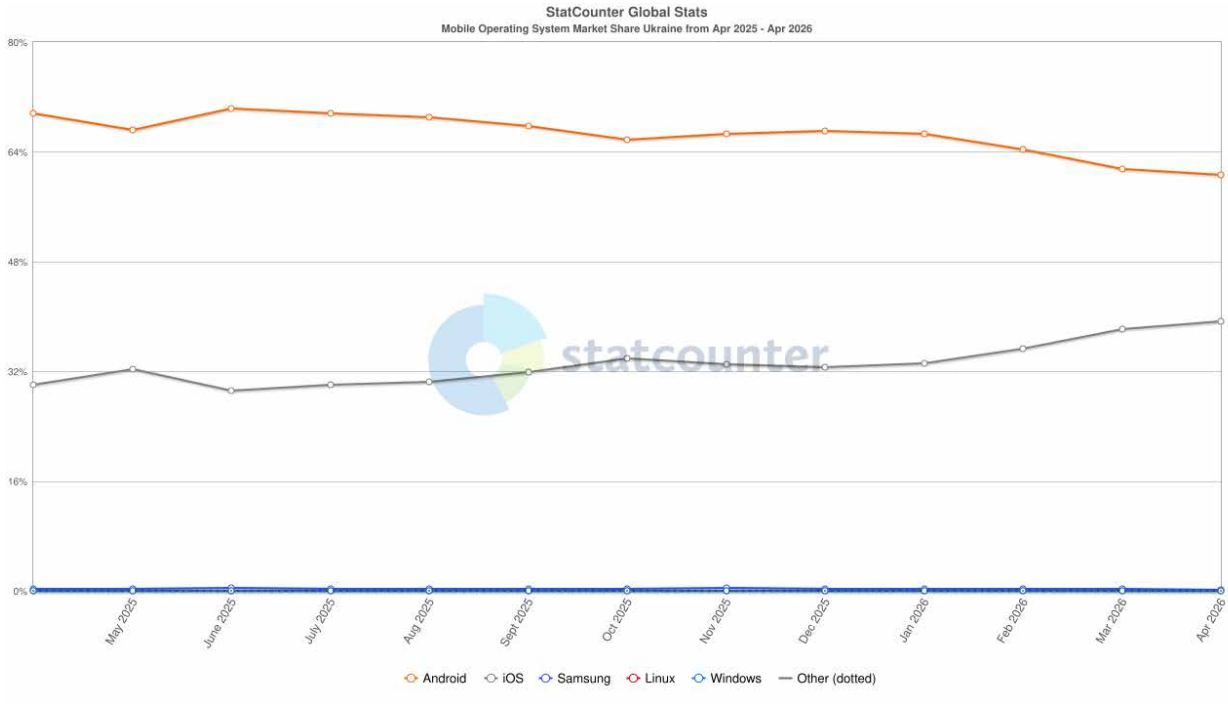

Рис. 1.1 Статистика використання мобільних операційних систем

1.2.2 Шифрування даних. Шифрування - це спосіб скремтування даних, щоб зрозуміти інформацію могли лише уповноважені сторони. У технічному плані це процес перетворення простого тексту в шифротекст. Простіше кажучи, шифрування приймає читабельні дані та змінює їх, щоб вони виглядали випадковими. Принцип шифрування наведено на рис. 1.2.

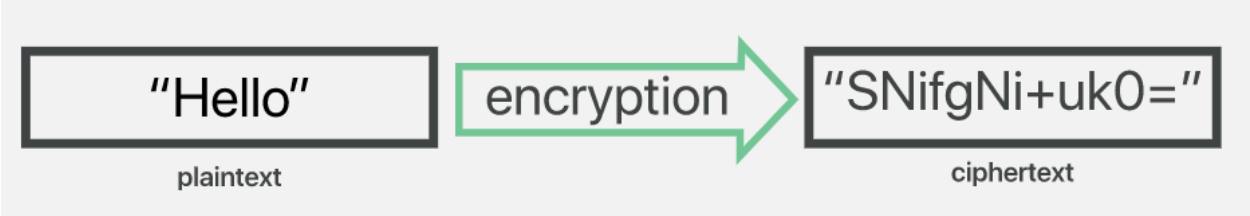

Рис. 1.2 Принцип цифрування даних

Існує два основних види шифрування - це симетричне шифрування та асиметричне шифрування. Асиметричне шифрування також відоме, як шифрування відкритого ключа.

У симетричному шифруванні є лише один ключ, і всі комунікаційні сторони використовують один і той же ключ для шифрування та дешифрування. В шифруванні асиметричного або відкритого ключа є два ключі: один ключ використовується для шифрування, а інший ключ - для дешифрування. Будь-який ключ може бути використаний для будь-якої дії, але дані, зашифровані першим ключем, можна розшифрувати лише за допомогою другого ключа, і навпаки. Один ключ зберігається приватним, тоді як один ключ публічно використовується для будь-якого користувача - звідси і назва "відкритого ключа". Асиметричне шифрування є основоположною технологією для SSL (TLS).

Шифрування забезпечує:

- Конфіденційність: Шифрування гарантує, що ніхто інший не може читати пересилку чи дані, крім призначеного одержувача чи власника даних. Це заважає кіберзлочинцям, рекламним мережам, постачальникам послуг Інтернету, а в деяких випадках урядам не перехоплювати та читати конфіденційні дані;

- Безпека: шифрування допомагає порушувати дані, незалежно від того, перебувають вони в режимі транзиту чи в спокої. Якщо корпоративний пристрій загублено або вкрадено, а його жорсткий диск належним чином зашифровано, дані на цьому пристрої, ймовірно, все ще будуть захищені. Аналогічно, зашифровані комунікації дозволяють сторонам, що спілкуються, обмінюватися чутливими даними, не протікаючи. Шифрування також допомагає запобігти зловмисній поведінці, такі як атаки через посередника;
- Аутентифікація: шифрування відкритого ключа, серед іншого, встановлює, що сервер-джерело веб-сайту володіє приватним ключем і тому законно видано сертифікат SSL.

Брутфорс - це коли зловмисник, який не знає ключа для розшифровки, намагається визначити ключ, роблячи тисячі чи мільйони здогадок. Брутфорс атаки набагато швидше з сучасними комп'ютерами, тому шифрування повинно бути надзвичайно сильним і складним. Більшість сучасних методів шифрування в поєднанні з високоякісними паролями стійкі до грубої атаки, хоча вони можуть бути в майбутньому, оскільки комп’ютери стають все більш потужними. Слабкі паролі все ще сприйнятливі до цього типу атак.[3]

Тому алгоритми шифрування повинні бути якомога складнішим для запобігання швидкого взлому шифрування.

1.2.3Порівняння з існуючим аналогом. Програмою для порівняння із власним проєктом прийнята програма « **EDS Lite** ».

Програма, EDS Lite , хоч і молода, але дуже перспективна. По-перше, для її роботи не потрібні права root, що дуже важливо для багатьох користувачів. По-друге, програма підтримує контейнери TrueCrypt, програма TrueCrypt зараз працює на всіх основних настільних платформах. Але програма не ідеальна. Шифрування здійснюється не «на льоту» і з зашифрованим контейнером не можна працювати як зі звичайною папкою. Однак є вбудований файловий менеджер, який підтримує всі операції над файлами. Підтримуються два алгоритму шифрування - AES 256 і SHA-512.

Конкурентоспроможність продукту, що розробляється, можна оцінити провівши аналіз і порівняння з обраним аналогом за функціональним призначенням, основними технічними та експлуатаційними параметрами, областям застосування. Подібний аналіз здійснюється за допомогою оцінки експлуатаційно-технічного рівня продукту, що розробляється.

Експлуатаційно-технічний рівень (ЕТР) продукту, що розробляється - це узагальнена характеристика його експлуатаційних властивостей, можливостей, ступеня новизни, що є основою якості продукту. Для визначення ЕТР продукту можна використовувати індекс експлуатаційно-технічного рівня J_{ETP}, який розраховується як сума приватних індексів, куди входять показники якості програмного продукту. Для обліку значимості окремих параметрів застосовується бально-індексний метод.

Тоді $$J_{ETP} = \sum_{j=1}^{n} Bj \times Xj,$$

де J_{ETP} - комплексний показник якості продукту по групі показників;

n – число розглянутих показників;

B_j – коефіцієнт вагомості j-го показника в долях одиниці, який призначається відповідно до потреб організації-замовника програмного продукту; X_j – відносний показник якості, що встановлюється експертним шляхом за обраною шкалою оцінювання. Розрахунок параметрів бально-індексним методом наведено у табл. 1.

Таблиця 1

Розрахунок бально-індексним методом

Показники якості	Коефіцієнти вагомості, B_j	Проект		Аналог	
		X_j	$B_j \times X_j$	X_j	$B_j \times X_j$
1. Зручність роботи (користувацький інтерфейс)	0,05	2	0,1	3	0,15
2. Новизна (відповідність сучасним вимогам)	0,15	4	0,6	3	0,45

Таблиця 1 (завершення)

3. Надійність (захист даних)	0,30	3	0,9	3	0,9
4. Швидкість доступу до даних	0,18	3	0,54	4	0,72
5. Гнучкість	0,07	3	0,21	3	0,21
6. Функції обробки інформації	0,15	2	0,3	2	0,3
7. Співвідношення ціна/можливості	0,1	3	0,3	2	0,2
Узагальнений показник якості $J_{ЕТР}$		$J_{ЕТР1}$ = 2,95		$J_{ЕТР2}$ = 2,87	

В таблиці представлені результати розрахунку бально-індексним методом за п'ятибальною шкалою оцінювання.

Відношення двох знайдених індексів називають коефіцієнтом технічного рівня A_k першого програмного продукту по відношенню до другого

$$A_k = \frac{J_{ЕТР1}}{J_{ЕТР2}} = \frac{2,95}{2,87} = 1,03$$

Оскільки коефіцієнт більший за 1, то розробка проекту з технічної точки зору виправдана.

1.3 Моделювання предметної області

UML - це уніфікований графічна мова моделювання для опису, візуалізації, проектування та документування об'єктно-орієнтованих систем. Моделі на UML використовуються на всіх етапах життєвого циклу ПС, починаючи з бізнес-аналізу і закінчуючи супроводом системи.

UML забезпечує:

- ієрархічний опис складної системи шляхом виділення пакетів;
- деталізацію вимог до системи шляхом побудови діаграм діяльностей і сценаріїв;

- виділення класів даних і побудова концептуальної моделі даних у вигляді діаграм класів;
- виділення класів, що описують призначений для користувача інтерфейс, і створення схеми навігації екранів;
- опис поведінки об'єктів у вигляді діаграм діяльностей і станів;
- опис програмних компонент і їх взаємодії через інтерфейси;
- опис фізичної архітектури системи. [4]

Для моделювання процесів, які відбуваються у системі було розроблено діаграму прецедентів, розгортання, взаємодії та логічну модель БД.

