Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

_____Інститут електроенергетики_____
(інститут)
_____факультет інформаційних технологій_____
(факультет)
Кафедра_____інформаційних технологій та комп'ютерної інженерії _____
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня ___магістра__
(бакалавра, спеціаліста, магістра)

Студента _____Дудки Сергія Миколайовича_____
(ПІБ)
академічної групи _____126м-20-1_____
(шифр)
спеціальності _____126 «Інформаційні системи та технології»_____
(код і назва спеціальності)
за освітньо-професійною програмою _____

_____«Інформаційні системи та технології»_____
(офіційна назва)
на тему _____Розробка інформаційної технології для обміну миттєвими_____

_____повідомленнями на основі сервіса «GetMessage»_____

(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	проф. Коротенко Г.М.			
розділів:				

Рецензент	доц. Галушко О.М.			

Нормоконтролер	проф. Коротенко Г.М.			

Дніпро
2022

ЗАТВЕРДЖЕНО:
завідувач кафедри

___інформаційних технологій___
_____та комп'ютерної інженерії___
(повна назва)

_____ _Гнатушенко В.В._
(підпис) (прізвище, ініціали)

«_____»_____ 2021_року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня ____магістр _____
(бакалавра, спеціаліста, магістра)

студенту _____Дудка С.М. **академічної групи** ____126м-20-1___
(прізвище та ініціали) (шифр)

спеціальності _____126 «Інформаційні системи та технології»_____

за освітньою-професійною програмою _____

_____«Інформаційні системи та технології»_____

на тему _____Розробка інформаційної технології для обміну миттєвими_____
_____повідомленнями на основі сервіса «GetMessage»_____

затверджену наказом ректора НТУ «Дніпровська політехніка» від 10.12.2021 №1036-с

Розділ	Зміст	Термін виконання
Розділ 1	Аналіз стану області рішення задачі	1.10.2021 – 31.10.2021
Розділ 2	Моделі та методи розв'язання задачі	1.11.2021 – 30.11.2021
Розділ 3	Програмна реалізація розроблених фрагментів	1.12.2021 – 21.12.2021

Завдання видано _____ _____Коротенко Г.М._____
(підпис керівника) (прізвище, ініціали)

Дата видачі _____1.10.2021 р._____

Дата подання до екзаменаційної комісії ____23.12.2021 р.____

Прийнято до виконання _____ _____Дудка С.М._____
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 85 с., 23 рис., 1 табл., 4 дод., 33 джерела.

Об'єкт дослідження: клієнт-серверна інформаційна технологія для операційної системи Android.

Мета кваліфікаційної роботи: створення інформаційної технології для обміну миттєвими повідомленнями для ОС Android.

У вступі розглядається сучасний стан проблеми, конкретизується мета кваліфікаційної роботи актуальність та галузь її застосування, уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі наведено: призначення інформаційної технології, опис застосованих математичних методів, опис використаних технологій та мов програмування.

У третьому розділі приведено опис структури програмного забезпечення підтримки інформаційної технології, алгоритмів її функціонування та детальний опис роботи розробленого програмного продукту.

Практичне значення полягає у створенні додатка, що надає можливість безпечно обмінюватися миттєвими повідомленнями користувачам в режимі реального часу при підключенні до мережі Інтернет.

Актуальність сервісу визначається великою кількістю користувачів, які користуються подібними сервісами.

Список ключових слів: СМАРТФОН, МЕСЕНДЖЕР, ПОВІДОМЛЕННЯ, ДОДАТОК, ANDROID, БАЗА ДАНИХ, СЕРВЕР, КЛІЄНТ.

ABSTRACT

Explanatory note: 85 pages, 23 figures, 1 table, 4 appendices, 33 sources.

Object of research: client-server information technology for the Android operating system.

The purpose of the qualification work: the creation of information technology for instant messaging for Android.

The introduction considers the current state of the problem, specifies the purpose of the qualification work, the relevance and scope of its application, clarifies the task.

In the first section the analysis of the subject area is carried out, the urgency of the task and the purpose of development are determined, the task statement is developed, the requirements to the software implementation, technologies and software are set.

The second section provides: the purpose of information technology, a description of the applied mathematical methods, a description of the technologies used and programming languages.

The third section describes the structure of software support for information technology, algorithms for its operation and a detailed description of the developed software product.

The practical value is to create an application that allows you to securely exchange instant messages to users in real time when connected to the Internet.

The relevance of the service is determined by the large number of users who use such services.

List of keywords: SMARTPHONE, MESSAGER, MESSAGE, APP, ANDROID, DATABASE, SERVER, CLIENT.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ 7
ВСТУП 8
РОЗДІЛ 1. АНАЛІЗ СТАНУ ОБЛАСТІ РІШЕННЯ ЗАДАЧІ 11
1.1. Загальні відомості з предметної галузі 11
1.2. Призначення розробки та галузь застосування 14
1.3. Постановка завдання 14
1.4. Вимоги до програми або програмного виробу 15
1.4.1. Вимоги до функціональних характеристик 15
1.4.2. Вимоги до інформаційної безпеки 15
1.4.3. Вимоги до складу та параметрів технічних засобів 17
1.4.4. Вимоги до інформаційної та програмної сумісності 19
РОЗДІЛ 2. МОДЕЛІ ТА МЕТОДИ РОЗВ'ЯЗАННЯ ЗАДАЧІ 20
2.1. Функціональне призначення системи 20
2.2. Опис математичних методів необхідних для створення сервісу 20
2.3. Опис необхідних технологій та мов програмування 20
2.4. Обгрунтування та організація вхідних та вихідних даних програми 35
РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ РОЗРОБЛЕНИХ ФРАГМЕНТІВ 36
3.1. Опис структури сервісу та алгоритмів функціонування 36
3.2. Використані технічні засоби 38
3.3. Використані програмні засоби 39
3.4. Виклик та завантаження програми 42
3.5. Опис інтерфейсу користувача 43
ВИСНОВОК 57

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 58

ДОДАТОК А. Відомість матеріалів кваліфікаційної роботи 62

ДОДАТОК Б. Лістинг серверної частини 63

ДОДАТОК В. Відгук керівника кваліфікаційної роботи 82

ДОДАТОК Г. Рецензія 84

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IT – інформаційні технології;

ОС – операційна система;

ПК – персональний комп’ютер;

БД – база даних;

СУБД – система управління базами даних;

API – application programming interface;

HTTPS – hypertext transfer protocol secure;

SSD – solid-state drive;

SQL – structured query language;

PHP – hypertext preprocessor;

SSL – secure sockets layer;

JVM – java virtual machine;

UI – user interface;

MVVM – model-view-viewmodel;

MVC – model-view-controller;

CLI – call level interface;

ЕОМ – електронно-обчислювальна машина.

ВСТУП

Інформаційна технологія для обміну повідомленнями в режимі реального часу – телекомунікаційна служба для обміну текстовими повідомленнями між комп'ютерами або іншими пристроями користувачів через комп'ютерні мережі та інші засоби зв’язку. Зазвичай і від початку, це були невеликі текстові повідомлення. Але з розвитком компонентів комп’ютерної техніки з’являлись все нові і нові функції, такі як передавання файлів, зображень, звукових сигналів та повідомлень, відео, а також здійснення спільних дій, таких як малювання або ігри.

Для користування цим видом комунікації необхідна клієнтська програма. Клієнтську програму системи миттєвих повідомлень часто називають інтернет-пейджером або месенджером [30]. Відмінність миттєвих повідомлень від, наприклад, електронної пошти тут в тому, що обмін повідомленнями відбувається в реальному часі. При відправлені повідомлення електронною поштою повідомлення зберігається у поштовій скриньці на сервері. Для того, щоб отримати повідомлення, отримувач повинен сам перевірити свою поштову скриньку і забрати їх. У месенджерах зв'язок між користувачами утримується постійно і відправлене повідомлення одразу передається користувачу. Обмін повідомленнями може бути або між двома, або між декількома співрозмовниками (конференція, чат).

Система миттєвих повідомлень працює за деяким протоколом. Протоколи бувають серверні або безсерверні. Найпоширенішими є серверні протоколи, коли месенджери не працюють самостійно, а підключаються до центрального комп'ютера мережі обміну повідомленнями, який називають сервером. Тому месенджери й називають клієнтами (клієнтськими програмами).

В сучасному світі месенджери користуються великою популярністю серед інтернет користувачів. В останні роки сталося кардинальне зрушення в поведінці користувачів: люди витрачають більше часу в месенджерах, ніж в

соціальних мережах. І якщо ви будуєте бізнес в інтернеті, вам потрібно бути там, де знаходяться ваші клієнти, тобто в месенджерах. Люди люблять спілкуватися в месенджерах не тільки тому, що це зручно і швидко, але і тому, що вони ще не забиті спамом. Чат в месенджері - це канал зв'язку зі сформованою культурою довіри. Отримувати повідомлення і обмінюватися інформацією з іншими людьми тут легко і приємно. Це миттєвий обмін інформацією. Відкрити повідомлення в месенджері - справа однієї секунди, на відміну від стомлюючого пошуку потрібного листа в поштовій скриньці. А ще інформація, якою люди обмінюються через месенджери - корисна. Месенджер задовольняє потреби сучасної людини в свіжій, актуальній інформації, і люди звикли думати, що все, що приходить на телефон і відображати на екрані у вигляді повідомлення з месенджера - важливо і потребує їх уваги прямо зараз.

Інформаційна технологія на основі сервісу «GetMessage» призначена для обміну миттєвими повідомленнями в режимі рального часу між користувачами пристроїв на базі операційної системи Android.

Інформаційна технологія може використовуватись в будь-якій галузі, так як вона надає можливість користувачам встановити контакт на відстані і обмінюватися інформацією для будь-яких цілей і стосовно будь-яких сфер життя і діяльності людини. Для цього користувачу необхідно мати тільки підключення до мережі Інтернет.

Об’єктом досліджень є інформаційна технологія для передавання повідомлень.

Предметом - інформаційна технологія на основі сервісу «GetMessage».

Мета роботи - створення інформаційної технології для передавання повідомлень.

Для досягнення поставленої мети в роботі ставились та вирішувались такі основні завдання:

- аналіз існуючих на ринку програмних продуктів з необхідним функціоналом;

- уточнення вимог з урахуванням потреб користувачів;
- розробка алгоритму роботи інформаційної системи;
- розробка клієнтського застосунку для ОС Android;
- розробка серверної частини;
- розгортання серверної частини.

Відповідно до проведеного аналізу в роботі поставлені такі основні функціональні задачі та вимоги до розроблюваної інформаційної системи:

- наявність зручного інтуїтивно зрозумілого інтерфейсу;
- можливість безпечного обміну приватними повідомленнями з іншими користувачами;
- забезпечення високої швидкодії для комфортного користування;
- робота на смартфонах різних характеристик та конфігурацій під управлінням сімейства ОС Android.

РОЗДІЛ 1. АНАЛІЗ СТАНУ ОБЛАСТІ РІШЕННЯ ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Месенджери стали масово використовуватися у 1990-х. Однак, сама концепція для спілкування великої кількості користувачів між собою за допомогою електронних пристроїв, об'єднаних в уніфіковану мережу, сягає своїм корінням середини 1960-х. Система обміну даними в режимі, наближеному до реального часу, під назвою Compatible Time-Sharing System була створена в Массачусетському технологічному інституті ще в 1961 році [1].

В рамках цього експериментального проєкту до 30 користувачів могли одночасно увійти в спільне інформаційне середовище і обмінюватися повідомленнями між собою. До 1965 року вона стала досить популярним внутрішнім інструментом комунікації в Массачусетському технологічному інституті. Поява peer-to-peer протоколу в 1970-х дозволило підключати різних користувачів до одного і того ж комп'ютера, обмінюватися даними, і перемикатися між різними користувачами одного і того ж комп'ютера.

У 1980 запускається Zephyr Notification Service і так званий «проєкт Athena». Він використовував Unix для роботи з повідомленнями користувачів. Деякі наукові організації в США на початку 2010-х років XXI століття все ще продовжували використовувати цю систему для обміну листами та повідомленнями [31].

У той же період з'явилася система так званих «дошок для спілкування» - BBS. Користувачам достатньо було скористатися програмою-терміналом для того, щоб завантажувати файли і обмінюватися прямими повідомленнями з іншими користувачами [31].

Історія месенджерів у вигляді універсальних настільних додатків, а не систем з обмеженим доступом, починається 1996 року. Саме тоді ізраїльська компанія Mirabilis запустила ICQ. Відмінністю цієї програми стали чати

розраховані на велику кулькість користувачів, підтримка передачі файлів, пошук по базі користувачів і ряд інших опцій, яких раніше не було в месенджерів 1980-х років.

Далі власні розробки презентували в Yahoo Pidgin. Потім з`явився xMSN Messenger від Microsoft, який був представлений 1999 року, а до 2009 року він налічував вже понад 330 млн активних користувачів щомісяця.

Відмінною рисою стала мультіпротокольність і розробка ними власних платформ і чатів. Однак був у цьому і недолік: якщо ви хотіли мігрувати на інший месенджер, ви не могли автоматично синхронізувати історію повідомлень між пристроями. Також в той час ще не було хмарних сервісів і широких можливостей роботи з файлами для мобільних користувачів.

Зміни в сприйнятті месенджерів, як засобу комунікації наступили в момент, коли швидкості і стабільність роботи бездротових мереж дозволили людям спілкуватися не тільки за монітором комп'ютера або екраном ноутбука. Після ряду тестових спроб в 2010-х майже всі основні месенджери (від Skype і Facebook до Line і WeChat) запустили власні версії для мобільних браузерів або окремі додатки для смартфонів.

А все це сталося завдяки появі iPhone від Apple і смартфонів на ОС Android від Google. Справжню революцію здійснив WhatsApp, створений нащадком емігрантів з України Яном Борисовичем і екс-інженером Yahoo Брайаном Актон. Цей месенджер став першим, що мав прив'язку до номера мобільного телефону і першим отримав підтримку push-повідомлень для смартфонів Apple.

Сучасний користувач має великий вибір серед багатьох месенджерів, найпупулярнішими з яких є WhatsApp, Viber, Telegram, Facebook Messenger.

WhatsApp є окремим Android додатком від Facebook, який, перш за все, орієнтований на бізнес-клієнтів. Спеціальні бізнес акаунти і профілі дозволяють компаніям отримувати доступ до зручного набору інструментів.

Примітно те, що крім спілкування зі своїм списком контактів, власники акаунтів можуть влаштовувати розсилки, включати функцію автовідповідача

на найбільш часті питання і аналізувати свою статистику. Додаток має велику популярність і навіть в російськомовному секторі займає одне з лідируючих місць.

Viber один з найбільш зручних і популярних мобільних додатків. Головною перевагою є інтеграція безпосередньо в книгу контактів, дозволяючи оперувати номерами телефонів при використанні будь-яких функцій. Примітною є можливість змінювати номер телефону, не втрачаючи при цьому дані на своєму акаунті. Відправка файлів є однією з найбільш затребуваних функцій, особливо часто респонденти користуються послугами, як в побутових, так і в бізнес цілях. В іншому ж це зручне і просте засіб для листувань, здійснення дзвінків та обміну файлами.

Telegram був розроблений і випущений за участю творця популярної соціальної мережі «Вконтакте» - Павла Дурова. Це один з найшвидших месенджерів, який до того ж відрізняється простим дизайном і набором корисних функцій. Користувачі мають можливість обмінюватися між собою файлами і документами практично будь-якого формату, а також передавати координати свого місця розташування за допомогою геолокації. Перевагами Telegram також є швидка і проста реєстрація, висока надійність, безпеку архівів листувань і функція автовидалення повідомлень по заданому таймеру.

Facebook Messenger безпосередньо пов'язаний з Facebook, дозволяє швидко і легко листуватися в окремому додатку з усім своїм списком контактів. Має простий дизайн, позбавлений непотрібних функцій і відрізняється високою швидкістю. Однією з найбільш примітних функцій є груповий відеочат. Відеотрансляцію в такому випадку зможуть вести не більше шести осіб, але інші зможуть використовувати голос.

1.2. Призначення розробки та галузь застосування

Сервіс на основі застосунку «GetMessage» призначений для обміну миттєвими повідомленнями в режимі реального часу між користувачами пристроїв на базі операційної системи Android.

Сервіс може використовуватись в будь-якій галузі, так як вона надає можливість користувачам встановити контакт на відстані і обмінюватися інформацією для будь-яких цілей і стосовно будь-яких сфер життя і діяльності людини. Для цього користувачу необхідно мати тільки підключення до мережі Інтеренет.

1.3. Постановка завдання

Метою даної кваліфікаційної роботи є створення інформаційної системи та відповідної технології, що доволить користувачам пристроїв на базі операційної системи Androd обмінюватися приватними повідомленнями в режимі реального часу, маючи підключення до мережі Інтернет.

Відповідно до проведеного аналізу в роботі поставлені такі основні функціональні задачі та вимоги до розроблюваної інформаційної системи:

- наявність зручного інтуїтивно зрозумілого інтерфейсу;
- можливість безпечного обміну приватними повідомленнями з іншими користувачами;
- забезпечення високої швидкодії для комфортного користування;
- робота на смартфонах різних характеристик та конфігурацій під управлінням сімейства ОС Android.

