Київський національний торговельно-економічний університет
Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЄКТ

на тему:

«Модель обміну захищеними даними з використанням звукових сигналів»

Студента 2м курсу, 2 групи, спеціальності 121 «Інженерія програмного забезпечення» спеціалізації «Інженерія програмного забезпечення» _____ підпис студента — Маркевича Володимира Степановича

Науковий керівник pHd, доцент кафедри інженерії програмного забезпечення та кібербезпеки _____ підпис керівника — Десятко Альона Миколаївна

Гарант освітньої програми доктор економічних наук, професор кафедри інженерії програмного забезпечення та кібербезпеки _____ підпис керівника — Токар Володимир Володимирович

КИЇВ - 2021

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення та кібербезпеки
Освітній ступінь магістр
Спеціальність 121 «Інженерія програмного забезпечення»
Спеціалізація «Інженерія програмного забезпечення»

Затверджую
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
"10" листопада 2020 р.

Завдання
на випускний кваліфікаційний проєкт студента

Маркевича Володимира Степановича
(прізвище, ім'я, по батькові)

Тема випускного кваліфікаційного проєкту «Модель обміну захищеними даними з використанням звукових сигналів»

Затверджена наказом ректора від "30" листопада 2020 р. № 3923

2. Строк здачі студентом закінченої проєкту 25 листопада 2021

3. Цільова установка та вихідні дані до проєкту

Мета проєкту розробка мобільного додатку, який використовує звукові сигнали як спосіб обміну даними між користувачами, а також й інші методи за допомогою яких можна здійснювати обмін даними

Об'єкт дослідження виступають обмін захищеними даними з використанням звукових сигналів

Предметом дослідження є мобільний додаток з обміну захищеними даними

4. Консультанти проєкту із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускного кваліфікаційного проєкту (перелік питань за кожним розділом)

Вступ

РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ РОЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ ОБМІНУ ДАНИМИ

1.1. Розвиток ринку месенджерів

1.2. Обґрунтування засобів технічної реалізації

1.3. Архітектура мобільного додатку

1.4. Висновки до розділу

РОЗДІЛ 2. ОГЛЯД МЕТОДІВ СТВОРЕННЯ МОБІЛЬНИХ ДОДАТКІВ

2.1. Аналіз архітектурних шаблонів для розробки мобільного додатку

2.2. Аналіз технологій обміну даними

2.3. Аналіз стандартів шифрування даних

2.4. Висновки до розділу

РОЗДІЛ 3. РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

3.1. Опис ідеї проекту

3.2. Архітектура клієнт-серверного мобільного додатку

3.3. Розробка алгоритмів роботи програми

3.4. Розробка основних екранів додатку

3.5. Висновки до розділу

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ТЕХНІЧНЕ ЗАВДАННЯ

ДОДАТКИ

6. Календарний план виконання проєкту

№ по р.	Назва етапів випускного кваліфікаційного проєкту	Строк виконання етапів проєкту	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускного кваліфікаційного проєкту*	*21.09.2020*	*21.09.2020*
2.	*Розробка та затвердження завдання на проєкт магістра*	*22.12.2020*	*22.12.2020*
3.	*Вступ та перелік літературних джерел*	*27.02.2021*	*27.02.2021*
4.	*Розробка технічного завдання*	*20.03.2021*	*20.03.2021*
5.	*Розділ 1. КОРПОРАТИВНИЙ ВЕБ-ПОРТАЛ ЯК ОСНОВА ПОБУДОВИ ВІДЕО КАНАЛУ*	*16.04.2021*	*16.04.2021*
6.	*Розділ 2. ПІДХОДИ ДО ОРГАНІЗАЦІЇ ВІДЕО-КОНФЕРЕНЦЗВ'ЯЗКУ ЯК ГОЛОВНОЇ КОМПОНЕНТИ ВІДЕОКАНАЛУ КОРПОРАТИВНОГО ВЕБ-ПОРТАЛУ*	*24.05.2021*	*24.05.2021*
7.	*Розділ 3. МОДЕЛЮВАННЯ ТЕХНОЛОГІЙ ПРОВЕДЕННЯ ВІДЕО-КОНФЕРЕНЦЗВ'ЯЗКУ*	*21.06.2021*	*21.06.2021*
8.	*Розробка програми та методики тестування*	*18.10.2021*	*18.10.2021*
9.	*Написання наукової статті*	*22.05.2021*	*22.05.2021*
10.	*Керівництво користувача*	*21.10.2021*	*21.10.2021*
11.	*Висновки та пропозиції*	*01.11.2021*	*01.11.2021*
12.	*Здача випускного кваліфікаційного проєкту на кафедру (перша перевірка)*	*03.11.2021*	*03.11.2021*
13.	*Підготовка автореферату та презентації доповіді*	*03.11.2021*	*03.11.2021*
14.	*Попередній захист випускного кваліфікаційного проєкту*	*22.11.2021 – 25.11.2021*	*24.11.2021*
15.	*Здача зброшурованої випускного кваліфікаційного проєкту*	*25.11.2021*	*25.11.2021*
16.	*Зовнішнє рецензування випускного кваліфікаційного проєкту*	*26.11.2021*	*26.11.2021*
17.	*Підготовка до публічного захисту випускного кваліфікаційного проєкту*		

7. Дата видачі завдання « 10 » листопада 2021 р.

8. Науковий керівник випускного кваліфікаційного проєкту Десятко А.М.

(прізвище, ініціали, підпис)

9. Гарант освітньої програми Токар В.В.

(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Маркевич В.С.

(прізвище, ініціали, підпис)

11. Відгук керівника випускного кваліфікаційного проєкту

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Науковий керівник випускного кваліфікаційного проєкту

_____

(підпис, дата)

Відмітка про попередній захист _____

(ПІБ, підпис, дата)

12. Висновок про випускний кваліфікаційний проєкт

Випускний кваліфікаційний проєкт студента _____Маркевича В.С.___

(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми ___Токар В. В._____

(прізвище, ініціали, підпис)

Завідувач кафедри ___Криворучко О. В._____

(підпис, прізвище, ініціали)

«_____»_____20 ___ р.

АНОТАЦІЯ

Метою роботи є розробка мобільного додатку, який використовує звукові сигнали як спосіб обміну даними між користувачами, а також й інші методи за допомогою яких можна здійснювати обмін даними.

У процесі роботи досліджувалися методи та платформи для створення мобільних додатків..

Результатом роботи мобільний додаток.

Обсяг роботи: 48 сторінок, 24 ілюстрації, 1 формула, 4 таблиці, 32 використаних джерела.

Ключові слова: мобільний додаток, користувач, месенджер, звукові сигнали.

ABSTRACT

The purpose of the work is to develop a mobile application that uses audio signals as a way to exchange data between users, as well as other methods by which you can exchange data.

In the course of work methods and a platform for creation of mobile Applications were investigated.

The result is a mobile application.

Volume of work: 48 pages, 24 illustrations, 1 formula, 4 tables, 32 used sources.

Keywords: mobile application, user, messenger, sound signals.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

СУБД — Система управління базами даних.

ООП — Об'єктно-орієнтоване програмування.

MVC — Model View Controller.

MVP — Model View Presenter.

MVVM — Model View View-Model.

VIPER — View Interactor Presenter Entities Router.

UI — User Interface.

API — Application Programming Interface.

ARPANET — Advanced Research Projects Agency Network.

ICANN — Internet Corporation for Assigned Names and Numbers.

IETF — Internet Engineering Task Force.

БД — База Даних.

ЗМІСТ

ВСТУП

Мобільні додатки стали одним з головних трендів у розвитку інформаційних технологій за останні роки. Основною перевагою мобільних додатків є те, що користувач може отримати доступ до них, перебуваючи в будь-якому місці за допомогою смартфону.

Одним з найпопулярніших типів додатків для мобільних пристроїв є месенджери. Різноманітність функцій та можливостей такого додатку буде найголовнішим чинником у виборі месенджера користувачем. Не завжди у будь-якому місці та в будь-якій ситуації є доступ буде до інтернету або мобільного оператора, тому кожен месенджер має використовуати додаткові ресурси.

Метою роботи є розробка мобільного додатку, який використовує звукові сигнали як спосіб обміну даними між користувачами, а також й інші методи за допомогою яких можна здійснювати обмін даними.

Для досягнення поставленої мети необхідне вирішення наступних завдань:

- проаналізувати сутність задачі вибору програмного забезпечення та проблеми її вирішення;
- дослідити методи розробки мобільних додатків;
- провести аналіз архітектурних шаблонів для розробки мобільного додатку;
- створити програмне забезпечення.

Об'єктом дослідження виступають обмін захищеними даними з використанням звукових сигналів.

					КНТЕУ 121 02-09.МР		
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*			
Зав. каф.		Криворучко О.В.		*27.02.21*	*«Модель обміну захищеними даними з використанням звукових сигналів»*	*Стадія*	*Арку*	*Аркуші*
Керівник		Десятко А.М.		27.02.21		*В*	*4*	*50*
Гарант		Токар В.В.		27.02.21		*Факультет інформаційних технологій 2м курс, 2 група*		
Розробив		Маркевич В.С.		27.02.21	*Вступ*			

Предметом дослідження є мобільний додаток з обміну захищеними даними.

Методи досліджень. Для розв’язання зазначеної проблеми в роботі застосовано методи моделювання архітектури мобільного додатку, композиції логічних структур даних, тестування, та логічного узагальнення отриманих результатів.

Практичне значення одержаних результатів полягає в отриманні результату після реалізації мобільного додатку, що може використовуватись в інших операційних системах, або веб-сервісах.

РОЗДІЛ 1
ТЕОРЕТИЧНІ АСПЕКТИ РОЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ ОБМІНУ ДАНИМИ

1.1. Розвиток ринку месенджерів

Месенджер – це мобільний додаток або веб-сервіс для миттєвого обміну повідомленнями.

Інтернет перестав вважатися розкішшю, смартфони можуть виконувати по кілька завдань одночасно, а месенджерами можна користуватися на будь-яких пристроях. Практично кожна велика інтернет-компанія і соціальна мережа має на своєму сайті маленький і докучливий месенджер, який вічно спливає в правому нижньому кутку екрану [1].

Першим месенжером є KakaoTalk, його показник дорівнює 318,3 кілобита в секунду. Це в 2,7 рази менше, самого витратного Skype.

Друге зайняв WhatsApp компанії Facebook. Саме ідеальне співвідношення у Viber - відмінна якість і відносно невеликі витрати в 497 Кб / сек, що в два рази менше того ж Skype. Tango і Hangouts вже можна віднести до вимогливих, 650 Кб / сек.

Різні типи мобільних додатків заробляють по-різному. На грі, яка подобається користувачам, заробляти можна практично відразу. З проектами, в яких основний капітал - самі користувачі (соціальні мережі, месенджери і т.п.), складніше [2].

Досвід успішних проектів говорить про те, що заробляти їх доведеться, швидше за все, не рекламою. Топові месенджери виключають рекламні

оголошення і банери зі своїх додатків, оскільки вважається, що це знижує інтерес користувачів.

Метод монетизації, використовуваний в WhatsApp, який стягує з користувачів плату в $ 1 [3].

Досвід месенджера LINE показав очевидну, на перший погляд, річ: користувачі готові платити за можливість використовувати при спілкуванні красиві картинки-стікери. Монетизувати любов аудиторії до різного роду смайликів - хороша ідея. LINE тільки на продажу стікерів заробляє десятки мільйонів доларів за квартал [4]. Якщо вдалося створити популярний додаток, то непоганим способом заробітку можуть стати спеціальні акаунти для брендів і зірок. Різні месенджери намагаються монетизувати такі профілі через їх прив'язку до тих же стікері.

Перспективний напрямок монетизації месенджерів – створення власних ігрових продуктів і їх подальша інтеграція в додаток для спілкування. Корейська месенджер Kakao Talk йде цим шляхом і досягає успіхів. Китайське додаток WeChat дає своїм користувачам можливість прив'язувати до аккаунту кредитну карту для здійснення покупок прямо з вікна месенджера.