1.3.1 Діаграма прецедентів. Діаграма прецедентів створюється для виявлення і формального уявлення вимог замовника до проектованої системі, тобто вона описує, які можливості надаватиме система кінцевому користувачеві, яка інформація необхідна для обробки запиту користувача. При цьому механізм функціонування системи від користувача приховано і на діаграмі прецедентів не показується.

Між компонентами діаграми прецедентів можуть існувати різні відносини. Відносини можуть бути між користувачами і прецедентами, між кількома користувачами. Користувач може взаємодіяти з декількома прецедентами.

Існує два основних види відносин між компонентами на діаграмах прецедентів:

- Відношення розширення (extend relationship) визначає взаємозв'язок прецеденту з прецедентом, можливості якого він може використовувати. Графічно позначається пунктирною стрілкою з позначкою «extend» від доповнює прецеденту до розширюватися;
- Відношення включення (include relationship) вказує на включення прецеденту в інший прецедент в якості його складової частини. Один і той же прецедент може бути включений в кілька більших прецедентів. Графічно дане

відношення позначається суцільною лінією зі стрілкою, спрямованої від базового прецеденту до такого, що включається з позначкою «include».

Підсистема захисту інформації на мобільних пристроях на базі ОС Android має два основних актори:

Користувач – має доступ до всіх функцій системи, а саме:

- **завантаження файлу з локального носія або з БД;**
- **використовувати усі доступні алгоритми шифрування;**
- **захищати файли шляхом шифрування;**
- **здійснювати блокування файлів за допомогою паролю;**
- **здійснювати скидання паролю з заблокованих файлів;**
- **переглядати список заблокованих файлів;**
- **зберігати файли/папки в усіх доступних форматах.**

Неавторизований користувач – має обмежений доступ до функцій системи, а саме:

- **здійснення реєстрації/авторизації;**
- **завантаження файлу з локального носія;**
- **використовувати деякі основні алгоритми шифрування;**
- **захищати файли шляхом шифрування;**
- **здійснювати блокування файлів за допомогою паролю;**
- **здійснювати скидання паролю з заблокованих файлів;**
- **зберігати захищені файли тільки локально;**
- **зберігати файли/папки тільки в певному форматі.**

Діаграма прецедентів підсистеми захисту інформації на мобільних пристроях на базі ОС Android зображена на Плакаті 1. Діаграма показує взаємодію акторів (користувач, неавторизований користувач) із системою за допомогою прецедентів.

1.3.2Діаграма послідовності. Діаграма послідовності (sequence diagram) є основним способом відображення взаємодії об'єктів в часі. Незважаючи на те, що діаграма послідовності спочатку орієнтована на програмні проекти з

використанням ООП, вона застосовується значно ширше, в тому числі у випадках, коли не передбачені відповідні методи опису «динаміки» програми. У подібних випадках повинна здійснюватися адаптація поняття об'єкта.

Діаграма послідовності, як і інші діаграми, що виділяються стандартом UML, повинна зображуватися відповідно до нього.

Основні моменти, на які слід звернути увагу при роботі над діаграмою послідовності:

1. Взаємодіючі об'єкти (objects), як правило, зображуються у вигляді прямокутників з суцільними межами і розміщуються по горизонталі.

Усередині прямокутника вказується ім'я об'єкта, за якими через двокрапку може слідувати ім'я класу даного об'єкта. Імена об'єкта і класу підкреслюються.

Серед об'єктів можуть виділятися так звані актори (actors), Актори повинні зображувати не особливим чином за допомогою символу «чоловічка», а так, як звичайні об'єкти.

2. Лінія життя (lifeline) об'єкта зображується за допомогою штриховий лінії, яка проводиться вертикально вниз від середини його нижньої межі.

За допомогою лінії життя показується період часу, протягом якого об'єкт існує в системі і, отже, може потенційно брати участь у взаємодіях.

Активність (activity) об'єкта, тобто період часу, протягом якого він бере участь у взаємодії, може збігатися з фокусом управління (focus of control) і відображається тонким вертикальним прямокутником відповідної тривалості на лінії його життя. Об'єкти - ініціатори взаємодій рекомендується зображати на кресленні лівіше.

3. Повідомлення (messages), якими обмінюються об'єкти в процесі взаємодії, показуються різними лініями зі стрілками між лініями життя об'єктів, спрямованими в бік передачі. [5]

Термін повідомлення має максимально широкий сенс і може означати будь-який вид передачі управління або даних.

Діаграма послідовності взаємодії користувача з підсистемою захисту інформації на мобільних пристроях на базі ОС Android представлена на Рис.1.3.

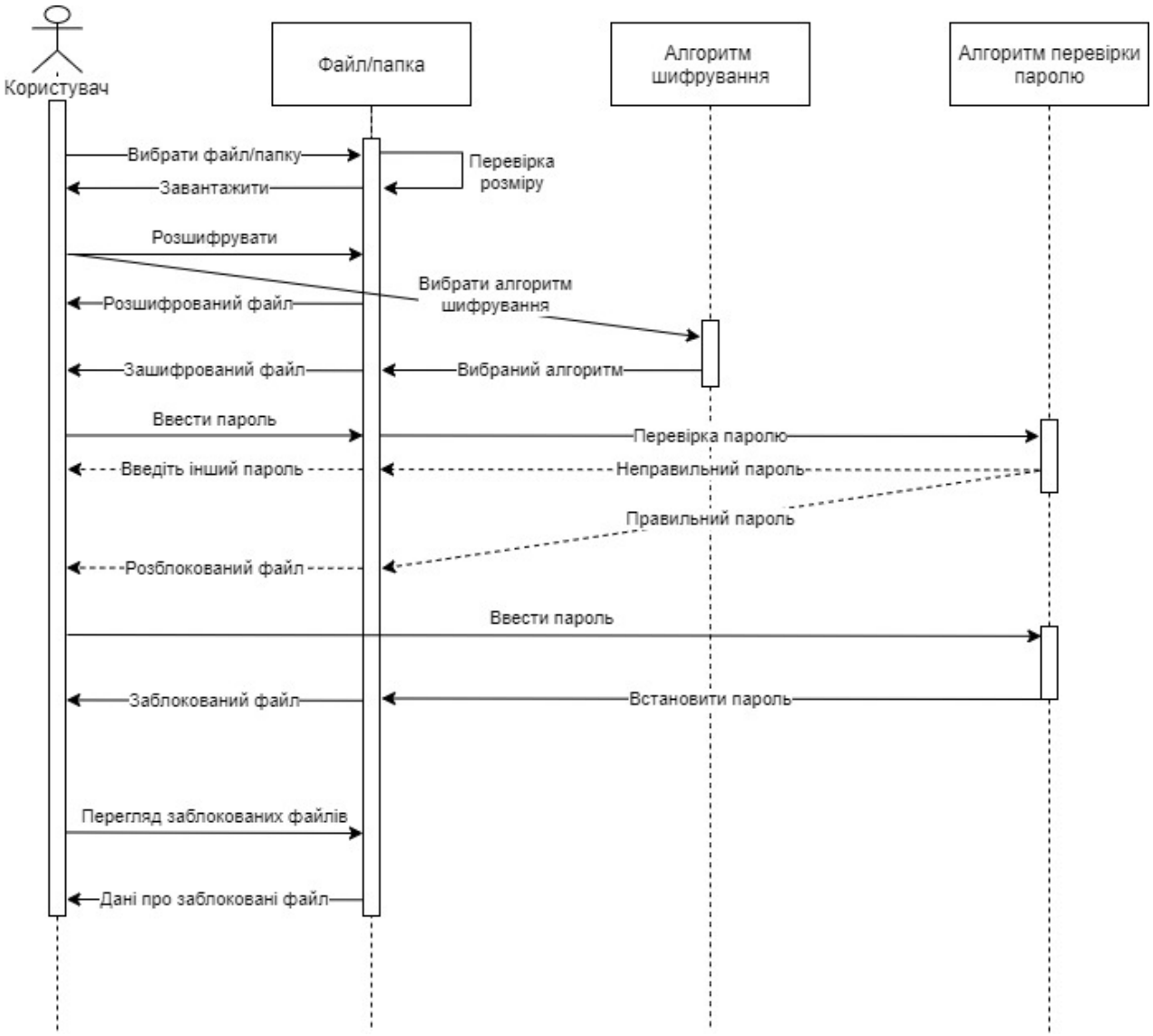

Рис. 1.3 Діаграма послідовності

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ ПРОЕКТУ

2.1 Вибір методів та засобів для реалізації інформаційного забезпечення системи

Для розробки проекту «Підсистема захисту інформації на мобільних пристроях на базі ОС Android» було вибрано систему управління базою даних – MySQL.

По-перше, важливо відзначити її відкритість. Код, який використовується MySQL, є відкритим, що дозволяє додавати будь-який функціонал без будь-яких перешкод. Є дуже багато розширень, створених різними авторами для MySQL, що також вигідно виділяє цю СУБД серед

схожих програм.

На сьогодні версії програми СУБД MySQL існують для більшості поширених комп'ютерних платформ, таких, як ОС Windows, Linux, MacOS. Також плюсом MySQL вважається її швидкодія. В MySQL реалізована багатопоточність, що гарантує високу швидкість роботи бази даних.

На відмінну від деякий інших СУБД, має клієнт-серверну архітектуру, що дуже важливо при взаємодії мобільного додатка з базою даних.

Для реалізації клієнт – серверної архітектури використовується безкоштовний веб-сервер Apache HTTP Server. Apache добре і зручно налаштовуэться, оскільки має модульну структуру. Модулі дозволяють адміністраторам сервера включати або вимикати додаткову функціональність. У Apache є модулі безпеки, кешування, редагування URL, аутентифікації за допомогою паролю та інші.

Переваги Apache серед інших веб-серверів:

- безкоштовний навіть для використання в комерційних цілях;

- надійний, стабільне програмне забезпечення;
- часто оновлюваний, регулярні патчі безпеки;
- гнучкий, завдяки своїй модульній структурі.;
- багатоплатформовий (працює однаково добре на Unix і на Windows серверах);
- велика спільнота і легко доступна підтримка у разі будь-якої проблеми. [6]

Для взаємодії додатку з базою даних використовуються скриптова мова PHP. На відмінну від інших скриптових мов, PHP-скрипти виконуються на сервері та генерують HTML, який пересилається клієнту. Стандартні бібліотеки PHP мають усі необхідні функції для взаємодії з MySQL.

2.2 Логічна модель БД

Логічна схема - модель бази даних, виражена в поняттях моделі даних. Цим відрізняється від концептуальної моделі, яка описує семантику предметної області без вказівки технології (конкретних методів реалізації), і від фізичної моделі, яка описує конкретні фізичні механізми, що застосовуються для зберігання даних в накопичувачах.

