НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)

“___” _____2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп’ютерні системи та компоненти»

зі спеціальності **123 «Комп'ютерна інженерія»**

на тему: Android-месенджер з криптографічним захистом

Виконав: студент IV курсу, групи КВ-62

Додяк Дмитро Іванович _____ _____
(підпис)

Керівник ст. викл. Дробязко І.П._____ _____
(підпис)

Консультант з нормоконтролю, доц. каф. СПСКС, к.т.н. Клятченко Я.М._____
(підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»

Освітньо-професійна програма «Комп’ютерні системи та компоненти»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___»_____2019 р.

ЗАВДАННЯ
на дипломний проєкт студента
Додяка Дмитра Івановича

1. Тема проєкту «Android-месенджер з криптографічним захистом», керівник проєкту Дробязко І.П, старший викладач, затверджені наказом по університету від «___»_____ 2020 р. №_____

2. Термін подання студентом проєкту: 25 травня 2020 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - аналіз існуючих Android-месенджерів з криптографічним захистом;
   - аналіз технологій розроблення Android месенджерів;
   - аналіз і вибір засобів реалізації криптографічного захисту;
   - розробка Android-месенджера з криптографічним захистом

5. Перелік графічного матеріалу
   - Структурна схема Android-месенджера.
   - Алгоритм взаємодії з користувачем

- Алгоритм шифрування програмного коду
- Діаграма класів проєкту.
- Презентація за темою роботи.

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., доцент		

7. Дата видачі завдання: “30” жовтня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вивчення літератури за тематикою проєкту	18.11.2019	
2.	Розроблення та узгодження технічного завдання	30.11.2019	
3.	Аналіз існуючих рішень	10.01.2020	
4.	Підготовка матеріалів першого розділу проєкту	18.01.2020	
5.	Підготовка матеріалів другого розділу проєкту	16.02.2020	
6.	Підготовка матеріалів третього розділу проєкту	14.03.2020	
7.	Підготовка графічної частини	01.04.2020	
8.	Оформлення документації дипломного проєкту	20.04.2020	
9.	Попередній огляд матеріалів проєкту на кафедрі	01.05.2020	

Студент _____ Дмитро ДОДЯК

Керівник проєкту _____ Ірина ДРОБЯЗКО

*Консультантом не може бути зазначено керівника дипломного проєкту.

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (51 с., 8 рис., 2 табл., 4 додатки).

Об’єкт розробки – Android-месенджер з криптографічним захистом, що забезпечує захист даних користувача.

Основні характеристики додатку:

- безпечна реєстрація та логінізація;
- захищений програмний код;

В системі передбачені механізми захисту декомпіляції коду. Для розробки використано мову програмування Java з використанням фреймворку Android та середовище розробки JetBrains Android. У якості шифратора обрано шифратор Allatori. У якості бази даних та серверу використано Firebase.

В ході виконання дипломного проекту:

- проведено аналіз існуючих рішень;
- визначено архітектуру системи;
- створено Android-месенджер.

Впровадження цього додатку дозволить забезпечити приватність даних користувачів при передаванні повідомлень.

Ключові слова: АНДРОЇД ДОДАТОК, ПЕРЕДАЧА ПОВІДОМЛЕНЬ, ШИФРУВАННЯ, ЗАХИСТ КОДУ, JAVA, ANDROID FRAMEWORK, JETBRAINS ANDROID STUDIO, MESSENGER, FIREBASE.

ANNOTATION

Qualification work includes an explanatory note (51 pages, 8 figures, 2 tables, 4 appendices).

The object of development is an Android messenger with cryptographic protection, which provides protection of user data.

The main characteristics of the application:

- secure registration and login;
- protected program code;

The system provides mechanisms for protecting code decompilation. The Java programming language using the Android framework and the JetBrains Android development environment were used for development. The Allatori encoder is selected as the encoder. Firebase is used as the database and server.

During the implementation of the diploma project:

- the analysis of existing decisions;
- the architecture of the system is defined;
- Android messenger created.

The implementation of this application will ensure the privacy of user data.

Keywords: ANDROID APPENDIX, MESSAGE TRANSMISSION, ENCRYPTION, CODE PROTECTION, JAVA, ANDROID FRAMEWORK, JETBRAINS ANDROID STUDIO, MESSENGER, FIREBASE.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ. 467100.002 ТЗ	Android-месенджер	4		
			з криптографічним			
			захистом.			
			Технічне завдання			
	А4	ІАЛЦ. 467100.003 ТП	Android-месенджер	2		
			з криптографічним			
			захистом.			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ. 467100.004 ПЗ	Android-месенджер	51		
			з криптографічним			
			захистом.			
			Пояснювальна записка			
	А4	ІАЛЦ. 467100.005 Д1	Android-месенджер	1		
			з криптографічним			
			захистом.			
			Схема структурна			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ. 467100.001 ОА		
Розробив		Додяк Д.І.			Android-месенджер з криптографічним захистом Опис альбому	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Дробязко І.П.					
Н. контроль		Клятченко Я.М.				НТУУ "КПІ" ФПМ КВ-62	
Затвердив		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ. 467100.006 Д2	Android-месенджер	1		
			з криптографічним			
			захистом.			
			Взаємодія з користувачем.			
			Схема алгоритму.			
	А4	ІАЛЦ. 467100.007 Д3	Android-месенджер	1		
			з криптографічним			
			захистом.			
			Шифрування коду.			
			Схема алгоритму.			
	А4	ІАЛЦ. 467100.008 Д4	Android-месенджер	1		
			з криптографічним			
			захистом.			
			Діаграма класів.			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал			
			Презентація проєкту			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ. 467100.001 ОА	Арк. 2

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Найменування роботи – «Android-месенджер з криптографічним захистом».

Область застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування та спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є розробка Android месенджеру з криптографічним захистом.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- швидкий користувацький інтерфейс;
- безпечна автентифікація і авторизація користувачів;
- можливість відновлення паролю;
- можливість шифрування повідомлень перед відправоенням
- захист програмного коду за допомогою криптографії;
- реалізація повного функціоналу месенджера;
- безпечне кешування даних.

5.2. Вимоги до апаратного забезпечення

- Процесор: 4-ядерний процесор;
- Оперативна пам’ять: 32Гб;
- Наявність доступу до мережі Internet (Ethernet);

5.3. Вимоги до програмного та апаратного забезпечення користувача

- Операційна система Mac OS X.
- Комплект розробника застосунків Java Development Kit 8.
- Шифратор коду Allatori
- База даних Firebase.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	15.11.2019
2.	Розроблення та узгодження технічного завдання	30.11.2019
3.	Аналіз існуючих рішень	05.02.2020
4.	Підготовка матеріалів першого розділу дипломного проекту	05.05.2020
5.	Підготовка матеріалів другого розділу дипломного проекту	07.05.2020
6.	Підготовка матеріалів третього розділу дипломного проекту	10.05.2020
7.	Підготовка графічної частини дипломного проекту	20.05.2020
8.	Оформлення документації дипломного проекту	23.05.2020
9.	Попередній огляд матеріалів диплому на кафедрі	25.05.2020

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ. 467100.004 ПЗ	Android-месенджер	51		
			з криптографічним			
			захистом.			
			Пояснювальна записка			
	А4	ІАЛЦ. 467100.005 Д1	Android-месенджер	1		
			з криптографічним			
			захистом.			
			Схема структурна			
	А4	ІАЛЦ. 467100.006 Д2	Android-месенджер	1		
			з криптографічним			
			захистом.			
			Взаємодія з користувачем.			
			Схема алгоритму.			
	А4	ІАЛЦ. 467100.007 Д3	Android-месенджер	1		
			з криптографічним			
			захистом.			
			Шифрування коду.			
			Схема алгоритму.			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ. 467100.003 ОА		
Розробив		Додяк Д.І.			Android-месенджер з криптографічним захистом **Відомість дипломного проєкту**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Дробязко І.П.					
Н. контроль		Клятченко Я.М.				НТУУ "КПІ" ФПМ КВ-62	
Затвердив		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ. 467100.008 Д4	Android-месенджер	1		
			з криптографічним			
			захистом.			
			Діаграма класів.			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал			
			Презентація проєкту			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ. 467100.003 ОА	Арк. 2

Пояснювальна записка
до дипломного проєкту

на тему: Android-месенджер з криптографічним захистом

Київ – 2020 року

ЗМІСТ

					ІАЛЦ.467100.005 ПЗ		
Зм	***Лист***	***№ докум.***	***Підп.***	***Дата***			
Розроб.		Додяк Д.І.			Android-месенджер з криптографічним захистом ***Пояснювальна записка***	***Літ.***	***Лист*** 1 / ***Листів*** 51
Перев.		Дробязко І.П.					
Н. контр.		Клятченко Я.М.				**НТУУ «КПІ ім. І. Сікорського»,ФПМ,КВ-62**	
Затв.		Романкевич В.О.					

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ANDROID – операційна система смартфону;

JAVA – мова програмування додатків на Android;

ПЗ – Програмне забезпечення;

UI – User Interface – користувацький інтерфейс;

AUTH – Authorization – авторизація;

ALLATORI– шифратор коду;

MVVM – Model-View-ViewModel – шаблон проєктування;

SDK – Software Development Kit – набір інструментів розроблення;

API – Application Programming Interface – прикладний програмний інтерфейс;

FIREBASE – серверна база даних;

ВСТУП

Сьогодні месенджери займають перше місце в світі за кількістю їх завантажень користувачами смартфонів. Користувачі встановлюють по два-п'ять месенджерів, які використовуються в тій чи іншій мірі. Проте більшість з них не мають шифрування програмного коду, повідомлень, а також безпечної автентифікації. Користуючись додатками, значна кількість людей може навіть не здогадуватися, що їх приватна інформація може бути без дозволу використана сторонніми особами. Як показує статистика магазину додатків Android, лише 10% програм мають хоча б елементарний захист програмного коду.