Популярність глобальних месенджерів продовжує стрімко зростати – WhatsApp і Facebook Messenger в сукупності обслуговують більше 1,3 млрд [5]. Користувачів, китайські месенджери QQ Mobile і WeChat - понад 1 млрд.

Спектр бізнес-моделей простягається від найбільш очевидних і часто використовуються – платна підписка (WhatsApp, QQ Mobile), дзвінки на мобільні та фіксовані номери (SkypeOut, Viber Out), до продажу брендованих сувенірів (LINE, KakaoTalk) і партнерств з медіакомпанії (Snapchat).

Сервіси, що надаються через Інтернет, відчувають вибухове зростання. Ця тенденція збережеться, чому сприяє збільшення числа мобільних пристроїв (смартфонів, планшетів, ноутбуків). На ринку присутня величезна кількість вендорів, що надають свої послуги в цій галузі, але особливою популярністю користуються сервіси, що дозволяють безпечно організовувати спілкування

між організаціями. Простота і гнучкість використання є ключовою вимогою [6].

Cisco пропонує кілька сервісів, що дозволяють користувачам, в тому числі і мобільним, спілкуватися в Інтернеті. Webex доповнено новим рішенням Collaboration Meeting Room (CMR), що дозволяє до одної конференції підключитися абсолютно будь-якому користувачеві незалежно від того, який пристрій йому доступно в даний час (професійна відеоконференцсістема або звичайний смартфон) і в якому місці він знаходиться, - важливо лише наявність Інтернету.

Інтернет все більше проникає в найвіддаленіші куточки світу, тому популярність сервісів, що дозволяють комфортно спілкуватися між собою, буде тільки зростати. Причому вартість сервісів – вторинний аспект: все частіше на перше місце виходить якість, надійність і зручність. Але безкоштовні базові сервіси також будуть популярні серед малих підприємств і недосвідчених користувачів [7].

Cisco чітко відстежує і розуміє потреби ринку, тому в деяких країнах вже запущений проект Spark, що дозволяє безкоштовно спілкуватися один на один і в групах, як в чаті, так і з використанням відео, як через браузер, так і через спеціалізований додаток. Можна вибрати як безкоштовний додаток, так і платне – для більш вимогливих користувачів.

Продовжують розвиватися технології, що дозволяють по-новому поглянути месенджери. Тепер це не просто персональні чати, а багатофункціональні програми з можливістю передачі голосу, відео, інформації з позиціонування та присутності, також здатні надавати загальну середу для спільної роботи [8]. Все це робить роботу користувачів набагато ефективнішою, до того ж в ряді випадків, виходить, значно економити на поїздках – досить включити відео, і проблема вирішена: завдання вирішуються швидше.

1.2. Обґрунтування засобів технічної реалізації

Firebase - Backend-as-a-Service - BaaS, який стартував як стартап YC11 і виріс в платформу розробки додатків наступного покоління на платформі Google Cloud Platform.

Firebase дозволяє розробникам зосередитися на розробці клієнтської частини програми, що позначається на якості розроблюваного ПЗ і враження користувачів. Firebase – це сервер, API та сховище даних. Все написано з урахуванням можливосте його модифікації відповідно до більшісті потреб. Іноді доведеться використовувати інші шматочки Google Cloud для просунутих додатків.

Основні можливості:

- Робота в режимі реального часу (Realtime) - є хмарної базою даних. Дані зберігаються як JSON і синхронізуються в реальному часі з кожним підключеним клієнтом. Коли створюєтбся крос платформні додатки з iOS, Android і JavaScript SDK, всі клієнти діляться одним екземпляром бази даних Realtime і автоматично отримують оновлення з новітніми даними [9].
- Робота в автономному режимі (Offline) - додатки Firebase працюють також в автономному режимі, оскільки SDK Firebase Realtime Database кеширует поточні дані на пристрої. Після відновлення підключення до мережі Інтернет клієнтський пристрій отримує пропущені зміни, синхронізуючи їх з поточним станом сервера.
- Доступ з клієнтських пристроїв (Accessible from Client Devices) - доступ до Firebase Realtime Database можна отримати безпосередньо з мобільного пристрою або веб-браузера, немає необхідності в написанні сервера для додатків. Безпека і перевірка даних доступні через Firebase Realtime Database Security Rules, гнучкий мову правил на основі виразів, які виконуються при читанні або запису даних.

- Firebase Cloud Messaging (FCM) – це кроссплатформне рішення для обміну повідомленнями, за допомогою якого можна відправляти повідомлення пристроїв [10].

Використовуючи FCM, можливо повідомити клієнтську програму про те, що для синхронізації доступні нові електронні або інші дані. Для таких випадків, як обмін миттєвими повідомленнями, повідомлення може передавати корисне навантаження до 4 КБ в клієнтську програму.

Основні можливості FCM:

- відправляти повідомлення або інформацію з даними;
- вказати конкретного користувача або групу, як точку призначення повідомлення;
- відправляти повідомлення з клієнтського застосування.

Реалізація FCM включає в себе два основних компоненти для відправки та отримання (рис. 1.1):

- довірене середовище, таке як хмарні функції для Firebase або сервер додатків, на яких можна створювати, настроювати і відправляти повідомлення.
- додаток iOS, Android або веб-додаток (JavaScript), яке отримує повідомлення.

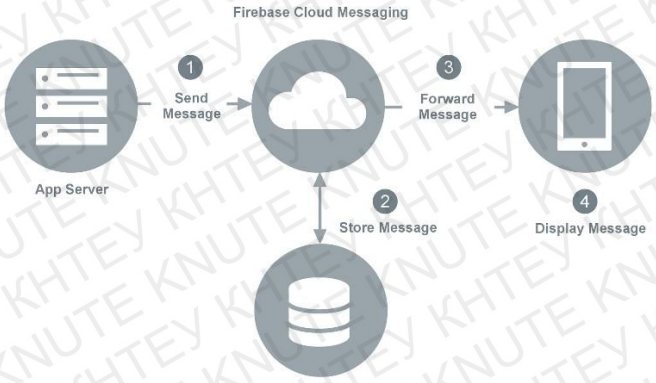

Рис. 1.1. Принцип роботи FCM

- Firebase Storage – хмарне сховище, створене для розробників додатків, яким необхідно зберігати і обслуговувати завантажуються користувачами будь-якого контенту, зображення або відеофайли [11].

- Cloud Storage for Firebase – це масштабна, легко освоювана служба збереження медіа даних. Що включає в безкоштовному тарифі до 10 гб. вільного простору в хмарі.

Програміст може використовувати SDK для читання і збереження аудіофайлів, зображень, відеофайлів або будь-якого іншого формату даних, які можуть знадобитися для відображення контенту. Завантажені файли, можна переглянути в Firebase Console.

Аутентифікация Firebase. Багато додатків повинні ідентифікувати користувача. Ідентифікація дозволяє додатку безпечно зберігати і структурувати призначені для користувача дані в хмарі, забезпечуючи синхронізацію на різних операційних системах пристроїв користувача. Також ідентифікація дозволяє відображати релевантні дані для користувача [12].

Firebase Authentication надає базові служби, прості у використанні SDK і готові бібліотеки призначеного для користувача інтерфейсу для аутентифікації користувачів у додатку. Firebase Authentication підтримує аутентифікацію з використанням:

- пароля і електронної пошти;
- телефонного номера;
- соціальної мережі google;
- соціальної мережі facebook;
- соціальної мережі twitter;
- соціальної мережі gitHub.

Firebase підтримує анонімний вхід. Firebase Authentication безпосередньо інтегрується з іншими сервісами Firebase, використовуючи галузеві стандарти, такі як OAuth 2.0 і OpenID Connect, що дозволяє легко інтегрувати з будь-яким сервером [13].

Визнання компанією Google мови програмування Kotlin, як основного для написання програми під ОС Android, дало сильний поштовх розвитку цієї

мови програмування. Будучи статичним універсальна мова програмування для сучасних мультиплатформених додатків.

Kotlin походить від індустрії, а не від академічних кіл. Kotlin вирішує проблеми, з якими стикаються програмісти. Як приклад система типів допомагає уникнути виключень з нульового покажчика (Null Pointer Exception - NPE). Дослідницькі мови, як правило, не мають нульового значення, але це марно для людей, які працюють з великими кодовими базами і API-інтерфейсами.

Дана мова програмування повністю сумісний з Java і Android. Перевагами мови програмування Kotlin є:

- сумісність: Kotlin сумісний з JDK 6. Даний факт гарантує, що нові або переписані на Kotlin додатки будуть повністю підтримуватися усіма пристроями на Android.
- продуктивність: факт, що структура байт-коду схожа, додаток, написане на Kotlin не поступається додатком, написаним на Java. Підтримка Kotlin вбудованих функцій, використовують lambda - вираження, позначається на додатку в кращу сторону (рис. 1.2).

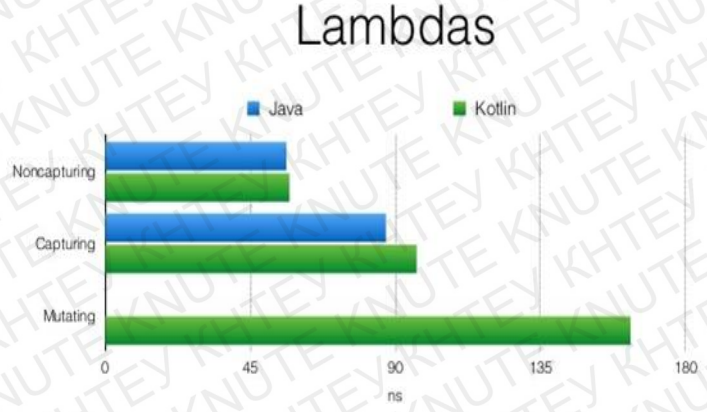

Рис. 1.2. Порівняння швидкості виконання функції

- взаємодія: сумісність між Java і Kotlin сто відсоткова, що дозволяє використовувати бібліотеки, написані на Java, використовувати в Kotlin додатку і навпаки [14].

- footprint: Kotlin володіє дуже компактною бібліотекою часу виконання коду, показали швидкості роботи якої, можна ще поліпшити за допомогою ProGuard. У реальному додатку розмір apk - файлу збільшується на 100 КБ в розмірі файлу.
- час компіляції: Kotlin використовує ефективну інкрементного збірку, що за фактом збільшувати швидкість збірки проекту. Java проект може збирати і по 30 хвилин, Kotlin збере проект за пару хвилин.
- поріг входження: для програміста, який знає мову програмування Java, не важко буде почати програмувати на мові програмування Kotlin.

Приклад компіляції проекту, написаного на двох мовах програмування: Kotlin (файли з розширенням *.kt) і Java (файли з розширенням *.java), можна побачити на рис. 1.3 [15].

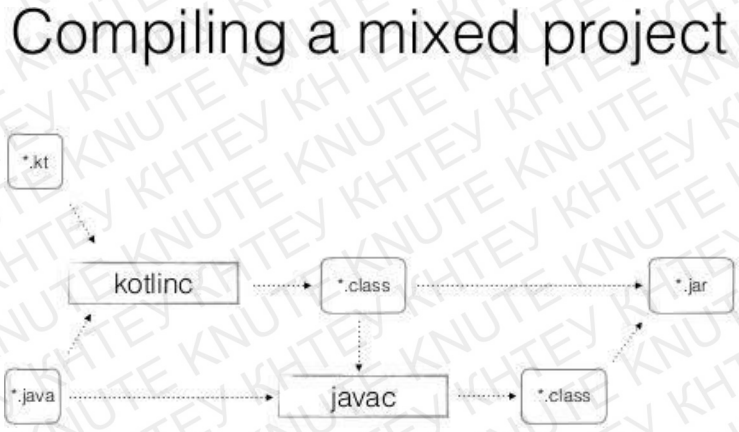

Рис. 1.3. Компіляція проекту

Наступною популярною мовою для написання мобільних додатків є Android Studio – офіційна середовище для розробки Android додатків. Заснована середовище розробки на базі IntelliJ IDEA. Вона була створена для прискорення процесу розробки і якості додатків під операційною системою Android. Основні можливості [16]:

- система збирання Gradle;
- швидкий, що настроюється і багатофункціональний емулятор;
- розробка під все Android-пристрої в одному середовищі;

- Instant App - миттєвий запуск програми, після внесених змін;
- база готових додатків, інтеграція з GitHub, щоб створювати додатки в команді, ділиться кодом і імпортувати код;
- Lint - інструмент для відстеження стану програми, його продуктивності, сумісності з іншими версіями;
- підтримує мову програмування C ++ і NDK;
- інтегрована підтримка сервісів Google, що дозволяє швидко інтегрувати Firebase, Google Cloud Messaging.