Метою побудови логічної моделі є отримання графічного представлення логічної структури досліджуваної предметної області. Логічна модель предметної області ілюструє сутності, а також їх взаємовідносини між собою.[7]

Логічна модель підсистеми захисту інформації на мобільних пристроях на базі ОС Android представлена на Плакаті 2.

Дана логічна модель знаходиться у третій нормальній формі, оскільки:

- в таблицях знаходяться атомарні дані (кожен атрибут містить єдине значення);
- відсутня транзитивна функціональна залежність (кожен

неключовий атрибут залежить тільки від ключового атрибуту);

- наявна функціональна залежність (дані у не ключових стовпцях повністю залежать від первинного ключа).

2.3 Створення інформаційної бази

Щоб створити нову базу даних на мові SQL потрібно використати команду CREATE SCHEMA.

Створення БД для даної підсистеми відбувається за команди зображеної на рис. 2.1.

```
CREATE DATABASE `protect`
```

Рис. 2.1 Запит на створення бази даних

2.3.1 Створення таблиць. Для створення таблиць програмним способом використовують оператор CREATE TABLE. Для цього потрібно вказати наступні дані:

- ім'я таблиці, яке вказується після ключового слова CREATE TABLE;
- імена та визначення стовпців таблиці, відокремлені комами.

Приклад створення таблиці для БД даного проекту наведено на рис. 2.2:

```
CREATE TABLE `object` (
  `id_obj` int(15) NOT NULL AUTO_INCREMENT,
  `name` varchar(30) CHARACTER SET utf8 NOT NULL,
  `path` varchar(100) NOT NULL,
  `status` enum('free','block','unblock','encrypted','decrypted') NOT NULL,
  `login` varchar(30) DEFAULT NULL,
  `id_alg` tinyint(4) DEFAULT NULL,
  PRIMARY KEY (`id_obj`),
  KEY `fk_alg_id_idx` (`id_alg`),
  KEY `fk_user_login_idx` (`login`),
  CONSTRAINT `fk_alg_id` FOREIGN KEY (`id_alg`) REFERENCES `algorithm` (`id_alg`),
  CONSTRAINT `fk_user_login` FOREIGN KEY (`login`) REFERENCES `user` (`login`) ON DELETE NO ACTION ON UPDATE NO ACTION
)
```

Рис. 2.2 Запит на створення таблиці «Object»

Після виконання запиту створилася таблиця із наступною структурою (рис. 2.3)

Column	Type	Default Value	Nullable	Character Set	Collation	Privileges	Extra
id_obj	int(15)		NO			select,insert,update,references	auto_increment
name	varchar(30)		NO	utf8	utf8_general_ci	select,insert,update,references	
path	varchar(100)		NO	latin1	latin1_swedish_ci	select,insert,update,references	
status	enum('free','block','u...		NO	latin1	latin1_swedish_ci	select,insert,update,references	
login	varchar(30)		YES	latin1	latin1_swedish_ci	select,insert,update,references	
id_alg	tinyint(4)		YES			select,insert,update,references	

Рис. 2.3 Структура таблиці «Object»

SQL – запити на створення інших таблиць знаходяться у додатку А.

2.3.2 Створення тригерів. Тригер - це відкомпільована SQL-процедура, виконання якої обумовлено настанням певних подій всередині реляційної бази даних. Застосування тригерів здебільшого зручно для користувачів бази даних.

Тригер являє собою спеціальний тип збережених процедур, що запускаються сервером автоматично при спробі зміни даних в таблицях, до яких тригери прив'язані. Кожен тригер прив'язується до конкретної таблиці. Всі вироблені їм модифікації даних розглядаються як одна транзакція. У разі виявлення помилки або порушення цілісності даних відбувається відкат цієї транзакції. Тим самим внесення змін забороняється. Скасовуються також всі зміни, вже зроблені тригером.

Приклад створення тригеру для відображення в таблиці «Log» змін у таблиці «Object» наведено на рис. 2.3.

```
CREATE DEFINER=`root`@`localhost` TRIGGER `protect`.`object_AFTER_INSERT`
AFTER INSERT ON `object` FOR EACH ROW
BEGIN
INSERT into log Set action ='insert', id_obj = NEW.id_obj, login = New.login;
END
```

Рис. 2.4 Запит на створення тригеру

3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Організаційна структура програмного забезпечення

Будь-яка інформаційна система повинна мати мінімум три основні функціональні частини – модулі зберігання даних, обробки і інтерфейсу з користувачем. Кожна з яких може бути реалізована незалежно від двох інших.

Виходячи із вимог до розроблюваної підсистеми найбільш доцільним є використання клієнт-серверної архітектури, що дозволить три основні функціональні частини розподілити по двох фізичних модулях.[8]

В основі взаємодії клієнт-серверної архітектури лежить принцип того, що таку взаємодію починає клієнт, сервер лише відповідає клієнту і повідомляє про те чи може він надати послугу клієнтові і якщо може, то на яких умовах. Клієнтське програмне забезпечення та серверне програмне забезпечення зазвичай встановлено на різних машинах, але також вони можуть працювати і на одному комп'ютері. Дана концепція взаємодії була розроблена впершу чергу для того, щоб розділити навантаження між учасниками процесу обміну інформацією. Архітектура клієнт-сервер визначає лише загальні принципи взаємодії між комп'ютерами, деталі взаємодії визначають різні протоколи. [9]

Програмне забезпечення, що відповідає за зберігання даних, розташовується на сервері, інтерфейс з користувачем – на стороні клієнта, а обробку даних доводиться розподіляти між клієнтською і серверною частинами.

3.2 Вибір інструментарію для створення прикладного програмного забезпечення

Для створення мобільного застосунку «FileLocker Control» під ОС Android було використано середовище розробки Android Studio. Це офіційна IDE від Google, яка завдяки інтеграції з компонентом WebView дозволяє компілювати вебтехнології у повноцінний нативний APK-файл.

Основні переваги використання Android Studio для цього проєкту:

- гнучка система збірки (Gradle) для зручної інтеграції необхідних плагінів;
- вбудований емулятор для швидкого тестування мобільного інтерфейсу на різних екранах;
- інструменти аналізу продуктивності, які є критично важливими для стабільної роботи алгоритмів локального шифрування файлів;
- зручна генерація та підпис готового APK-файлу для його встановлення на пристрої.

Серверна частина реалізована за допомогою скриптової мови PHP та бази даних MySQL. PHP виконує роль зв’язуючого API-інтерфейсу: він приймає запити від мобільного додатка і відповідає за безпечну авторизацію та збереження даних картотеки користувачів на сервері.

3.3 Алгоритмізація та програмування програмних модулів

3.3.1 Реалізація шифрування файлів розглядається на прикладі Advanced Encryption Standard (AES) — найбільш поширеного та визнаного у світі алгоритму симетричного шифрування. Практичні дослідження доводять, що AES працює щонайменше в шість разів швидше за свій аналог — потрійний DES.

В основі алгоритму лежить криптографічний принцип "мережі підстановок та перестановок". Процес шифрування складається з послідовності математичних операцій: частина з них відповідає за заміну вхідних даних на інші значення (підстановка), а інша — за зміщення елементів за певними правилами (перестановка).

Особливістю AES є побайтова, а не побітова обробка інформації. Стандартний 128-бітний блок відкритого тексту розглядається алгоритмом як 16 байт. Для зручності обчислень ці 16 байт організовуються у матрицю розміром 4x4 (чотири рядки та чотири стовпці).

На відміну від стандарту DES, кількість раундів перетворення в AES не є фіксованою і прямо залежить від обраної довжини ключа. Зокрема, для 128-бітних ключів виконується 10 раундів, для 192-бітних — 12, а для 256-бітних — 14 раундів. Під час кожного такого раунду використовується унікальний 128-бітний підключ, який генерується з початкового секретного ключа AES. Принцип роботи шифрування та дешифрування методо AES наведено на рис. 3.1.

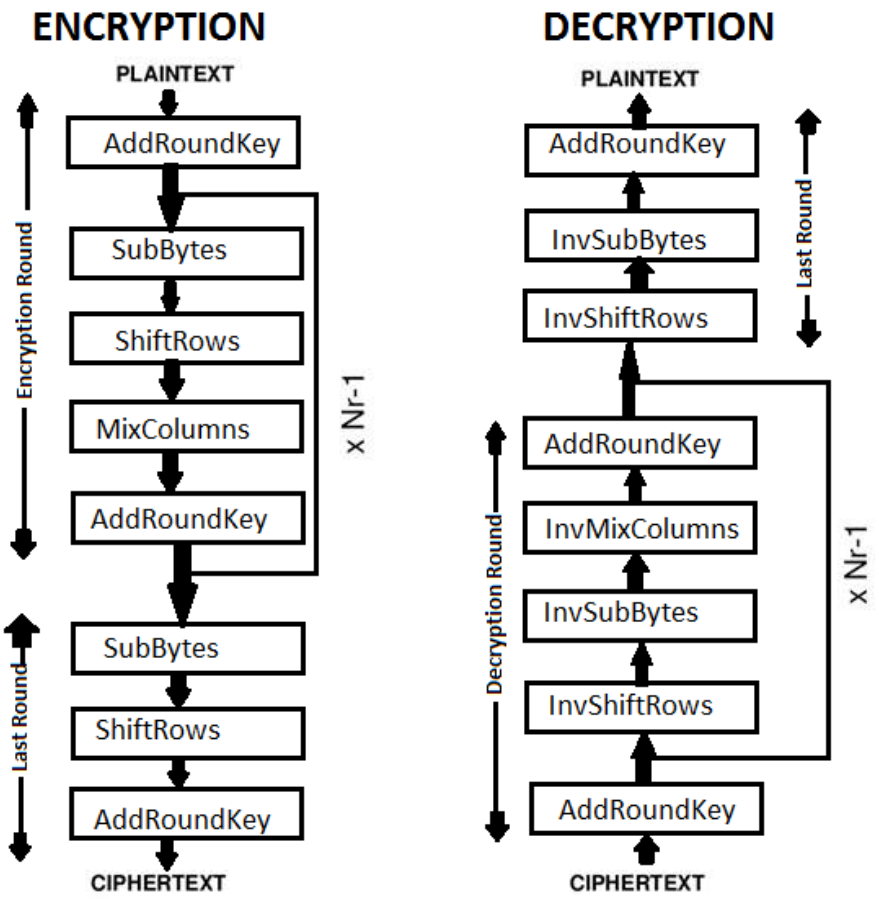

Рис. 3.1 Схема роботи алгоритму AES

Далі зображен процес типового раунду шифрування AES. Кожен раунд складається з чотирьох підпроцесів. Процес першого раунду зображений на рис. 3.2.