1.4. Вимоги до програми або програмного виробу

1.4.1. Вимоги до функціональних характеристик

Сервіс на основі застосунку «GetMessage» повинен надавати користувачам такий функціонал, як:

1. Реєстрація нового аккаунту.
2. Авторизації по логіну і паролю.
3. Відновлення паролю в разі його втрати.
4. Встановлення інформації про аккаунт, такої як:
   а) фото аккаунту;
   б) статус аккаунту;
   в) ім’я аккаунту.
5. Можливість додати друга по його адресі електронної пошти. В разі якщо користувач з введеною електронною поштою не є користувачем ІС, надати можливість запросити його шляхом відправлення запрошення на його електронну пошту.
6. Можливість прийняти або відхилити запрошення дружби від іншого користувача.
7. Відправка текстових повідомлень та зображень користувачам з розділу «Друзі».
8. Видалення уже відправлених повідомлень.
9. Перегляд аккаунтів користувачів з розділу «Друзі».
10. Можливість перегляду останньої активності користувачів з розділу «Друзі».
11. Відображення push-повідомлень.

1.4.2. Вимоги до інформаційної безпеки

Інформаційна технологія на основі сервісу «GetMessage» повинна забезпечувати безпеку персональних даних користувачів, зберігати таємницю

переписки, обмежити доступ посторонніх осіб до приватних повідомлень і діяльності користувача, забезпечити неможливість перехвату персональних данних при передачі їх через мережу інтернет. Для того щоб виконати всі вимоги до інформаційної системи використовується авторизація користувача по його логіну і паролю, які він вказує при реєстрації. Після авторизацій користувач отримує токен, який в подальшому кожен раз перевіряється при звертанні клієнту на сервер.

Передача данних через мережу Інтернет відбувається різним протоколам передачі даних.

HTTPS - це протокол передачі даних, що забезпечує безпечний і конфіденційний обмін інформацією між сервером і клієнтом [32].

Особливості використання HTTPS:

– Всі дані шифруються. Так зловмисники не зможуть дізнатися, яка інформація передається, і відстежити дії користувачів. Для шифрування використовується загальний секретний ключ, який при установці безпечного з'єднання вибирають сервер і комп'ютер. Всі ключі одноразові, перехопити і підібрати дуже складно - довжина ключа перевищує 100 знаків.

– Фіксація всіх змін або спотворень даних, навіть якщо це було зроблено випадково.

– Аутентифікація гарантує, що клієнт отримає дані с того серверу, з якого потрібно, і захищає від атаки посередника. Для цього на сервері повинен бути спеціальний цифровий сертифікат.

Захищеність інформації досягається шляхом використання SSL - стандартного протоколу, що забезпечує безпечне підключення при доступі до веб-ресурсів і робить неможливим перегляд переданих даних сторонніми. Під час установки з'єднання на основі протоколу HTTPS відбувається створення випадкового секретного ключа, який буде відомий тільки серверу і клієнту. Потім, при використанні цього ключа (який генерується заново при кожному новому сеансі зв'язку) вся передана інформація шифрується.

Отримати доступ до переданих дани шляхом підбору секретного ключа неможливо, оскільки він складається більше ніж з 100 символів. Використовується ще один додатковий елемент підвищення безпеки з'єднання на основі протоколу HTTPS - це цифровий сертифікат, який використовується для ідентифікації сервера. Він підтверджує факт керування сервером особою, якій був виданий сертифікат. У ньому міститься вся необхідна інформація про його власника і присутній цифровий підпис, який використовується для підтвердження автентичності. Тільки в разі проходження перевірки автентичності цифрового сертифікату сервера починається обмін даними між сервером і клієнтом користувача.

Існують різні способи придбання SSL-сертифіката для серверу. Найдоступніший варіант - це використання самоподпісного сертифіката self-signed. Він генерується власником серверу самостійно, і в багатьох панелях управління хостингом ця функція включена за замовчуванням. Плюси такого способу - доступність і відсутність оплати за використання. Але при цьому не можна говорити про високу безпеку атестата, а більшість браузерів і сервісів будуть видати помилку і попередження про те, що сервер не підтверджений. Self-signed сертифікати найчастіше застосовуються для використання всередині компанії.

Інший спосіб отримання сертифікату - це звернення до постачальника послуг хостингу і налаштування протоколу HTTPS. Також можна налаштувати атестат, скориставшись послугами стороннього провайдера або отримати його від центру сертифікації.

1.4.3. Вимоги до складу та параметрів технічних засобів

Інформаційна технологія на основі сервісу «GetMessage» є клієнт-серверною інформаційною технологією та складається з трьох умовних частин:

1. Android-додаток, або клієнт з яким безпосередньо взаємодіє користувач і отримує доступ до всіх функцій.

2. Сервер, що надає API для клієнта і виконує операції збереження інформації в БД.

3. БД – база даних в якій зберігаються всі дані.

Вимоги до технічних засобів кожної частини інформаційної системи наведено в таблиці 1.1.

Таблиця 1.1

Вимоги до технічних засобів

Назва сервісу	Технічні засоби
Androd-додаток	Пристрій[1] на базі ОС Android
API	1 Сервер[2]
БД	1 Сервер бази даних[3]

Примітки:

1. Під «Пристроєм» мається на увазі апаратна платформа на базі ОС Android, що має не менше одного гігабайту оперативної памяті та 20 мегабайт вільного дискового простору.

2. Під «Сервером» тут мається на увазі апаратна платформа, що має забезпечувати швидку обробку данних при великій кількості користувачів, тому потребується наявність чотирьохядерного процесору з тактовою частотою 2.4ГГц і розрядністю 64 біти та оперативною памяттю 8 гігабайт. Також по мірі зростання навантаження на сервер, ці характеристики необхідно буде збільшувати.

3. Під «Сервером бази даних» мається на увазі апаратна платформа, що має забезпечувати збереження всіх даних для успішної роботи інформаційної системи і мати не менше 10 гігабайт вільного дискового простору, який

необхідно буде збільшувати по мірі зростання користувачів і збільшення потоку данних.

1.4.4. Вимоги до інформаційної та програмної сумісності

Мінімальними вимогами для встановлення і ефективної роботи Android-додатку є наявність на апаратному забезпеченні операційної системи Android версії не нижче ніж 4.2.

Для розгортання серверної частини ІС, на апаратному забезпеченні повинно бути встановлене програмне забезпечення Apache HTTP Server з підтримкою PHP версії 5.6.40.

Для створення і обслуговування бази даних, на апаратному забезпеченні повинна бути встановлена система управління базами даних MySQL та програмне забезпечення phpMyAdmin для адміністрування БД.

1.5. Висновок

Отже, основною метою виконання кваліфікаційної роботи є створення зручного інструменту, який дозволить користувачам обмінюватися текстовими повідомленнями в будь-який при підключенні до мережі Інтернет. Розроблюваний сервіс повинен відповідати всім встановленим в пункті 1.4 вимогам. А саме повинен мати інтуїтивно зрозумілий інтерфейс, завдяки якому технологію можна використовувати одразу після інсталяції, без прочитання додаткових інструкцій, мати підтримку різних версій операційної системи Android, бути захищеної від можливого перехоплення повідомлень та дій користувачів через мережу Інтернет. Для виконання всіх вимог встановлених для сервісу необхідні використати найефективніші технології створення програмних продуктів та інформаційних систем та застосувати їх для досягнення поставлених цілей.

РОЗДІЛ 2. МОДЕЛІ ТА МЕТОДИ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1. Функціональне призначення системи

Основним функціональним призначенням сервісу на основі застосунку «GetMessage» є надання користувачеві можливості обміну миттєвими повідомленнями з іншими користувачами при підключенні до мережі Internet.

2.2. Опис математичних методів необхідних для створення сервісу

При розробці ІТ «GetMessage» використовувався протокол Діффі-Геллмана, який є частиною SSL [33]. SSL - криптографічний протокол, який використовує асиметричну криптографію для аутентифікації ключів обміну, симетричне шифрування для збереження конфіденційності, коди аутентифікації повідомлень для цілісності повідомлень.

Протокол Діффі-Геллман - це метод обміну криптографічними ключами. Один з перших практичних прикладів узгодження ключа, що дозволяє двом учасникам, що не мають жодних попередніх даних один про одного, отримати спільний секретний ключ з використання незахищеного каналу зв'язку. Цей ключ можна використати для шифрування наступний сеансів зв'язку, что використовують шифр з симетричним ключем[2].

2.3. Опис необхідних технологій та мов програмування

Для розробки різних частин сервісу «GetMessage» використовувалося різні мови програмування та технології.

Для розрбки Android-додатку застосовувались такі мови програмування, технології та архітектурні стилі:

- Kotlin – основна мова програмування.

- Clean Architecture – архітектурний стиль.
- Model-View-ViewModel - шаблон проєктування.
- Kotlin coroutines – технологія багатопоточноті.
- LiveData – архітектурний компонент Android.
- Room – архітектурний компонент Android.
- ViewModel – архітектурний компонент Android.
- Data Binding – архітектурний компонент Android.
- Dagger 2 – фреймворк для впровадження залежностей Andorid.
- SQL – декларативна мова програмування.
- Firebase Cloud Messaging – технологія для відправки push-повідомлень.

Для розробки серверної частини інформаційної системи застосовувалась мови програмування та технології:

- PHP – скриптова мова програмування.
- SQL – декларативна мова програмування.
- Firebase Cloud Messaging – технологія для відправки push-повідомлень.

Для розробки Android-застосуну основною мовою програмування було обрано Kotlin. Kotlin (Ко́тлін) — статично типізована мова програмування, що працює поверх JVM і розробляється компанією JetBrains. Також компілюється в JavaScript. Мову названо на честь острова Котлін у Фінській затоці, на якому розміщена частина Кронштадту.

Автори ставили перед собою ціль створити лаконічнішу та типобезпечнішу мову, ніж Java, і простішу, ніж Scala. Наслідками спрощення, порівняно з Scala стали також швидша компіляція та краща підтримка IDE.

Мова розробляється з 2010 року, публічно представлена в липні 2011. Сирцевий код було відкрито в лютому 2012. В лютому було випущено milestone 1, який містив плагін для IDEA. У червні — milestone 2 з

підтримкою Android. У грудні 2012 року вийшов milestone 4 та забезпечив підтримку Java 7. Станом на листопад 2015 року основні можливості мови стабілізовані, готується реліз версії 1.0. В грудні 2015 року з'явився реліз-кандидат версії 1.0, а 15 лютого 2016 року відбувся реліз версії 1.0.

З 17 травня 2017 року входить в список офіційно підтримуваних мов для розробки застосунків для платформи Android.

З 7 травня 2019 року є рекомендованою мовою для розробки Android застосунків.

Як і Java, C і C ++, Kotlin - це статично типізований мова. Він підтримує як об'єктно-орієнтоване, так і процедурне програмування. За аналогією з вищезгаданими мовами, основний код Kotlin-програми пишеться в функції main, якій передається масив аргументів командного рядка.

Основні можливості та переваги Kotlin:

– Компілюється в байткод JVM або в JavaScript.

– Програми можуть використовувати всі існуючі Java-фреймворки і бібліотеки. Kotlin можна інтегрувати з Maven, Gradle і іншими системами збірки.

– Мова дуже проста для вивчення.

– Вихідний код відкритий.

– В IntelliJ доступна автоматична конвертація Java-коду в Kotlin і навпаки.

– Мова null-безпечна – при спробі привласнення або повернення null код не скомпілюється. Проте, в мові є підтримка Nullable-типів. Задати таку змінну або функцію можна, приписавши «?» до назви типу.

– Легко читається синтаксис – прості функції і структури можна оголосити одним рядком, геттери і сеттери задаються за лаштунками для інтероперабельності з Java-кодом. Додавання data-анотації до класу активує автоматичну генерацію різних шаблонів.

– Підтримка функціонального програмування – Kotlin заточений під функціональне програмування. Він надає велику кількість корисних можливостей, наприклад, функції вищого порядку, лямбда-вирази, перевантаження операторів і ледачі обчислення логічних виразів.

– Наявність функцій-розширень – Kotlin дозволяє розширювати функціональність існуючих класів, не вдаючись до наслідування. Це робиться за допомогою функцій-розширень. Для оголошення такої функції до її імені потрібно приписати префікс у вигляді розширюваного типу.

Більш детально з усіма можливостями мови Kotlin можна ознайомитися в роботах [3, 4, 14, 16, 17].

Існує досить багато підходів для побудови складних систем з хорошою архітектурою. Незважаючи на невеликі відмінності цих підходів, у них багато спільного. Вони всі задають способи розбиття програми на окремі модулі. При цьому в кожній системі як мінімум є модулі, що містять бізнес-логіку програми, і модулі для відображення даних. І кожен підхід в підсумку дозволяє побудувати систему, яка задовольняє наступним принципам:

– Архітектура повинна бути незалежна від різних фреймворків. В сучасному світі ми не можемо обходитися без бібліотек, які дозволяють вирішувати завдання набагато швидше і частіше ефективніше, ніж це зробили б ми в разі самостійної реалізації. Але тут важливо розуміти, що бібліотека повинна вбудовуватися в вашу архітектуру, а не архітектура повинна підлаштовуватися під обрану бібліотеку. Потрібно використовувати бібліотеки тільки в якості допоміжних інструментів.

– Система повинна бути протестована. При цьому ви повинні мати можливість тестувати як модулі системи окремо, так і тестувати взаємодію цих модулів між собою і інтеграцію їх в системі. Крім того необхідно тестувати систему без UI, реального сервера і роботи з базою даних, тобто архітектура повинна бути незалежна від оточення.

– З попередніх пунктів плавно випливають і такі принципи, які говорять про те, що ваш додаток має бути незалежним від усього: від інтерфейсу користувача, від роботи з базою даних, від роботи сервера і від інших елементів оточення. Незалежність архітектури від оточення дуже важлива, так як це дозволяє змінювати різні компоненти оточення без зміни самої архітектури. Що мається на увазі під зміною компонентів архітектури? ІЩе може бути зміна у виборі бази даних (або ж взагалі відмова від неї) або ж зміна в UI-частини програми (наприклад, потрібно змінити зовнішній вигляд екрану).

Для побудови архітектури системи розглянемо наступну схему (рис. 2.1), де кожна концентрична окружність є певним компонентом системи:

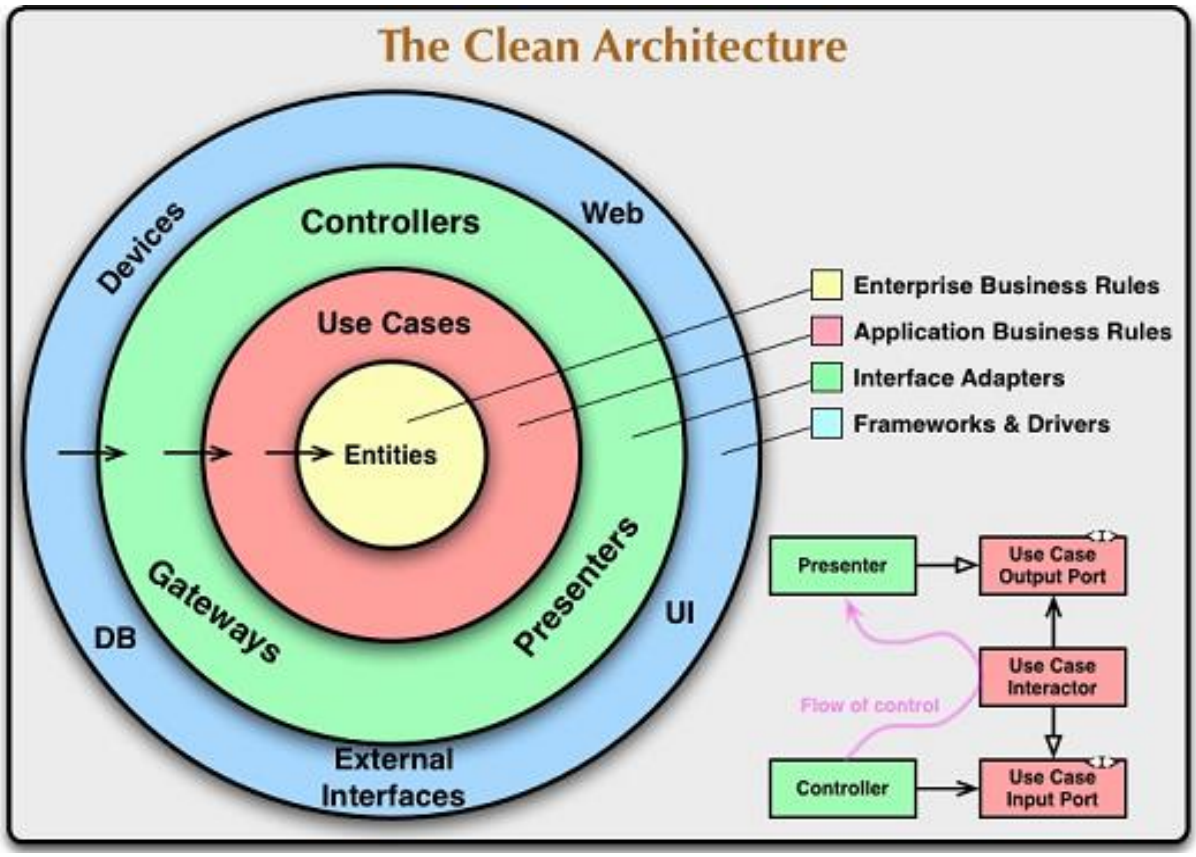

Рис. 2.1. Схема побудови чистої архітектури

Як видно зі схеми, система складається з бізнес-об'єктів, об'єктів для керування даними і бізнес-логікою, з шару уявлення і шару високорівневих

фреймворків. Можна сказати, що в загальному випадку це все таке ж розбиття на подання даних і логіку роботи з ними.

Такий поділ системи на шари є досить логічним і зрозумілим. І поки воно не привносить нічого нового. Головним є правило залежностей - жоден внутрішній шар не повинен знати нічого про зовнішній. Саме це і дозволяє будувати незалежну архітектуру, принципи якої були описані вище.