Всі проекти в Android Studio включають в себе один або більше модулів з вихідним кодом, також містить дані ресурсів, такі як розмітка сторінок, рядки, стилі і загальні константи. Модулі бувають трьох типів:

- модулі додатків для Android;
- бібліотеки, написані в якості модуля (рис. 1.4);
- модулі Google App Engine.

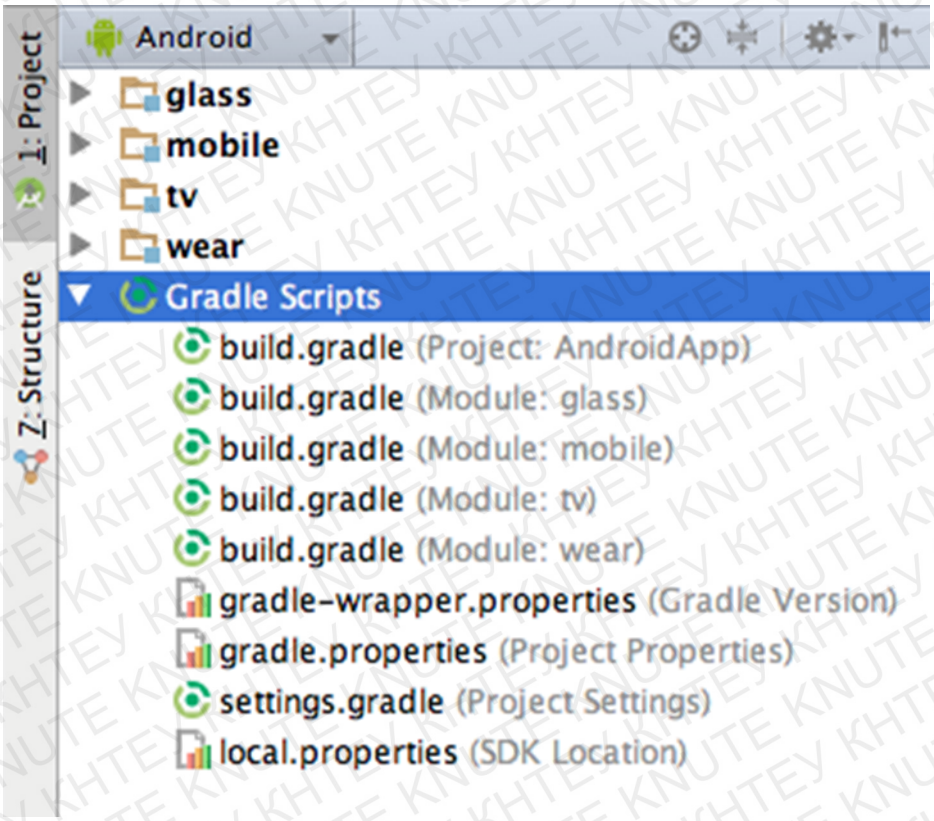

Рис.1.4. Бібліотеки, написані в якості модуля

1.3. Архітектура мобільного додатку

Вибір архітектури для мобільного додатка - це один з найважливіших етапів проектування мобільного додатка. Так, як від зробленого вибору буде залежати його подальша масштабованість та можливість розвитку.

Існує безліч статей і прикладів архітектури MVP (Model- View-Presenter), та існує безліч різних реалізацій. Спільнота розробників постійно прагне адаптувати цей шаблон до Android найкращим чином. Взаємодія класів показано на рис. 1.5.

Model: це інтерфейс, відповідальний за управління даними. Обов'язки моделі включають використання API-інтерфейсів, кешування даних, управління базами даних. Модель також може бути інтерфейсом, який взаємодіє з іншими модулями, що відповідають за ці обов'язки. Якщо використовується шаблон Repository, модель може бути Repository. Якщо використовується Clean Architecture, натомість модель може бути Interactor.

Presenter: Presenter - прошарок між Model і View. До нього належить вся логіка уявлення. Presenter відповідає за запит Model і оновлення View, реагуючи на взаємодію користувачів з оновленням моделі.

View: несе відповідальність тільки за подання даних так, як це було прийнято наказано Presetner. Подання може бути реалізовано за допомогою Activities, Fragments, будь-якого віджета Android або все, що може виконувати операції, такі як показ ProgressBar, оновлення TextView, заповнення RecyclerView [17].

Рис. 1.5. Схема «спілкування» Model-View-Presenter

Однією з найбільших проблем Android є те, що Activities та Fragments непросто перевірити через складність структури. Щоб вирішити цю проблему,

потрыбно реалізували Passive View шаблон. Реалізація цього шаблону зменшила поведінку уявлення до абсолютного мінімуму за допомогою контролера Presenter. Цей вибір значно покращує тестування.

Presenters, використовують тільки залежності Java, з двох причин: по-перше, абстрагуємося Presenter від деталей реалізації (Android framework), і, отже, можливо писати не інструментальні тести для Presenter (навіть без Robolectric - фреймворк для тестування Android модулів), швидше запускати тести на локальній JVM і без емулятора. Якщо раптом знадобиться мати в Presenter Android-модулі, то, по-перше, спочатку треба переконатися, що це дійсно необхідно, а, по-друге, щоб це зробити потрібно використовувати DI - принцип, щоб підвищити тестованих за допомогою Robolectric [18].

На рис. 1.6 представлені класи, що реалізують шаблон проектування MVP на прикладі реалізації екрану чату. У ChatActivity - Kotlin клас. ChatActivity потрібен для хостингу фрагмента ChatFragment. ChatFragment - відображає отримані повідомлень, які не реалізуючи ніякої логіки.

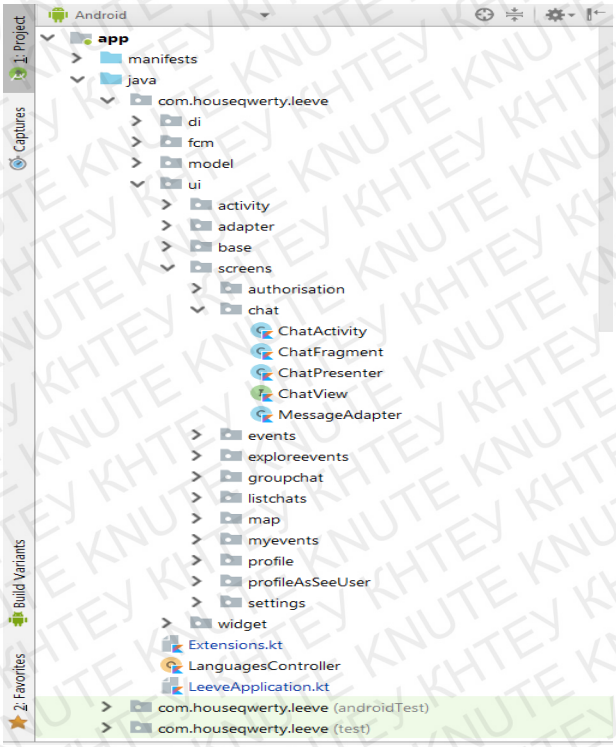

Рис. 1.6. Класи, шаблону MVP

Даний клас реагує на команди від ChatPresenter. MessageAdapter служить для відображення повідомлення на екрані. ChatPresenter реагує на зміну в базі даних. (рис. 1.7).

```
package com.houseqwerty.leeve.ui.screens.chat

import ...

/**
 * Created by Houseqwerty on 11.10.2017.
 */
@StateStrategyType(AddToEndSingleStrategy::class)
interface ChatView : BaseView {

    fun addMessage(message: Message)
    fun showEmptyMessageError()
    fun showErrorSendMessage()
    fun clearSendMessageField()
    fun setCurrentUserPhoto(url: String)
    fun setUserPhoto(url: String)
    fun setUserLogin(login: String)

}
```

Рис. 1.7. Команди, ChatFragment

1.4. Висновки до розділу 1

В першому розділі розглядались теоретичні аспекти розробки мобільних додатків обміну даними. Проаналізовано ринок месенджерів та етапи розвитку ринку.

В розділі «обґрунтування засобів технічної реалізації» були розглянуті основні технічні засоби реалізації даної задачі та переваги кожної з систем та мов програмування.

В розділі «Архітектура мобільного додатку» була розглянута класична архітектура мобільного додатку та методи її реалізації.

РОЗДІЛ 2

ОГЛЯД МЕТОДІВ СТВОРЕННЯ МОБІЛЬНИХ ДОДАТКІВ

2.1. Аналіз архітектурних шаблонів для розробки мобільного додатку

Існує безліч різних архітектурних шаблонів для розробки мобільних IOS додатків. Вибір архітектури майбутнього додатки дуже важливий етап проектування. Правильний вибір архітектури, дозволить проводити пошук помилок та швидко реагувати на зміни всередині класу. Існує безліч варіантів шаблонів проектування архітектури:

- MVC;
- MVP;
- MVVM;
- VIPER.

MVC, MVP, MVVM припускають включення об'єктів додатки в одну з трьох категорій:

- Models - відповідальні за дані домену або рівень доступу до даними, який маніпулює даними;
- Views - відповідальні за рівень представлення (Graphics User Interface), всі елементи з префіксом "UI";
- Controller, Presenter, ViewModel - виконує роль посередника між Model і View, що відповідає за зміни Model. Реагуючи на дії користувача, що виконуються під View і оновлюючи View зі змінами з Model.

При використанні шаблону традиційного MVC, View не має стану, він просто відображається контролером після зміни моделі [19]. Таким чином, цей шаблон на увазі повне перезавантаження сторінки після будь-якої дії користувача. Таким чином, реалізація традиційного MVC в додатку IOS не має сенсу через архітектурної проблеми - всі три об'єкта пов'язані один з другом і кожна сутність знає про двох інших (рис.2.1).

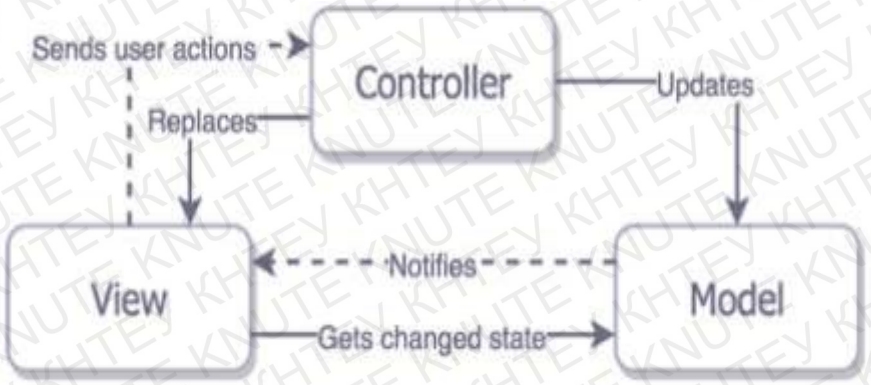

Рис. 2.1. Традиційний MVC

У шаблони архітектури Apple MVC контролер буде посередником між View і Model, що б ці два об'єкти не знали про існування один одного [20]. Найбільш часто використовуваним об'єктом буде бути контролер. У цьому є плюси, тому що в додатку повинно бути місце для всієї складної бізнес-логіки, яка не вписується в Model (рис. 2.2.).