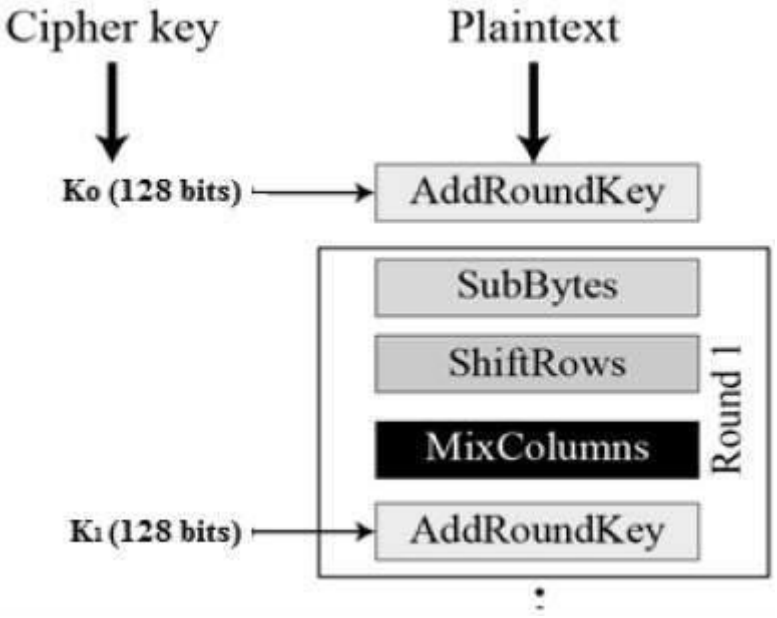

Рисунок 3.2. Процес першого раунду кодування

Кожен з чотирьох рядів матриці зміщується вліво. Будь-які записи, які "відпадають", знову вставляються в правій частині рядка. Зсув здійснюється наступним чином:

- перший ряд не зміщується;
- другий ряд зміщений на одну (байтну) позицію вліво;
- третій ряд зміщений на дві позиції вліво;
- четвертий ряд зміщується на три положення вліво.

В результаті виходить нова матриця, що складається з тих же 16 байт, але зміщених відносно один одного.

Процес дешифрування тексту AES аналогічний процесу шифрування у зворотному порядку. Кожен раунд складається з чотирьох процесів, що проводяться у зворотному порядку:

- додати зовнішній ключ;
- змішати стовпчики;
- змішати ряди;
- заміна байтів.

Оскільки підпроцеси в кожному раунді відбуваються у зворотному порядку, на відміну від фейстельської шифри, алгоритми шифрування та дешифрування повинні бути реалізовані окремо, хоча вони дуже тісно пов'язані. [10]

Для програмної реалізації алгоритму шифрування строки було використано OpenPGP API – бібліотека CryptoLight. Бібліотека забезпечує коректну роботу на мобільних пристроях з мінімальною версією ОС Android - 4.4+.

Для використання функцій проекту CryptoLight потрібно додати у файл build.gradle залежність та вказати посилання на репозиторій (рис. 3.3).

```
allprojects {
 repositories {
  ...
  maven { url 'https://jitpack.io' }
 }
}
```

Рис. 3.3 Посилання на репозиторій

Також потрібно вказати відповідну залежність (рис. 3.4).

```
dependencies {
        implementation 'com.github.dori871992:CryptoLight:1.1.0'
}
```

Рис. 3.4 Необхідна залежність для CryptoLight

3.3.2 Реалізація інтерфейсу користувача. Для реалізації інтерфейсу користувача було використано графічний редактор Adobe Photoshop CS6 та XML.

Стандарт XML визначає набір базових лексичних та синтаксичних правил для побудови мови описання інформації шляхом застосування простих тегів. Він був створений для структурування, зберігання і передачі інформації.

Для зручності роботи корисутвача з програмою був обран макет додатку Navigation Drawler.

Дизайн програми був розроблений за методико Material design. Material design - це мова візуальних образів, яку не так давно створила корпорація Google для уніфікації інтерфейсів всіх її продуктів і сервісів. Брендбук, який включає в себе всі елементи даного напрямку в дизайні, постійно розвивається і доповнюється, при цьому зберігаючи фундаментальні основи незмінними.

Стратегічне бачення компанії полягає в створенні нового користувацького досвіду і проникнення сервісів в усі аспекти життєдіяльності користувача. В цьому і полягає ідея єдиного інтерфейсу - об'єднати весь різноманітний і навіть розколоту сервіс в єдиному ключі, щоб створити цілісний призначений для користувача досвід.

Поверхні і контури елементів в даному напрямку дизайну створюють візуальні образи і сигнали, які передають підказки і допомагають інтуїтивно орієнтуватися, як якщо б це відбувалося в реальному світі.

Використання знайомих тактильних характеристик і реалістичне освітлення допомагають користувачеві візуально відокремити головні об'єкти від другорядних, зрозуміти ставлення об'єкта до його оточенню і визначити його призначення.

Material design грунтується і на принципах друкованого дизайну. І не тільки для краси, а й для розстановки акцентів і фокусування уваги користувача на потрібному елементі, для спрощення навігації серед ієрархії конструкцій інтерфейсу, для інтуїтивної передачі їх сенсу.[11]

Перехід до конкретних вкладок реалізован за допомогою фрагментів та відповідних макетів, тобто кожен екран інтерфейсу розроблений окремо. Для розмітки кожного макету використовувалась мова XML.

В меню доступні наступні вкладки:

- Вхід – можливість авторизуватися вже зареєстрованому користувачу;
- Реєстрація – можливість створити новий аккаунт, необхідно ввести потрібні дані (логін/e-mail, пароль);
- Шифрування – можливість за вибраним алгоритмом зашифрувати/розшифрувати файл;
- Блокування – можливість заблокувати файл, необхідно ввести пароль;

Профіль програмного додатку зображене на рис. 3.5.

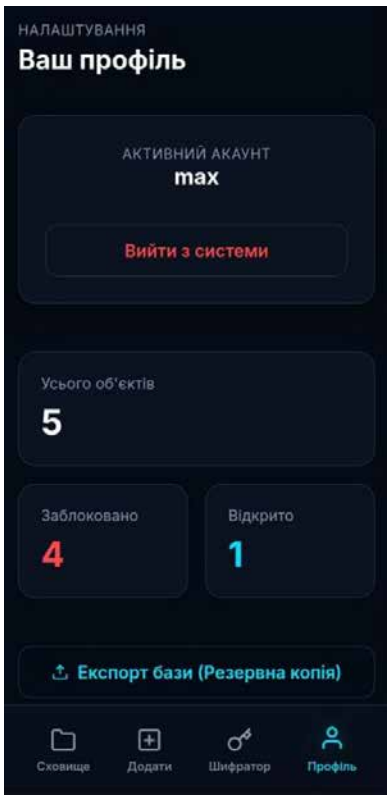

Рис. 3.5 Профіль

Екрани авторизації та реєстрації зображені на рисунка 3.6 - 3.7 відповідно.

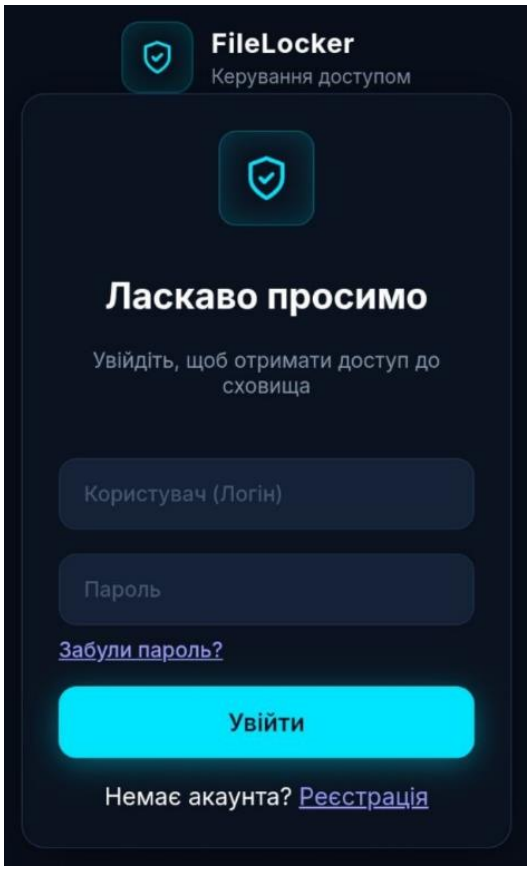

Рис. 3.6 Екран авторизації

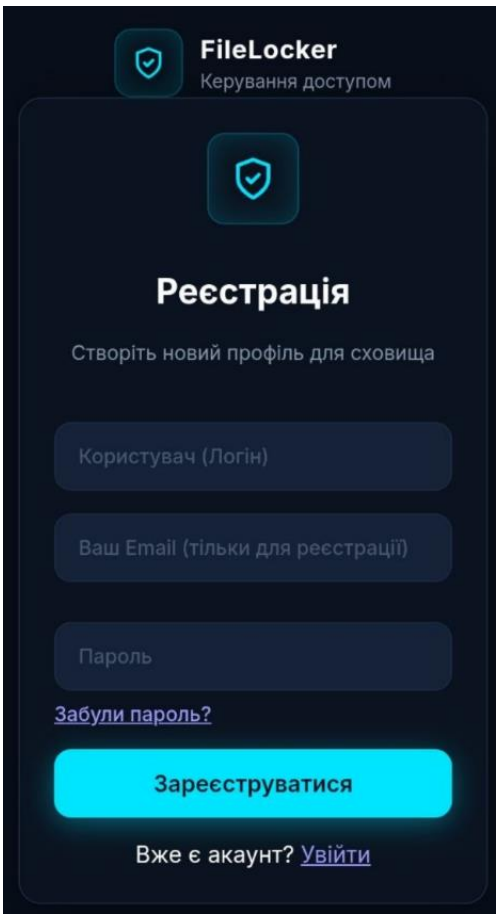

Рис. 3.7 Екран реєстрації

3.3.3 Забезпечення інтерфейсу з базою даних. Зі сторони програми для взаємодії з серверною частиною використовується бібліотека Volley.

Volley це бібліотека, яка робить мережеві додатки для Android простіше і, найголовніше, швидше. Вона управляє обробкою і кешуванням мережевих запитів, і це заощаджує дорогоцінний час розробників від написання одного і того ж коду мережевого запиту / зчитування з кеша знову і знову. Volley автоматично складає всі мережеві запити. Volley буде приймати на себе всі мережеві запити вашого застосування виконувати їх для вилучення відповіді або зображення з веб-сайтів. Бібліотека забезпечує потужне API для скасування запиту. Можна скасувати один запит або встановити кілька запитів для скасування.[12]

Для підключення бібліотеки до проекту потрібно потрібно додати у файл build.gradle залежність наведену на рис. 3.8.