У даному випадку, дуже важливим є використання криптографічних алгоритмів, адже вони можуть дозволити безпечно передавати дані між додатком та сервером, а також допоможуть вберегти програмний код. Хоча криптографічні алгоритми не дають стовідсоткову гарантію захисту, але все ж вони набагато ускладнюють будь-які спроби дістати корисну інформацію з додатку, а на повне зняття шифрування можуть піти роки.

Більшість месенджерів мають проблеми з безпекою через відсутність захищеної взаємодії з сервером та відсутність шифрування повідомлень. Тому створення месенджера з безпечною серверною частиною, шифруванням повідомлень, а також захищеним програмним кодом є надзвичайно важливим і становить завдання даного дипломного проєкту.

1 АНАЛІЗ ANDROID-МЕСЕНДЖЕРІВ З КРИПТОГРАФІЧНИМ ЗАХИСТОМ

1.1 Криптографія і завдання криптографічного захисту систем

Криптографією називають науку, що забезпечує такі методи:

– приватность,

– автентифікація

Криптографія – це наука, що вивчає і описує моделі інформаційної безпеки даних. Вона дозволяє вирішити багато проблем, що притаманні інформаційній безпеці мережі: конфіденційність, аутентифікація, контроль і цілісність взаємодіючих учасників [1].

Криптографія – одна з найдавніших наук, її історія простягається на тисячі років назазд. На початку свого існування наука криптографія займалась вивченням методів шифрування інформації, тобто оборотного перетворення тексту на основі певного алгоритму в шифрований текст (шифротекст). Традиційна криптографія містить в собі криптосистеми, де процеси кодування і декодування здійснюється з одним і тим самим серкетним ключем [2].

Криптографічним захистом інформації - механізм захисту за допомогою шифрування даних для забезпечення інформаційної безпеки [3]. Криптографічні методи захисту інформації знайшли застосування в сучасному житті при зберіганні, обробці та передачі інформації мережами зв'язку і на різних носіях.

Сьогодні криптографічний захист інформації може забезпечити найдійний зхист передачі інформаційних повідомлень на великі відстані за допомгою шифрування.

Шифрування дає змогу трансформувати інформаційні дані в форму, яка буде нечитабельною для більшості програмних алгоритмів і людини без наявності спеціального ключа шифрування-дешифрування. Ключовою метою криптографічного захисту інформації є забезпечення конфіденційності та захисту інформаційних даних комп'ютерних мереж в процесі передачі її між користувачами системи мережею. Захист конфіденційної інформації, який заснований на криптографічному захисті, шифрує інформаційні дані за допомогою оборотних перетворень, кожне з яких описується ключем.

Найбільш важливим елементом криптографічного захисту інформації є спецальний ключ, який відповідає за методику трансформації. Ключ – це сукупність певних символів, який дозволяє конфігурувати, шифрувати і дешифрути визначений алгоритм системи криптозахисту інформації [4].

Всі трансформації визначається ключем, що задає певний крипто алгоритм, який дозволяє забезпечити надійну безпеку інформаційної системи і інформації в цілому. Алгоритм криптографічного захисту інформації працює в різних режимах, які мають як переваги так і недоліки, що впливає на надійність інформаційної безпеки.

Існують різні методи криптографічного захисту, до них можемо віднести програмні, апаратні та програмно-апаратні засоби, що реалізують криптографічні алгоритми інформації з метою:

- захисту інформаційних даних (повідомлень) при їх обробці, використанні та передачі;
- забезпечення цілісності (відсутності спортворень) та достовірності інформації при її зберіганні, обробці та передачі (у тому числі, із застосуванням алгоритмів цифрового підпису);
- створення інформації, яка використовується для автентифікації та ідентифікації суб'єктів, користувачів і пристроїв;

- створення інформації, яка використовується для захисту елементів автентифікації при їх зберіганні, створенні, обробці та передачі.

На сьогодні криптографічні методи захисту інформації для забезпечення надійної автентифікації сторін інформаційного обміну є базовими. Вони передбачають шифрування і кодування інформації. Розрізняють два основні методи криптографічного захисту інформації:

– симетричний;

– асиметричний.

Симетричні методи використовують для операцій шифрування/дешифрування один і той самий ключ, що є секретним. Крім цього, існують досить ефективні методи симетричного шифрування.

В асиметричних методах криптографічного захисту інформації використовуються два ключі:

- Несекретний, який може публікуватися разом з іншими відомостями про користувача, що є відкритими. Цей ключ застосовується для шифрування.
- Секретний, який відомий тільки одержувачу, і використовується для розшифрування.

Серед асиметричних методів найбільш відомим є метод шифрування RSA (базується на операціях з великими (100-значними) простими числами).

Завдяки застосуванню криптографічних методів можна надійно контролювати цілісність окремих порцій інформаційних даних та їх наборів, гарантувати неможливість відмов дій, а також визначати справжність джерел даних.

Алгоритми шифрування мають виконовувати такі основні функції: приватність та відсутність спотвореннь даних. Приватність, в даному

випадку, означає, що під час обміну даними між особами, лише вони будуть здатні декодувати ці дані. Ані інтернет-провайдер, ані розробник месенджера чи якась третя сторона не повинні мати технічної можливості виконувати декодування. Повідомлення, цілісність якого було втрачено, не буде оброблюватись приймаєчею стороною і буде повністю відхилине, це забезпечує додатковий захист від атак які спрямовані на зміну даних .

На сьогодні, в багатьох систамах обміну інформацією, реалізована велика кількість функцій, що підвищують безпеку передачі. У протоколі Signal та схожих технологіях: передача даних в другорядному потоці, а також наявність обох видів секретності.

У месенджерах здійснюється доставка пропущених повідомлень. Вони приходять навіть у тому випадку, якщо була розмова в груповому чаті, і раптово відбулось довге відключення під час виклику. Повідомлення відпарвляютсья в другорядних потоках, це дозволяє забезпечити незалежний процес шифрування. Варто зазначити, що за рахунок точок синхронізації, послідовність залишається сталою та зберігається логіка виконання .

Секретність(пряма), дозволяє , при компрометації ключа шифрування поточного повідомлення за його допомогою можна буде провести процес декодування минулої генерації повідомлення. Тому реаалізовна періододична зміна сесійнийних ключів, кожен з яких шифрує власну частину даних.

Секретність(зворотна) дозволяє забезпечити шифрування настпного покоління даних за допомогою генерації нових ключів.

Обидва типи секретності лежать в осонові алгоритмів ключового управління. Наприклад в Signal для досягення цієї мети застосований спеціальний алгоритм - «Подвійний храповик».

За аналогією з цим, DR алгоритм також перешкоджає повторному використанню попередніх станів шифросистеми. DR часто змінює сесійні ключі, при цьому не дозволяючи повторне використання раніше

згенерованих. Завдяки цьому алгоритм реалізує обидва типи секретності, і таким чином захищає дані.

1.2 Характеристики ОС Android

Система Android є вільною операційною системою (ОС) для різних пристроїв, основана частина яких - смартфони, а також планшети, телевізори та інші пристрої, які працюють на базі ядра Linux. Перша версія системи з'явилася у вересні 2008 року. Оновлення версій ОС Android представлено на рис. 1.1.

Рисунок 1.1 – Оновлення версій Android

До основних переваг операційної системи Android можна віднести наступні:

- Операційна система Android є повністю відкритою, а отже, отримавши root права, але навіть без цього користувач може зберігати файли на пристрої;
- На відміну від iOS, Android надає можливість завантажувати різні додатки, які дають можливість дуже суттєво змінити системну конфігурацію ОС і пришвидшити її роботу.
- Система Android дозволяє встановлювати різні види UI лаунчерів, які дають змогу кардинально змінити інтерфейс всієї системи і налаштувати його для користувача індивідуально.

- Android дозволяє запускати будь-які програми і працювати вони будуть навіть після того як їх згорнули(поки оперативна пам'ять не буде переповнена) .
- Android надає можливість підключення практично будь-якого периферійного пристрою комп'ютера, навіть жорсткого диску ємністю 500 Гб за допомогою OTG-кабелю;
- Для встановлення нового додатку достатньо мати потрібний APK файл і встановити його на пристрої.

До недоліків ОС Android можна віднести:

- Необхідність досить частого зарядження пристрою на базі цієї операційної системи, що пов'язано з неекономним використанням системою наданих їй апаратних ресурсів. Частково цей недолік можна виправити налаштуванням енергозбереження, відключенням непотрібних функцій;
- Проблема оновлень системи, адже кожна компанія налаштовує Android під свої пристрої(змінює програмний код системи) і тому пристрої підтримувати важко і вони дуже швидко втрачають підтримку оновлень.
- Велика кількість налаштувань, що може бути досить складним для багатьох користувачів.

Певний час Android була не тільки найпоширенішою, але й найменш безпечною мобільною ОС. До версії 4.4 (2013 рік) безпека системи була дуже слабкою: шифрування було повністю відсутнє за замовчуванням і слабкі опціональні алгоритми робили смартфони з цією ОС уразливими для цілого ряду атак. Шифрування за зомовчуванням та мінімально прийнятний рівень безпеки в Android був досягнутий тільки з версією 6.0, причому тільки в тих пристроях, які випускались вже з встановленою Android 6.0. У

наступних версіях системи Google послідовно посилював вимоги до безпеки та впроваджував алгоритми шифрування на системному рівні для підвищення рівня безпеки ОС . Останнім нововведенням, яке з'явилося в Android 9, стало шифрування «хмарних» резервних копій пристроїв ключем, який залежав від коду блокування екрану. Статистика розповсюдженості версій ОС представлена нижче в табл. 1.1 [7].