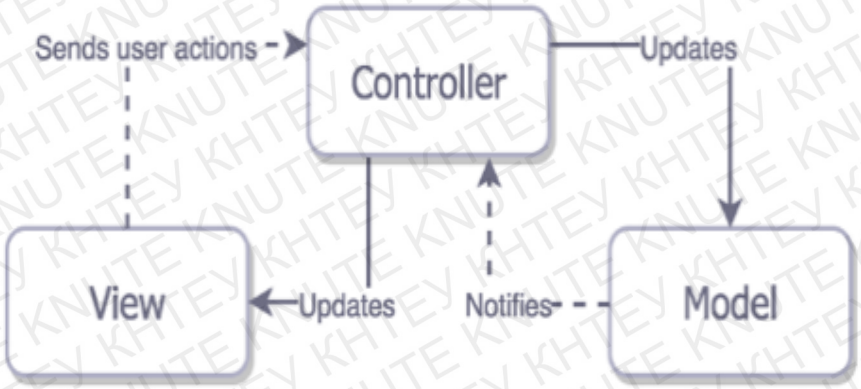

Рис. 2.2. Apple MVC

В реальності шаблон Apple MVC працює по-іншому. Перевантажений контролер містить в собі велику кількість бізнеслогікі, перетворюючи Model View-Controller в Massive-View-Controller. Cocoa MVC заохочує писати Massive-View-Controller, тому що вони дуже сильно залучені в життєвий цикл View, що призводить до нероздільності цих двох сутностей (рис.2.3).

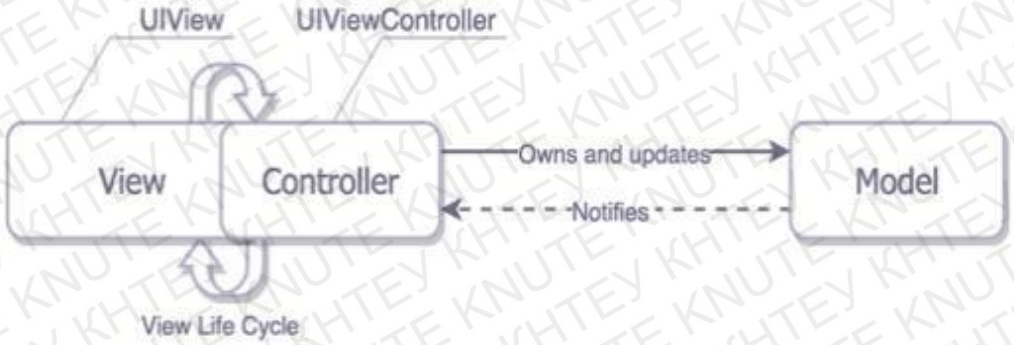

Рис. 2.3. Практичний Apple MVC

MVP (Cocoa MVC's promises delivered) схожий на Apple MVC, але це не так [21]. В Apple MVC - View пов'язаний з контролером, в той час як медіатор MVP - Presenter, не має нічого спільного з життєвим циклом контролера, тому в Presenter немає коду для установки макета екрану, але він відповідає за оновлення View з актуальними даними і станом. MVP це архітектурний шаблон, який показує проблеми збірки, яка виникає через наявність трьох фактично розділених шарів (рис.2.4).

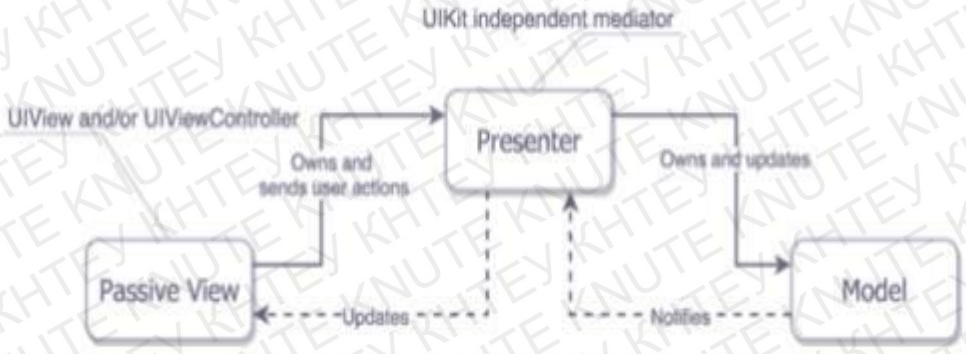

Рис. 2.4. MVP

Існує й інший формат MVP - MVP Supervising Controller [22]. Цей варіант включає пряму залежність View і Model, в той час як Presenter (Supervising Controller) все ще обробляє дії з View і здатний його змінювати. Поділ невиразною відповідальності, є неприпустимим, так само як і сильна зв'язок View і Model (рис. 2.5).

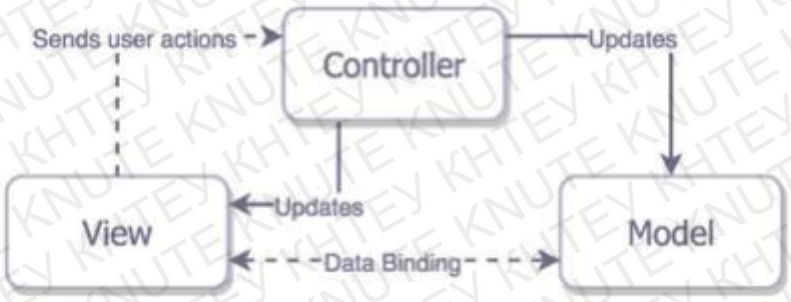

Рис. 2.5. MVP (With Bindings and Hooters)

MVVM розглядає контролер View, як View, а так само має на увазі відсутність зв'язку між View і Model [2]. View-Model забезпечує незалежне уявлення View і його стану. View-Model викликає зміни в Model і оновлює себе з оновленою Model, а оскільки View і View-Model з'єднані, View відповідно теж оновлюється (рис. 2.6).

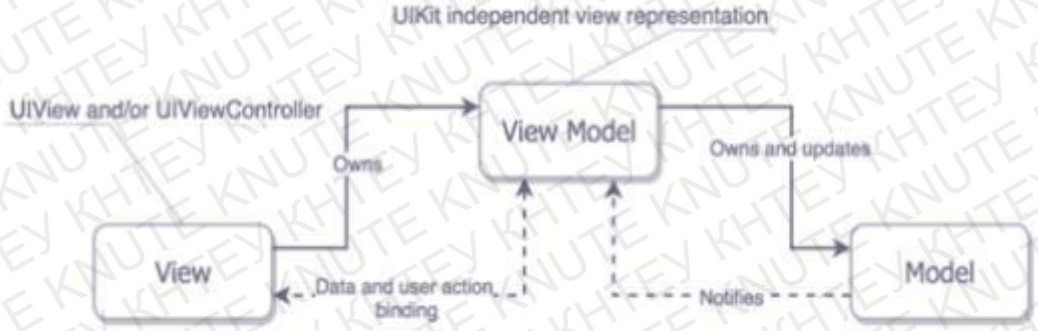

Рис. 2.6. MVVM

VIPER дозволяє повністю розділити обов'язки, і забезпечити повну незалежність модулів [24]. Він складається з:

- View - відповідає за виконання сталися в Presenter змін;

- Interactor - містить бізнес-логіку, пов'язану з даними (Entities) або мережевими пристроями. У його обов'язки входять такі операції, як створення нових екземплярів об'єктів, або вибірку їх з сервера. Для цих цілей використовуються служби і менеджери, які не є частиною модуля VIPER - вони є зовнішньої залежністю;
- Presenter - містить пов'язану з UI (незалежну від UIKit) бізнеслогіку, викликає методи в Interactor;
- Entities - в цьому класі зберігаються прості об'єкти даних;
- Router - відповідає за переходи між різними модулями VIPER.

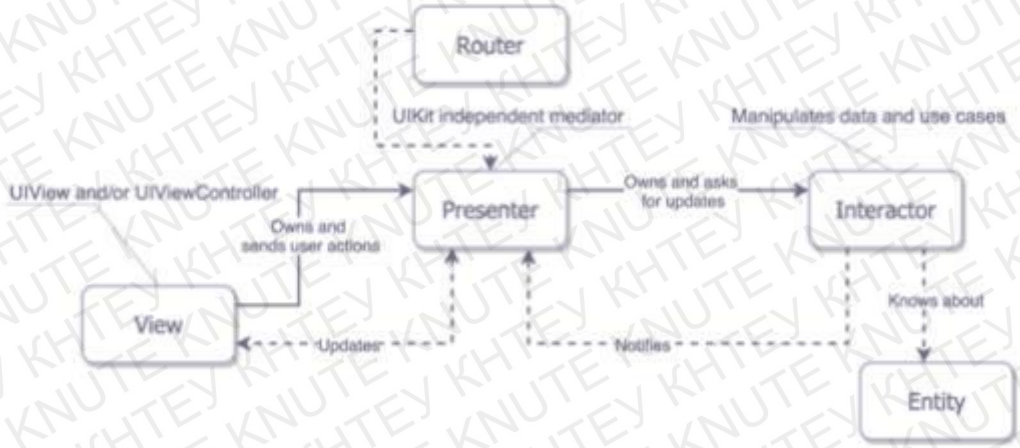

Рис. 2.7. VIPER

Модуль VIPER може відповідати як одному екрану в додатку, так і всьому додатком, чітких вказівок як вибрати розмір модуля не наводиться. VIPER - це перший шаблон в якому явно розглядається навігаційна відповідальність, яка вирішується за допомогою Router.

2.2. Аналіз технологій обміну даними

Force (IETF) - проводить технічне обґрунтування і стандартизацію основних протоколів (IPv4 та IPv6). Це некомерційна організація, являє собою

відкриту міжнародну спільнота учених, проектувальників, мережевих операторів і постачальників послуг [25].

Інтернет побудована на маршрутизації пакетів даних і використанні протоколу IP. В наш час ця мережа слугує фізичною основою доступу до вебсайтів і багатьох систем (протоколів) передачі даних, а також відіграє важливу роль у створенні інформаційного простору глобального суспільства [25].

Звукові сигнали є новим та цікавим способом обміну даними. Технологія Chirp Connect дозволяє додаткам надсилати та отримувати інформацію за допомогою звуку. Chirp кодує масив байтів як аудіосигнал, який може бути переданий будь-яким пристроєм з динаміком і отриманий будь-яким пристроєм з мікрофоном і Chirp Connect SDK. Він розроблений таким чином, щоб бути надійним на відстані декількох метрів у шумному, повсякденному середовищі.

Оскільки передача відбувається цілком за допомогою аудіосигналів, не потрібно підключення до Інтернету або попереднього з'єднання, а будь-який пристрій у межах діапазону слуху може отримувати дані.

Сигнали Chirp Connect можуть бути згенеровані на пристрої з корисної навантаження динамічних даних або записуються як аудіофайл для подальшого відтворення, наприклад звуковий штрих-код [26].

Ця технологіє є адаптивною для багатьох платформ та технологій такі як iOS, Android, Arm, JavaScript, Python, macOS, Windows, WebAssembly, Audio API [27].

2.3. Аналіз стандартів шифрування даних

Однією з основних цілей криптографії є захист інформації від несанкціонованого доступу та використання при передачі і зберіганні даних. Для реалізації даної можливості інформація проходить процедуру

шифрування, в процесі якої відбувається перетворення відкритих даних в зашифровані з використанням параметрів, визначених для обраної системи шифрування. При процесі розшифрування, відбувається зворотне перетворення зашифрованих даних у відкриті. Як в прямому, так і в зворотному процесі використовуються ключі.

Існуючі алгоритми шифрування можна розділити на два класи:

- симетричні – створені для шифрування і дешифрування. Використовується один і той же ключ;
- асиметричні – створені для шифрування і дешифрування. Використовується два ключі, відкритий і секретний, відповідно.