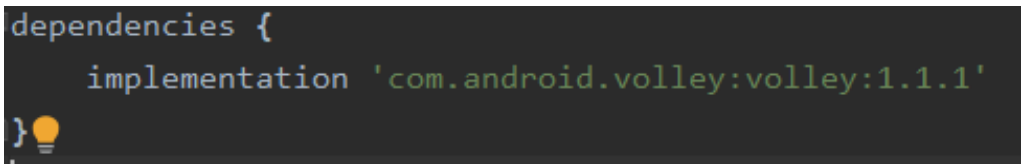

Рис. 3.8 Залежність для бібліотеки Volley

Інтерфейс з базою даних реалізован за допомогою PHP – скриптів, які обробляють клієнтські запити на серверній частині. Усі SQL – запити які передаються на сервер функціями бібліотеки Volley, приймає певний PHP – скрипт та надсилає готовий запит на сервер БД. Для встановлення з'єднання з БД виконується скрипт зображений на рис. 3.9.

Рис. 3.9 Скрипт для з'єднання з БД «protect»

Основні PHP – скрипти для взаэмодії з БД знаходяться у додатку Б.

3.3.4 Формування звітної інформації. Звітна інформація формується у вигляді списку файлів, які вже використовувалися додатком та відображається назва та статус файлу. PHP – скрипт, який оброблює запит на вибірку даних зображений на рис. 3.10.

```php
<?php
$login = filter_input(INPUT_POST, "login");

require_once 'connect.php';

$result = array();
$result['data'] = array();

$sql = "select name, status from object where login ='$login'";
$response = mysqli_query($conn, $sql);

while ($row = mysqli_fetch_array($response)) {

    $index['name'] = $row[0];
    $index['status'] = $row[1];

    array_push($result['data'], $index);
    # code...
}

$result["success"] = "1";
echo json_encode($result);
mysqli_close($conn);

?>
```

Рис. 3.10 Скрипт для запиту на вибірку даних про файли

Скрипт отримує значення Логіну користувача та створює SQL – запит на виведення назви та статусу файлів, які належать саме цьому користувачу. Отримує масив значень та повертає до клієнтської частини програми. На клієнтській стороні масив оброблюється та виводиться у звичній для List View формі. Для коректного відображення списку файлів було розроблено власний адаптер, код якого зображно на рис. 3.11.

```
    class MyAdapter extends ArrayAdapter<File>{
        Context context;
        List<File> arraylistFile;
        MyAdapter(Context c, List<File> arraylistFile){
            super(c,R.layout.row,arraylistFile);
            this.context = c;
            this.arraylistFile = arraylistFile;
        }
        @NonNull
        @Override
        public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
            View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.row,  root: null,  attachToRoot: true);
            TextView myTitle = view.findViewById(R.id.titleV);
            TextView mySubtitle = view.findViewById(R.id.subtitle);
            myTitle.setText(arraylistFile.get(position).getName());
            mySubtitle.setText(arraylistFile.get(position).getStatus());
            return view;
        }
    }
}
```

Рис. 3.11 Код створення адаптеру для відображення списку файлів

Результат виконання відображення списку файлів зображено на рис. 3.12.

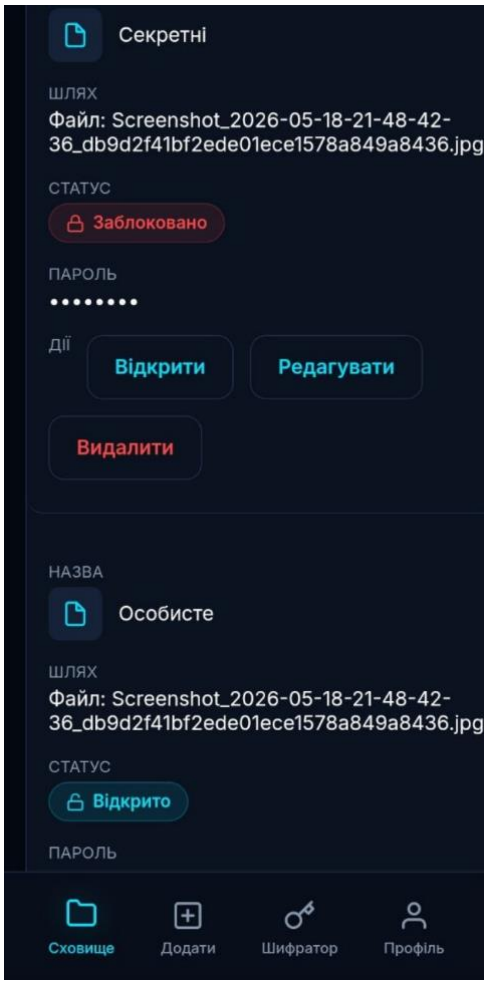

Рис. 3.12 Список файлів використовуваних програмою

4. РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Діаграма розгортання

Діаграма розгортання - діаграма на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонентів. Діаграма розгортання відображає робочі екземпляри компонентів, а діаграма компонентів, натомість, відображає зв'язки між типами компонентів. Діаграма розгортання в UML моделює фізичне розгортання артефактів на вузлах. Вузли представляються, як прямокутні паралелепіпеди з артефактами, розташованими в них, зображеними у вигляді прямокутників. Вузли можуть мати підвузли, які представляються, як вкладені прямокутні паралелепіпеди. Один вузол діаграми розгортання може концептуально представляти безліч фізичних вузлів, таких, як кластер серверів баз даних.

Існує два типи вузлів:

- вузол пристрою;
- вузол середовища виконання.

Вузли пристроїв - це фізичні обчислювальні ресурси зі своєю пам'яттю і сервісами для виконання програмного забезпечення, такі як звичайні ПК, мобільні телефони. Вузол середовища виконання - це програмний обчислювальний ресурс, який працює всередині зовнішнього вузла і який надає собою сервіс, який виконує інші виконувані програмні елементи.[13]

Діаграма розгортання інформаційної системи системи захисту файлів представлена на Плакаті 3.

4.2 Тестування системи

Тестування програмного забезпечення – це процес перевірки відповідності заявлених до продукту вимог і реально реалізованої функціональності, здійснюваний шляхом спостереження за його роботою в штучно створених ситуаціях і на обмеженому наборі тестів, обраних певним чином. [14]

При завантаженні додатку з’являється екран авторизації для вже існуючого користувача. Тестування будемо проводити з аккаунта користувача «alex» (рис. 4.1).

Рис. 4.1 Дані користувача «maksym»

При спробі авторизуватися, за допомогою скрипту перевіряється наявність даного користувача у системі. У разі введення неіснуючого користувача або невірного паролю, програма видасть помилку (рис. 4.2)

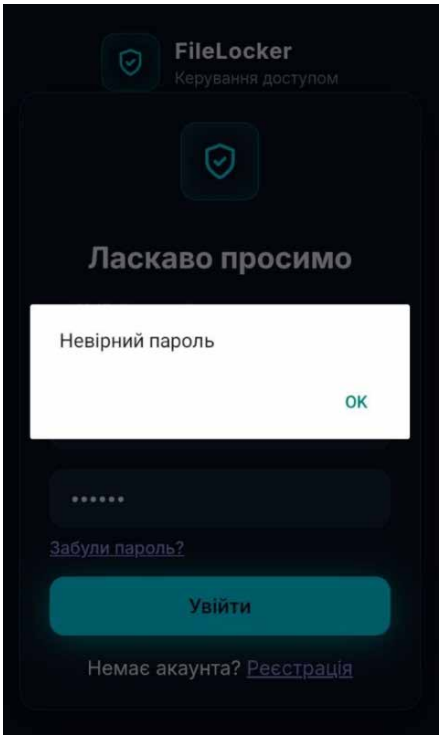

Рис. 4.2 Введено некоректні дані

Після успішного проходження етапу авторизації програма запам’ятовує авторизованого користувача та з’являється екран вітання (рис.4.3).

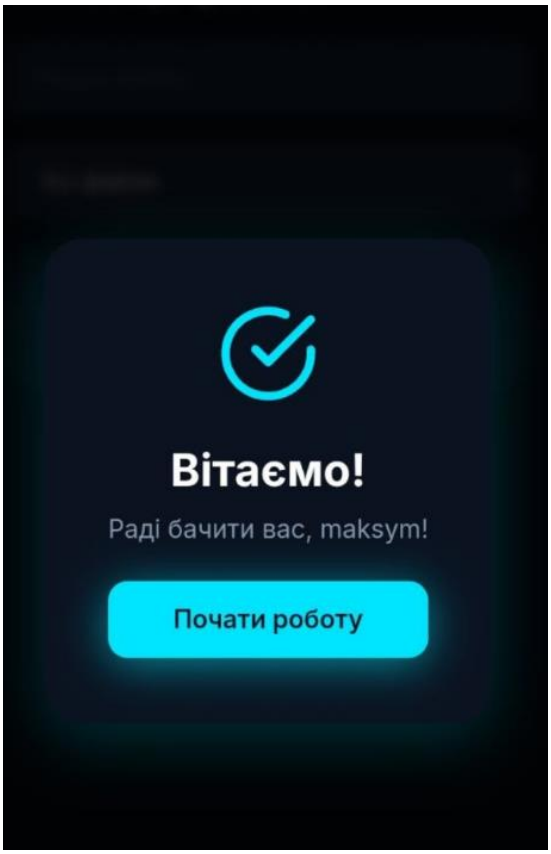

Рис. 4.3 Успішна авторизація

Для проходження реєстрації потрібно ввести логін, пароль та email (рис. 4.5). Після успішного процесу реєстації користувача програма перенаправляє на екран авторизації. В системі з’являється новий користувач (рис. 4.4).

Рис. 4.4 Успішна реєстрація

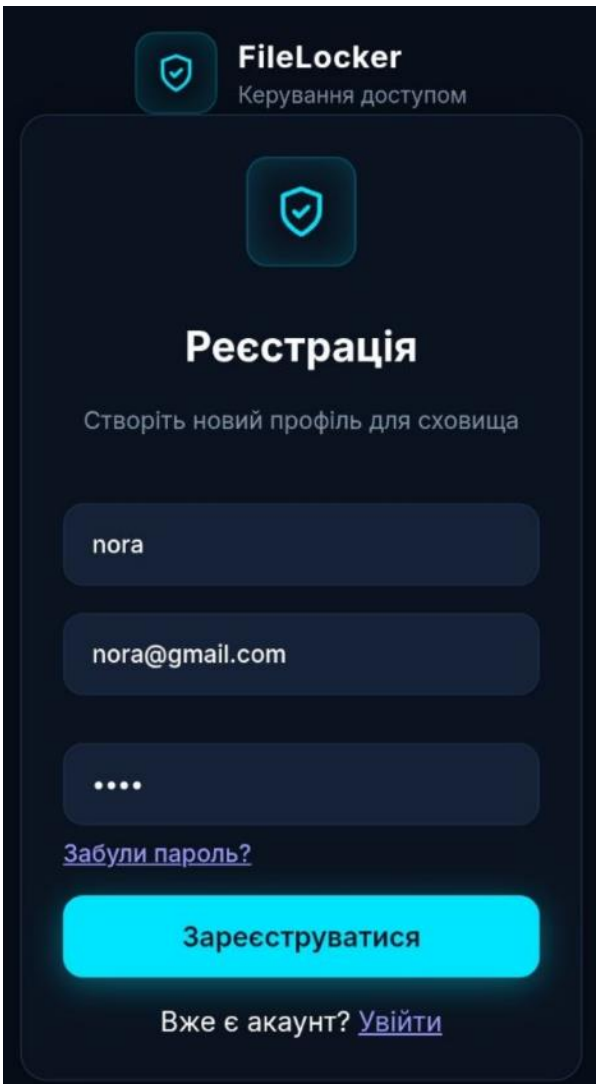

Рис. 4.5 Реєстрація нового користувача

На вкладці «Шифрування» програма пропонує обрати потрібний файл для здійснення шифрування (рис. 4.6) та алгоритм (рис. 4.7), за яким відбудеться процес захисту.