Щодо вразливості, виявленої в коді операційної системи, для смартфонів на платформі Android передбачені перводичні(щомісячні) патчі безпеки, що послідовно усувають нововиявлені загрози, а iOS Apple може опублікувати оновлення в будь-який момент. Процент пристроїв Apple минулих поколінь які отримують оновлення і пдтримуються, набагато більший ніж у Google. Різниця в тому, що якщо Apple послідовно розвиває і покращує актуальну версію iOS (не торкаючись відмінностей між версіями системи для Apple TV, HomePod і планшетів з iPad OS), то Google доводиться підтримувати (або, навпаки, відмовлятися від підтримки) цілого спектру версій Android та смартфонів: тобто більшість користувачів смартфонів під управлінням Android отримують оновлення безпеки не завжди регулярно і оперативно, на відміну від деяких виробників, наприклад Sony, Nokia, Essential.

Таблиця 1.1 – Розповсюдження версій на травень 2019 року

Версія	Назва	Рік	Частка
2.3.3 - 2.3.7	Gingerbread	2010	0,3 %
4.0.3 - 4.0.4	Ice Cream Sandwich	2011	0,3 %
4.1.x	Jelly Bean	2012	1,2 %
4.2.x		2012	1,5 %

4.3		2013	0,5 %
4.4	KitKat	2013	6,9 %
5.0	Lollipop	2014	3 %
5.1		2015	11,5 %
6.0	Marshmallow	2015	16,9 %
7.0	Nougat	2016	11,4 %
7.1		2016	7,8 %
8.0	Oreo	2017	12,9 %
8.1		2017	15,4 %
9.0	Pie	2018	10,4 %
10.0	Q	2019	< 0,1 %

Шифрування на палтформі Android – це процес кодування всієї інформації та даних користувачів на пристрої Android за допомогою симетричних ключів шифрування. Після шифрування пристрою всі створені користувачем дані автоматично шифруються перед записом їх на диск, а при зчитуванні автоматично розшифровуються, перш ніж повернути їх у процес виклику. Шифрування гарантує, що навіть при намаганні сторонньої особи отримати доступ до даних, вона не зможе їх прочитати, це суттєво покарщує рівень безпеки системи.

Платформа Android підтримує два способи шифрування пристрою: шифрування на основі файлів та шифрування на диску.

В нових версіях Android (7.0 та новіші версії) є підтримка файлового шифрування. Файлове шифрування дозволяє виконати процес шифрування різних файлів різними ключами, які можна розблокувати самостійно. Пристрої з підтримкою файлового шифрування також можуть підтримувати Direct Boot, що дозволяє зашифрованим пристроям завантажуватися прямо

на заблокований екран, тим самим забезпечуючи швидкий доступ до важливих функцій пристрою, таких як послуги доступності та тривоги.

За допомогою файлового шифрування та API, які дають додаткам інформацію про шифрування, програми можуть працювати в обмеженому контексті. Це може статися до того, як користувачі надали свої повноваження, захищаючи інформацію приватних користувачів.

Починаючи з версії Android Pie ОС Android отримала підтримку шифрування метаданих, але лише на тих простоях де є апаратна підтримка. Завдяки шифруванню метаданих, один ключ, присутній під час завантаження, шифрує будь-який вміст, що не був зашифрований FBE, наприклад, розміри файлів, дозволи та час створення / модифікації. Цей ключ захищений Keymaster, який, у свою чергу, захищений перевіреним завантаженням.

Android 5.0 до Android 9 підтримують шифрування на повному диску. В даному шифрувані використовується всього один ключ (захищений паролем пристрою користувача) для захисту всього розділу даних користувача. Інформація на пристрої стане доступною тільки після того як користувач введе відповідні дані.

Це є суттєвим плюсом для безпеки, проте означає, що більшість основних функціональних можливостей телефону одразу недоступні, коли користувачі перезавантажують свій пристрій. Оскільки доступ до їх даних захищений за єдиним обліковим записом користувача, такі функції, як алерти, не можуть працювати, служби доступності є недоступними, а телефони не можуть приймати дзвінки.

1.3 Огляд існуючих месенджерів з криптографічним захистом

На сьогодняшній день на ринку месенджерів є Telegram, Viber, WhatsApp та Facebook Messenger. Розглянемо ступінь їх захисту.

Головною перевагою Telegram є наявність алгоритму шифрування MTProto, який оснований, відразу на декількох технологіях: стандарт, прийнятий урядом США – симетричне (для виконання операцій шифрування і розшифрування використовується один ключ) шифрування AES, шифрування RSA; класичний алгоритм для обміну секретними ключами Діффі-Хеллмана, який забезпечує міст між AES і RSA.

Тобто, для використання усіх можливостей шифрувальних технологій Telegram потрібно користуватися виключно секретними чатами.

Також до переваг Telegram відноситься двофакторна автентифікація за замовчуванням, яка дозволяє прив'язати акаунт користувача до номера телефону, і можливість обмежити доступ до самого додатку.

Для обходу базового захисту зловмисникам доведеться вирішувати питання з сім-картою користувача або ж клонувати її. Ще один шар захисту на випадок, якщо зловмисники отримають доступ до чужого смартфону – це пароль на запуск програми в системі [10].

Viber, який спочатку був ізраїльською розробкою, з нещодавна належить японській корпорації Rakuten. Viber – це найпопулярніший месенджер в Україні, який, за різними даними, має тут близько 25 млн активних користувачів. У додатку реалізовані різні методи шифрування, чати зі функціями автоматичного видалення повідомлень, заборони скріншотів. Також реалізований хороший захист від пересилань повідомлень.

При цьому, на відміну від Telegram, в Viber повне шифрування за замовчуванням включено для всіх чатів. В цілому, згідно заяв розробників Viber, технології шифрування даних в месенджері дуже схожі на ті, що у

Telegram. Більше того, використовуються технології месенджера Signal, які вважаються еталоном в цій сфері.

Для створення резервної копії листування Viber пропонує використовувати Google Drive. І Viber стверджує, що при створенні таких бекапів програма не несе відповідальності за їх приватність (що все залежить від Google).

У мессенджері використовують сучасні методи шифрування, але їх незалежний аудит не проводився.

Технології шифрування в месенджері WhatsApp: двофакторна автентифікація, наскрізне шифрування, налаштування з обмеженням доступу до додатку [11].

Навіть описані методи шифрування можна поставити під сумнів після недавнього злому WhatsApp, через який могли постраждати мільйони користувачів.

Незважаючи на вищезазначене, WhatsApp залишається найпопулярнішим месенджером в світі і йому належать дані близько 1,5 млрд користувачів. Потрібно зауважити, що тільки 5% користувачів звертають увагу на безпеку менеджеру, інші розраховують на масовість і зручність використання.

Facebook Messenger є ще одним продуктом Facebook. Прив'язування в додатку до акаунту Facebook, яке працює як двофакторна автентифікація, можна вважати як позитивним, так і негативним (якщо хтось зламав FB, то то він відразу отримує і доступ до месенджера). Зараз в компанії Facebook заявляють про більшу увагу до безпеки персональних даних користувачів.

Наскрізне шифрування – основний метод шифрування даних в Skype. Перевагою Skype є відсутність резервного копіювання файлів.

Показовим був недавній витік інформації, згідно з яким співробітники Microsoft могли прослуховувати розмови в Skype.

Існують ще сервіси iMessage і Facetime компанії Apple, керівництво якої постійно заявляє про пріоритетність захисту персональних даних своїх клієнтів. Повідомлення в цих додатках дійсно надійно зашифровані, проте проблема в тому, що ключі шифрування зберігаються на серверах.

1.4 Обґрунтування теми та формулювання завдань розробки месенджера

В сучасному світі месенджери стали важливою частиною життя кожної людини, мільярди людей користуються ними, але нажаль дуже мало з них має достойний рівень захисту. Особливо великою проблемою є відсутність шифрування коду програми, що дає можливість отримати повний контроль над програмним інтерфейсом. Таким чином, створення месенджера з криптографічним захистом є актуальним сьогодні.

Статистика кількості користувачів різних месенджерів показує, що з часом окремі додатки-месенджери втрачають лідерські позиції, поступаючись місцем новим. Так, за один рік Skype з першого місця, яке він займав кілька років, перемістився на четверте.

На ранньому етапі месенджери створювалися для певної мети: як чати, наприклад WhatsApp, або як додаток для дзвінків – Skype, Viber. Пізніше в месенджери стали додавати функції, яких спочатку не було. Так, в WhatsApp додалися функції аудіозвонков, потім відео. Далі з'явилися відкриті API для створення ботів, маски, статуси, прийоми платежів, публічні канали. Однак впровадити новий функціонал або змінити структуру, коли у месенджера мільйони користувачів, складно. У тому ж WhatsApp досі немає API і пошукових роботів спамів.

Основною складністю при створенні програми для відправки повідомлень на Android є розробка архітектури. Структуру додатка потрібно розробити таким чином, щоб було нескладно розширювати його можливості.

При розробці месенджера слід закладати в його основу не тільки те, що вже є в інших додатках, але й те, що може бути затребуване в майбутньому.