Для реалізації симетричних алгоритмів характерні наступні основні ознаки:

- для шифрування і розшифрування застосовується один і той ж алгоритм;
- для шифрування і розшифрування застосовується один і той ж секретний ключ.

Симетричні алгоритми поділяються на [28]:

- потокові, в яких відбувається шифрування потоку даних. Такий підхід необхідний в тих випадках, коли інформацію неможливо розбити на блоки - скажімо, якийсь потік даних, кожен символ яких повинен бути зашифрований і відправлений куди-небудь, не чекаючи інших даних, достатніх для формування блоку. Тому алгоритми поточного шифрування шифрують дані побітово або посимвольний. Деякі класифікації не поділяють блочне і потокове шифрування, вважаючи, що потокове шифрування – це шифрування блоків одиничної довжини.
- блокові - дані шифруються по блоках. Інформація розбивається на блоки фіксованої довжини (наприклад, 64 або 128 біт), після чого ці блоки по черзі шифруються. Причому, в різних алгоритмах шифрування або навіть в різних режимах роботи одного і того ж алгоритму блоки можуть шифруватися незалежно один від одного або "зі зчепленням" - коли

результат шифрування поточного блоку даних залежить від значення попереднього блоку або від результату шифрування попереднього блоку.

- шифри перестановки (permutation, P-блоки) – перестановка елементів тексту, які залишаються незмінними;
- шифри заміни (підстановки, substitution, S-блоки) – окремі символи вихідної інформації замінюються на інші зберігаючи при цьому своє становище в потоці інформації:
  - моноалфавитной (код Цезаря),
  - поліалфавітних (шифр Вижинера, циліндр Джефферсона, диск Уетстоун, Enigma) [29].

Складові:

- DES (Data Encryption Standard, США);
- AES (Advanced Encryption Standard, Rijndael);
- FEAL-1, а потім FEAL-4, FEAL-8, (Fast Enciphering Algoritm, Японія);
- IDEA / IPES (International Data Encryption Algorithm);
- Improved Proposed Encryption Standard (фірма Ascom-Tech AG, Швейцарія);

Також блочне шифрування можливо:

- Без зворотного зв'язку (ОС). Кілька бітів (блок) оригінального тексту шифруються одночасно, і кожен біт оригінального тексту впливає на кожен біт шифротексту. Однак взаємовпливу блоків немає, тобто два однакових блоку вихідного тексту будуть представлені однаковим шифротексту. Тому для шифрування випадкової послідовності бітів (наприклад, ключів) можна використовувати тільки подібні алгоритми.

З зворотним зв'язком. Зазвичай ОС організовується так: попередній шифрований блок складається по модулю 2 з поточним блоком. В якості першого блоку в ланцюзі ОС використовується ініціалізуючі значення. Помилка в одному біту впливає на два інших блоки - помилковий і наступний

за ним. Приклад - DES в режимі CBC (Cipher Block Chaining, ланцюжок блоків) [30].

В системах захисту інформації використовуються складові алгоритми, які поєднують в собі як перестановку, так і заміну (підстановку) блоків даних, що дозволяє зробити систему більш стійкою до злому.

Коротка інформація з історичною довідкою про деякі сучасних стандартах шифрування представлена в таблиці (4.1). дані про характеристиках: розмірність ключів, блоків інформації, кількість циклів зміни даних, деяких алгоритмів шифрування наведені в таблиці (4.2).

Мережа Фейстеля. Блокові шифри, які використовують послідовність перестановок і підстановок, називають мережею перестановок-підстановок або SP-мережею.

Більшість сучасних блокових алгоритмів, відносяться до так званим мереж Фейстеля. Це алгоритми або стандарти DES, Lucifer, FEAL, CAST, Blowfish і інші [31], [32].

На вхід блочного алгоритму шифрування подається блок відкритого тексту довжиною m-бітів і ключ K. Блок відкритого тексту поділяється на дві частини - подблоки N1 і N2. Якщо розміри частин рівні, таку архітектуру називають класичною або збалансованої мережею Фейстеля. Якщо частини N1 і N2 не рівні, то алгоритм називають розбалансованою мережею Фейстеля.

Всі раунди обробки проходять по одній і тій же схемі. Спочатку виконується операція підстановки. Вона полягає в застосуванні до подблоков N2 деякої функції F використовує значення ключа, і подальшому додаванні отриманого результату з подблоков N1 за допомогою логічної операції XOR. Для всіх раундів функція раунду має одну і ту ж структуру, але залежить від параметра - підключа раунду. Підключи раундів відрізняються від загального ключа і один від одного і обчислюються на основі загального ключа K. Після підстановки виконується перестановка, що представляє собою обмін місцями двох половин блоку даних.

Результат і-го раунду визначається за результатом попереднього раунду:

$$\begin{cases} N1(i+1) = N2(i) \\ N2(i+1) = N1(i) \oplus F\big(N2(i), K(i)\big) \end{cases} \quad (2.1)$$

де

і – раунд;

N1, N2 - подблоки відкритого тексту;

K (i) - ключ раунду I;

Якщо параметри функції F змінюються після шифрування кожного наступного символу, то мережу Фейстеля є гетерогенною. Якщо параметри функції F не змінюються, то мережу - гомогенна.

Шифр, який використовує таку конструкцію звернемо. Процес розшифрування шифру Фейстеля принципово не відрізняється від процесу шифрування. Застосовується той же алгоритм, але на вхід подається шифрований текст, а підключи використовуються в зворотній послідовності.

Якщо розглядати криптоаналітичних стійкість алгоритму мережі Фейстеля, який є базою для симетричних алгоритмів шифрування, то можна виділити наступні основні параметри, від яких буде залежати стійкість алгоритму до злому:

- Розмір блоку. Чим більше розмір блоку, тим вище надійність шифру, але швидкість виконання операцій шифрування і розшифрування знижується.
- Розмір ключа. Чим довше ключ, тим вище надійність шифру, але при цьому швидкість виконання операцій шифрування і розшифрування знижується.
- Число раундів обробки. Чим більше число раундів шифрування, тим більше ускладнюється криптоаналіз шифру. У загальному випадку число раундів повинен вибиратися так, щоб всі відомі методи

криптоаналізу вимагали більше зусиль, ніж аналіз за допомогою перебору всіх ключів.

- Алгоритм обчислення подключей. Чим складніше алгоритм обчислення подключей, тим більшою мірою ускладнюється криптоаналіз шифру.
- Функція раунду. Застосування гетерогенних і розбалансованих мереж Фейстеля може значно поліпшити характеристики шифру.

Основним недоліком симетричного шифрування є необхідність передачі ключів "з рук в руки". Цей недолік дуже серйозний, оскільки унеможливлює використання симетричного шифрування в системах з необмеженим числом учасників.

Стандарт DES. Блоковий алгоритмом з закритим ключем DES використовує мережу Фейстеля.

Наразі алгоритм DES широко використовувався при зберіганні і передачі даних між різними обчислювальними системами; в електронних системах, в поштових системах креслень, при електронному обміні комерційною інформацією.

DES володіє двома дуже важливими якостями блокових шифрів: лавини і повнотою:

Лавинний ефект або розсіювання означає, що невеликі зміни в початковому тексті або ключі викликають значні зміни в зашифрованому тексті.

Ефект повноти полягає в тому, що кожен біт зашифрованого тексту повинен залежати від багатьох бітів вихідного тексту. При шифруванні алгоритмом DES повідомлення ділиться на блоки по 64 біта. Для кожного раунду відбувається генерація ключів з 64-бітного вихідного ключа, в якому значущими є тільки 56 бітів (7 байтів або 7 символів в ASCII). У процесі роботи алгоритму використовуються таблиці, за допомогою яких здійснюється шифрування інформації. Блок схема алгоритму DES представлена на рис. 2.8.

Для кожного раунду алгоритму відбувається генерація ключів з початкового ключа. Під час перетворення відбувається циклічний зсув вліво на один або два біта в залежності від раунду, потім застосовується таблиця перестановок, після чого для остаточного раундового ключа відбираються 48 біт з 56 біт, тобто відбувається стиснення ключа. До вихідного блоку інформації застосовується початкова перестановка, для здійснення якої використовується таблиця початкової перестановки (в реалізації des_IP_table). Після цього блок інформації розбивається на дві частини по 32 біта: ліва і права частини, до яких застосовується мережу Фейстеля протягом 16 раундів, при яких дані об'єднуються з ключем. На заключному етапі після 16 раунду права і ліва частини об'єднуються, виконується зворотна перестановка, для здійснення якої використовується таблиця зворотного перестановки (des_invIP_table).

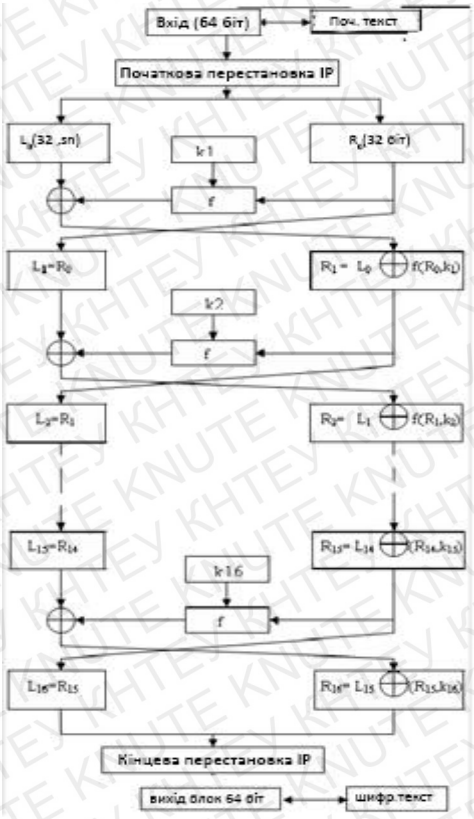

Рис. 2.8. Блок-схема алгоритму DES. Шифрування і розшифрування

На рис. 2.9 представлена схема перетворень, що відбуваються в одному раунді алгоритму.

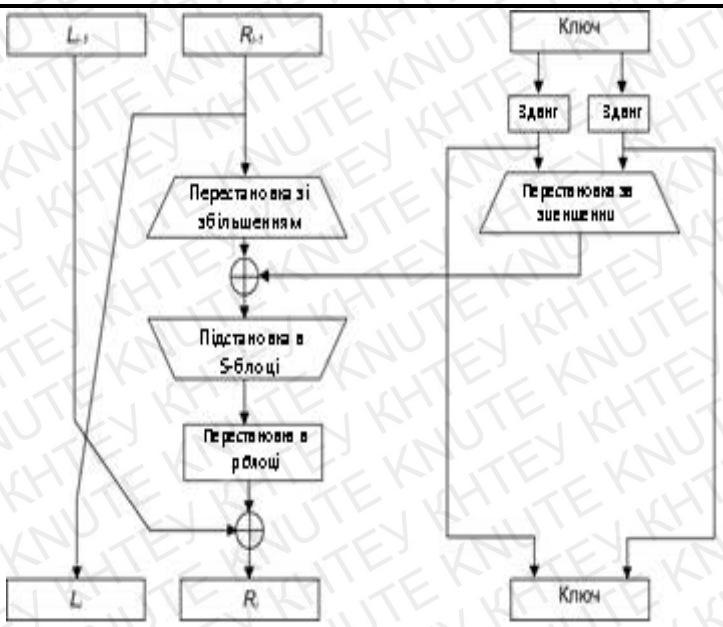

Рис. 2.9. Раунд алгоритму DES

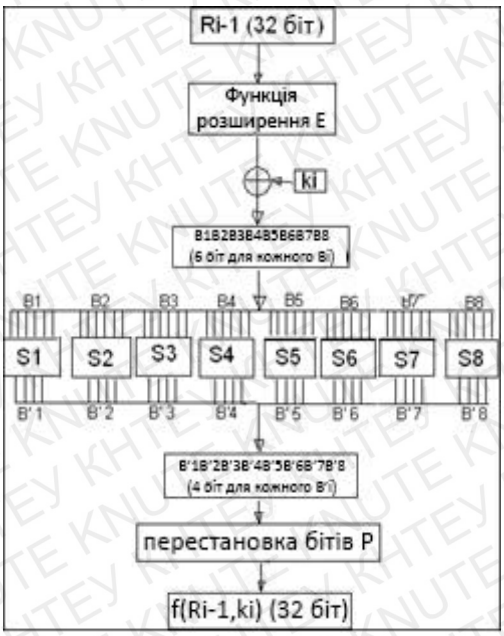

Рис. 2.10. Представлення функції F алгоритму DES

Функція F описує дії в раунді являє собою послідовні перетворення, при яких дані об'єднуються з ключем (формула 1.1).