На рівні бізнес-обїєктів з точки зору роботи з системою в результаті ми завжди працюємо з певними сутностями, які визначаються вимогами системи. По суті бізнес-об'єкти - це класи моделей з певними методами або ж набір якихось структур даних. Ці класи відповідають логіці вашого застосунку, і вони повинні визначати найзагальніші правила поведінки. Оскільки це внутрішній шар, то він буде змінюватися тільки в крайньому випадку, коли ви вирішите змінити саму суть системи. І, зрозуміло, він залишиться незайманим, коли буде змінюватися, наприклад, спосіб роботи з даними або інтерфейс користувача.

Шар сценаріїм взаємодії містить реалізацію основних методів для роботи системи і організовує роботу з даними і бізнес-об'єктами. Він використовує бізнес-об'єкти і їх логіку для того, щоб виконати свої завдання. Цей шар можна розглядати як деякий посередник між бізнес-об'єктами і безпосередньо шаром представлення даних (таке визначення є умовним, так як за правилом залежностей цей посередник не повинен знати нічого про шари уявлення).Також в силу правила залежностей цей шар також не змінюватиметься при зміні якогось з елементів оточення. І зміни на цьому шарі в свою чергу, не будуть зачіпати бізнес-об'єкти.

На шарі представлення здійснюється перетворення даних з формату, який використовують бізнес-об'єкти або сценарії взаємодії, в формат, необхідний для роботи системи. Під роботою системи в даному випадку мається на увазі передача даних для відображення користувачеві або іншій службі.На цьому шарі реалізуються такі архітектурні патерни як Model-View-Controller, Model-View-Presenter або Model-View-ViewModel. Наприклад,

Controller може передавати дані в сценарії взаємодії, отримувати результат і передавати його для відображення у View.

Система не повинна залежати ні від яких фреймворків. Саме тому шар фреймворків на схемі є самим зовнішнім. Оскільки саме на цьому рівні зникає вся абстракція, і ми використовуємо конкретні засоби для вирішення певних завдань. Під конкретними засобами тут мається на увазі база даних, фреймворк для UI та інші служби. Оскільки ми дотримуємося правило залежностей, внутрішні шари нічого не знають про конкретні використовуваних фреймворками, що дозволяє легко змінити будь-який з них без зміни внутрішніх шарів.

Всі переваги використння чистої архітектури та шляхи реструктурзації проєктів для побудови чистої архітектури описані в роботі [18].

Сервіс розроблявся з дотриманням вимог Clean Architecture для того, щоб створити модульний код, окремі модулі якого максимально незалежні один від одного. Схема архітектури застосунків, в тому числі «GetMessage», з дотриманням вимог Clean Architecture представлена на рис 2.2 [29].

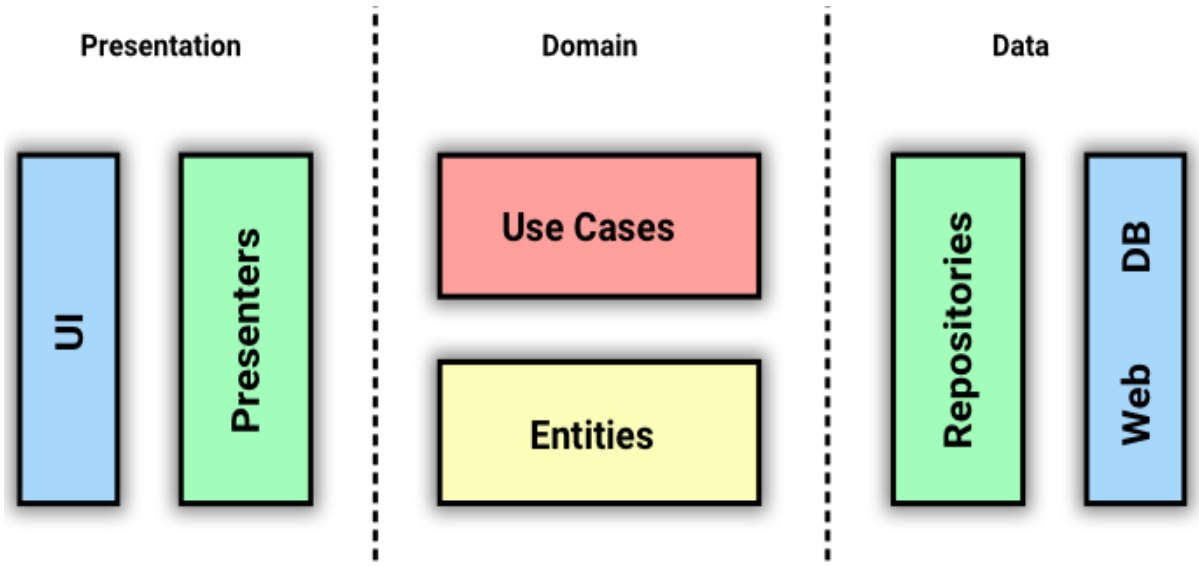

Рис. 2.2. Схема архітектури застосунків Clean Architecture

Структура коду проєкту Android-застосунку складається з трьох

частин, кожна з яких також ділиться на частини:

– Presentation – відповідає за відображення даних, де UI відповідає за взаємодію з користувачем, а Presenters визначає, як відображати дані користувачеві.

– Domain відповідає за бізнес-логіку, де Use Cases містить логіку додатка, а Entities визначає форму даних.

– Data – відповідає за роботу з даними, де Repositories визначає джерело даних, а Web і DB - код, який працює безпосередньо з даними з мережі або БД відповідно.

Процес руху даних в нашому додатку зображений на рис. 2.3.

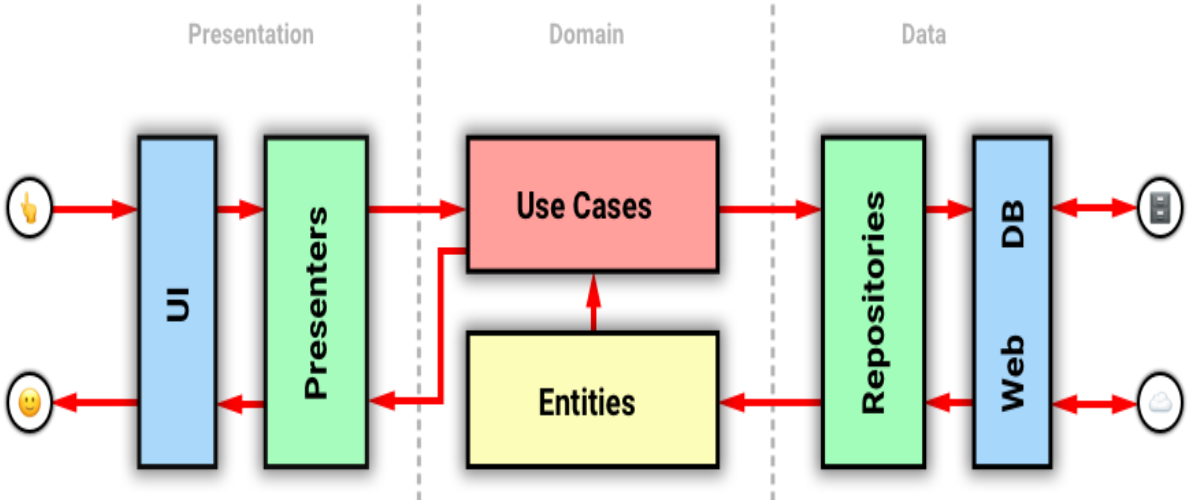

Рис. 2.3. Схема руху даних в додатку

Дані ходять від UI до backend'у або базі даних і назад. Подія користувача йде в Presenter, той передає в Use Case. Use Case робить запит в Repository. Repository отримує дані, створює Entity, передає його в UseCase. Так Use Case отримує всі необхідні йому Entity. Потім, застосувавши їх і свою логіку, отримує результат, який передає назад в Presenter. А той, у свою чергу, відображає результат в UI.

Model-View-ViewModel — це шаблон проєктування, що полегшує відокремлення розробки графічного інтерфейсу від розробки бізнес логіки

(бек-енд логіки), відомої як модель (можна також сказати, що це відокремлення представлення від моделі). Модель представлення є частиною, яка відповідає за перетворення даних для їх подальшої підтримки і використання. З цієї точки зору, модель представлення більше схожа на модель, ніж на представлення і оброблює більшість, якщо не всю, логіку відображення даних.

Шаблон MVVM ділиться на три частини:

– Модель (Model), як і в класичному шаблоні MVC, Модель являє собою фундаментальні дані, що необхідні для роботи застосунку.

– Вид/(Вигляд) (View) як і в класичному шаблоні MVC, вигляд — це графічний інтерфейс, тобто вікно, кнопки тощо.

– Модель вигляду (ViewModel, що означає «Model of View») з одного боку є абстракцією Вигляду, а з іншого надає обгортку даних з Моделі, які мають зв'язуватись. Тобто вона містить Модель, яка перетворена до Вигляду, а також містить у собі команди, якими може скористатися Вигляд для впливу на Модель. Фактично ViewModel призначена для того, щоб здійснювати зв'язок між моделлю та вікном відслідковувати зміни в даних, що зроблені користувачем відпрацьовувати логіку роботи View (механізм команд).

При розробці використовувався шаблон MVVM, але програміст може використовувати також MVC, MVP тощо [15].

Android - це однопоточна платформа, і за замовчуванням все працює на основному потоці (UI-поток або інтерфейс користувача). Коли настає час роботи, не повязаної з використанням інтерфейсу користувача (наприклад, вихід в мережу, робота з БД, операції вводу або виводу, або прийом задачі в будь-який момент), ми розподіляємо задачі між різними потоками і, якщо потрібно, передаємо результат назад у головний потік.

Android має свої механізми для виконання завдань у інших потоках, такі як: AsyncTask, Handler, Services і т.д. Ці механізми включають зворотні

визови, методи публікації та інші прийоми для передачі результатів між потоками.

Важливо уникати блокування основного потоку. Основний потік - це окремий потік, який обробляє всі оновлення призначеного для користувача інтерфейсу. Це також потік, який викликає всі обробники кліків та інші зворотні виклики для користувача інтерфейсу. Він повинен працювати безперебійно, щоб гарантувати відмінний користувальницький досвід.

Щоб додаток відображався користувачеві без видимих пауз, основний потік повинен оновлювати екран кожні 16 мс або частіше, що становить близько 60 кадрів в секунду. Багато звичайних завдання займають більше часу, наприклад, аналіз великих наборів даних JSON, запис даних в базу даних або вибірка даних з мережі. Таким чином, виклик подібного коду з основного потоку може привести до припинення, заїкання або навіть зависання програми. І якщо ви заблокуєте основний потік занадто довго, додаток може навіть аварійно завершити роботу і відобразити діалогове вікно «Додаток не відповідає».

Для вирішення проблеми блокування головного потоку в розробці використовувалася технологія Kotlin Coroutines або корутини. Kotlin Coroutines в додатку для Android - новий спосіб управління фоновими потоками, який може спростити код за рахунок зменшення потреби в зворотніх викликах. Корутини - це функція Kotlin, яка перетворює асинхронні зворотні виклики для тривалих завдань, таких як доступ до бази даних або мережі, в послідовний код.

Коли користувач натискає ярлик програми, система запускає Activity (вікно), яке зазначено як стартове в маніфесті додатка. За замовчуванням це MainActivity, але ми можемо зробити стартовим будь-яке Activity. Після запуску ми бачимо це Activity на екрані. Ми можемо звернути його або відкрити іншу програму, екран налаштувань і т.д. При цьому наше Activity йде на другий план і перестає бути видимим. Однак воно не знищується, а продовжує працювати у фоновому режимі, і ми можемо в будь-який момент

до нього повернутися. Ми можемо зовсім закрити програму, або його може закрити система, якщо їй буде не вистачати оперативної пам'яті. Тоді Activity припинить свою роботу і буде знищено.

Activity в процесі роботи може знищуватися і створюватися заново. Це відбувається при подіях зміни конфігурації, таких як поворот пристрою з портретної орієнтації в альбомну (і назад), зміна розміру екрану або мови інтерфейсу.

При цьому Activity проходить низку станів, які складають життєвий цикл Activity (рис. 2.4).

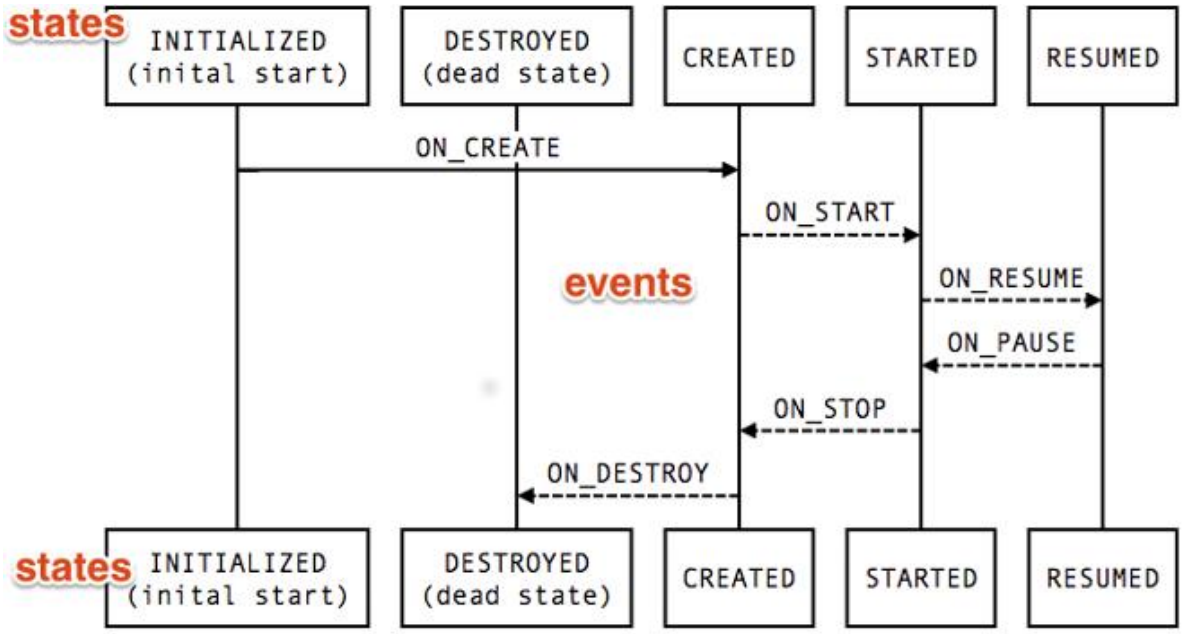

Рис. 2.4. Схема життєвого циклу Activity

Якщо користувацький інтерфейс відображає якісь дані, при зміні конфігурації вони будуть знищені разом з Activity. Це стає проблемою, якщо ці дані отримані в результаті запиту до БД або з мережі. При простому повороті пристрою нам доведеться знову виконувати дорогі операції завантаження даних. Але цього можна уникнути, якщо використовувати архітектурні компоненти ViewModel і LiveData.

Допоміжний клас ViewModel з набору Android Jetpack

використовується для контролера, який відповідає за підготовку даних для призначеного для користувача інтерфейсу. Об'єкти ViewModel автоматично зберігаються під час змін конфігурації, тому що містяться в них дані відразу ж стають доступні для наступного примірника активують або фрагмента.

На рис. 2.5 відображено, як ViewModel взаємодіє з життєвим циклом Activity.

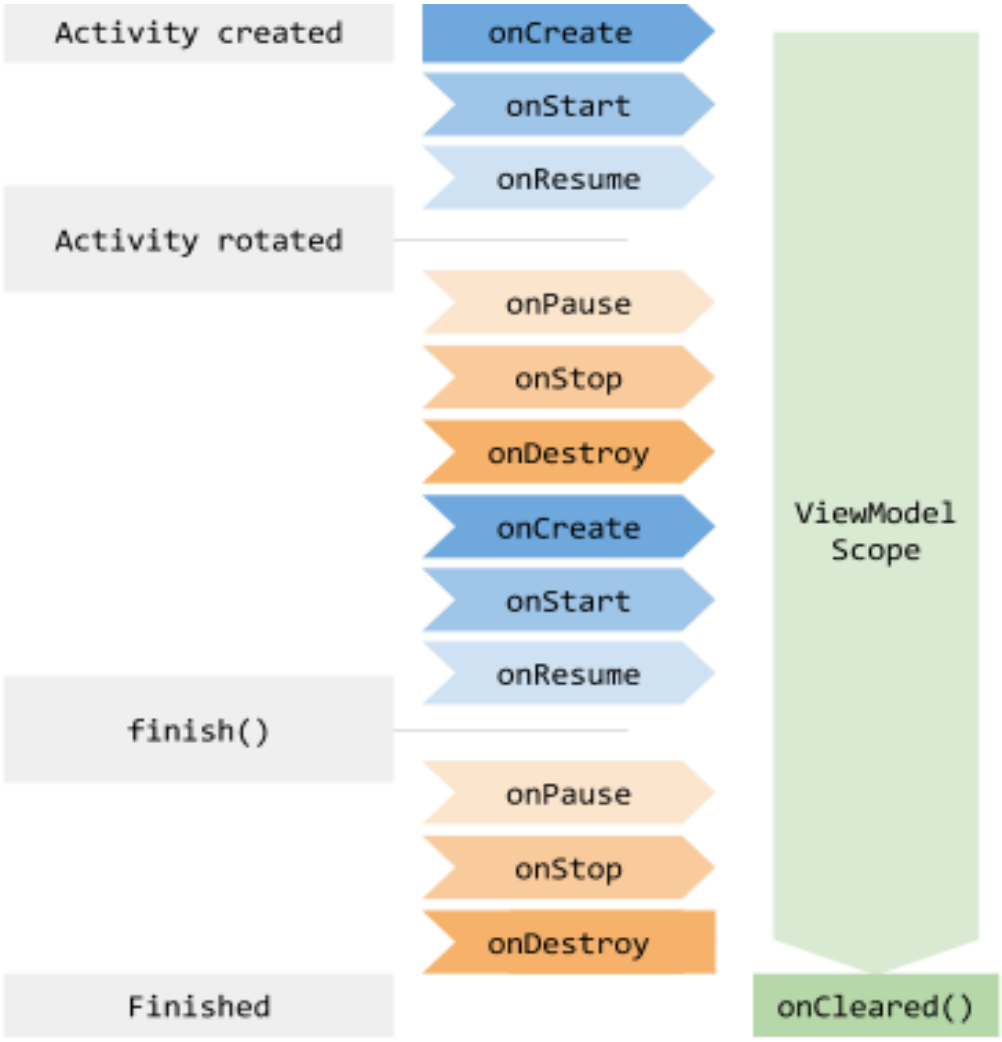

Рис. 2.5. Схема взаємодії ViewModel з життєвим циклом Activity

LiveData – клас, сховище даних, що працює за принципом паттерна Observer (спостерігач). Це сховище вміє робити дві речі:

1) В нього можна помістити деякий об’єкт.