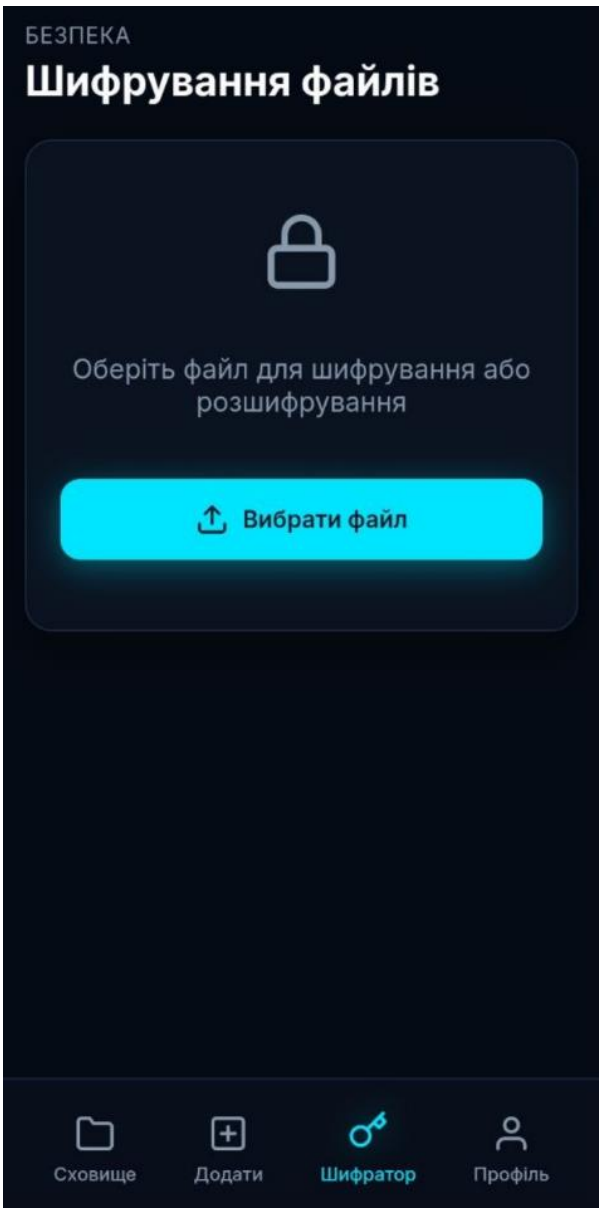

Рис. 4.6 Вибір файлу для шифрування

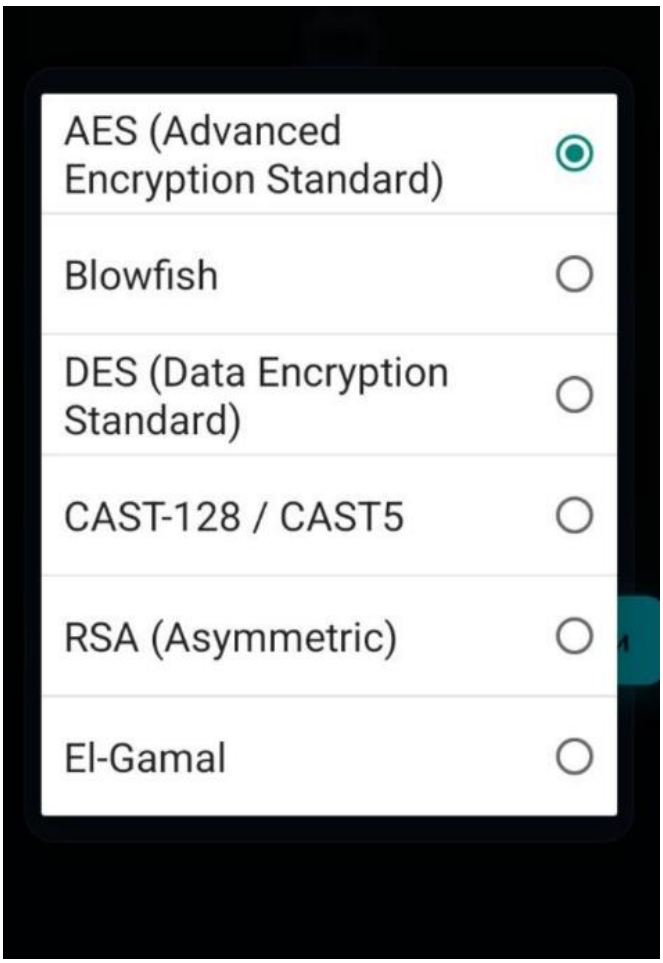

Рис. 4.7 Вибір алгоритму шифрування

Для шифрування алгоритмом AES був обраний файл фрази.txt. Вміст цього файлу зображено на рис. 4.8.

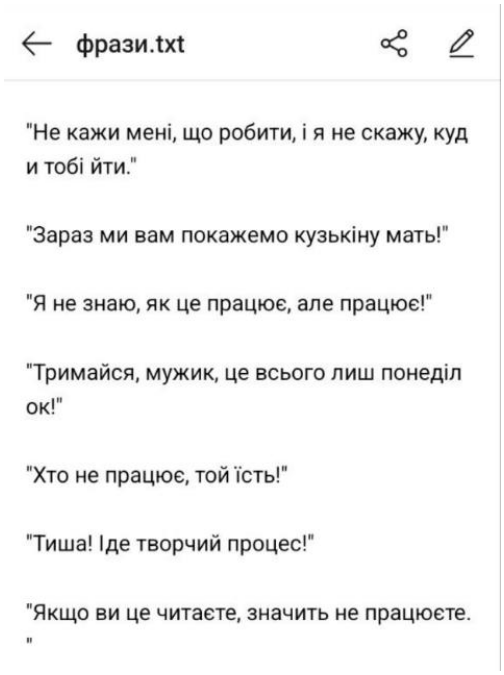

Рис. 4.8 Вміст файл «фрази.txt» до шифрування

Після натискання кнопки «Зашифрувати» виконується скрипт, що перевіряє наявність даного файлу у БД, якщо файл існує – статус цього файлу змінюється на «Зашифрований», якщо ні – додається файл у БД. Якщо файл зашифрований його назва містить «[e]», що свідчить про його статус. Вміст зашифрованого файлу зображений на рис. 4.9.

Рис. 4.9 Вміст зашифрованого файлу

Для текстових файлів, розмір яких перевищує 10 МБ, даний алгоритм не є найоптимальнішим, оскільки, в зашифрованому вигляді буде займати дуже великий об’єм.

Після натискання на кнопку розшифрування, перевіряється чи даний файл був зашифрований, відбувається процес розшифрування та виконується скрипт на зміну статусу в БД. Після розшифрування вміст файлу має вигляд, як показано на рис. 4.10.

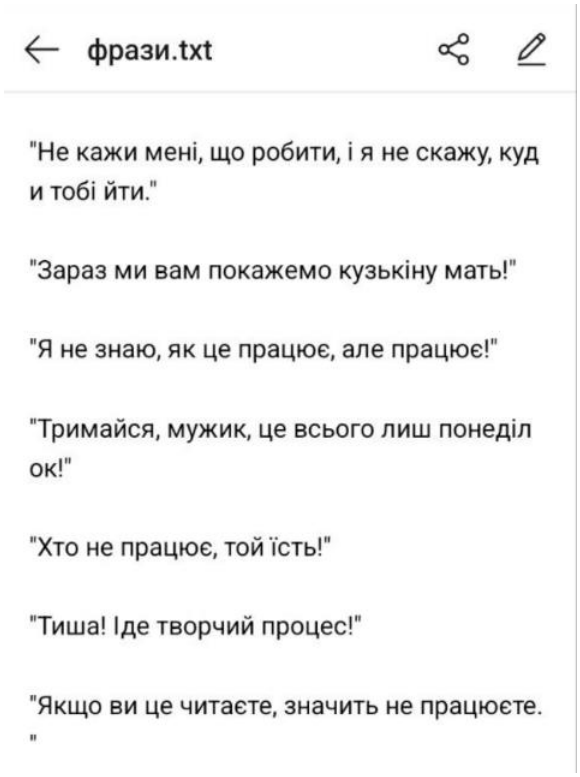

Рис. 4.10 Вміст файлу після розшифрування

Як було показано на прикладі, вміст розшифрованого файлу співпадає зі змістом початкового, тому процеси шифрування та дешифрування працюють правильно.

Для того щоб заблокувати файл, потрібно обрати сам файл та встановити паролі. Програма перевіряє співпадіння введених паролів, а також наявність вибраного файлу у БД. Якщо файл вже знаходиться у БД в заблокованому вигляді, програма, за допомогою скрипту, перевіряє відповідність введнего паролю та паролю, що знаходиться в БД. При співпадінні цих паролів програма дозволяє зробити розблокування файлу.

Для прикладу заблокуємо файл Registration.txt (рис. 4.11).

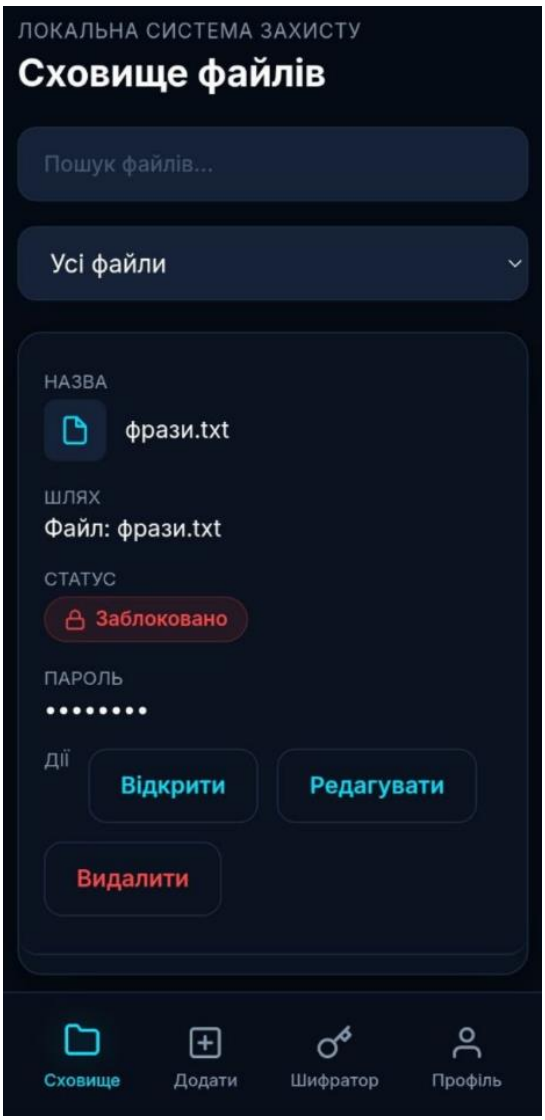

Рис. 4.11 Блокування файлу

Після блокування файлу він становиться недоступним для перегляду для файлових менеджерів (рис 4.12), тому розблокування файлу відбувається тільки з додатку.