При цьому основними критеріями при розробці є:

- Модульність
- Масштабованість
- Гнучкість
- Тестованість
- Продуктивність
- Можливість роботи з великою кількістю користувачів
- Можливість роботи з великими навантаженнями
- Ергономічність інтерфейсу месенджера
- Зручність всередині чату
- Пошук у чаті
- Конфіденційність, шифрування

Розробник повинен забезпечити повну анонімність користувача, навіть без прив'язування до номера телефону, і можливості якимось чином розпізнати особистість. Крім того, потрібно виключити ризики злому.

Для отримання функції анонімного чату співрозмовники без прив'язки до конкретних ознак (номеру телефону, імені, локації) повинні ідентифікувати один одного. Для цього потрібно використовувати одноразовий шифр, яким можуть користуватися всі, але він не буде повторюватися двічі. Запрошення людей до такої бесіди також відбувається

за допомогою "ключа", який працює тільки раз і задається тільки самим користувачем.

Основні функції розроблюваного Android-месенджера з криптографічним захистом:

- швидкий користувацький інтерфейс;
- безпечна автентифікація і авторизація користувачів;
- можливість відновлення паролю;
- можливість шифрування повідомлень перед відправленням;
- захист програмного коду за допомогою методів криптографії;
- реалізація повного функціоналу месенджера;
- безпечне кешування даних.

2 ВИБІР ЗАСОБІВ РОЗРОБЛЕННЯ МОБІЛЬНОГО ДОДАТКУ

2.1 Теоретичні основи розроблення Android-месенджера з криптографічним захистом

Для написання Android-додатків використовують багато різних мов програмування, основа з яких - java. Основним елементом для розробки Android-додатків є Android Sdk, даний набір інстментів дозволяє компілювати файли програми та встановлює прямий зв'язок з системними функціями платформи Android. Всі додатки Android мають формат APK або AAB, останній з’явився досить недавно і представляє собою оптимізовану версію APK з меншим розміром. Встановлення додатків відбувається саме завдяки цим файлам. Додатки можуть встановлюються на всіх пристроях з ОС Android.

Кожен додаток, працює у власному програмному середовищі:

• платформа Android розрахована на велику кількість користувачів;

• Кожній програмі присвоюється унікальний номер uuid, подібно до системи Linux.

• система встановлює повноваження для всіх файлів додатку, щоб забезпечити доступ до них тільки користувачам з uuid, який відповідає тільки цьому додатку;

• Всі додатки працюють в різних ізольованих середовищах, тому вони являються незалежними.

• Кожному додатку відповідає свій процес. При завершенні виконання Android завершує процес [12].

За допомогою таких принципів в Android реалізовано систему обмежених прав, адже кожен додаток виконується ізольовано від інших. Це означає, що програма використовує тільки ті права, які вона потребує. Якщо

програмі необхідні додаткові права – вона повинна запросити їх у користувача. Однак, додатки можуть передавати дані іншим програмам.

• В системі Android є можливість присвоєння однакового унікального ідентифікатору декільком додаткам одночасно;

• Для використання смс-повідомлень, карти-пам’яті, геолокації та інших системних функцій, додаток повинен запросити на це дозвіл у користувача. У разі відмови користувача, дана функція не зможе використовуватись у додатку.

Основні компоненти Android програми:

• базові компоненти, які описані вище;

• файл маніфесту, в якому оголошуються необхідні для додатку компоненти, а також визначаються права додатку;

• ресурси, які існують окремо від коду програми і дозволяють додаткам оптимізувати представлення інформації відповідно до типу пристрою.

Android додаток складається з окремих компонентів. Точки входу в програму різні для всіх компонентів. Більшість компонентів є залежними один від одного і являються точками входу для різних саб-компонентів, але навіть в такій системі кожен компонент є унікальним і відіграє свою особливу роль в даній системі.

Основних системних Android компонентів існує всього чотири – компоненти які прописуються в маніфест файлі. Кожен з них має свій власний цикл життя і виконує свою особливу функцію.

Типи:

- Активності

Даний компонент являє собою екран користувача. В даному елементі описана вся логіка взаємодії з користувачем, а також ініціалізація та керування всіми фрагментами даного екрану. Даний клас також відповідає за

ініціалізацію всіх елементів інтерфейсу користувача. Встановлення базового контенту активності відбувається за допомогою стандартного методу setContentView(). Всі активності успадковуються від стандартного класу Activity, а також імплементують його методи які дозволять визначити цикл життя даної активності. Зазвичай додаток має безліч екранів, а зв'язок між встановлюється за допомогою класу Intent, який дозволяє здійснити перехід від одного екрану до іншого.

- Служби

Даний компонент виконується у фоновому потоці й дозволяє реагувати на системні події, а також працює навіть коли додаток знаходиться не в активному стані. Наприклад, за допомогою служби ми можемо слухати музику, навіть коли екран пристрою заблокований, а також отримувати push-повідомлення. Служба може бути статичною (ініціалізованою в маніфест-файлі), викликатись системою або бути викликаною через компонент активності, в останньому випадку служба буде залежати від циклу життя даної активності. Всі служби успадковуються від стандартного класу Service, а також імплементують його методи. Служби мають можливість викликати інші служби, механізм дуже схожий на перехід між активностями, який відбувається за допомогою стандартного класу Intnet. Виклик служби відбувається методом startService(). Сервіси також можуть прив’язувати до інших компонентів за допомого методу bindService().

- Постачальники контенту

Даний компонент (Content provider) забезпечує комунікацію між програмами, дані зберігаються в бд SQLite. За допомогою даного компоненту база даних може використовуватися декількома додатками одночасно й таким чином відбувається обмін інформацією між програмами. Для отримання доступу до постачальника контенту іншої програми у даної

програми має бути відповідний дозвіл. Приведемо приклад: додаток може містити номера телефонів певних користувачів, а інші програми можуть отримати до неї доступ через постачання контенту й використовувати її у своїх цілях. Content Provider дозволяє додаткам формувати спільні бази даних, без прямих взаємодій між ними. Даний компонент має обов'язковий метод Query() для взаємодії з бд, а доступ до даних відбувається безпосередньо за допомогою класу ContentResolver.

- Приймачі широкомовних повідомлень

BroadcastReceiver дозволяє реагувати на певні події, які відбуваються по всій системі додатку (відключення, змін стану сім-карти, вимкнення екрану, розрядження акумулятора і т.д.). також є можливість реалізації прослухування не системних подій (завантаження файлу з інтернету). BroadcastReceiver може бути статичним або ж викликатись іншим компонентом(активність, служба).

Даний компонент також дозволяє реалізовувати комунікацію між різними компонентами додатку, за рахунок відправки повідомлень.

Android SDK має власний приймач широкомовних повідомлень, який походить від класу BroadcastReceiver. В той час клас Intent забезпечує надання повідомлень.

Однією із основних характеристик платформи Android є те, що абсолютно всі програми мають здатність використовувати складники інших додатків. Для прикладу можна навести ситуацію, якщо у користувача є необхідність використати функцію відео зйомки, то можна викликати додаток (за умови, що додаток встановлений), здатний виконати цю дію, замість розроблення операції відео зйомки у своєму додатку. Немає необхідності для повторного впровадження програмного модуля камери. Для виконання даної операції можна почати запуск стандартного додатку камери

ОС Android. Одразу після виконання даного виклику відео буде надіслано безпосередньо в додаток і стане можливим для використання без будь-яких обмежень, але при цьому користувач навіть не помітить переходу між програмами.

При запуску складника системою, вона ініціює створення окремого процесу для цього додатку (якщо його ще не запущено) і створює потрібні цьому складнику екземпляри класів. У випадку запуску додатком операції відео зйомки, за допомогою системному додатку камери, то ця операція буде виконуватиметься в процесі цього стороннього додатку, а не в процесі основного модуля програми. Таким чином, порівнюючи з додатками інших систем, у Android прослідковується відсутність визначеної точки входу.

Враховуючи той факт, що у ОС Android кожна програма виконується в окремо визначеному процесі, то це сричиняє конфлікт прав доступу та дозволів між програмами. Саме тому в даній системі відсутня можливість прямого виклику складників із іншої програми. Ця функція виконується виключно платформою Android. Для виконання таких завдань використовується системний клас Intent, так як саме він відповідає за запуск сторонніх компонентів.

До складу додатку Android входять не лише програмні файли. Додаток потребує додаткові ресурси, наприклад: текстові файли, стилі, шрифти, зображення та інше. Для забезпечення інтерфейсу користувача, макети екранів відображаються в XML-файлах. Дані ресурси є статичними і вони не змінюються під час виконання програми. Основна функція таких ресурсів – це оптимізація додатку для будь-яких фізичних характеристик пристроїв, та пришвидшення роботи програми.

Для всіх ресурсів, які є складовими частинами додатку Android, задано спеціальний id, що призначений для можливості посилання на даний ресурс з будь-яких модулів додатку. Наочним прикладом є стандартний стиль

програми Android, який можна викликати за посиланням R.style.AppTheme. Всі ідентифікатори ресурсів додатку Android генеруються на етапі компіляції програми й визначені в системному класі R. Даний підхід суттєво спрощує роботу з ресурсами.

Як вже було зазначено вище, одним із основних завдань впровадження ресурсів є оптимізація під різні пристрої. Так, для прикладу, директорія values може використовуватися для повної локалізації Android-додатку. Даний каталог містить в собі файл strings.xml, в якому зберігаються рядкові константи та їх переклади різними мовами. Під час виконання програми, Android додаток, в залежності від регіону пристрою, обраного користувачем, буде використовувати відповідний переклад.