2) На нього можна підписатися і отримувати об'єкти, які в нього

поміщають.

Тобто з одного боку хтось поміщає об'єкт в сховище, а з іншого боку хтось підписується і отримує цей об'єкт. Як правило обєкти класу LiveData є частиною ViewModel, тому вони не знищуються при знищенні Activity і використовуються для зберігання даних різних типів. LiveData вміє визначати активний підписник чи ні, і відправляти дані буде тільки активним підписникам. Передбачається, що підписниками LiveData будуть Activity і фрагменти. А їх стан активності буде визначатися за допомогою їх Lifecycle об'єкта.

Якщо перед розробником постає питання зберігання великого обсягу даних, то рішення буде цілком очевидним - потрібно використовувати базу даних. Для цих цілей в Android є підтримка бази даних SQlite засобами якої, можна вирішити переважну більшість завдань, проте використовувати її часто не зовсім зручно. Навіть для найпростіших операцій вставки/отримання даних, доводиться писати багато однотипного коду, що може привести не тільки до незручностей, але і до помилок. Тому для зручності при розробці використовувалася бібліотека Room. Room – це бібліотека, що надає нам зручну обгортку для роботи з базою даних SQLite, тобто це прошарок між API для взаємодії з базою і вашим кодом.

Бібліотека Data Binding Library, є частиною Android Jetpack, дозволяє прив'язувати компоненти користувальницького інтерфейсу в макетах до джерел даних в додатку, використовуючи декларативний формат, а не програмно. Іншими словами, Data Binding допоможе організувати роботу з View так, щоб нам не довелося писати купу методів findViewById, setText, setOnClickListener і т.п.

Dagger 2 – це повністю статичний фреймворк для впровадження залежностей в Kotlin, Java і Android, що працює під час компіляції та реалізує шаблон Dependency Injection.

Dependency Injection - це програмний шаблон, який реалізує принцип об'єктно-орієнтованого програмування «Інверсія управління (Inversion Of

Control)». Реалізація цього шаблону передбачає зниження «пов'язаності коду», відповідно, виходить код, який легше використовувати повторно і супроводжувати, тобто, зміна компонент однієї частини програми не викликає помилок в іншій частині або необхідності значних каскадних змін.

Плюси використання технології Dagger 2:

– Доводиться писати менше шаблонного коду.
– Допомагає структурувати залежності.
– Значною мірою спрощує роботу коли залежностей багато.
– Код стає простим для читання.

Отримати детальний опис Dependency Injection та Dagger 2 та методи використання цих технологій можна в роботі [13].

SQL – декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних та її модифікації, системи контролю за доступом до бази даних. Сама по собі SQL не є ані системою керування базами даних, ані окремим програмним продуктом. На відміну від дійсних мов програмування, SQL може формувати інтерактивні запити або, бувши вбудованою в прикладні програми, виступати як інструкції для керування даними. Окрім цього, стандарт SQL містить функції для визначення зміни, перевірки та захисту даних. SQL — це діалогова мова програмування для здійснення запиту і внесення змін до бази даних, а також керування базами даних. Багато баз даних підтримує SQL з розширеннями до стандартної мови. Ядро SQL формує командна мова, яка дозволяє здійснювати пошук, вставку, оновлення і вилучення даних за допомогою використання системи керування і адміністративних функцій. SQL також включає CLI (Call Level Interface) для доступу і керування базами даних дистанційно.

Мова SQL буде використовуватися для створення і управління базою даних як для створення андроід-застосунку, так і для створення серверної частини сервісу.

Детально з синтаксисом та принципами роботи мови SQL можна ознайомитися в роботах [10,11,12,23].

Firebase Cloud Messaging - це кроссплатформне рішення для обміну хмарними повідомленнями та push-сповіщеннями від Google, які дозволяють розробникам відправляти push-сповіщення своїм кінцевим користувачам через Firebase Notification Composer або через набор API. FCM підтримує два типи повідомлень: сповіщення та повідомлення з даними.

Сповіщувальні повідомлення - це явні push-повідомлення, які показуються на пристрої користувача для інформування або запиту про участь. Повідомлення з даними навпаки, безпосередньо оброблюються в фоновому режимії і можуть використувуватись для оновлення табличного представлення даних.

FCM може відправляти повідомлення про окремі пристрої, групи пристроїв або пристроїв, підписані на теми.

PHP - скриптова мова програмування, створена для генерації HTML-сторінок на веб-сервері і роботи з базами даних. В даний час підтримується переважною більшістю представників хостингу.

В галузі програмування для мережі PHP - один з найпопулярніших скриптових мов завдяки своїй простоті, швидкості виконання, багатій функціональності і розповсюдженню початкових кодів на основі ліцензії PHP.

PHP відрізняється наявністю ядра і модулів, «розширень» для роботи з базами даних, сокетами, динамічною графікою, криптографічними бібліотеками, документами формату PDF і т. п. Будь-який бажаючий може розробити своє власне розширення і підключити його. Існують сотні розширень, проте в стандартну поставку входить лише кілька десятків, які

добре зарекомендували себе. Інтерпретатор PHP підключається до веб-серверу або через модуль, створений спеціально для цього сервера.

Детальний опис і особливості використання PHP та способи взаємодії з мовою SQL описані в роботах [20, 22].

2.4. Обгрунтування та організація вхідних та вихідних даних програми

Вхідними даними інформаційної системи є текстова інформація яку вводить користувач Android-застосунку в відповідні поля для вводу тексту та діалогові вікна, а також графічна інформація, тобто зображення з галереї.

Вихідними даними інформаційної системи є текстова інформація, яка відображається в графічних елементах Android-застосунку, а також графічна інформація, тобто зображення, які користувач отримує від інших користувачів та при перегляданні аккаунтів інших користувачів.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ РОЗРОБЛЕНИХ ФРАГМЕНТІВ

3.1. Опис структури сервісу та алгоритмів функціонування

Загалом, сервіс складається з трьох основних частин – це:

1. Android-застосунок.
2. Сервер.
3. База даних.

Android-застосунок – це єдина частина з якою безпосередньо взаємодіє користувач. Тобто це встановлений додаток на пристрої користувача з ОС Android, який виступає в ролі клієнта інформаційної системи. За допомогою цього застосунку користувач отримує доступ до всіх функціональних можливостей інформаційної системи. Він може виконати реєстрацію або вхід до вже існуючого аккаунту, відправити повідомлення, додати або видалити друга і т.д. Так як інформаційна технологія є клієнт-серверною, то клієнт, тобто Android-застосунок, має взаємодіяти з серверною частиною. Ця взаємодія відбувається за рахунок виконання HTTPS-запитів від клієнта до сервера. Кожен Android-застосунок має свій унікальний ідентифікатор – токен в інформаційній системі, який йому надає Firebase Cloud Messaging.

В разі відправлення клієнтом-відправником запиту на додавання нових даних до БД, це може бути відправка нового повідомлення або запиту на додавання в друзі, сервер ідентифікує клієнта по його токену та перевіряє права доступу клієнта до БД. Якщо клієнт має доступ на додавання нових даних, сервер звертаєтья до БД, виконує необхідний запит на мові SQL, після чого дані в БД оновлюються. Відправляє клієнту-відправнику відповідь про успішність виконання операції. Після чого формує запит до Firebase Cloud Messaging на відправку push-повідомлення до клієта-отримувача по його токену.

Клієнт-отримуач отримує push-повідомлення, яке відображується на пристрої користувача, після чого виконує запит до сервера на отримання

оновлених даних з БД(які до цього додав в БД клієнт-відправник), сервер ідентифікує клієнта-отримувача, звертається до БД, отримує дані з БД, та відповідає клієнту-отримувачу необхідними даними.

Схема взаємодії структурних компонентів інформаційної системи зображена на рис. 3.1.

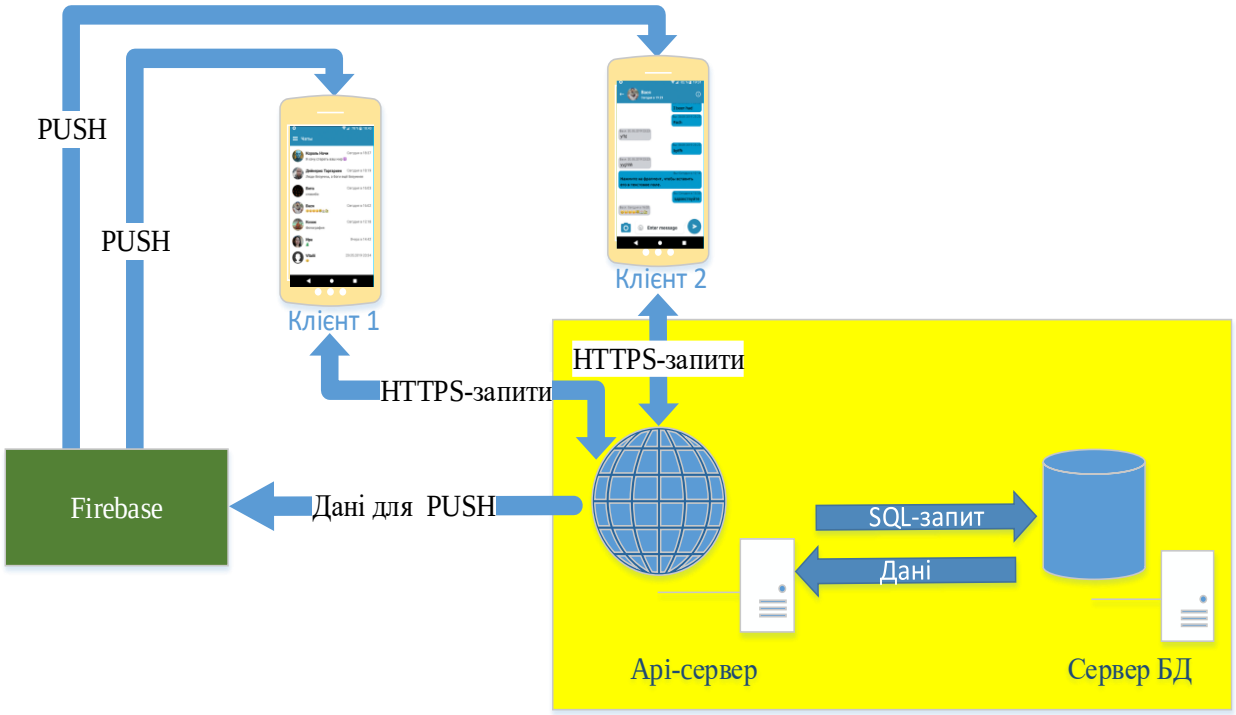

Рис. 3.1. Схема взаємодії структурних компонентів «GetMessage»

Структуру бази даних зображено на рис. 3.2.

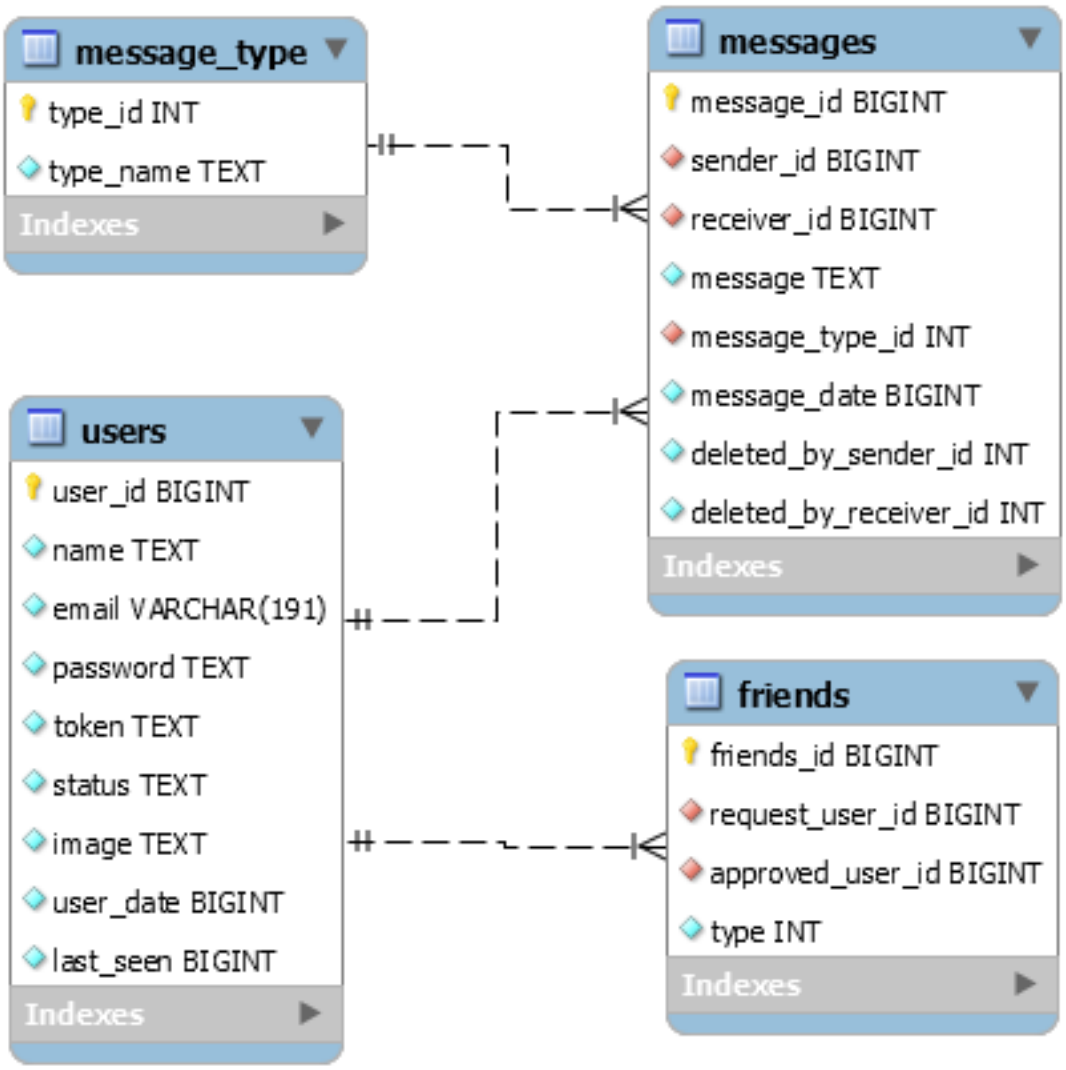

Рис. 3.2. Структура бази даних «GetMessage»

3.2. Використані технічні засоби

Для розробки інформаційної системи «GetMessage» були використані такі технічні засоби:

1. Ноутбук Acer Aspire 5 A515-54G-56DQ.
2. Серверний компьютер Dell PowerEdge R710.
3. Смартфон Xiaomi Redmi Note 5 Plus.

Ноутбук Acer Aspire 5 A515-54G-56DQ використувався для розробки Android-застосунку та програмного забезпечення для серверної частини, а

також для віддаленого розгортання серверної частини інформаційної системи.

Ресурси серверного компютеру Dell PowerEdge R710 були взяті в оренду у постачальника послуг веб-хостингу «Regery» для розгортання серверного програмного забезпечення.

Смартфон Xiaomi Redmi Note 5 Plus використовувався для тестування функцій Android-застосунку.

3.3. Використані програмні засоби

Для розробки «GetMessage» були використані такі програмні засоби:

1. Android Studio – інтегроване середовище розробки Android-застосунку.
2. PhpStorm – інтегроване середовище розробки для PHP.
3. Apache HTTP Server – серверне оточенням для виконання php-коду.
4. PhpMyAdmin – веб-додаток для адміністрування СУБД MySQL.

Android Studio (рис 3.3) - інтегроване середовище розробки (IDE) для роботи з платформою Android, анонсоване 16 травня 2013 року на конференції Google.

Дане середовище розробки перебувало у вільному доступі починаючи з версії 0.1, опублікованій в травні 2013, а потім перейшла в стадію бета-тестування, починаючи з версії 0.8, яка була випущена в червні 2014 року. Перша стабільна версія 1.0 була випущена в грудні 2014 року, тоді ж припинилася підтримка плагіна Android Development Tools (ADT) для Eclipse.