1. Застосування функції розширення E. Одна з двох частин інформації, наприклад, права, розміром 32 біт розширюється до 48 біт. Дублювання біт вихідної інформації здійснюється відповідно до таблиці розширення.

2. Операція XOR даних, отриманих на першому кроці, і ключа. Отриманий результат представляється у вигляді 8 блоків по 6 біт.
3. Застосування таблиці перетворення до отриманий блокам. Кожен з 8 блоків по 6 біт за допомогою таблиці перетворення трансформується в блок по 4 біта.
4. Застосування таблиці перестановки. До отриманих 32 бітам (8 блоків по 4 біта) застосовується таблиця перестановки і в результаті буде отримано перетворений блок. При переході до наступного раунду права і ліва 32-бітові частини міняються місцями. Кроки з 1 по 4 повторюються.

Після закінчення всіх раундів перетворення, отримані 64 біта піддаються інверсії, для здійснення якої використовується таблиця зворотного перестановки (des_invIP_table) і, таким чином, буде отриманий зашифрований текст. Раундові ключі формуються з початково ключа. Додаються біти в позиції 8, 16, 24, 32, 40, 48, 56, 64 ключа k таким чином, щоб кожен байт містив непарне число одиниць. Далі до ключу застосовується таблиця перестановки, для кожного раунду застосовується циклічний зсув вліво, після чого із значущих 56 біт ключа вибирається 48 біт відповідно до таблиці. Розшифрування відбувається в зворотному порядку з використанням того ж ключа.

З самого початку використання алгоритму криптоаналитики усього світу докладали багато зусиль для злому DES. Фактично DES дав різнобічний розвиток криптоанализа. Багато праці, присвячені різним методам криптоаналізу саме в додатку до алгоритму DES, а також деталей самого алгоритму і їх впливу на криптостойкость. Можна стверджувати, що в результаті проведених досліджень з'явилися цілі напрямки криптоанализа, такі як:

- лінійний криптоаналіз;
- диференціальне криптоаналіз;

- аналіз залежностей між відкритим текстом і шіфротекста;
- аналіз залежностей між співвідношеннями двох або більше відкритих текстів і відповідних їм шіфротекста;
- кріптоаналіз на пов'язаних ключах;
- пошук та аналіз залежностей між шіфротекста, отриманими на шуканому ключі і ключах, пов'язаних передбачуваним співвідношенням з шуканим ключем; проте DES виявився невразливий до даного виду атак, так як по ключовому розкладом циклічний зсув бітів ключа виконується на різну кількість позицій в різних раундах;
- наявність слабких ключів. Зусилля по злому DES робилися з 90-х років минулого століття: Японський фахівець Міцуру Мацуї (Mitsuru Matsui), винахідник лінійного криптоаналізу, в 1993 році показав, що обчислити ключ шифру можна методом лінійного криптоаналізу при наявності у атакуючого 247 пар «Відкритий текст - шифротекст» [20].

Криптологи з Ізраїлю, винахідники диференціального криптоаналізу Елі Біхам (Eli Biham) і Аді Шамір (Adi Shamir) в 1991 році представили атаку, в якій ключ шифрування обчислювався методом диференціального криптоаналізу за умови, що атакуючий має 247 спеціально обраних пар «відкритий текст - шифротекст» [20].

Надалі ці атаки були кілька посилені (наприклад, атака лінійним Криптоаналіз при наявності 243 пар замість 247).

Як видно, всі атаки вимагають наявності величезної кількості пар «Відкритий текст - шифротекст» (таблиця 4.3), отримання яких на практиці є настільки трудомісткою операцією, що найбільш простий атакою на DES все ще можна вважати повний перебір.

2.4. Висновки до розділу 2

В розділі «Огляд методів створення мобільних додатків» розглядались засоби та методи створення мобільних додатків та обирався найоптимальніший метод створення месенжера.

В підрозділі «Аналіз архітектурних шаблонів для розробки мобільного додатку» здійснювався аналіз класичних систем та методів реалізації програми. Для створення мобільного додатку було обрано Firebase від Google, як найоптимальніша система з максимальної кількістю можливості.

В підрозділі «Аналіз технологій обміну даними» аналізувались технології обміну даними. Зважаючи на те, що в дана система повинна виконувати функцію «обміну даними з використанням звукових сигналів» було знайдено найоптимальнішей метод обміну звуковими даними.

В підпрозділі «Аналіз стандартів шифрування даних» були розглянуті найпопулярніші методи шифрування даних. Зважаючи на те, що програма повинна виконувати функцію «обміну захищеними даними» було обґрунтовано використання системи Firebase, як тієї, що автоматично оновлює алгоритми шифрування з урахуванням стану кібербезпеки в інтернеті.

РОЗДІЛ 3

РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

3.1. Опис ідеї проекту

Для виконання завдання створення «Мобільного додатку обміну захищеними даними з використанням звукових сигналів" використовується платформа Firebase на Google Cloud Platform. Обґрунтування використання даної системи є те, що дана система є найоптимізованішою у порівнянні з іншими мультиплатформеними системами.

Розділ має на меті ознайомлення з економічними та функціональними характеристиками майбутнього проекту, економічними аспектами його реалізації та впровадження у використання.

Основною ідеєю майбутнього проекту є:

- підвищення швидкості обміну даними, покращення взаємодії між користувачами;
- обмін даними в місцях без доступу до інтернету, полегшення взаємодії між користувачами;
- мобільний додаток відрізняється від аналогів тим що має ще одну технологію для обміну даними – звукові сигнали.

Масштаб даного проекту є невеликим. Його доцільно використовувати як корпоративний месенжер для невеликих підприємств чи компаній з обмеженою кількістю робітників.

3.2. Архітектура клієнт-серверного мобільного додатку

Розроблений мобільний додаток представляє собою двуланкову клієнт-серверну архітектуру (рис. 3.1). Двуланковою називають архітектуру, яка розподіляє три основні компоненти між клієнтом і сервером.

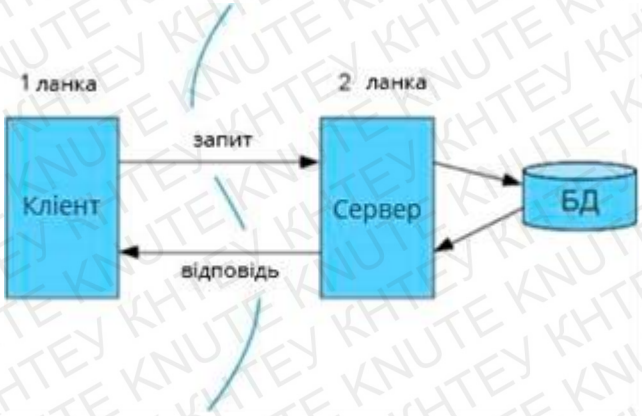

Рис. 3.1. Архітектура програми

Клієнт - це програмне забезпечення, що надає користувачеві інтерфейс для взаємодії з системою. Клієнт не взаємодіє з базою даних безпосередньо, так само він не містить великої кількості бізнес-логіки. Клієнт відповідає за отримання і відображення даних отриманих з сервера. Так само клієнт відповідає за відправку отриманих від користувача даних, або виконує запит до даних по команді користувача.

У додатку взаємодія з користувачем відбуваються за допомогою сенсорного дисплея.

Архітектура програми складається з 4 шарів:

1. Cocoa Touch - містить в собі безліч фреймворків, які відповідають за зовнішній вигляд програми, а так само забезпечує базову інфраструктуру додатки і підтримку основних технологій, таких як: багатозадачність, дотики, а push- повідомлення;

2. Media layer - містить в собі аудіо і відео технології, використовувані для реалізації мультимедійних функцій програми;

3. Core Services - складається з основних системних сервісів для додатків, де основними сервісами є Core Foundation і Foundation frameworks, що визначають основні типи використовуються усіма додатками, а так само він містить в собі технології для роботи з місцем розташування та інтернет мережами;

4. Core OS layer - включає в себе такі сервіси як безпеку, локальна ідентифікація і Bluetooth.

Рис 3.2. Архітектура програми.

Сервер - розташований на другому рівні логіки, містить велику частину бізнес-логіки системи. Сервер взаємодіє з базою даних, після цього генерує JSON моделі і відправляє їх клієнту через інтерфейс API. Так само відбувається процес обміну інформацією в іншу сторону, коли користувач змінює дані в додатку клієнта, потім вони відсилаються на сервер, після чого змінюється база даних

Сервер Firebase взаємодіє з додатком для отримання такої інформації:

Під час реєстрування нового користувача:

- інформацію про ПІБ користувача;
- інформацію «про себе» від користувача;
- інформацію про телефон користувача;

- інформацію про email користувача;
- інформацію про місце проживання користувача;
- інформацію про веб-сайт користувача;
- головне фото користувача;
- бекграунд фото користувача.
- ідентифікатор користувача;
- ідентифікатор пристрою користувача.

Під час пошуку нових користувачів через інтернет:

- інформацію про ідентифікатор відправника користувача;
- інформацію про місцезнаходження користувача;
- інформацію про радіус пошуку користувача.

Під час додавання нових користувачів за допомогою звукових сигналів:

- інформацію про звуковий ідентифікатор відправника користувача;
- інформацію про звуковий ідентифікатор отримувача користувача;
- інформацію місцезнаходження під час додавання користувача.
- інформацію про час додавання користувача.

3.3. Розробка алгоритмів роботи програми

Була проведена розробка наступних алгоритмів. Алгоритм пошуку користувачів через інтернет. Даний алгоритм показує всю бізнес логіку пошуку користувачів через інтернет . У ньому представлено три аспекти алгоритму для Firebase, Backend а також самого мобільного додатку. Алгоритм представлений в Додатку А.