Android дає змогу встановлювати різноманітні кваліфікатори для своїх ресурсів. Найкращим прикладом на підтвердження цього є можливість створення конфігурацій, які відповідають певній орієнтації дисплею. Наведемо приклад: у випадку горизонтальної орієнтації екрана, завдяки встановленому кваліфікатору, буде використовуватися відповідний ресурс, що забезпечить коректне відображення інформації в даній орієнтації. Дане твердження є також справедливим і для вертикальної орієнтації. Кваліфікатори забезпечують адаптацію відображення інформації для великої кількісті пристроїв одночасно.

Android використовує функцію шифрування dm-crypt. Це стандартна система шифрування диску в ядрі Linux. Технологія dm-crypt використовується різними збірками Linux.

2.2 Системні вимоги до розробки Android додатків

Перед розробкою додатків для Android готується відповідна інфраструктура, тобто обираються технічні засоби, необхідні для

розроблення програм для Android. Основним інтегрованим середовищем розробки додатків на Android є Android Studio, дана IDE має дуже широкий функціонал і мастить весь набір SDK для побудови додатків на Android.

В розділі «System requirements» офіційної документації Android Studio для Windows вказана наступна специфікація:

- Windows 7/8/10 будь-якої розрядності або Mac OS;
- мінімум 3 Гб ОЗП (+2 Гб для емулятора), рекомендований обсяг – 8 Гб;
- 4 Гб на диску;
- монітор з роздільною здатністю хоча б 1280 x 800.

На практиці, такі мінімальні вимоги не є достатніми для стабільної роботи цієї IDE і розгортання системи з такими параметрами не гарантує комфортної та швидкої роботи.

Як результат, реально системні параметри виглядають наступним чином:

- реальний мінімум ОЗП – 8 Гб;
- процесор класу Core i5;
- SSD диск.

Реальна статистка показує, що потрібно:

- орієнтуватися саме на процесори Intel, а не AMD, оскільки Android Studio повноцінно користується ресурсами багатоядерних Intel процесорів;
- підняти роздільну здатність монітора до 1440 x 900.

У підсумку, машина розробника додатків під Android, що працює в Android Studio, повинна для комфортної роботи мати наступну нижню межу конфігурації:

- Windows 7/8/10 (32 або 64 біт)/Mac OS X;
- процесор Intel Core i5-8400;

- материнська плата на база чіпсета Z370;
- мінімум 12 Гб ОЗП;
- SSD диск;
- відеокарта, що підтримує дозвіл 1440 x 900 (і відповідний монітор).

2.3 Технологічні та програмні засоби розроблення додатку

Архітектурний патерн MVVM

Model-View-ViewModel (MVVM) – шаблон (патерн) проектування архітектури додатку , який був презентований ще у 2005 році. На сьогоднішній день цей шаблон став одним із основних патернів для побудови клієнт-серверних додатків.

MVVM доповнює класичний шаблон MVC(коли у платформи розробки є зв'язування даних). Основною відмінністю від MVVM від MVC є те, що у останнього зміни в інтерфейсі не впливають на Model (Модель), а попередньо йдуть через Controller (Контролер).

Патерн MVVM дозволяє відокремити логіку додатків від візуальної частини (View) і є архітектурним, тобто він задає загальну архітектуру програми [18].

MVVM складається з трьох компонентів: Model, ViewModel і Model, представлених на рис. 2.1.

Модель описує дані, що використовуються додатком та включає в себе логіку, яка очевидно має зв'язок з цими даними. Основною відмінністю між моделлю та представленням є відсутність у моделі логіки, яка відповідає за представлення цих даних користувачеві .

INotifyPropertyChanged – один із основних інтерфейсів патерну MMVM. Даний інтерфейс допомагає швидко реагувати на зміни характеристик моделі.

Рисунок 2.1 – Взаємозв’язок компонентів моделі

View (UI) відповідає за інтерфейс системи. В даному описана логіка завдяки якій відбувається взаємодія між користувачем і додатком [15].

ViewModel – це компонент патерна, який забезпечує зв’язок між Model та View, завдяки ситемі DataBinding. Інтерфейс INotifyPropertyChanged відображає зміну даних в моделі й безпосередньо встановлює зв’язок між ViewModel та Model.

Команди визначають взаємодію між ViewModel та Model в даному патерні, так як елементи інтерфейсу працюють без залучення подій.

Наведемо приклад роботи патерну MVVM: користувач натискає на текстове поле, через яке відбувається надсилання команди в компоненту ViewModel, після чого відбувається оновлення моделі, за рахунок отриманих даних.

Як результат застосування патерна MVVM програму можна розділити на три умовні частини, які у подальшому модифікувати і підтримувати.

Особливості мови програмування Java

За результатами щорічного звіту State of the Octoverse, який випускає Github, мова програмування Java в 2019 році займає третє місце в списку найбільш популярних мов і стабільно займає найвищі позиції у провідних рейтингах на протязі багатьох років.

Java – мова програмування загального призначення. Відноситься до об'єктно-орієнтованих мов програмування, а також до мов програмування з сильною типізацією.

Це завдання вирішується завдяки компіляції написаного на Java коду у байт-код.

У Java застосовується особливий механізм для управління та очищення пам'яті, який називається збиранням сміття (garbage collector). Розробник створює об'єкти, а JRE за допомогою збирача сміття сама очищає пам'ять, коли об'єкти перестають використовуватися. Цей підхід суттєво відрізняється від C++, де в об’єктів необхідно реалізовувати деструктори для подальшого очищення пам’яті.

Синтаксис мови Java схожий на синтаксис інших Сі-подібних мов.

Мова Java успадкувала у мови C, C ++:

- основний синтаксис від мови C;
- Об'єктно-орієнтовану базу від мови C ++.

Проблеми перенесення і безпеки програм при використанні мови Java вирішені наступним чином:

- Програма, яка написана, мовою програмування Java, сама по собі, не є виконуваним кодом, а представляє байт-код і називається Java додатком;
- виконання байт-коду відбувається тільки під управлінням спеціальної програми, яка називається віртуальною Java машиною;

- перенесення та виконання байт-коду можливе на всіх операційних системах, де встановлена Java-машина.

Програма, написана мовою Java:

- представляє собою один або декілька файлів .java;
- після компіляції програми трансформуються у файли .class;
- байт-код – це системні інструкції для JVM, які являють собою певні команди;
- JVM трансформує байт-код у відповідні інструкції для процесора.

У Java використовують поняття «пакети»:

- програма завжди складається з одного або декількох класів;
- кожному класу відповідає окремий файл;
- для угрупування класів використовуються пакети.

Пакет в Java – це сукупність класів, які використовуються для встановлення незалежності простору імен і регуляції доступу до класів.

Пакети додатку утворюють певний модуль. Скомпільований модуль програми – це не великий набір класів, а всього лише один файл, архів JAR (Java Archive, архів Java). Архів має теж саме ім'я, що і додаток.

Програми Java можна розділити на декілька основних категорій:

- Додаток (application) – аналог «звичайної» прикладної програми;
- Серверний додаток (Enterprise application) – застосовується для виконання серверної частини додатку;
- Аплет (applet) – спеціалізований додаток(має обмежені можливості), що виконується на базі браузера;
- Сервлет (servlet) – спеціалізований додаток(має обмежені можливості), що виконується в WWW на серверній частині;
- Мидлет (midlet) – програма, що запускається і виконується в мобільному середовищі;

- Бібліотека (Java Class Library – бібліотека класів або модуль платформи NetBeans).

Java відноситься до мов програмування загального призначення.

Наприклад, переважна більшість всіх великих компаній так чи інакше використовують Java. Переважна більшість використовує мову для написання серверних програм. Наприклад, програми для фінансових організацій, які забезпечують проведення транзакцій, фіксацію торгових операцій тощо.

Java застосовується у багатьох веб-додатках: від ecommerce-проектів до великих порталів, від освітніх платформ до урядових ресурсів [16].

Мобільна розробка – ще одна область застосування мови програмування Java. Цією мовою пишуть програми для пристроїв, що працюють під управлінням платформи Android.

На Java створюють також і клієнтські програми, наприклад IDE NetBeans повністю написано на Java.

Тобто на Java універсальна мова, з її використанням можна писати різні типи додатків: веб, мобільний і десктопні програми, ігри тощо. Традиційно у цієї мови сильні позиції у промисловому програмуванні, в сегменті великих компаній.

Firebase Sdk

Firebase - сервіс від компанії Google, який представляє хмарну базу даних, яка, в свою чергу, оновлюється в режимі реального часу, може містити великий об’єм даних, а також здатна витримувати великі навантаження. Firebase SDK реалізовано на багатьох платформах, в тому числі і на Android та представляє API для виконання операцій читання/запису

бази даних. Інтеграція відбувається з допомогою .json файлу. Передбачено API для шифрування даних [12].

Крім цього, компанією випущений під ліцензією MIT веб-редактор коду Firepad, що забезпечує одночасну спільну роботу декільком користувачам з одним документом, який став основою редакторів Stash Realtime Editor.

Завдяки Firebase ми маємо можливість швидко розробляти додатки високої якості. Також слід зауважити, що Firebase має високий рівень захисту даних своїх користувачів.

Шифратор програмного коду Allatori

Allatori – це обфускатор Java другого покоління, який пропонує повний спектр захисту інтелектуальної власності.

Більшість обфускаторів другого покоління забезпечує гідний рівень захисту, але в Allatori реалізовані додаткові методи обфускації та більше можливостей модифікації алгоритмів, щоб зробити декомпіляцію коду майже неможливою.

Allatori не просто шифрує, а також мінімізує розмір програми та збільшує швидкість її виконання. Allatori, як і кожен сучасний обфускатор Java, має повний набір функцій, що дозволяє потенційно правильно ліцензувати програмне забезпечення.