Android Studio, заснована на програмному забезпеченні IntelliJ IDEA від компанії JetBrains, - офіційне засіб розробки Android додатків. Дане середовище розробки доступне для Windows, macOS і GNU / Linux]. 17 травня 2017, на щорічній конференції Google I / O, Google анонсував підтримку мови Kotlin, використовуваного в Android Studio, як офіційної

мови програмування для платформи Android на додаток до Java і C ++[25].

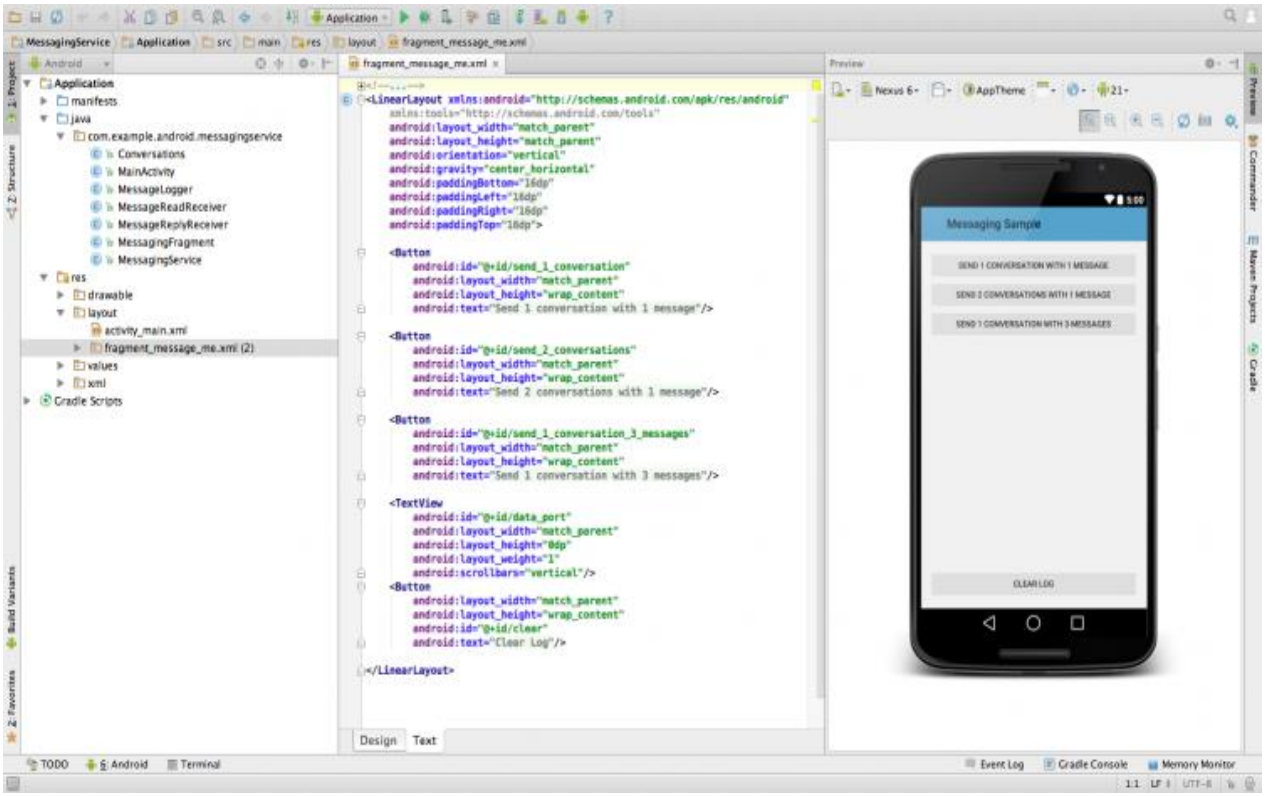

Рис. 3.3. Інтерфейс Android Studio

PhpStorm (рис. 3.4) - комерційне крос-платформне інтегроване середовище розробки для PHP. Розробляється компанією JetBrains на базі платформи IntelliJ IDEA.

PhpStorm являє собою інтелектуальний редактор для PHP, HTML та JavaScript з можливостями аналізу коду на льоту, запобігання помилкам у коді та автоматизованими засобами рефакторингу для PHP та JavaScript. Автодоповнення коду в PhpStorm підтримує специфікацію PHP 5.3, 5.4, 5.5, 5.6, 7.0, 7.1, 7.2, 7.4 та 8.0. Є повноцінний SQL-редактор із можливістю редагування отриманих результатів запитів.

PhpStorm розроблено на основі платформи IntelliJ IDEA, написаної на Java. Користувачі можуть розширити функціональність середовища розробки з допомогою установки плагінів, розроблених для платформи IntelliJ, або написавши власні плагіни[26].

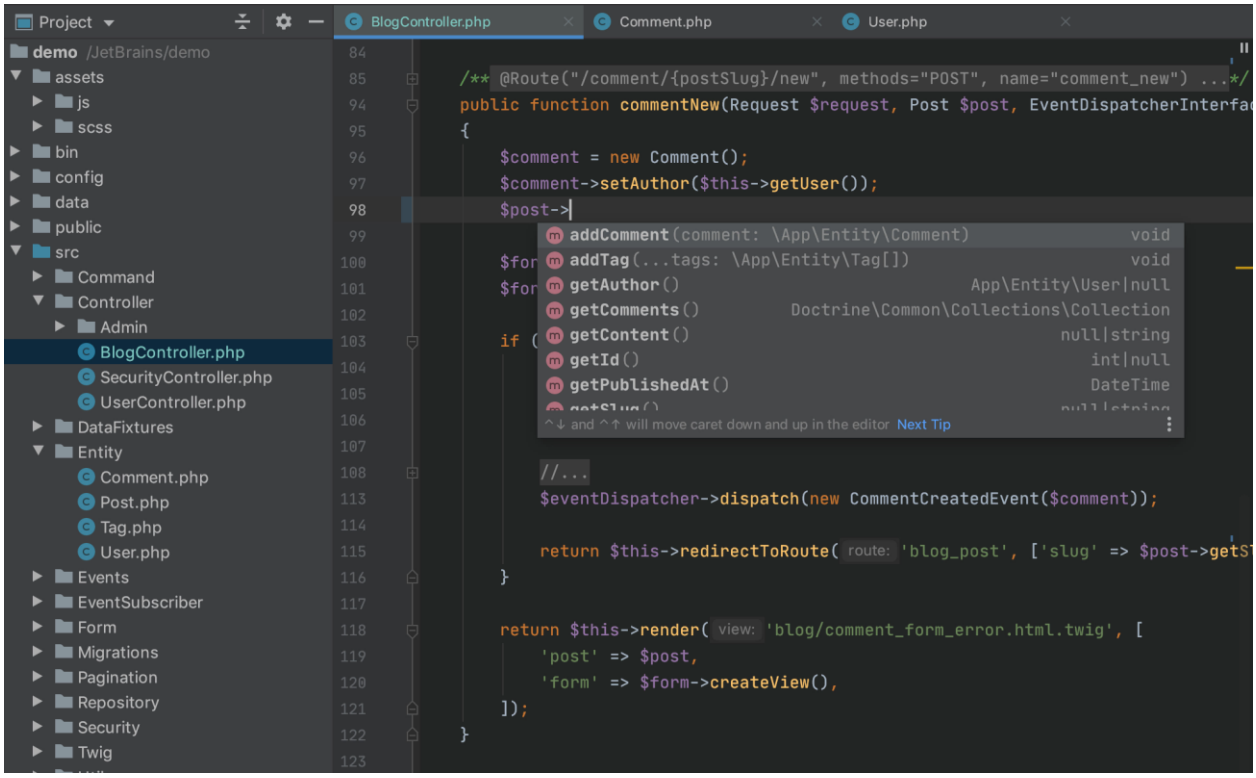

Рис. 3.4. Інтерфейс PhpStorm

Apache HTTP-сервер — відкритий веб-сервер Інтернет для UNIX-подібних, Microsoft Windows, Novell NetWare та інших операційних систем.

Apache розроблюється та підтримується спільнотою розробників відкритого програмного забезпечення під керівництвом Apache Software Foundation.

В 1996 році Apache обійшов NCSA HTTPd із того часу є найбільш популярним веб-сервером у світі. Станом на червень 2013 року Apache встановлений на 53.34% (358 974 045 серверів) для порівняння на другому місці Microsoft IIS їхня частка 17.22% (115 920 681 серверів)[27].

PhpMyAdmin (рис. 3.5) - веб-додаток з відкритим кодом, написаний на мові PHP і представляє собою веб-інтерфейс для адміністрування СУБД MySQL. PhpMyAdmin дозволяє через браузер і не тільки здійснювати адміністрування сервера MySQL, запускати команди SQL і переглядати вміст таблиць і баз даних. Додаток користується великою популярністю у веб-

розробників, так як дозволяє управляти СУБД MySQL без безпосереднього введення SQL команд.

Додаток поширюється під ліцензією GNU General Public License і тому багато інших розробники інтегрують його в свої розробки, наприклад XAMPP, Denwer, AppServ, Open Server.

Проект на даний момент часу локалізований на більш ніж 62 мовах[28].

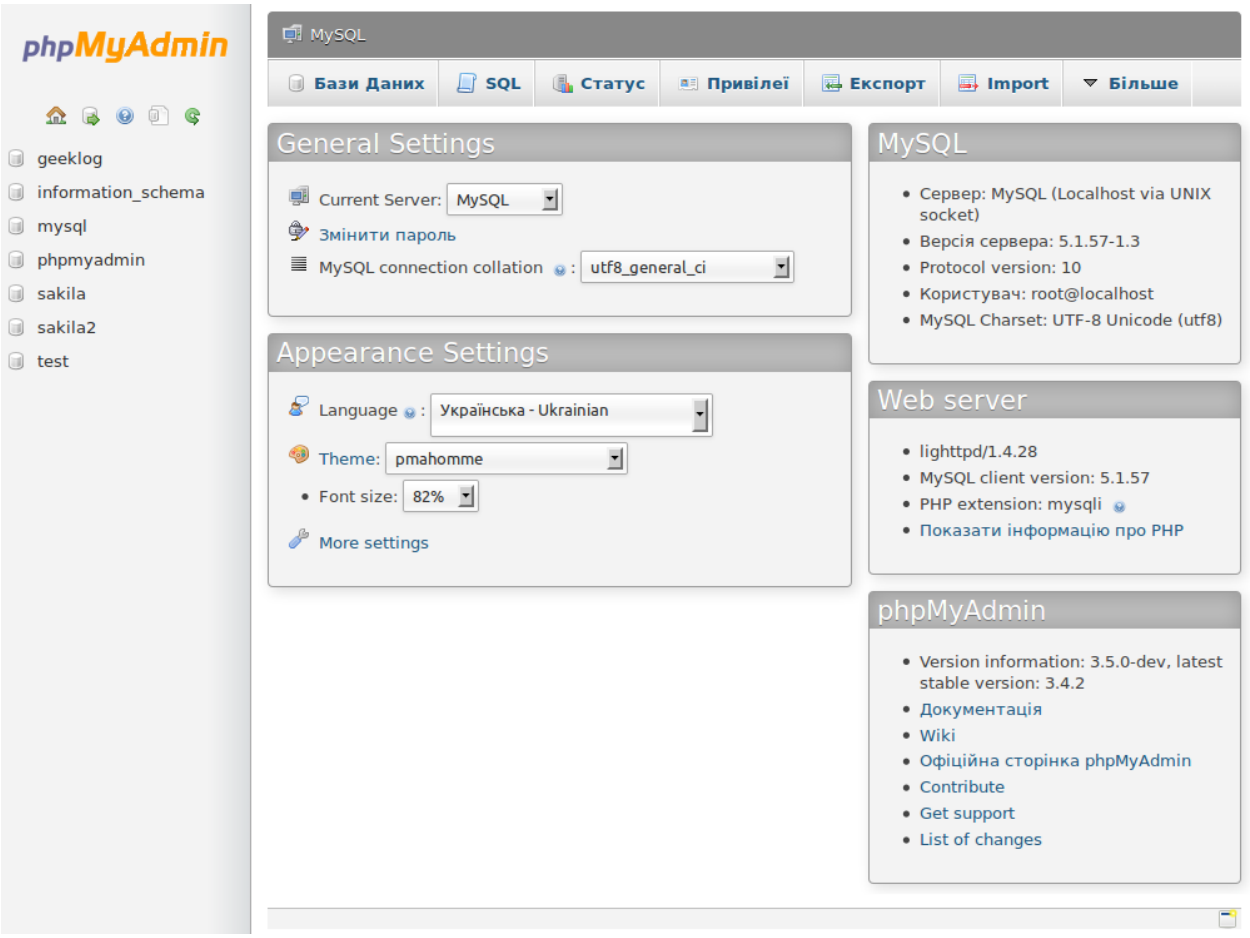

Рис. 3.5. Інтерфейс PhpMyAdmin

3.4. Виклик та завантаження програми

Для встановлення Android-застосунку необхідно попередньо перенести файл формату apk на пристрій користувача та почати його завантаження в систему.

Для запуску Android-застосунку необхідно відкрити застосунок звичайними засобами ОС Android.

3.5. Опис інтерфейсу користувача

При користуванні інформаційною системою «GetMessage» користувач безпосередньо взаємодіє лише з Android-застосунком. Тому цей компонент інформаційної системи повинен мати зручний інтуїтивно-зрозумійлий інтерфейс.

При першому відкритті додатку користувач потрапляє на стартове вікно авторизації додатку (рис. 3.6), де користувачеві запропоновано ввести його адресу електронної пошти та пароль для входу. Якщо користувач ще немає акаунту в системі, то він може відкрити вікно реєстрації (рис. 3.7) натиснувши кнопку «Ще немає акаунту?». Якщо користувач має акаунт, але він забув свій пароль, то він може відкрити вікно нагадування паролю (рис. 3.8) натиснувши кнопку «Забули пароль?». В вікні нагадування паролю користувач повинен ввести адресу електронної пошти, використаної при реєстрації для отримання паролю від облікового запису. Після введення електронної пошти і натискання кнопки «Відправити пароль» користувач отримає повідомлення на свою поштову скриньку зі своїм поточним паролем та зможе виконати вхід до свого облікового запису через вікно авторизації.