Таблиця 3.1

Код отримання даних через QR-код

```
-(void)captureOutput:(AVCaptureOutput *)captureOutput
didOutputMetadataObjects:(NSArray *)metadataObjects
```

```
fromConnection:(AVCaptureConnection *)connection
{
if (metadataObjects != nil && [metadataObjects count] > 0) {
AVMetadataMachineReadableCodeObject *metadataObj
=[metadataObjects
objectAtIndex:0];
if ([[metadataObj type]
isEqualToString:AVMetadataObjectTypeQRCode]){
dispatch_async(dispatch_get_main_queue(), ^{
[self stopReading];
[self createAlertWithParams:metadataObj.stringValue];
});
}
}
}
```

Таблиця 3.2

Код отримання даних після обробки звукових сигналів

```
[self.connect setReceivedBlock:^(NSData * _Nonnull data, NSUInteger
channel)
{
if (data){
NSLog(@"Received data: %@ on channel %lu", data, (unsigned long)
channel);
NSString *charlieSendString = [[NSString alloc] initWithData:data
encoding:NSUTF8StringEncoding];
[weakSelf.receivedLabel setText:charlieSendString];
}
```

```
else {
 NSLog(@"Decode failed.");
 [weakSelf.receivedLabel setText:@"Decode failed."];
 }
}];
```

Таблиця 3.3

Приклад отримання даних після обробки звукових сигналів

```
[self.connect setReceivedBlock:^(NSData * _Nonnull data, NSUInteger channel)
{
if (data){
NSLog(@"Received data: %@ on channel %lu", data, (unsigned long) channel);
NSString *charlieSendString = [[NSString alloc] initWithData:data encoding:NSUTF8StringEncoding];
[weakSelf.receivedLabel setText:charlieSendString];
}
else {
NSLog(@"Decode failed.");
[weakSelf.receivedLabel setText:@"Decode failed."];
}
}];
```

Таблиця 3.4

Приклад відправки даних для програвання звукових сигналів після Кодування

```
- (IBAction)sendButtonPressed:(id)sender {
if (self.textView.text.length > 0) {
```

```
NSData *payload = nil;
if (_textView.text.length <= (self.connect.maxPayloadLength - 1))
payload = [self.textView.text
dataUsingEncoding:NSUTF8StringEncoding];
else {
NSString *text = [self.textView.text
substringToIndex:(self.connect.maxPayloadLength - 1)];
payload = [text dataUsingEncoding:NSUTF8StringEncoding];
}
NSError *error = [self.connect send:payload];
if (error){
NSLog(@"Error sending data: %@", error);
}
} else {
NSUInteger length = 1 + arc4random() %
(self.connect.maxPayloadLength-
1);
NSData *payload = [[NSString stringWithFormat:@"%lu", length]
dataUsingEncoding:NSUTF8StringEncoding];
NSError *error = [self.connect send:payload];
if (error){
NSLog(@"Error sending data: %@", error);
}
}
}
```

3.4. Розробка основних екранів додатку

Розробка програми проводилася в середовищі Kotlin. Розмітка елементів екранів проводилася за допомогою Constranit, розмітка масштабується залежно від розміру екрана за допомогою «Auto-Layout» (рис. 3.3).

Екран привітання дозволяє користувачу обрати наступний крок створити нового користувача, або увійти в старий аккаунт. Екран привітання містить три Lablel, дві Button, одну секцію ScrollView, один ImageView.

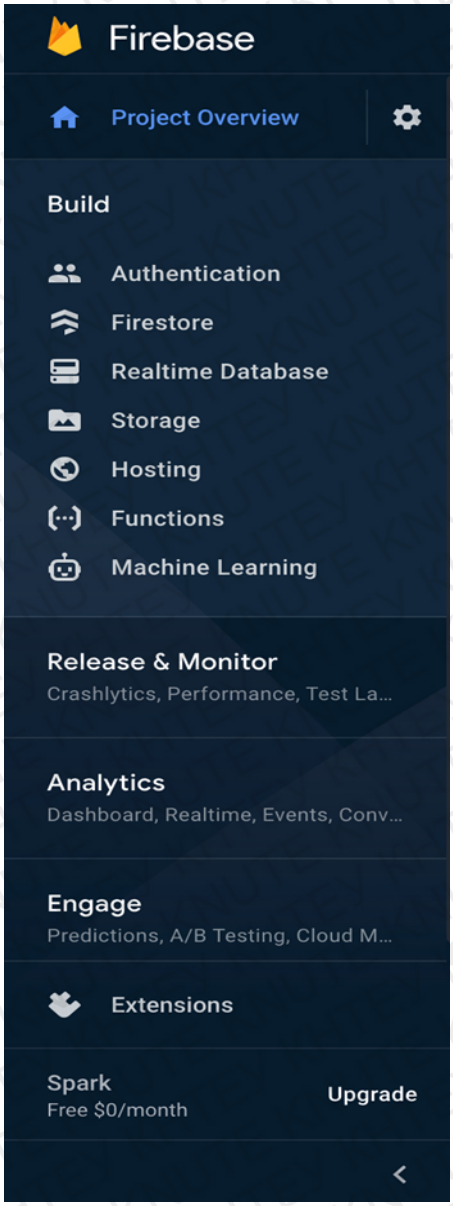

Рис. 3.3. Знімок екрану із сервісу Firebase

Для входу в особистий аккаунт використовується класичне вікно входу. Адміністратор надає користувачу дані, по яким користувач потрапляє до особистого аккаунту (рис.3.4). Можливість реєстрації відсутня. Це зроблено для того, щоб стороні користувачі не потрапили до закритої мережі. Ввівши дані (пошту та пароль) користувач потрапляє у вікно діалогу (рис. 3.6).

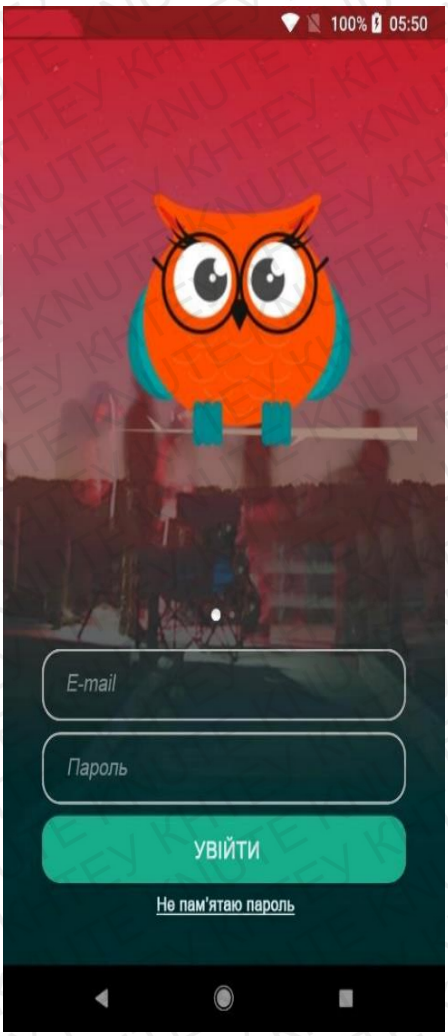

Рис. 3.4. Вікно входу

Кожен авторизований користувач може змінити особисті дані. Це допоможе іншим користувачам зрозуміти з ким вони спілкуються. Для зміни доступні наступні значення: ПІБ, стать, телефон, e-mail, сайт, країна, про себе. (рис. 3.5)

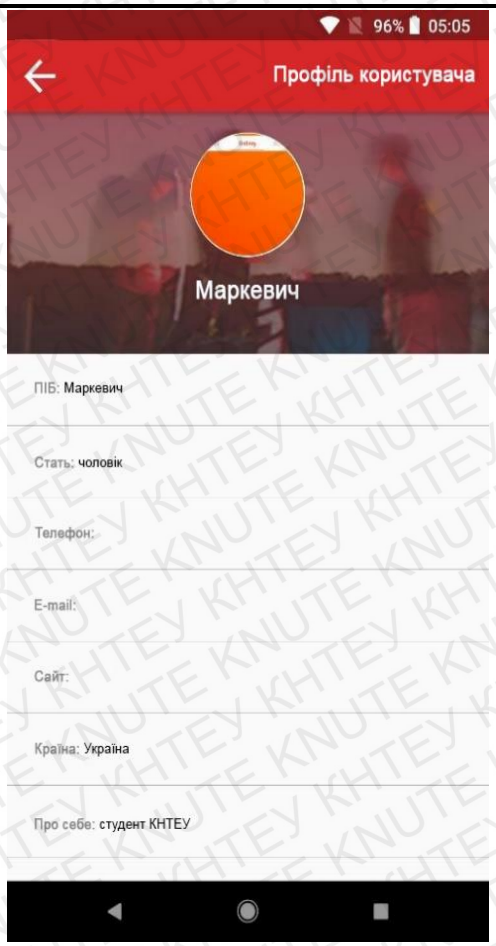

Рис. 3.5. Налаштування профілю

Рис. 3.6. Вікно вхідних повідомлень

Вікно нового діалогу представлене у вигляді звичного месенжера. Для користувачів доступні наступні функції: написати та відіслати текстове повідомлення, записати та прослухати аудіо повідомлення. Ліміт на запис аудіоповідомлення встановлений на 1 хвилину. Це зумовлено тим, що ліміт на даному сервері встановлений на 10 Гб (рис. 3.6).

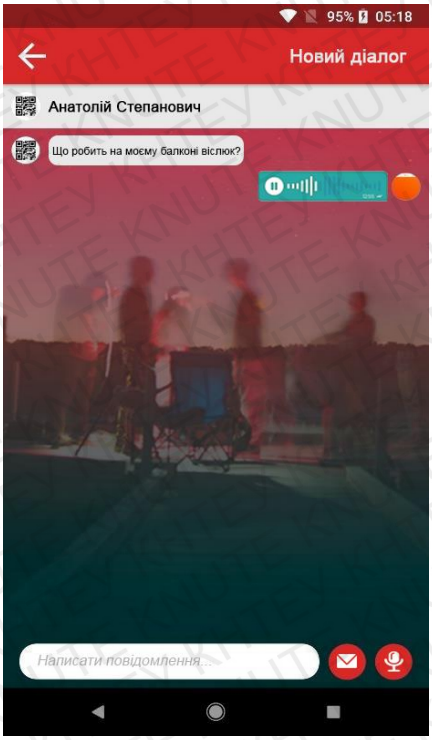

Рис. 3.7. Вікно діалогу

3.5. Висновки до розділу 3

В розділі «Розробка мобільного додатку» здійснювалась розробка мобільного додатку з урахуваннями вимог до сучасного (на момент написання роботи) програмного забезпечення мобільних додатків.

Було розроблено архітектуру клієнт-серверного мобільного додатку, що є кросплатформеною. Це зумовлено економією ресурсів сервера та покращеною можливістю просування. Розробка алгоритмів роботи програми дозволила окремо створити та доповнити функціонал кожного з блоків та додати оптимальні рішення у вже готові шаблони.

Розробка основних екранів додатку дозволила створити інтуїтивно-зрозумілий інтерфейс, що відповідає звичному загальноприйнятому інтерфейсу месенжерів.

Результатом роботи є мобільний додаток обміну захищеними даними з використанням звукових сигналів.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

Для вирішення поставленої в роботі мети було проведено аналіз мов програмування для створення мобільних додатків. В процесі виконання випускної кваліфікаційної роботи було зроблено наступне:

1. Провівши аналіз технологій було визначено технології, за допомогою яких найзручніше реалізувати даний мобільний додаток.
2. Проведений аналіз архітектури мобільних додатків та обрано найдоцільнішу, що відповідає завданню випускної кваліфікаційної роботи.

В другому розділі був проведений аналіз архітектурних шаблонів для розробки мобільного додатку та аналізувалися технології обміну даними. Дослідивши методи шифрування та визначено найоптимальніші для використання.

В третьому розділі було здійснювалась розробка мобільного додатку обміну захищеними даними з використанням звукових сигналів. Розроблено архітектуру клієнт-серверного мобільного додатку що дозволяє обробляти запити без перенавантаження серверу. Здійснювалась розробка алгоритмів роботи програми, що дозволило оптимізувати процес відповідно до вимог випускної кваліфікаційної роботи. На останньому етапі здійснювалась розробка основних екранів додатку. Інтерфейс програми інтуїтивно-зрозумілий та відповідає класичному інтерфейсу месенжерів.

За результатами розробки отримано мобільний додаток, який можна використовувати на підприємстві різного масштабу.