Якщо потрібно захистити програмне забезпечення і якщо потрібно зменшити як його розмір, так і час обробки, то Alfori Obfuscator є хорошим рішенням [13].

3 РОЗРОБКА ANDROID-МЕСЕНДЖЕРА З КРИПТОГРАФІЧНИМ ЗАХИСТОМ

3.1 Основні компоненти месенджера з криптографічним захистом

Як визначено вище, розроблюваний месенджер використовує архітектуру проектування MVVM.

Тобто основі модулі програми це – View, ViewModel, Model.

Основними складовими додатку є:

• Activity / Діяльність

В програмі реалізовані наступні Activity:

OpenActivity – стартовий екран програми; користувач обирає між логіном та реєстрацією. Також у даному класі йде перевірка на те, чи логіниться користувач за допомогою FirebaseAuth.

EnterActivity – екран логінізації; для успішного логіну потрібно ввести пошту та пароль.

MainActivity – основний екран програми; являє собою контейнер для допоміжних екранів (фрагментів).

UserMainActivity – профіль користувача, надає можливість завантажити користувачу фото.

RegistrationActivity – екран реєстрації; завдання даного екрану – зв’язок

із FirebaseAuth та валідацію полів реєстрації за допомогою регулярних виразів.

ResetPasswordActivity – екран зміни пароля користувача.

MessageActivity – екран листування з іншим користувачем; цей екран показує користувачу історію листування, а також дає можливість відправляти повідомлення.

• Services / Служби

Виконує задачі в другорядному потоці.

FirebaseMessagingService – основний сервіс програми, який забезпечує відображення повідомлень користувачу, працює в фоновому режимі і показує повідомлення на екрані смартфона, навіть коли програма не активна. Сервіс інтегровано з Firebase, а також має можливість відправлення сповіщень користувачу.

• Content Provider / Контент-провайдери.

Дозволяє отримувати дані з інших програм, а також безпечно переміщати інформацію. Використовується інтерфейсами програми для безпечної передачі даних між модулями програми.

• Intents / Наміри

Наприклад, додаток може запитати через Інтент контакт у програми контактів (телефонної / записної книжки) апарату. Intent використовується програмою для запусків сервісів, а також переходу між екранами інтерфейсу.

• Broadcast Receiver / Широкомовний приймач (далі – Приймач)

Приймає системні повідомлення і неявні Intent. Використовується програмою для отримання системних повідомлень.

Основні моделі програми:

UserModel – містить дані про користувача (id, ініціали, посилання на його фото, статус, та його опис)

ChatModel – містить дані про відправника та приймача повідомлень чату, їх текст, а також про час відправлення цих повідомлень.

TokenModel – містить інформацію про токен доступу в програму, при кожному запуску йде перевірка до серверу з метою з метою підтвердження дійсності токену.

ResponseModel – містить код відповіді від серверу та код даної відповіді.

Основний системний файл месенджера – «AndroidManifest.xml». У цьому файлі оголошені всі активні служби, приймачі і контент-провайдери додатку. Також він повинен містити дозволи, що необхідні додатку.

Каталог «gen» в Android-проєкті містить згенеровані значення.

Тоді як каталог "res" зберігає структуровані значення, які відомі платформі Android, каталог "assets" використовується для зберігання файлів.

3.2 Автентифікація користувачів

Безпечна автентифікація користувачів – дуже важлива складова захисту всього додатку. Автентифікація користувачів у додатку відбувається за допомогою Firebase Auth, який є частиною Firebase Sdk. Дана технологія дозволяє автентифікувати користувача за його паролем та поштою. Після перевірки даних користувач отримує токен, який має час життя і оновлюється під часу кожного заходження в додаток. Якщо додаток не використовується тривалий час, потрібна буде повторна автентифікація.

Самі дані для перевірки автентифікації зберігаються у шифрованому вигляді в Firebase Realtime Database.

Процедура автентифікації у додатку має наступний вигляд:

```
FirebaseAuth token =  FirebaseAuth.getInstance();
token.signInWithEmailAndPassword(email, password)
                        .addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
                            @Override
                            public void onComplete(@NonNull Task<AuthResult>
task) {
                                if (task.isSuccessful()){
```

```
                    Intent i = new Intent(EnterActivity.this,
MainActivity.class);
                    if(message !=null){
                    message.dismiss(); }
                    startActivity(i);
                    finish();
                } else {
                    if(message !=null){
                    message.dismiss();
                    }
                }
            }
        });
```

Автентифікація відбувається за допомогою методу signInWithEmailAndPassword(email, password) класу FirebaseAuth. Після чого, за допомогою класу Task<AuthResult>, проводиться перевірка на успішність виклику FirebaseAuth (за допомогою методу task.isSuccessful()). Тільки після автентифікації є можливість для отримання даних користувача, а також можливість подальшої зміни цих даних. У програмному коді це виглядає наступним чином:

```
FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
if (user != null) {
    String name = user.getDisplayName();
    String email = user.getEmail();
    Uri image = user.getPhotoUrl();
    boolean v = user.isEmailVerified();
    String identifier = user.getUid();
}
```

3.3 Криптографічний захист програмного коду додатку

Захист програмного коду додатку реалізовано за допомогою професіонального обфускатора Allatori. Використання цієї технології є дуже важливим фактором захисту всієї програми, оскільки додаток на Android (Apk) можна легко і швидко декомпілювати і отримати, таким чином, весь код програми. Код програми може бути використаний для проведення атак на сервери, плагіату, а також недозволеної зміни логіки функціонування програми.

Алгоритм шифрування відображений на кресленні Д3 (див. Додатки). Шифратор послідовно застосовує до коду методи обфускації, відображені в файлі allatori.xml [17].

Налаштування шифрування відбувається за допомогою файлу allatori.xml:

```
//Шифрування рядків String
<property name="string-encryption" value="enable"/>
<property name="string-encryption-type" value="strong"/>
<property name="string-encryption-version" value="v4"/>

//Контрольно потокове шифрування
<property name="control-flow-obfuscation" value=""/>
<property name="extensive-flow-obfuscation" value="maximum"/>

//Переміщення класів в default package
<property name="default-package" value=""/>
<property name="force-default-package" value="enable"/>
```

```
//Перейменування класів
<property name="packages-naming" value="compact"/>
<property name="classes-naming" value="abc"/>
<property name="methods-naming" value="abc"/>
<property name="fields-naming" value="abc"/>
<property name="local-variables-naming" value="optimize"/>
<property name="update-resource-names" value="enable"/>
<property name="update-resource-contents" value="enable"/>

//Додатові методи
<property name="line-numbers" value="keep"/>
<property name="generics" value="keep"/>
<property name="inner-classes" value="keep"/>
<property name="member-reorder" value="enable"/>
<property name="finalize" value="disable"/>
<property name="synthetize-methods/>
<property name="synthetize-fields"/>
<property name="remove-toString" value="disable"/>
<property name="remove-calls"/>

//Compression for optimization
<property name="output-jar-compression-level" value="9"/>
```

Далі представлено основні методи шифрування, які використані у додатку:

Шифрування рядків java.lang String

Сутність методу полягає тому, що в програмі створюється декілька локальних шифраторів, які шифрують всі рядкові константи коду, таким чином роблячи повторне використання дуже складним. Розшифрування рядків відбувається тільки під час виконання програми. Також треба зауважити, що від даної опції програма стає не дуже повільнішою, адже, як показано далі, для розшифрування рядків використовуються базові операції мови Java, а не складні і ресурсовитратні бібліотеки типу Crypto. Allatori пропонує декілька алгоритмів для шифрування рядків: “fast”,”medium”,”strong”. Для збереження швидкості роботи інтерфейсу у програмі використано найшвидший. Нижче надаються приклади констант до шифрування:

```
hashMap.put("status", "offline");
```

Приклад алгоритму шифрування:

```
hashMap.put(Data.pa("{5i5}2"), User.pa("<~5t:v6"));
```

Також локальна функція для розшифровування рядків:

```
public static String pa(String str) {
    String str2 = str;
    int length = str2.length();
    char[] cArr = new char[length];
    int i = length - 1;
    while (i >= 0) {
       int i2 = i - 1;
       cArr[i] = (char) (str2.charAt(i) ^ 'A');
       if (i2 < 0) {
          break;
       }
       i = i2 - 1;
       cArr[i2] = (char) (str2.charAt(i2) ^ 8);
```

```
        }
        return new String(cArr);
    }
```

Перейменування класів, методів, змінних

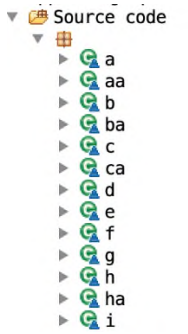
Головна ідея цього методу полягає в тому, щоб зробити код нерозбірливим. Адже втрачаються майже всі імена класів, методів, змінних (окрім тих, які потрібні системним бібліотекам Android). Allatori дозволяє конфігурувати, яким чином будуть змінюватись імена, які префікси будуть додаватися перед змінними. У програмі вибрана опція “abc”, що дозволяє оптимізувати програму, адже всі класи, методи та змінні будуть мати короткі імена типу a, b, c, а це доволі сильно пришвидшує програму і зменшує її розмір. Тому ця опція є одночасно і шифратором, і оптимізатором. Нижче представлено приклад, який ілюструє втрату імен класів, методів, і змінних:

```
class pa implements View.OnClickListener {
    public final /* synthetic */ MainActivity e;
    public final /* synthetic */ TabLayout r;

    public /* synthetic */ void onClick(View view) {
        this.r.getTabAt(2).select();
    }

    public /* synthetic */ pa(MainActivity mainActivity, TabLayout tabLayout)
{
        this.e = mainActivity;
        this.r = tabLayout;
    }
```

}

Перенесення всіх перейменованих класів у default package

Метод полягає в тому, щоб знищити пакетну архітектуру програми. Звичайно всі класи розподіляються за своїм функціональним призначенням в дереві пакетів (UI, Model, ViewModel). Даний метод групує всі класи в одному місці і, таким чином, робить визначення функціонального призначення даних класів дуже складним. Також треба зауважити, що класи переносяться не в новий пакет, а в root директорію java і, таким чином, всі класи знаходяться у default пакеті. Багато декомпіляторів навіть не здатні отримати класи з default пакету, тому це робить програму ще більш захищеною. Даний метод також виступає оптимізатором, адже через зменшення пакетного дерева розмір програми стає меншим. Наступний приклад показує, як виглядають класи у default пакеті:



Контрольно-потокове шифрування

Цей метод змінює семантику програмного коду таким чином, щоб неможливо було зрозуміти сутність логіки програмного коду. Побічною дією даного методу є сповільнення роботи програми, тому накладається у довільному порядку на 5-10% коду, тим самим змінюючи його структуру.