Рис. 3.6. Стартове вікно авторизації додатку

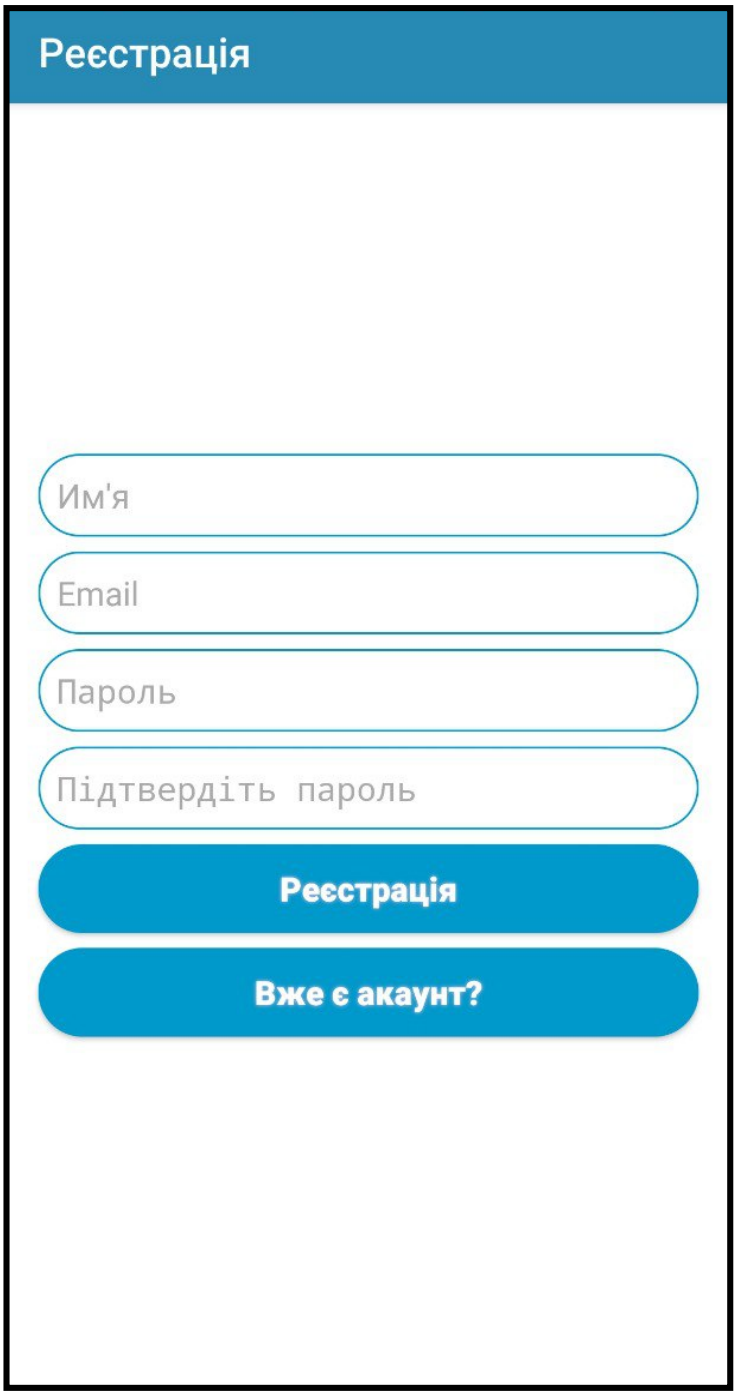

Рис. 3.7. Вікно реєстрації

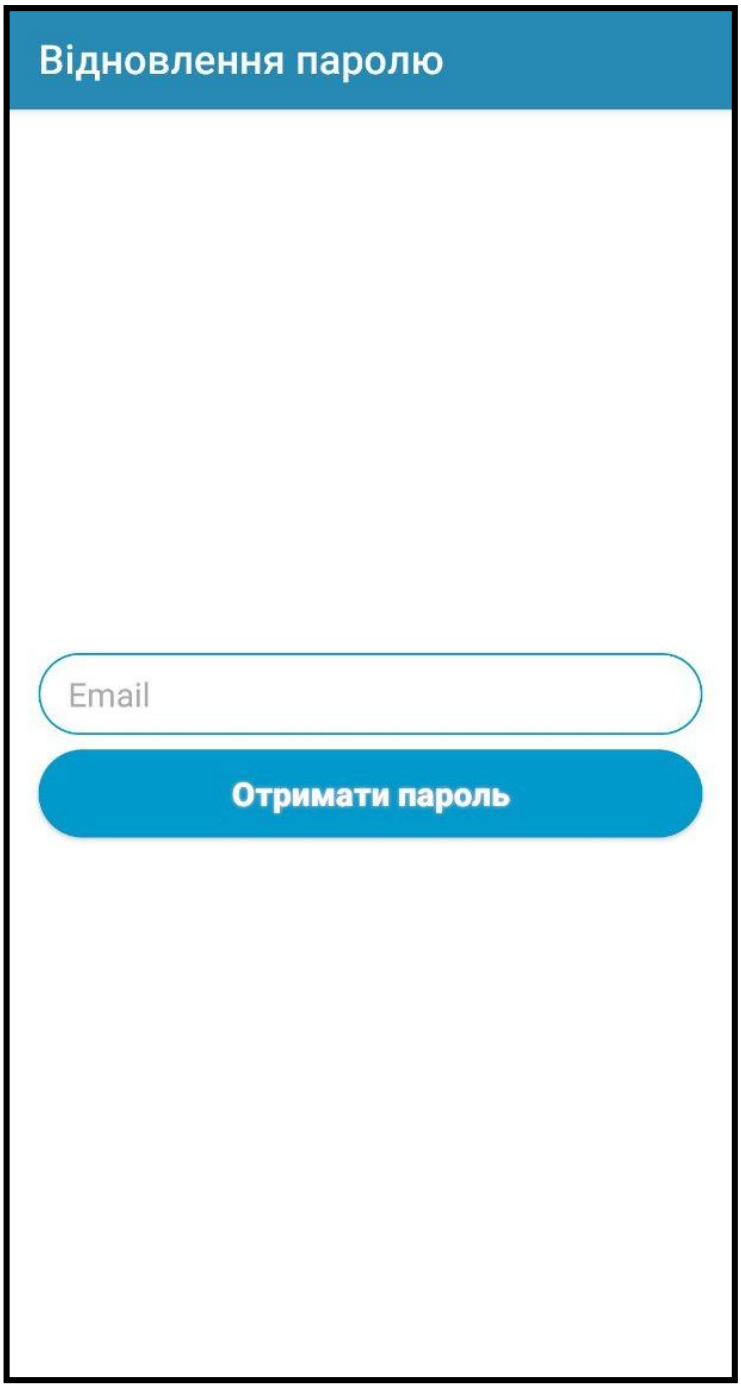

Рис. 3.8. Вікно нагадування паролю

Після входу в систему користувачеві відкривається головне вікно додадку (рис. 3.9) на якому відображаються діалоги з іншими користувачами. При проведенні пальцем зліва направо, або при натисканні кнопки в лівому верхньому вуглі відкриється головне меню (рис. 3.10).

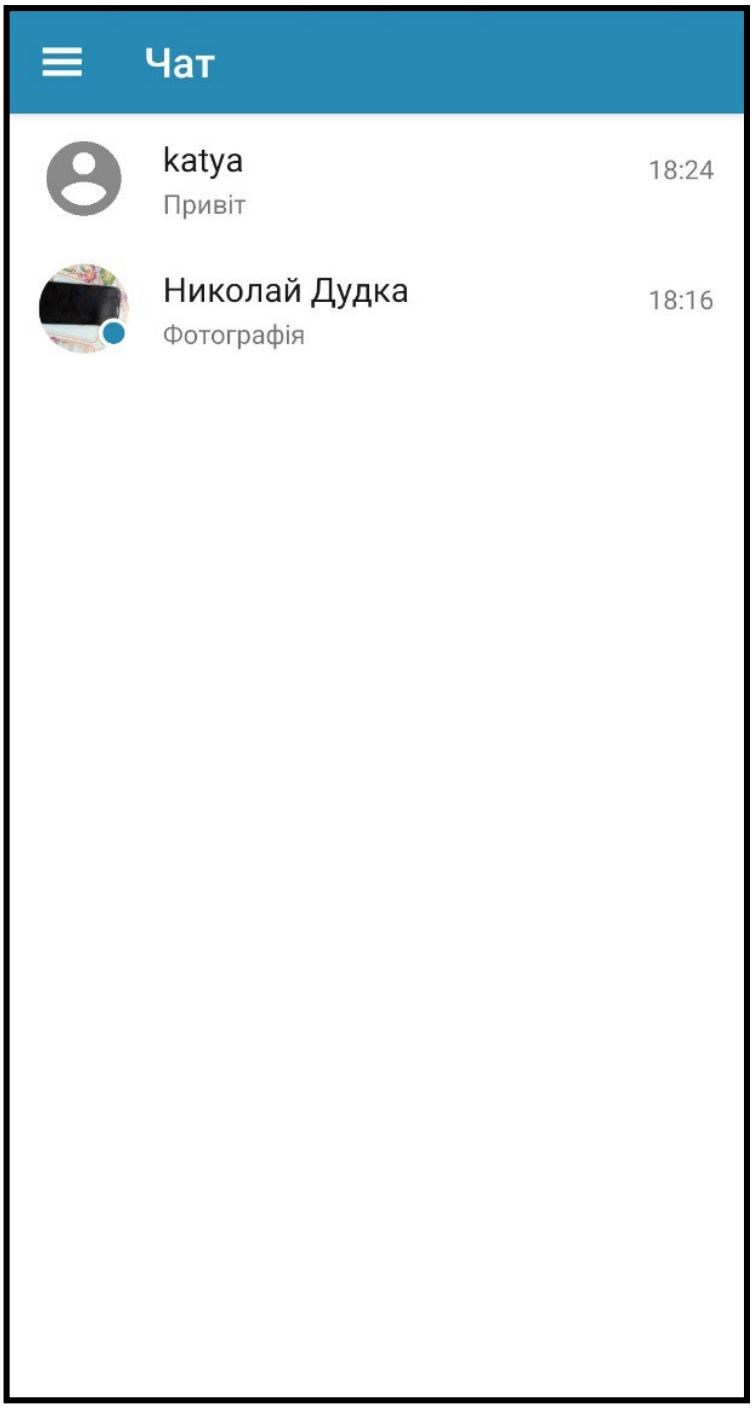

Рис. 3.9. Головне вікно додатку

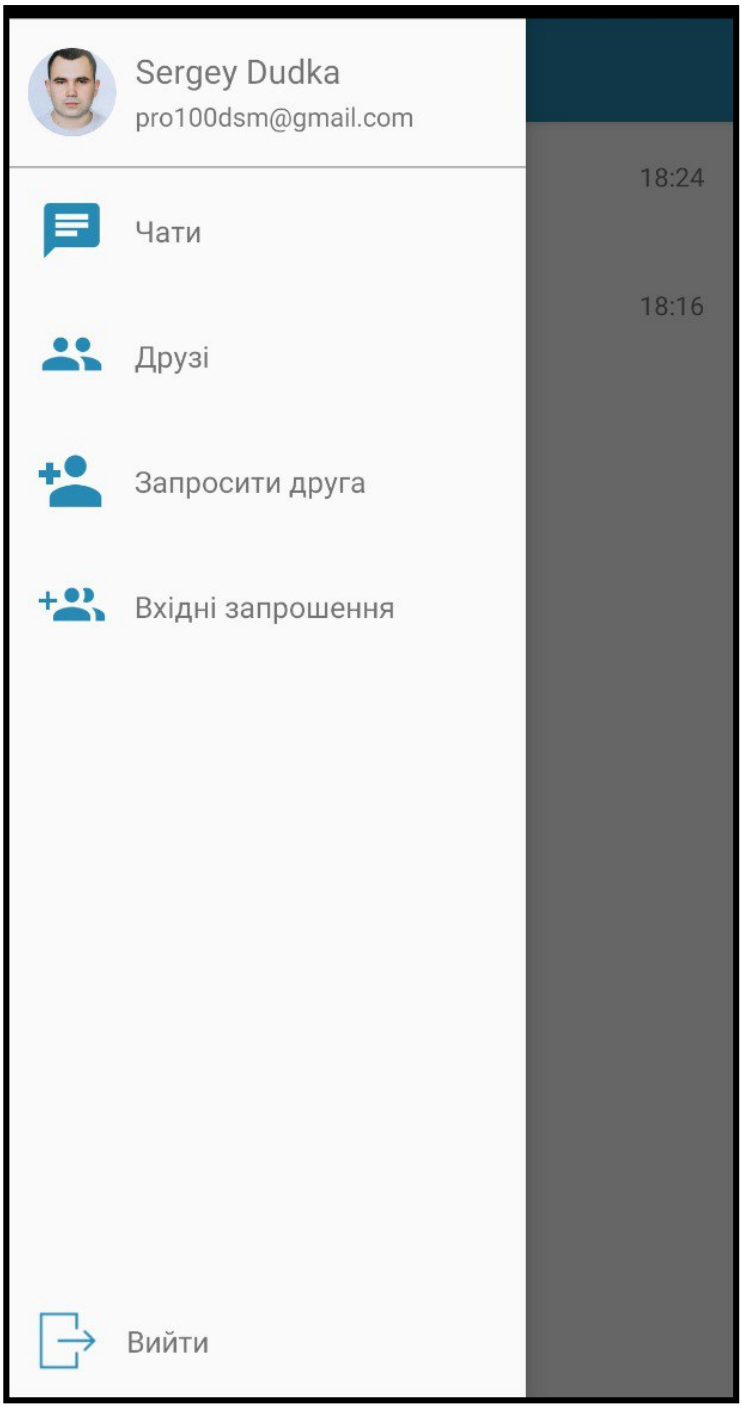

Рис. 3.10. Головне меню

В головному вікні також відображається статус друзів в межері. Якщо користувач в мережі, біля фото його профілю в правому нижньому вуглі буде відображатися спеціальне зображення.

При натисканні користувачем в головному меню на своє ім'я відкриється вікно редагування профілю користувача (рис. 3.11). В цьому вікні буде відображена основна інформація акаунту, а також фото

коистувача. Коритувач може встановити або змінити фото профілю натиснувши на відповідну іконку, після чого обрати фото з галереї або зробити його за допомогою камери свого пристрою. Для зміни інформації акаунту користувачу необідно заповнити відповідні поля і натиснути кнопку «Оновити акаунт». Після натискання кнопки «Оновити акаунт» оновлені дані будуть збережені на сервері і в подальшому для авторизації необхідно буде використовувати оновлені дані електронної пошти або паролю, якщо вони були змінені.

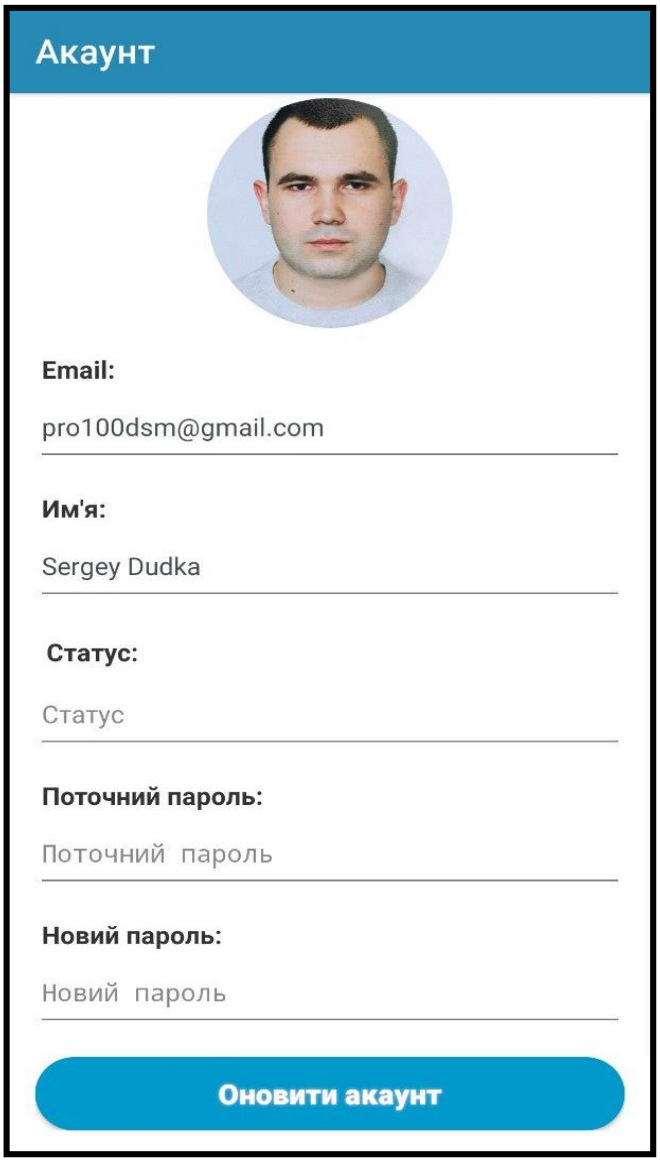

Рис. 3.11. Вікно редагування профілю користувача

При натисканні на діалог з користувачем в головному вікні додатку, де відображаються усі наявні діалоги з іншими користувачами, відкриється вікно діалогу к обраним користувачем (рис. 3.12).

Рис. 3.12. Вікно діалогу з користувачем

У вікні діалогу користувач може відправити іншому користувачу текстове повідомлення, ввівши його в відповідне поле та натиснувши кнопку

відправки. Також можливо відправити фото, для цього треба натиснути на відповідну кнопку та обрати фото з галереї або зробити фото за опомогою камери пристрою. При довгому натисканні на конкретному повідомленні, зявиться діалогове вікно підтвердження видалення повідомлення (рис. 3.13). Після підтвердження видалення, повідомлення буде видалено для обох користувачів. Таким чином можна видалити як відправлені, так і вхідні повідомлення. Тому користувач з яких ведеться діалог також має можливість видалити будь-яке повідомлення.

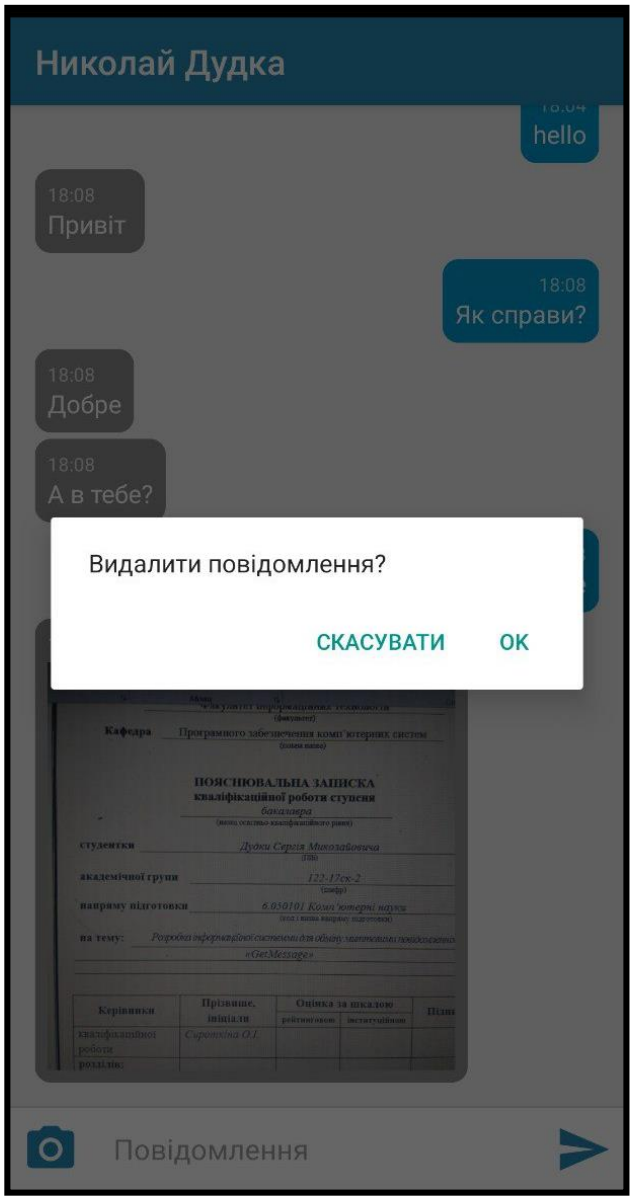

Рис. 3.13. Вікно підтвердження видалення повідомлення

При натисканні в головному меню на кнопку «Запросити друга» в головному меню зявиться додаткове поле для вводу електронної адреси користувача (рис. 3.14), якому необхідно відправити запрошення дружби, та додаткова кнопка для відправки запрошення. Якщо користувач з вказаною електронною адресою ще не зареєстрований в системі, зявиться діалогове вікно з пропозицією відправки запрошення на реєстрацію (рис. 3.15) на його електронну адресу. В разі згоди, користувача буде переправлено в додаток електронної пошти, що встановлений за замовчуванням.