					КНТЕУ 121 02-09.МР			
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*				
Зав. каф.		Криворучко О.В.		*01.11.21*	*«Модель обміну захищеними даними з використанням звукових сигналів»*	*Стадія*	*Арку*	*Аркуші*
Керівник		Десятко А.М.		01.11.21		*В*	*46*	*50*
Гарант		Токар В.В.		01.11.21		*Факультет інформаційних технологій 2м курс, 2 група*		
Розробив		Маркевич В.С.		01.11.21	*Висновки та пропозиції*			

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Цібульов П. М. Управління інтелектуальною власністю: монографія / Цибулів П. М., Чеботарьов В. П., Зінов В. Г., Суїні Ю., за ред. П. М. Цібульова. - К.: «К. І. С. », 2015. - 448 с.
2. Квашніна А. Як управляти портфелем технологій та інтелектуальною власністю: серія методичних матеріалів «Практичні керівництва для центрів комерціалізації технологій» / під рук. П Ліндхольм, проект EuropeAid «Наука і комерціалізація технологій», 2016. - 60 с.
3. Бланк, С. Стартап. Настільна книга засновника / С. Бланк, Б. Дорф; пер. з англ. Т. Гутман, І. Окунькова, Е. Бакушева. - 2-е вид. - Москва: Паблішер, 2014. - 614 с.
4. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. /Б.Шнайер/ [Текст] –М.: Издательство ТРИУМФ, 2018 – 816 с.
5. Swift - [Електронний ресурс] / Apple Inc. - Режим доступу: URL:https://www.apple.com/swift/.
6. Мова програмування Swift. - [Електронний ресурс] . -Режим доступу: URL: https://habr.com/post/225841/.
7. Під'єднання та обмін даними в Інтернеті - [Електронний ресурс]. Режим доступу: URL: https://mozok.click/644-pdyednannya-ta-obmn-danimi-vnternet.html.
8. Батрак В.І. Використання qr кодів [Електронний документ]. – Режим доступу: URL: https://www.google.com.ua/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1

					КНТЕУ 121 02-09.МР		
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*			
Зав. каф.		Криворучко О.В.		27.02.21	*«Модель обміну захищеними даними з використанням звукових сигналів»*	*Стадія*	*Арку*	*Аркуші*
Керівник		Десятко А.М.		27.02.21		*СД*	*47*	*50*
Гарант		Токар В.В.		27.02.21		*Факультет інформаційних технологій 2м курс, 2 група*		
Розробив		Маркевич В.С.		27.02.21	*Список використаних джерел*			

5&ved=2ahUKEwinn5XV7o3fAhUulIsKHWeLCxMQFjAOegQIARAC&url=http%3A%2F%2Fejournals.npu.edu.ua%2Findex.php%2Fikt%2Farticle%2Fdownload%2F20%2Fpdf_18&usg=AOvVaw2kEhetIXX9ZRyasIQsgUXc

9. Chirp Connect Overview [Електронний ресурс]. – Режим доступу: URL:https://chirp.io.
10. Шаблон проектування MVC [Електронний ресурс]. - Режим доступу: URL: https://webformyself.com/shablon-proektirovaniya-mvc/ .
11. Model-View-Controller (MVC) in IOS: A Modern Approach [Електронний ресурс]. - Режим доступу: URL: https://www.raywenderlich.com/132662/mvc-in-ios-a-modern-approach
12. IOS Architecture Patterns [Електронний ресурс]. - Режим доступу: URL: https://techblog.badoo.com/blog/2016/03/21/ios-architecture-patterns/ .
13. The Book of VIPER [Електронний ресурс]. - Режим доступу: URL: https://github.com/strongself/The-Book-of-VIPER
14. XCode - IDE - Apple Developer [Електронний ресурс] / Apple Inc. - Режим доступу: URL: https://developer.apple.com/xcode/ide
15. Core Data Programming Guide: Persistent Store Types and Behaviors [Електронний ресурс] / Apple Inc. - Режим доступу: URL: https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual / CoreData
16. XMPP is the open standard for messaging and presence [Електронний ресурс] / XMPP Community. - Режим доступу: URL: http://xmpp.org/
17. Пржіялковскій В.Oracle10: шифруємо дані, [Електронний документ]. - Режим доступу: URL: http://citforum.com/database/oracle/crypt_data/
18. Основні параметри. Rijndael (AES). [Електронний документ]. - Режим доступу: URL: http://www.enlight.com/crypto/algorithms/rijndael/rijndael00.htm

19. Шіфрованіе XTS [Електронний документ]. - Режим доступу: URL: http://www.kingston.com/ru/usb/encrypted_security/xts_encryption
20. Статистика вказала на умови для появи стартапів, успішних як Google і Facebook [Електронний ресурс]. - Режим доступу: URL: https://naked-science.ru/article/sci/statistika-ukazala-na-usloviya
21. Jurg van Vliet, Programming Amazon EC2, O'Reilly Media, 2011, 186c
22. Jurg van Vliet, Flavia Paganelli, Jasper Geurtsen, Resilience and Reliability on AWS, O'Reilly Media, 2013, 164c
23. Amazon Web Services, Amazon Elastic Compute Cloud (EC2) User Guide, Amazon Digital Services, Inc., 2012 892c
24. Rob Linton, Amazon Web Services: Migrating your .NET Enterprise Application, Packt Publishing, 2011, 311c
25. James Murty, Programming Amazon Web Services: S3, EC2, SQS, FPS, and SimpleDB, O'Reilly Media, 2009 604c
26. Danny Garber, Jamal Malik, Adam Fazio, Windows Azure Hybrid Cloud, Wrox, 2013, 246c
27. Tejaswi Redkar, Tony Guidici, Windows Azure Platform, Apress, 2011, 602c
28. Sriram Krishnan, Programming Windows Azure, O'Reilly Media, 2010 370c
29. Roger Jennings, Cloud Computing with the Windows Azure Platform, Wrox 2010, 360c
30. Google Hangouts [Електронний ресурс]. URL: https://hangouts.google.com/
31. Мобільний протокол MTProto [Електронний ресурс] // Документація Telegram [Електронний ресурс]. URL: https://tlgrm.ru/docs/mtproto
32. Ідентифікація та аутентифікація [Електронний ресурс] // Наукова бібліотека [Електронний ресурс]. URL: http://sernam.ru/ss_23.php
33. Mastering Android Development with Kotlin: Deep dive into the world of Android to create robust applications with Kotlin By Milos Vasic. Packt Publishing — ebooks Account, November 8, 2017; 378 pages;

ТЕХНІЧНЕ ЗАВДАННЯ

Мобільний додаток обміну захищеними даними з використанням звукових сигналів

ХАРАКТЕРИСТИЧНА ІНФОРМАЦІЯ

Мета в контексті: створення мобільного додатку обміну даними за допомогою аудіоповідомлення з можливістю шифрування на хмарному СУДБ, та дешифрування на пристрої, що приймає повідомлення.

Сфера застосування: мобільні додатки.

Передумови: користувач потрапляє до мобільного додатку через особистий акаунт. Користувач (адресант) відправляє аудіоповідомлення. Сервер шифрує дане повідомлення. Користувач потрапляє до мобільного додатку через особистий акаунт. Користувач (адресат) отримує повідомлення, що розшифровується мобільним додатком (акаунту адресата) в діалозі адресата з адресантом.

Невдале завершення стану: користувач не має доступу до інтернету та не може потрапити до особистого акаунту.

Основний актор: користувач, смартфон.

Тригер: відбувається обмін аудіоповідомленнями.

ГЕНЕРАЛЬНИЙ СЦЕНАРІЙ УСПІХУ

1. Користувач (адресант) потрапляє до мобільного додатку через особистий акаунт.
2. Користувач (адресант) додає контакти адресата.
3. Користувач (адресант) заходить в діалог з адресатом.

					КНТЕУ 121 02-09.МР		
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*			
Зав. каф.		Криворучко О.В.		20.03.21	*«Модель обміну захищеними даними з використанням звукових сигналів»*	*Стадія*	*Аркуш*	*Аркушів*
Керівник		Десятко А.М.		20.03.21		*ТЗ*	*51*	*50*
Гарант		Криворучко О.В.		20.03.21		*Факультет інформаційних технологій 2м курс, 2 група*		
Розробив		Маркевич В.С.		20.03.21	*Технічне завдання*			

4. Користувач (адресант) записує аудіоповідомлення та відправляє адресату.

5. Повідомлення потрапляє до БД та шифрується.

6. Користувач (адресат) потрапляє до мобільного додатку через особистий акаунт.

7. Користувач (адресат) заходить в розділ «Діалоги».

8. Користувач (адресат) заходить в діалог з Користувачем (адресантом).

9. Мобільний додаток дешифрує повідомлення користувача (адресанта) в момент запуску повідомлення на прослуховування.

10. Користувач (адресат) записує аудіоповідомлення у відповідь і надалі, до моменту отримання нового повідомлення стає Користувачем (адресантом).

11. Повідомлення від Користувача (адресанта) потрапляє до БД та шифрується.

12. Користувач (адресант) при отриманні нового повідомлення стає користувачем (адресатом). Користувач (адресат) запускає та прослухує повідомлення, яке дешифрується в на мобільному пристрої в акаунті отримувача.

13. Користувач (адресат) виходить з діалогу.

14. Користувач (адресант) виходить з діалогу.

Розширення

1а. Додається можливість обміну текстовими повідомленнями.

SUB-ВАРІАЦІЇ

2а. Додається можливість зміни особистих даних акаунту.

Відповідно до мети ставляться такі завдання:

1. Аналіз мов програмування для інтегрованого середовища розробки Firebase.

2. Аналіз архітектурних шаблонів для розробки мобільного додатку.

3. Аналіз технологій обміну даними.

4. Аналіз стандартів шифрування даних.

5. Аналіз сучасних хмарних СУБД.

6. Розробка алгоритму роботи мобільного додатку.

7. Створення мобільного додатку.

ВІДПОВІДНА ІНФОРМАЦІЯ

Пріорітет: ТОП.

Цільова ефективність: 2 акаунта за 1 сесію.

Частота: 100 повідомлення /день.

Служба надпорядкового використання: обмін захищеними аудіоповідомленнями.

Основний актор: користувач.

Вторинний актор: БД.

ВІДКРИТІ ПИТАННЯ

Створення корпоративного мобільного додатку для внутрішнього користування в компанії та обміну захищеними повідомленням.

Тестування додатка flybot

Тестування програмного забезпечення (англ. Software Testing) – техніка контролю якості, що перевіряє відповідність між реальною і очікуваною поведінкою програми завдяки кінцевому набору тестів, які обираються певним чином.

Техніка тестування Flybot також включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою оцінки. Він оцінюється за:

– відповідність вимогам, якими керувалися проектувальники та розробники;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з програмним забезпеченням та операційними системами;

– відповідність задачам замовника.

Якість не є абсолютною, це суб'єктивне поняття. Тому тестування, як процес своєчасного виявлення помилок та дефектів, не може повністю забезпечити коректність програмного забезпечення. Воно тільки порівнює стан і поведінку продукта зі специфікацією на ринку.

Однак, потрібно розрізняти тестування програмного забезпечення і забезпечення якості програмного забезпечення, до якого належать усі складові ділового процесу, а не тільки тестування.

Якість Flybot’a обмежується такими поняттями як коректність, надійність, практичність, безпечність, а також містить більше технічні вимоги, котрі описані у стандарті **ISO 9126**. Склад та зміст супутньої документації процесу тестування визначається стандартом **IEEE 829–1998 Standard for Software Test Documentation**.

Ми використовували багато підходів до тестування програмного забезпечення, але ефективне тестування складних продуктів – це по суті дослідницький та творчий процес.

ДОДАТКИ

Додаток А

Алгоритм пошуку користувачів через інтернет

```
- (UIImage *)createNonInterpolatedUIImageFromCIImage:(CIImage *)image
withScale:(CGFloat)scale
{
// Render the CIImage into a CGImage
CGImageRef cgImage = [[CIContext contextWithOptions:nil]
createCGImage:image fromRect:image.extent];
// Now we'll rescale using CoreGraphics
UIGraphicsBeginImageContext(CGSizeMake(image.extent.size.width * scale,
image.extent.size.width * scale));
CGContextRef context = UIGraphicsGetCurrentContext();
// We don't want to interpolate (since we've got a pixel-correct image)
CGContextSetInterpolationQuality(context, kCGInterpolationNone);
CGContextDrawImage(context, CGContextGetClipBoundingBox(context),
cgImage);
// Get the image out
UIImage *scaledImage = UIGraphicsGetImageFromCurrentImageContext();
// Tidy up
UIGraphicsEndImageContext();
CGImageRelease(cgImage);
return scaledImage;
}
```