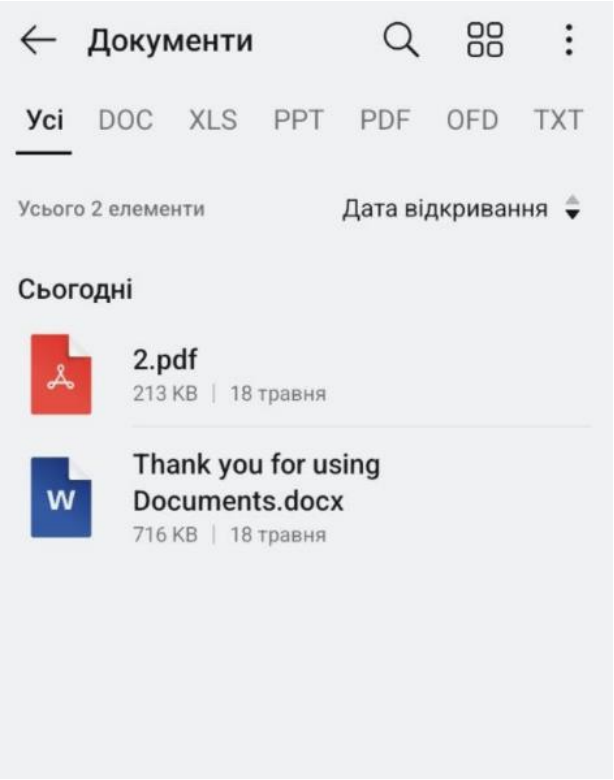

Рис. 4.12 Перегляд файлів після блокування

Розблокування файлу відбувається тільки з додатку при правильно введеному паролю.

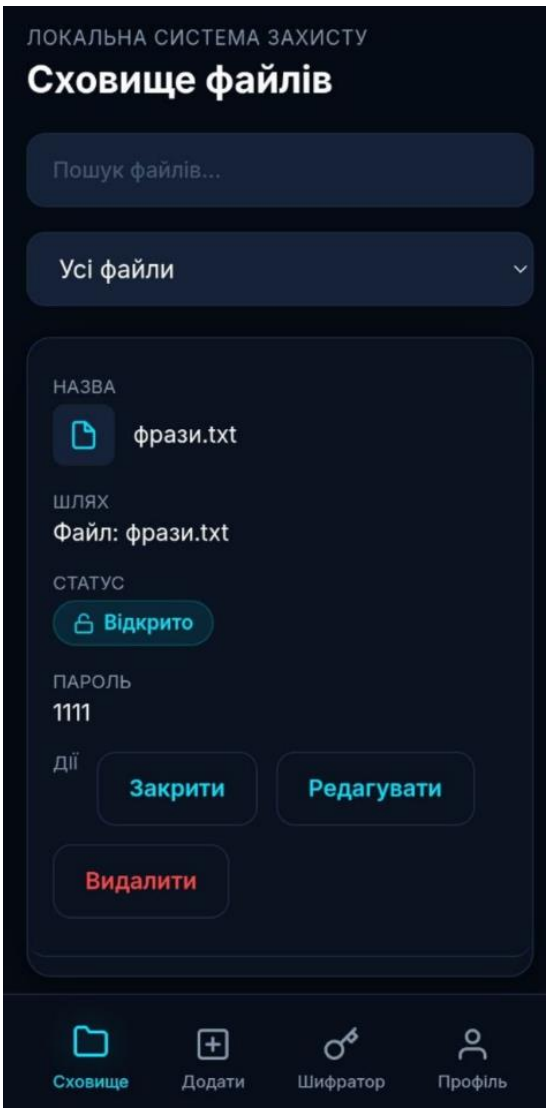

Рис. 4.13 Розблокування файлу

Після розблокування файлу, він знову стає доступним для перегляду з файлового менеджеру (рис. 4.14).

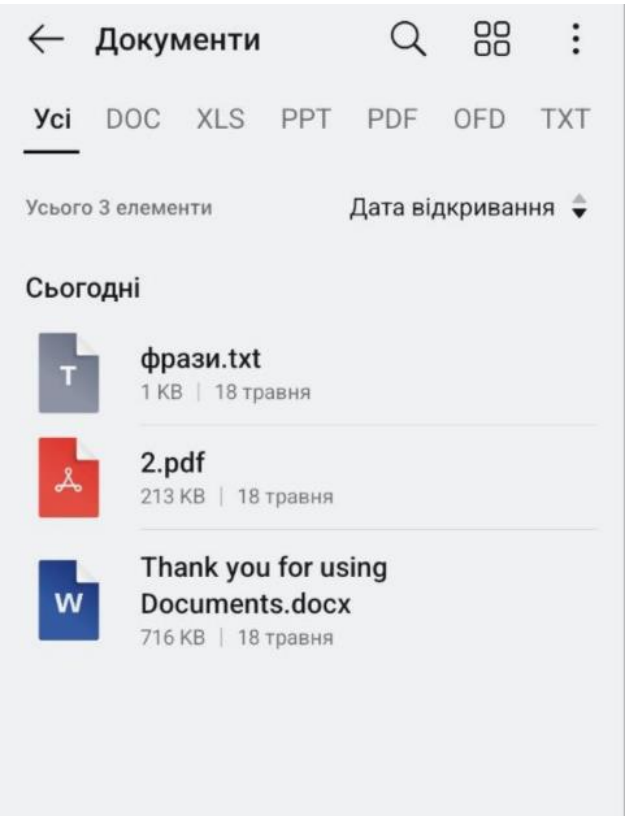

Рис. 4.14 Перегляд файлів після розблокування

4.3 Вимоги до апаратного та програмного забезпечення

4.3.1 Вимоги до апаратного забезпечення. Для коректного виконання програми рекомендується наступне **апаратне забезпечення:**

- Процесор – мінімальна частота 1.5 ГГц;
- Кількість ядер процесору – мінімальна 2;
- ОЗУ – мінімум 2 ГБ;
- Вбудована пам’ять – мінімум 3 ГБ вільної;
- Інтернет з’єднання – швидкість мережі від 20 Мбіт/с.

4.3.2 Вимоги до програмного забезпечення. Для інсталяції та запуску додатку необхідне наступне **програмне забезпечення**:

- ОС Android – мінімальна версія 4.4;
- SDK – мінімальна версія 19.

4.4 Діаграма пакетів

Діаграми пакетів забезпечують відмінну можливість візуального представлення залежностей між частинами вашої системи і часто використовуються для діагностики або визначення порядку компіляції. В пакетах можуть групуватися практично будь-які елементи UML (в тому числіі самі пакети). Існують два різні способи показу елементів, що входять в пакет. У першому варіанті елементи малюються всередині великого прямокутника. У другому варіанті записи пакет з'єднується з кожним містяться в ньому елементом суцільною лінією. [15]

Діаграма пакетів підсистеми захисту інформації на мобільних пристроях на базі ОС Android зображена на Плакаті 4.

На діаграмі зображено два основних пакети – Client та ServerDB

Пакет **Client** містить в собі 3 підпакети:

- Algorithms – бібліотека файлів з алгоритмами шифрування;
- ClientGUI – необхідні компоненти користувацького інтерфейсу;
- ClientNetworkWebServer – набір компонентів необхідних для взаємодії з сервером;

Пакет **WebServer** містить в собі 3 підпакети:

- ServerBusinessLogic – необхідні дані для реалізації бізне-логіки серверу;
- RequestDB – необхідні компоненти для реалізації доступу і логіки запитів до БД;
- ServerNetworkWebServer – набір компонентів необхідних для взаємодії з клієнтом.

Пакет **ServerDB** містить в собі підпакет Datebase в якому знаходяться дані БД.

ВИСНОВКИ

Результатом виконання курсової роботи є розроблена та спроектована підсистема захисту інформації на мобільних пристроях на базі ОС Android.

Для функціонування системи на базі ОС Android було розроблено:

- програмні засоби шифрування файлів;
- реалізовані алгоритми блокування файлів;
- інтерфейс з базою даних за допомогою PHP – скриптів;
- простий та зрозумілий користувацький інтерфейс;
- модулі авторизації та реєстрації;
- клієнт-серверну архітектуру додатку.

У першому розділі було проведено попередній аналіз предметної області, моделювання системи за допомогою уніфікованої мови UML.

Були використані засоби моделювання UML, такі як:

- діаграма прецедентів – для відображення відношень між акторами та прецедентами в системі;
- діаграма послідовності - відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність відправлених повідомлень;
- діаграма пакетів — відображає залежності між пакетами, з яких і складається модель;
- діаграма розгортання - відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах.

Другий розділ присвячений інформаційному забезпеченню системи. Був обгрунтован вибів засобів СУБД, виходячи з певних критеріїв, розроблено логічну модель даних, а також були створенні SQL – запити на створення відповідних таблиць та тригерів.

Третій розділ присвячений розробці програмного забезпечення та вибору інструментів для розробки. Для реалізації програмного забезпечення було використано IDE Android Stuidio. Для реалізації інтерфейсу з базою даних вмкористовуються на стороні користувача бібліотека Volley, на стороні серверу PHP – скрипти. Для реалізації інтерфейсу користувача було використано графічний редактор Adobe Photoshop CS6 та мову XML. Дизайн програми був розроблений за методико Material design.

Четвертий розділ присвячений впровадженню та дослідній експлуатації програмної системи. Визначено програмні та апаратні вимоги до мобільного девайсу. Проведена демонстрація тестування програмного продукту.

Отримана модель може бути використана для створення програмного продукту для підвищення рівня захисту мобільного девайсу та зменшити вірогідність стати жертвою зловмисників. Отже, можна зробити висновок, що мета та завдання дипломного проекту були виконані.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Android platform documentation [Електронний ресурс]. – Android Developers, 2026. – Режим доступу: https://developer.android.com/about – Дата звернення: 17.05.2026.