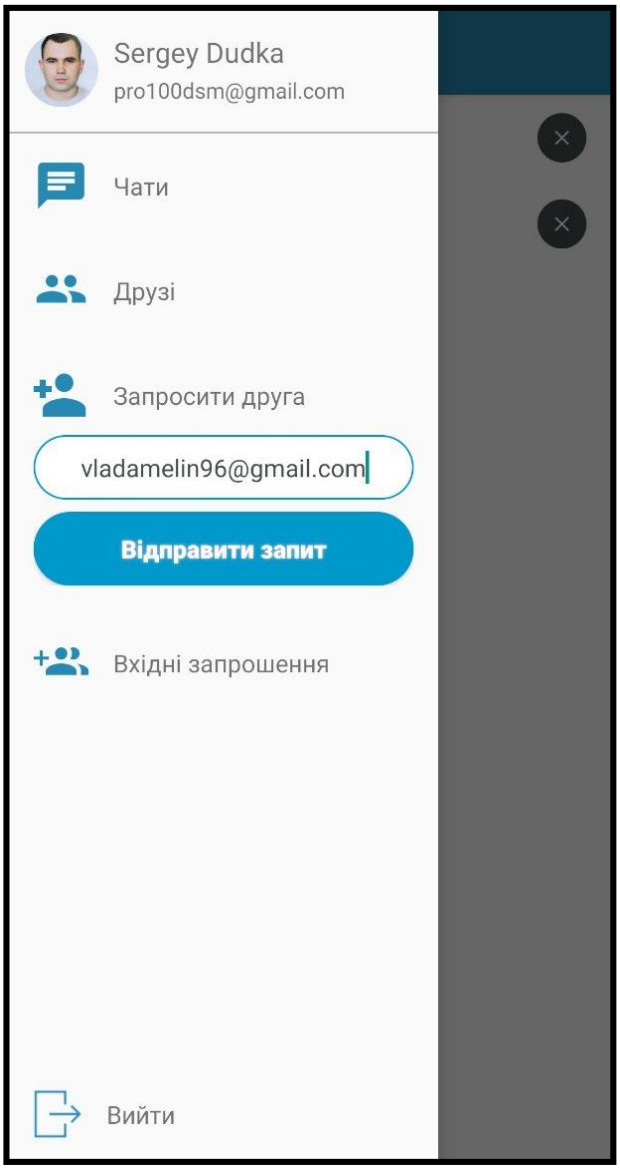

Рис. 3.14. Вікно відправки запрошення дружби

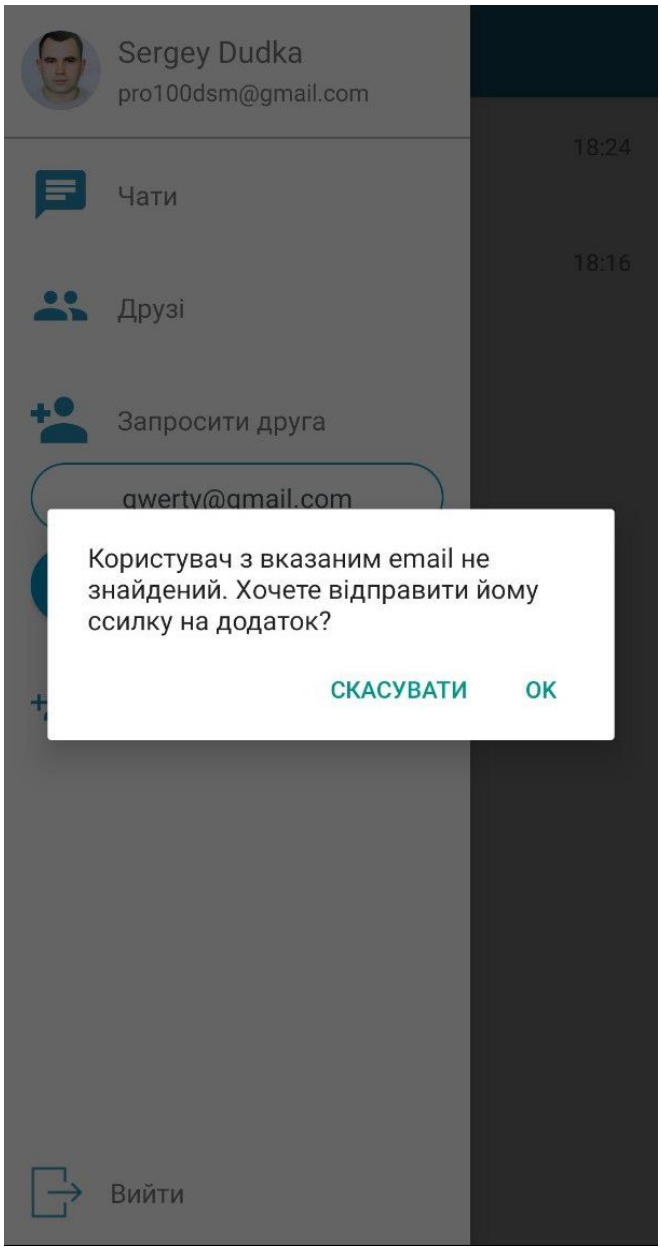

Рис. 3.15. Вікно відправки запрошення на реєстрацію

При натисканні кнопки «Вхідні запрошення» буде відображено вхідні запрошення на дружбу від інших користувачів (рис. 3.16). Запрошення можна підтвердити або відхилити натиснувши відповідні кнопки біля конкретного користувача, що відправив запрошення. Після підтвердження запрошення користувач, що його відправив зявиться в розділі «Друзі». Якщо вхідних запрошень немає, то на екрані зявиться відповдіне повідомлення.

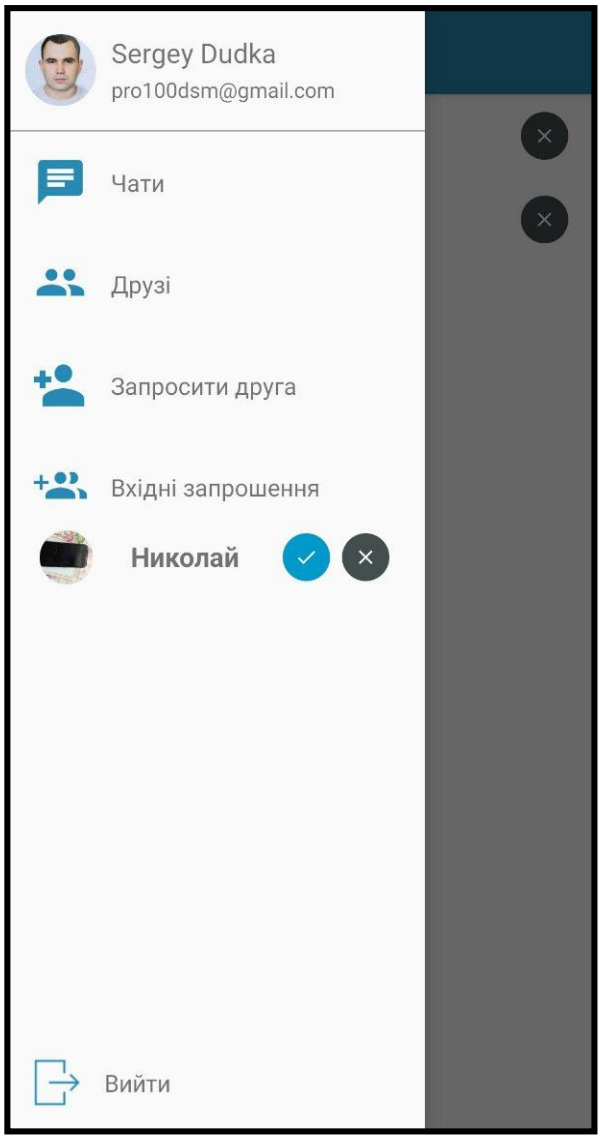

Рис. 3.16. Відображення вхідних запитів на дружбу

При натисканні кнопки «Друзі» користувачеві відкриється вікно в якому будуть відображені які були прийняті до друзів, або які прийняли запрошення до дружби (рис. 3.17). Біля фотографії профілю кожного друга буде відображено також його статус в мережі, а напроти друга буде кнопка видалення користвача з друзів. При натисканні на конкретного користувача відкриється вікно перегляду профілю користувача (рис. 3.18). З цього вікна можна потрапити до вікна діалогу з цим користувачем, натиснувши кнопку «Відправити повідомлення».

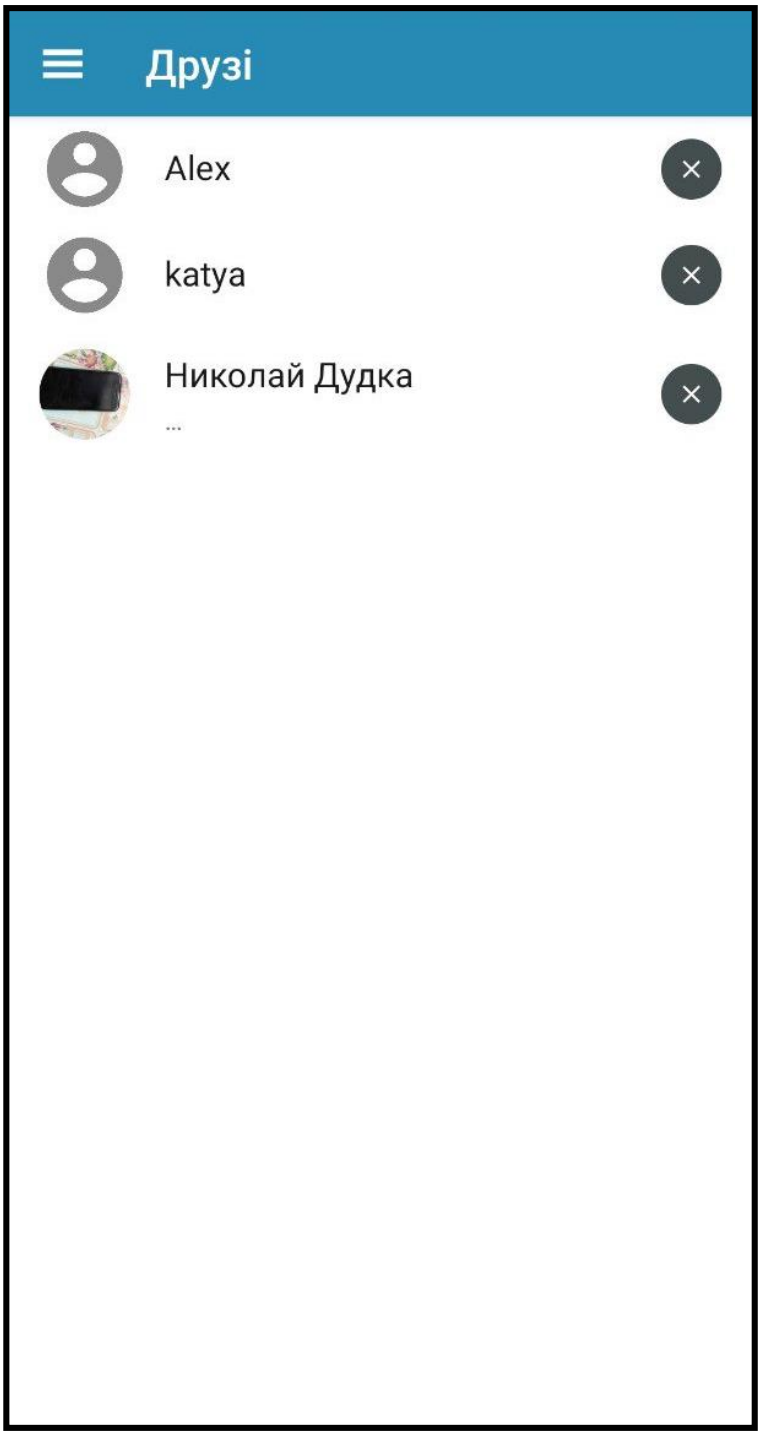

Рис. 3.17. Вікно відображення друзів

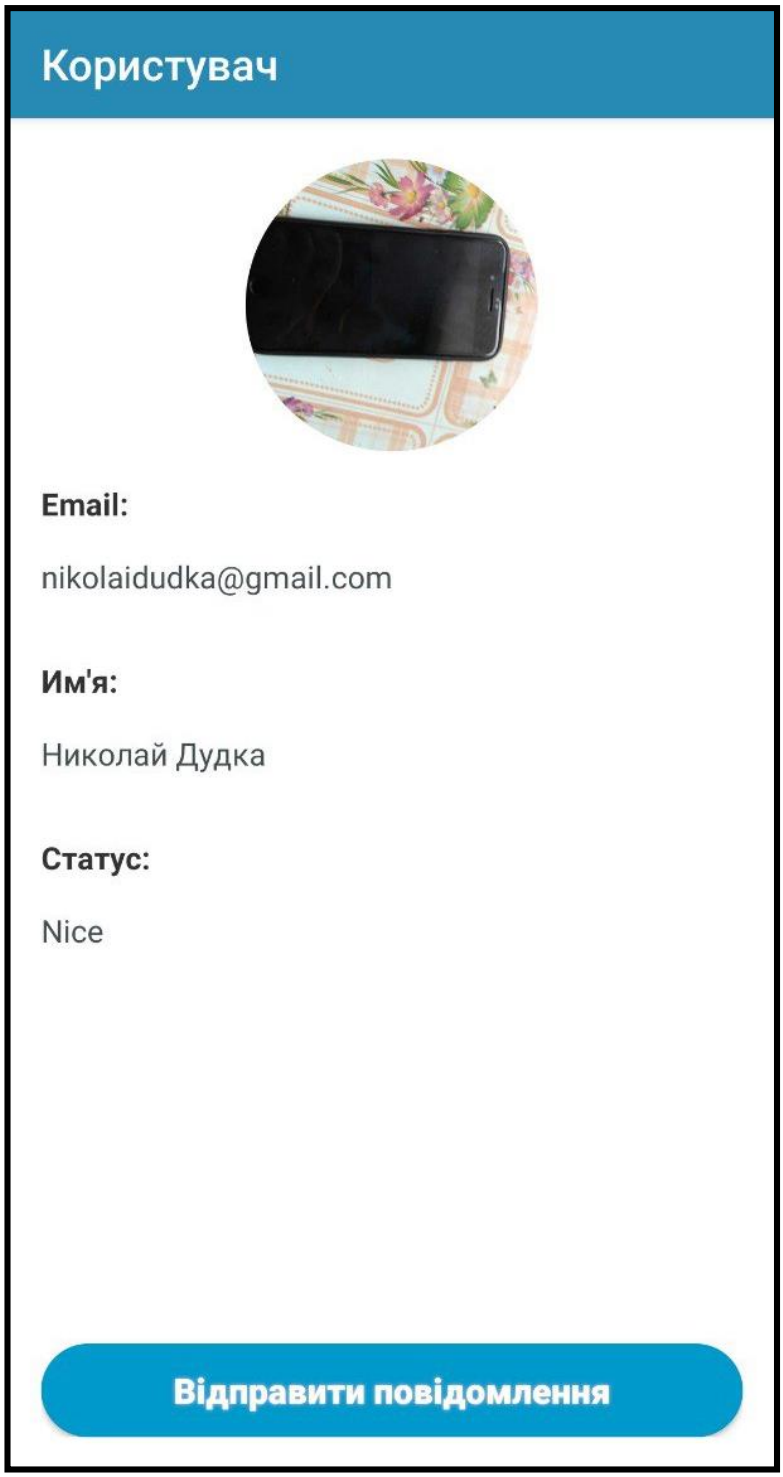

Рис. 3.18. Вікно перегляду профілю користувача

Також при натисканні кнопки «Вийти», користувач може вийти зі свого облікового запису, після чого знову буде відкрите стартове вікно авторизації.

ВИСНОВОК

Метою даної кваліфікаційної роботи було створення інформаційної технології у вигляді сервісу, який дозволить користувачам безпечно обмінюватись миттєвими повідомленнями в режимі реального часу при підключенні до мережі Інтернет.

Для того щоб розроблюваний отримав всі необхідні функціональні характеристики та відповідав всім вимогам, які висувались під час її проєктування, було використано стек сучасних технологій та паттернів програмування.

Кінцевий користувач безпосередньо взаємодіє лише з клієнтською частиною сервісу – Android-застосунком, тому саме ця частина є найважливішою і для її розробки було обрано сучасну мову програмування – Kotlin та використаний стек технологій, що рекомендований компанією «Google» саме для розробки під Android. Архітектура за якою розроблювався Android-застосунок називається чистою архітектурою, тому що вона легко масштабується та тестується. Це дозволить в майбутньому змінювати або додавати новий функціонал при мінімалних витратах.

Серверна частина була розгорнута на віддаленому компьтері наданого хостинг-провайдером. Тому сервер завжди онлайн, це забезпечує безперебійний доступ користувачів до актуальних даних.

Не менш важливим є таємниця листування. Тому повідомлення, які будуть відправляти одне одному користувачі, не повинні потрапити в руки третім особам. Для реалізації цієї задачі було використано шифрований протокол передачі даних. Завдяки цьому протоколу дані шифрувалися на стороні клієнту та могли бути розшифровані тільки на стороні серверу та навпаки. Це забезпечило необхідний рівень безпеки персональних даних користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Compatible Time-Sharing System. (Електрон. ресурс). Спосіб доступу: URL: https://uk.wikipedia.org/wiki/Compatible_Time-Sharing_System

2. Протокол Діффі-Геллмана. (Електрон. ресурс). Спосіб доступу: URL: https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%94%D1%96%D1%84%D1%84%D1%96_%E2%80%94_%D0%93%D0%B5%D0%BB%D0%BB%D0%BC%D0%B0%D0%BD%D0%B0#%D0%91%D1%96%D0%BB%D1%8C%D1%88%D0%B5_%D0%B4%D0%B2%D0%BE%D1%85_%D1%83%D1%87%D0%B0%D1%81%D0%BD%D0%B8%D0%BA%D1%96%D0%B2

3. Antonio Leiva, Kotlin for Android Developers: Learn Kotlin the easy way while developing an Android App / Antonio Leiva. South Carolina: CreateSpace Independent Publishing Platform, 2016. 150 p.