Зміна порядку коду

Даний метод ускладнює функціональну розбірливість програмного коду, і у довільному порядку змінює розташування методів, змінних і класів. Під час кожної компіляції порядок змінюється.

3.4 Інтерфейс користувача

Інтерфейс для діяльності (Activity) визначається за допомогою макетів. Під час виконання макети – екземпляри android.view.ViewGroups. Макет визначає елементи призначеного для користувача інтерфейсу, їх властивості та розташування. Елементи UI грунтуються на класі android.view.View.

Інтерфейс реєстрації у додатку представлено на рис. 3.1.
Реєстрація відбувається за допомогою 3 полів (ім’я, пошта, пароль).
При натисканні на кожне поле буде відкриватися клавіатура для вводу інформації у відповідне поле. Всі поля вводу реалізовані за допомогою класу EditText. Після заповнення всіх полів користувач має натискає на кнопку “Get Started” для подальшої валідації вхідних даних. Всі кнопки в програмі реалізовані за допомогою стандартного класу Button.

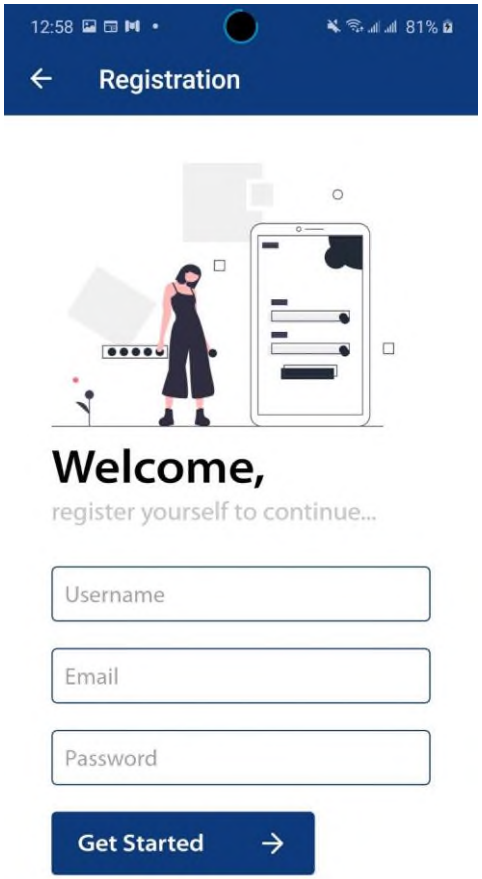


Рисунок 3.3 – Вікно реєстрації користувача у додатку

Інтерфейс входу зареєстрованого користувача до додатку представлено на рис. 3.2. Даний екран містить два поля(пошта та пароль). Поля реалізовані за допомогою технології EditText, аналогічно тому як реалізовано подібні поля на екрані реєстрації. Після натискання кнопки “Login” йде процес авторизації на стороні сервера. Після чого, в разі успішної авторизації, користувач отримує доступ до основного функціоналу програми(перегляду та відправки повідомлень). Всі картинки в програмі представлені класом ImageView.

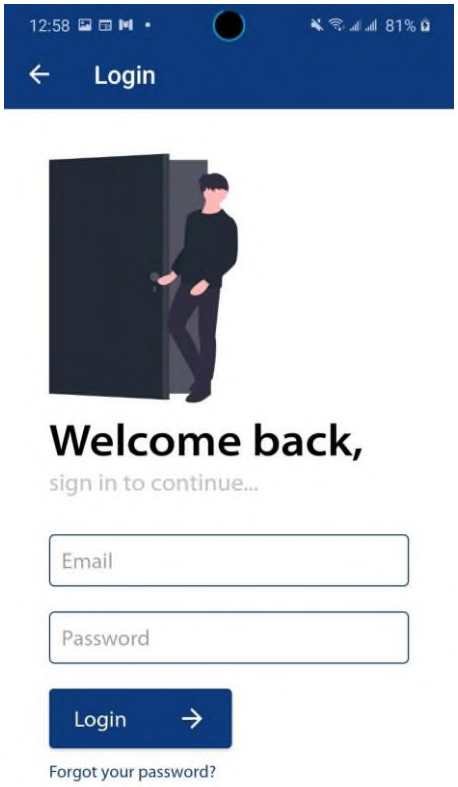


Рисунок 3.2 – Вхід до додатку

Налаштування профілю користувача представлено на рис. 3.3. Даний екран представляє можливості профіля користувача. На даній вкладці є можливість змінити логін, додати фотографію або зображення користувача, додати його опис. Також тут можливо вийти з профілю користувача, після чого користувач потрапляє на екран повторної авторизації.

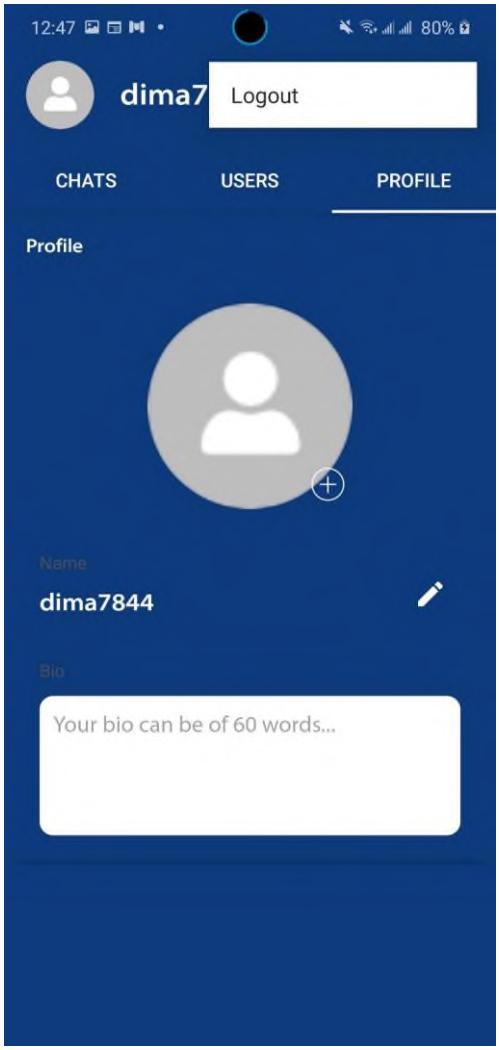


Рисунок 4.3 – Налаштування профілю користувача

На рис. 3.4 представлено основний екран програми з трьома основними вкладками(профіль, список чатів, список користувачів). Навігація між вкладками відбувається за допомогою свайпів. Контейнером для вкладок є клас ViewPager. Вкладка Users відображає список профілів користувачів з якими можна вести листування.

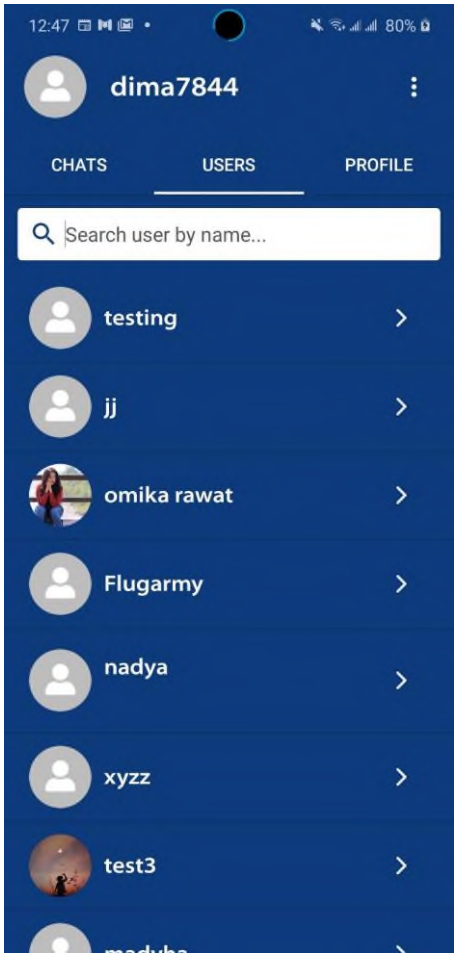


Рисунок 3.4 – Вкладка користувачів, з якими можна вести бесіду

Екран історії листувань виглядає наступним чином (рис. 3.5). Даний екран відображає всіх користувачів з якими відбувалось листування у відповідному порядку. При натисканні на користувача відкривається відповідний чат.