2 Android Studio documentation [Електронний ресурс]. – Android Developers, 2026. – Режим доступу: https://developer.android.com/studio – Дата звернення: 17.05.2026.

3 Browser Market Share Worldwide [Електронний ресурс]. – StatCounter Global Stats, 2026. – Режим доступу: https://gs.statcounter.com/browser-market-share – Дата звернення: 17.05.2026.

4 What is encryption? [Електронний ресурс]. – Cloudflare Learning Center, 2026. – Режим доступу: https://www.cloudflare.com/learning/ssl/what-is-encryption/ – Дата звернення: 17.05.2026.

5 Unified Modeling Language Specification [Електронний ресурс]. – Object Management Group, 2026. – Режим доступу: https://www.omg.org/spec/UML/ – Дата звернення: 17.05.2026.

6 UML Sequence Diagram Guide [Електронний ресурс]. – Visual Paradigm, 2026. – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/ – Дата звернення: 17.05.2026.

7 Apache HTTP Server Version 2.4 Documentation [Електронний ресурс]. – The Apache Software Foundation, 2026. – Режим доступу: https://httpd.apache.org/docs/current/ – Дата звернення: 17.05.2026.

8 MySQL 8.4 Reference Manual [Електронний ресурс]. – Oracle, 2026. – Режим доступу: https://dev.mysql.com/doc/refman/8.4/en/ – Дата звернення: 17.05.2026.

9 MySQL Workbench Manual: Database Design and Modeling [Електронний ресурс]. – Oracle, 2026. – Режим доступу:

https://dev.mysql.com/doc/workbench/en/wb-data-modeling.html – Дата звернення: 17.05.2026.

10 Material Design 3 [Електронний ресурс]. – Google, 2026. – Режим доступу: https://m3.material.io/ – Дата звернення: 17.05.2026.

11 Volley overview [Електронний ресурс]. – Google Open Source, 2026. – Режим доступу: https://google.github.io/volley/ – Дата звернення: 17.05.2026.

12 FIPS 197: Advanced Encryption Standard (AES) [Електронний ресурс]. – NIST Computer Security Resource Center, 2023. – Режим доступу: https://csrc.nist.gov/pubs/fips/197/final – Дата звернення: 17.05.2026.

13 OWASP Web Security Testing Guide [Електронний ресурс]. – OWASP Foundation, 2026. – Режим доступу: https://owasp.org/www-project-web-security-testing-guide/– Дата звернення: 17.05.2026.

14 Component Diagram syntax and features [Електронний ресурс]. – PlantUML, 2026. – Режим доступу: https://plantuml.com/component-diagram – Дата звернення: 17.05.2026.

15 Deployment Diagram syntax and features [Електронний ресурс]. – PlantUML, 2026. – Режим доступу: https://plantuml.com/deployment-diagram – Дата звернення: 17.05.2026.

16 Client-server overview [Електронний ресурс]. – MDN Web Docs, 2026. – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/Pages_sites_servers_and_search_engines – Дата звернення: 17.05.2026.

17 Security tips for Apache HTTP Server [Електронний ресурс]. – The Apache Software Foundation, 2026. – Режим доступу: https://httpd.apache.org/docs/current/misc/security_tips.html – Дата звернення: 17.05.2026.

18 Android app quality guidance [Електронний ресурс]. – Android Developers, 2026. – Режим доступу: https://developer.android.com/docs/quality-guidelines – Дата звернення: 17.05.2026.

19 PHP and MySQL web development documentation [Електронний ресурс]. – W3Schools, 2026. – Режим доступу: https://www.w3schools.com/php/php_mysql_intro.asp – Дата звернення: 17.05.2026.

20 Mobile app security verification standard [Електронний ресурс]. – OWASP MASVS, 2026. – Режим доступу: https://mas.owasp.org/MASVS/ – Дата звернення: 17.05.2026.

ДОДАТОК А

СТВОРЕННЯ ТАБЛИЦЬ ТА ТРИГЕРІВ

Сторінок – 2

КИЇВ – 2026

```
CREATE TABLE `user` (
  `id_user` int(11) NOT NULL,
  `login` varchar(30) NOT NULL,
  `password` varchar(30) NOT NULL,
  `email` varchar(45) NOT NULL,
  PRIMARY KEY (`id_user`)
)

CREATE TABLE `password` (
  `id_pass` tinyint(4) NOT NULL AUTO_INCREMENT,
  `text` varchar(30) NOT NULL,
  `length` tinyint(4) NOT NULL,
  `id_obj` int(11) NOT NULL,
  PRIMARY KEY (`id_pass`),
  KEY `fk_obj_id_idx` (`id_obj`),
  CONSTRAINT `fk_obj_id` FOREIGN KEY (`id_obj`) REFERENCES `object`
(`id_obj`)
)

CREATE TABLE `object` (
  `id_obj` int(11) NOT NULL,
  `name` varchar(30) NOT NULL,
  `size` int(11) NOT NULL,
  `path` varchar(100) NOT NULL,
  `status` enum('free','block','unblock','encrypted','decrypted') NOT NULL,
  `id_user` int(11) NOT NULL,
  `id_alg` tinyint(4) NOT NULL,
  PRIMARY KEY (`id_obj`),
  KEY `fk_user_id_idx` (`id_user`),
  KEY `fk_alg_id_idx` (`id_alg`),
  CONSTRAINT `fk_alg_id` FOREIGN KEY (`id_alg`) REFERENCES
`algorithm` (`id_alg`),
  CONSTRAINT `fk_user_id` FOREIGN KEY (`id_user`) REFERENCES `user`
(`id_user`)
)

CREATE TABLE `algorithm` (
  `id_alg` tinyint(4) NOT NULL,
  `name` varchar(45) NOT NULL,
  PRIMARY KEY (`id_alg`)
)

CREATE TABLE `log` (
  `id_log` int(11) NOT NULL AUTO_INCREMENT,
```

```
`datetime` timestamp NOT NULL,
  `action` varchar(100) NOT NULL,
  `id_obj` int(11) NOT NULL,
  `id_user` int(11) NOT NULL,
  PRIMARY KEY (`id_log`),
  KEY `fk_obj_id_idx` (`id_obj`),
  KEY `fk_user_id_log_idx` (`id_user`),
  CONSTRAINT `fk_obj_id_log` FOREIGN KEY (`id_obj`) REFERENCES `object` (`id_obj`),
  CONSTRAINT `fk_user_id_log` FOREIGN KEY (`id_user`) REFERENCES `object` (`id_user`)
)

CREATE DEFINER=`root`@`localhost` TRIGGER `object_AFTER_INSERT` AFTER INSERT ON `object` FOR EACH ROW BEGIN
	INSERT into log Set action ='insert', id_obj = NEW.id_obj, id_user = New.id_user;
END

CREATE DEFINER=`root`@`localhost` TRIGGER `object_AFTER_UPDATE` AFTER UPDATE ON `object` FOR EACH ROW BEGIN
	INSERT into log Set action ='update', id_obj = NEW.id_obj, id_user = New.id_user;
END

CREATE DEFINER=`root`@`localhost` TRIGGER `del_trigg_obj` BEFORE DELETE ON `object` FOR EACH ROW BEGIN
	INSERT into log Set action = concat('deleted ', old.id_obj, ' by userid ',old.id_user);
   DELETE FROM password WHERE id_obj = OLD.id_obj;
   DELETE FROM log WHERE id_obj = OLD.id_obj;
END
```

ДОДАТОК Б

ОСНОВНІ PHP - СКРИПТИ

Сторінок – 4

КИЇВ – 2026

Скрипт для логінізації

```
<?php
$login = filter_input(INPUT_POST, "login");
$password = filter_input(INPUT_POST, "password");
require_once 'connect.php';
$sql = "select * from user where login ='$login'";
$response = mysqli_query($conn, $sql);
$result = array();
$result['login'] = array();
if ( mysqli_num_rows($response) == 1) {
        $row = mysqli_fetch_assoc($response);
        if ( $password == $row['password']) {
                $index['login'] = $row['login'];
                $index['email'] = $row['email'];
                array_push($result['login'], $index);
                $result['success'] = "1";
                $result['message'] = "success";
                echo json_encode($result);
                mysqli_close($conn);
        }else{
                $result['success'] = "0";
                $result['message'] = "error";
                echo json_encode($result);
                mysqli_close($conn);
        }}?>
```

Скрипт для додавання та зміни файлів

```
<?php
$name = $_POST["name"];
$path = $_POST["path"];
$status = $_POST["status"];
$user = $_POST["user"];
require_once 'connect.php';
$sql1 = "SELECT Count(id_obj) FROM object WHERE name = '$name'";
$result1 = mysqli_query($conn, $sql1);
if ($result1) {
	$sql = "UPDATE object SET status = '$status' WHERE name = '$name'";
	$result = mysqli_query($conn, $sql);
	if ($result) {
	echo "Data Updated";
}else{
	echo "Failed";
}}else{
$sql2 = "insert into object(name, path, status, login, id_alg) values ('$name', '$path', '$status', '$user', '1')";
$result2 = mysqli_query($conn, $sql2);
if ($result) {
	echo "Data Inserted";
}else{
	echo "Failed";}}
mysqli_close($conn);?>
```

Скрипт для вибору даних про файл

```
<?php
$login = filter_input(INPUT_POST, "login");
require_once 'connect.php';
$result = array();
$result['data'] = array();
$sql = "select name, status from object where login ='$login'";
$response = mysqli_query($conn, $sql);
while ($row = mysqli_fetch_array($response)) {
        $index['name'] = $row[0];
        $index['status'] = $row[1];
        array_push($result['data'], $index);
        # code...
}
$result["success"] = "1";
echo json_encode($result);
mysqli_close($conn);
?>
```

Скрипт для реалізації блокування файлів

```
<?php
$name = $_POST["name"];
$path = $_POST["path"];
$status = $_POST["status"];
$user = $_POST["user"];
$password =$_POST["password"];
require_once 'connect.php';
$sql1 = "SELECT Count(*) FROM object WHERE name = '$name'";
$result1 = mysqli_query($conn, $sql1);
if ($result1) {
	$sql2 = "insert into object(name, path, status, login) values ('$name', '$path', '$status', '$user')";
$result2 = mysqli_query($conn, $sql2);
$sql3 = "insert into password(text, id_obj) values ('$password', (select id_obj from object where name = '$name'))";
$result3 = mysqli_query($conn, $sql3);
if ($result2 OR $result3) {
	echo "Data Inserted";
}else{echo "Failed";}
}else{	$sql = "UPDATE object SET status = '$status' WHERE name = '$name'";
	$result = mysqli_query($conn, $sql);
	if ($result) {echo "Data Updated";
}else{echo "Failed";
}}mysqli_close($conn);?>
```

ДОДАТОК В

ПЛАКАТИ

Сторінок – 4

КИЇВ – 2026р.