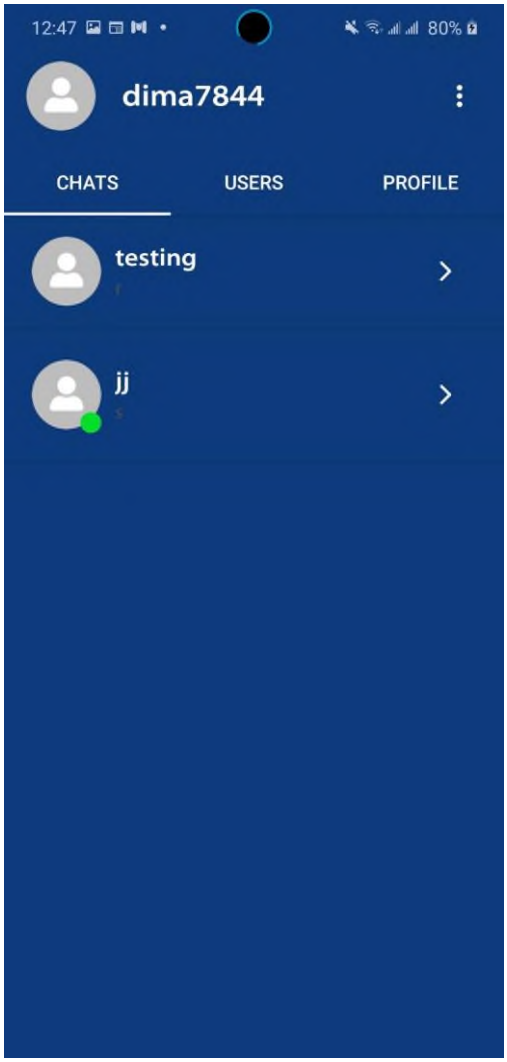


Рисунок 3.5 – Вікно історії листування у додатку

Екран повідомлень виглядає наступним чином (рис. 3.6). Даний екран відображає всю історію повідомлень з детальним відображень інформації(час відправлення, статус повідомлення) з відповідним користувачем. Також даний екран дає змогу відправляти повідомлення.

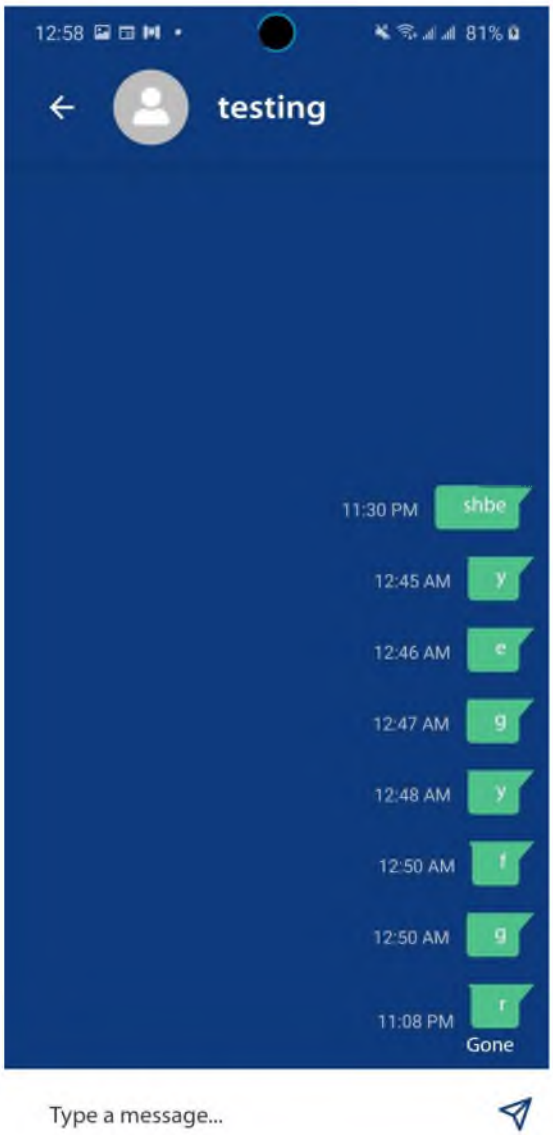


Рисунок 3.6 – Вікно чату додатку

Висновки

Метою даного дипломного проекту є розробка Android-месенджера з криптографічним захистом. Розробка є актуальною у зв’язку з необхідністю забезпечення захисту даних користувача, а також недостатнього рівня захисту у більшості месенджерів. Розроблений додаток надає користувачу можливість використовувати основні функціональні можливості звичайного месенджера, які доповнюються криптографічним захистом.

Аналіз існуючих рішень, показав, що у найбільш відомих мобільних додатків є декілька недоліків, які пов'язані з безпекою даних користувачів. Найбільшою проблемою є відсутність шифрування у таких додатках.

Основні функції та особливості розробленого додатку:

- швидкий користувацький інтерфейс;
- безпечна автентифікація і авторизація користувачів;
- можливість відновлення паролю;
- можливість шифрування повідомлень перед відправленням;
- захист програмного коду за допомогою засобів криптографії;
- реалізація повного функціоналу месенджера;
- безпечне кешування даних.

Розроблений мобільний додаток забезпечує ефективну комунікацію між користувачами.

В розробку месенджера закладена можливість подальшого розширення і модифікації додатку.

Список використаної літератури

1. Класифікація криптоалгоритмів [Електронний ресурс]. - Режим доступу: https://wiki.tntu.edu.ua/Класифікація_криптоалгоритмів. - Дата доступу: квітень 2020.

2. Криптографія [Електронний ресурс]. – Режим доступу: http://esu.com.ua/search_articles.php?id=1576. – Дата доступу: квітень 2020.

3. «Порівнюємо месенджери» [Електронний ресурс]. – Режим доступу: http://androidclub.com.ua/porivnyuyemo-mesendzhery/. – Дата доступу: квітень 2020.

4. «Що таке Android?» [Електронний ресурс]. – Режим доступу: http://ipkey.com.ua/uk/faq/912-android.html. – Дата доступу: квітень 2020.

5. «Порівняння ТОП 10 кращих месенджерів для Андроїд. Який вибрати?» [Електронний ресурс]. – Режим доступу: https: // androidapplications. ru / faq / 4792-10-luchshih-messendzherov-dlya-antroid.html. – Дата доступу: квітень 2020.

6. Д. В. Іртегов. Введення в операційні системи. / Д. В. Іртегов. - 2-е вид. - СПб. : БХВ-Петербург, 2012. - 1040

7. Акбердін Л. «Що таке месенджер? Популярні мобільні месенджери» [Електронний ресурс]. – Режим доступу http://fb.ru/article/139644/chto-takoe-messendjer-populyarnyie-mobilnyie-messendjeryi. – Дата доступу: квітень 2020.

8. Історія версій Android [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%82%D0%BE%D1%80%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D0%B9_Android . – Дата доступу: квітень 2020.

9. Єгоров. Д. Андроїд керівництво користувача-2010. - 430с.

10. Google I/O 2017 [Електронний ресурс]. – Режим доступу: https://events.google.com/io2017/ . – Дата доступу: квітень 2020.

11. Стеценко А. «Який месенджер крутіше - порівнюємо кращі програми для листування» від 9 жовтня 2015 г. [Електронний ресурс]. – Режим доступу: https:// uip. me / 2016/01 / messengers-comparison 2016 / . – Дата доступу: квітень 2020.

12. Firebase [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/database/android/start. – Дата доступу: квітень 2020.

13. Allatori [Електронний ресурс]. – Режим доступу: http://www.allatori.com. – Дата доступу: квітень 2020.

14. Краєва А. «Месенджери - що це таке? Список найпопулярніших. Який месенджер краще? Який вибрати? » [Електронний ресурс]. – Режим доступу: http://alenakraeva.com/new-digital-world/messen-dzhery-chto-eto-takoe/. – Дата доступу: квітень 2020.

15. Фелкер, Донн. Android: розробка додатків для чайників. : Пер. з англ. / Фелкер, Донн. - М.: ТОВ "І.Д. Вільямс ", 2012. - 336 с.

16. П. Дейтел, Х. Дейтел, Е. Дейтел, М. Моргано. Android для програмістів: створюємо додатки. / П. Дейтел, Х. Дейтел, Е. Дейтел, М. Моргано - СПб .: Питер, 2013. - 560 с.

17. Колісниченко Д. Н. Android для користувача. Корисні програми та поради. / Колісниченко Д. Н. - СПб .: БХВ-Петербург, 2013. - 256 с.

18. Патерн проектування MVVM [Електронний ресурс]. – Режим доступу: https://metanit.com/sharp/wpf/22.1.php – Дата доступу: квітень 2020.

19. Гарнаев, Андрій WEB-програмування на Java і JavaScript / Андрій Гарнаев, Сергій Гарнаев. - М .: БХВ-Петербург, 2012. - 179 с.

20. Голощапов, Олексій Google Android. Програмування для мобільних пристроїв / Олексій Голощапов. - М .: БХВ-Петербург, 2011. - 438 с.

21. Дерсі, Лорен Android за 24 години. Програмування додатків під операційну систему Google / Лорен Дерсі, Шейн Кондер. - М .: Рід Груп, 2011. - 464 с.

22. Майер, Рето Android 4. Програмування додатків для планшетних комп'ютерів і смартфонів / Рето Маєр. - М .: Ексмо, 2013. - 816 с.

23. 10. Мартін, К. Соломон Oracle. Програмування на мові Java / Мартін К. Соломон, Нірва Моріссо-Леруа, Джулі Басу. - М .: ЛОРИ, 2010. - 512 с.

24. Компіляція і збірка Android додатку [Електронний ресурс]. – Режим доступу: https://habr.com/ru/sandbox/63285/ . – Дата доступу: квітень 2020.