4. David Greenhalgh, Josh Skeen. Kotlin Programming: The Big Nerd Ranch Guide / Skeen J. Greenhalgh D. Atlanta: Big Nerd Ranch Guides, 2018. 384 p.

5. Venkat Subramaniam. Programming Kotlin: Create Elegant, Expressive, and Performant JVM and Android Applications. / Venkat Subramaniam / Pragmatic Bookshelf, 2019. 462 p.

6. Neil Smyth. Kotlin. Android Studio 3.0 Development Essentials – Android/ Smyth N. South Carolina: CreateSpace Independent Publishing Platform. 2017. 740 p.

7. Ted Hagos. Android Studio IDE Quick Reference: A Pocket Guide to Android Studio Development Paperback / Hagos T. New York: Apress, 2019. 200 p.

8. Barry Burd. Android Application Development All-in-One For Dummies / Burd B. New York: For Dummies. 2015. 268 p.

9. Josh Lockhart. Modern Php: New Features and Good Practices / Lockhart J. Newton: O'Reilly Media. 2015. 270 p.

10. Ben Forta. SQL in 10 Minutes, Sams Teach Yourself: Sams Teac Your SQL 10 Minu _4 4th Edition, Kindle Edition / Forta B. Karmel: Sams Publishing. 2012. 287 p.

11. Alan Beaulieu. Learning SQL / Beaulieu A. Newton: O'Reilly Media. 2009. 212 p.

12. Russell J.T. Dyer. Mysql in a Nutshell / Russell J.T. Dyer. Newton: O'Reilly Media. 2008. 564 p.

13. Mahmoud Ramadan. Master Dependency Injection for Android Using Dagger: learn Dagger 2 with Kotlin Step by Step Kindle Edition / Ramadan M. South Carolina: CreateSpace Independent Publishing Platform. 2019. 33 p.

14. Miguel Angel Castiblanco Torres. Learning Concurrency in Kotlin: Build highly efficient and robust applications / Miguel Angel Castiblanco Torres. Birmingham: Packt Publishing. 2018. 266 p.

15. Alexey Soshin. Hands-On Design Patterns with Kotlin: Build scalable applications using traditional, reactive, and concurrent design patterns in Kotlin/ Soshin A. Birmingham: Packt Publishing. 2018. 310 p.

16. Дмитрий Жемеров, Светлана Исакова. Kotlin в действии./ Дмитрий Жемеров, Светлана Исакова. Москва: ДМК Пресс, **2019**. 402 с.

17. Скин Джош, Гринхол Дэвид. Kotlin. Программирование для профессионалов./ Скин Джош, Гринхол Дэвид. Санкт-Петербург: Издательский дом «Питер», **2019**. 464 с.

18. Роберт Мартин. Чистая архитектура. Искусство разработки программного обеспечения / Мартин Р. Санкт-Петербург: Издательский дом «Питер», 2016. **352** с.

19. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования./ Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Санкт-Петербург: Питер, 2019. **368** с.

20. Маклафлин Б. PHP и MySQL. Исчерпывающее руководство./ Маклафлин Б.Санкт-Петербург: Питер, 2014. **512** с.

21. Б. Филлипс, К. Стюарт, К. Марсикано. Android. Программирование для профессионалов./ Филлипс, К. Стюарт, К. Марсикано. Санкт-Петербург: Питер, 2017. **688** с.

22. Мэтт Зандстра. PHP: объекты, шаблоны и методики программирования./ Зандстра М. Москва: Диалектика-Вильямс, 2019. **576** с.

23. Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. SQL: полное руководство./ Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Москва: Диалектика, 2020. **960** с.

24. Натан Марц, Джеймс Уоррен. Большие данные. Принципы и практика построения масштабируемых систем обработки данных в реальном времени./ Марц Н., Уоррен Д. Москва: Диалектика Вильямс, 2019. **368** с.

25. Android Studio. (Електрон. ресурс). Спосіб доступу: URL: https://ru.wikipedia.org/wiki/Android_Studio.

26. PhpStorm. (Електрон. ресурс). Спосіб доступу: URL: https://ru.wikipedia.org/wiki/PhpStorm.

27. Apache HTTP Server. (Електрон. ресурс). Спосіб доступу: URL: https://uk.wikipedia.org/wiki/Apache_HTTP_Server.

28. phpMyAdmin. (Електрон. ресурс). Спосіб доступу: URL: https://ru.wikipedia.org/wiki/PhpMyAdmin.

29. Заблуждения Clean Architecture. (Електрон. ресурс). Спосіб доступу: URL: https://habr.com/ru/company/mobileup/blog/335382.

30. Миттєві повідомлення. (Електрон. ресурс). Спосіб доступу: URL: https://uk.wikipedia.org/wiki/%D0%9C%D0%B8%D1%82%D1%82%D1%94%D0%B2%D1%96_%D0%BF%D0%BE%D0%B2%D1%96%D0%B4%D0%BE%D0%BC%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F.

31. Еволюція месенджерів. (Електрон. ресурс). Спосіб доступу: URL: https://nachasi.com/tech/2017/11/13/evolyutsiya-mesendzheriv/.

32. HTTPS. (Електрон. ресурс). Спосіб доступу: URL: https://uk.wikipedia.org/wiki/HTTPS.

33. Протокол Діффі — Геллмана. (Електрон. ресурс). Спосіб доступу: URL:https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%94%D1%96%D1%84%D1%84%D1%96_%E2%80%94_%D0%93%D0%B5%D0%BB%D0%BB%D0%BC%D0%B0%D0%BD%D0%B0.

ДОДАТОК А

Відомість матеріалів кваліфікаційної роботи

		Позначення	Найменування	Кількість	Примітка
	1				
	2		Документація	85	
	3				
	4		Пояснювальна записка	1	
	5				
	6		Диск CD-R з презентацією	1	

Зм	Лист	№ докум.	Підпис	Дата					
Розроб.		С.М. Дудка			**Матеріали кваліфікаційної роботи**	Літ.			Аркуш 1	Аркушів 1
Керівник		Г.М.Коротенко								
Рецензент		О.М.Галушко				НТУ «ДП» 8; 126м-20-1				
Н.контр.		Г.М.Коротенко								
Зав. каф.		В.В.Гнатушенко								

ДОДАТОК Б

Лістинг серверної частини

addFriend.php

```
<?php
// array for JSON response
$response = array();
include __DIR__ . "/GeneralFunctions.php";
include __DIR__ . "/utilities/firebaseNotification.php";
if (isset($_POST['request_user_id']) && isset($_POST['email']) && isset($_POST['token'])) {
    $request_user_id = $_POST['request_user_id'];
    $email = $_POST['email'];
    $token = $_POST['token'];
    $dbOperationsObject = new DBOperations();
    $generalFunctionsObject = new GeneralFunctionsClass();
    $firebaseNotificationObject = new FirebaseNotificationClass();
    $resultApprovedUser = $dbOperationsObject->getUserByEmail($email);

    $result_token = $dbOperationsObject->isTokenExist($request_user_id, $token);
    if (mysqli_num_rows($result_token) > 0) {

        if (mysqli_num_rows($resultApprovedUser) > 0) {
            $approvedUser = $generalFunctionsObject->getUserInfoSafety($resultApprovedUser);
            $approved_user_id = $approvedUser['user_id'];
            $friendInfo = $generalFunctionsObject->getFriendRelation($approved_user_id, $request_user_id);

            if ($friendInfo["friends_id"] == -1) {
                $request_user = $generalFunctionsObject->getUserById($request_user_id);
                $approved_user_token = $generalFunctionsObject->getUserRegestrationId($approved_user_id);
                $result = $dbOperationsObject->addFriend($request_user_id, $approved_user_id);
                if (mysqli_affected_rows($result) > 0) {
                    $friends_id = mysqli_insert_id($result);
                    $obj = new stdClass();
                    $obj->firebase_json_message = array(
                        "type" => 'addFriend',
                        "friends_id" => $friends_id,
                        "request_user" => $request_user
                    );

                    $messageFirbase = json_encode($obj);
                    $firebaseNotificationObject->send_notification($approved_user_token, $messageFirbase);
                    $response['success'] = 1;
                    $response['friends_id'] = $friends_id;
                    $response['approved_user'] = $approvedUser;
                    echo json_encode($response);
                } else {
                    $response['success'] = 0;
                    echo json_encode($response);
                }
            } else {
                $response['success'] = 0;
                if ($friendInfo["type"] == 0) {
                    if ($friendInfo["request_user_id"] == $request_user_id) {
                        $response["message"] = "you requested adding this friend before";
                    } else {
                        $response["message"] = "already found in your friend requests";
                    }
                } else
                    if ($friendInfo["type"] == 1) {
                        $response["message"] = "this contact is already in your friends list";
                    }
                echo json_encode($response);
            }
        } else {
```

```
            $response["success"] = 0;
            $response["message"] = "No Contact has this email";
            echo json_encode($response);
        }
    } else {
        $response["success"] = 0;
        $response["message"] = "Invalid token";
        echo json_encode($response);
    }
} else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";
    echo json_encode($response);
}
?>
```

approveFriendRequest.php

```
<?php
$response = array();
include __DIR__ . "/GeneralFunctions.php";
include __DIR__ . "/utilities/firebaseNotification.php";
if (isset($_POST['friends_id']) && isset($_POST['request_user_id']) && isset($_POST['user_id']) && isset($_POST['token'])) {
    $user_id = $_POST['user_id'];
    $token = $_POST['token'];
    $friends_id = $_POST['friends_id'];
    $request_user_id = $_POST['request_user_id'];

    $firebaseNotificationObject = new FirebaseNotificationClass();
    $generalFunctionsObject = new GeneralFunctionsClass();
    $dbOperationsObject = new DBOperations();
    $request_user = $generalFunctionsObject->getUserById($request_user_id);
    $approvedUser = $generalFunctionsObject->getUserById($user_id);
    $friendInfo = $generalFunctionsObject->getFriendRelation($request_user_id, $user_id);

    $result_token = $dbOperationsObject->isTokenExist($request_user_id, $token);
    if (mysqli_num_rows($result_token) > 0) {

        if ($friendInfo["friends_id"] == -1) {
            $response['success'] = 0;
            $response["message"] = "this contact cancelled this friend request ";
            echo json_encode($response);
        } else {
            if ($friendInfo["type"] == 1) {
                $response['success'] = 0;
                $response["message"] = "this contact is already in your friends list";
                echo json_encode($response);
            } else
                if ($friendInfo["type"] == 0) {
                    $request_user_token = $request_user["token"];
                    $result = $dbOperationsObject->approveFriendRequest($friends_id);
                    if (mysqli_affected_rows($result) > 0) {
                        $obj = new stdClass();
                        $obj->firebase_json_message = array(
                            "type" => 'approveFriendRequest',
                            "friends_id" => $friends_id,
                            "approved_user" => $approvedUser,
                        );
                        $messageFirbase = json_encode($obj);
                        $firebaseNotificationObject->send_notification($request_user_token, $messageFirbase);
                        $response["success"] = 1;
                        $response['friends_id'] = $friends_id;
                        $response['request_user'] = $request_user;
                        echo json_encode($response);
                    } else {
                        $response["success"] = 0;
                        $response["message"] = "Oops! An error occurred.";
```

```
                    echo json_encode($response);
                }
            }
        }
    } else {
        $response["success"] = 0;
        $response["message"] = "Invalid token";
        echo json_encode($response);
    }
} else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";
    echo json_encode($response);
}
?>
```

cancelFriendRequest.php

```
<?php

include __DIR__ . "/GeneralFunctions.php";
include __DIR__ . "/utilities/firebaseNotification.php";
$response = array();

if (isset($_POST['friends_id']) && isset($_POST['approved_user_id']) && isset($_POST['user_id']) && isset($_POST['token']))
{
    $user_id = $_POST['user_id'];
    $token = $_POST['token'];
    $friends_id = $_POST['friends_id'];
    $approved_user_id = $_POST['approved_user_id'];
    $firebaseNotificationObject = new FirebaseNotificationClass();
    $generalFunctionsObject = new GeneralFunctionsClass();
    $dbOperationsObject = new DBOperations();

    $result_token = $dbOperationsObject->isTokenExist($approved_user_id, $token);
    if (mysqli_num_rows($result_token) > 0) {

        $request_user = $generalFunctionsObject->getUserById($user_id);
        $approved_user = $generalFunctionsObject->getUserById($approved_user_id);
        $approved_user_token = $approved_user["token"];
        $connection = $dbOperationsObject->deleteFriend($friends_id);

        if (mysqli_affected_rows($connection) > 0) {
            $obj = new stdClass();
            $obj->firebase_json_message = array(
                "type" => 'cancelFriendRequest',
                "friends_id" => $friends_id,
                "request_user" => $request_user
            );

            $messageFirbase = json_encode($obj);
            $firebaseNotificationObject->send_notification($approved_user_token, $messageFirbase);
            $response["success"] = 1;
            $response['friends_id'] = $friends_id;
            echo json_encode($response);
        } else {
            $response["success"] = 0;
            echo json_encode($response);
        }

    } else {
```

```
        $response["success"] = 0;
        $response["message"] = "Invalid token";
        echo json_encode($response);
    }
}
?>
```

deleteFriend.php

```
<?php
include __DIR__ . "/GeneralFunctions.php";
include __DIR__ . "/utilities/firebaseNotification.php";
$response = array();

if (isset($_POST['friends_id'])
    && isset($_POST['user_id'])
    && isset($_POST['token'])
    && isset($_POST['approved_user_id'])
) {
    $friends_id = $_POST['friends_id'];
    $user_id = $_POST['user_id'];
    $approved_user_id = $_POST['approved_user_id'];
    $token = $_POST['token'];
    $firebaseNotificationObject = new FirebaseNotificationClass();
    $generalFunctionsObject = new GeneralFunctionsClass();
    $dbOperationsObject = new DBOperations();

    $result_token = $dbOperationsObject->isTokenExist($approved_user_id, $token);
    if (mysqli_num_rows($result_token) > 0) {

        $firebase_friend = $generalFunctionsObject->getUsersByFriendsId($friends_id);

        if ($user_id != $firebase_friend["approved_user_id"]) {
            $firebase_user = $generalFunctionsObject->getUserById($firebase_friend["approved_user_id"]);
            $firebase_user_token = $firebase_user["token"];
        } else {
            $firebase_user = $generalFunctionsObject->getUserById($firebase_friend["request_user_id"]);
            $firebase_user_token = $firebase_user["token"];

        }

        $delete_req_user = $generalFunctionsObject->getUserById($user_id);
        $connection = $dbOperationsObject->deleteFriend($friends_id);
        if (mysqli_affected_rows($connection) > 0) {
            $obj = new stdClass();
            $obj->firebase_json_message = array(
                "type" => 'deleteFriend',
                "friends_id" => $friends_id,
                "delete_req_user" => $delete_req_user
            );

            $messageFirbase = json_encode($obj);
            $firebaseNotificationObject->send_notification($firebase_user_token, $messageFirbase);
            $response["success"] = 1;
            $response['friends_id'] = $friends_id;
            echo json_encode($response);
        } else {
            $response["success"] = 0;
            echo json_encode($response);
```

```
        }

    } else {
        $response["success"] = 0;
        $response["message"] = "Invalid token";
        echo json_encode($response);
    }
}

?>
```

deleteMessagesByUser.php

```
<?php
//including the database connection file
include __DIR__ . "/GeneralFunctions.php";
// array for JSON response
$response = array();
if (isset($_POST['user_id']) && isset($_POST['messages_ids']) && isset($_POST['token'])) {
    $dbOperationsObject = new DBOperations();
    $generalFunctionsObject = new GeneralFunctionsClass();
    $user_id = $_POST['user_id'];
    $token = $_POST['token'];
    $messages_ids = $_POST['messages_ids'];

    $result_token = $dbOperationsObject->isTokenExist($user_id, $token);
    if (mysqli_num_rows($result_token) > 0) {

        $messages_ids = json_decode($messages_ids, true);
        $messagesIdsCount = count($messages_ids['messages']);
        $number = 0;
        foreach ($messages_ids['messages'] as $messages_ids_object) {
            $message_id = $messages_ids_object['message_id'];
            $numberUpdated = $dbOperationsObject->deleteMessageByUser($user_id, $message_id);
            if ($numberUpdated > 0) {
                $number++;
            } else {
                break;
            }
        }

        if ($number == $messagesIdsCount) {
            $response["success"] = 1;
            echo json_encode($response);
        } else {
            $response["success"] = 0;
            $response["message"] = "error in deleting";
            echo json_encode($response);
        }

    } else {
        $response["success"] = 0;
        $response["message"] = "Invalid token";
        echo json_encode($response);
    }
}
?>
```

editUser.php

```
<?php

$response = array();
include __DIR__ . "/GeneralFunctions.php";
include __DIR__ . "/utilities/encrypt_decrypt.php";
$uploadsFolder = "uploadsProfiles/";
$file_upload_url = '../' . $uploadsFolder;
if (isset($_POST['user_id']) && isset($_POST['name']) && isset($_POST['email'])
    && isset($_POST['password'])
    && isset($_POST['status'])
    && isset($_POST['token'])
    && isset($_POST['image_uploaded'])
    || isset($_POST['image_new'])
) {
    $generalFunctionsObject = new GeneralFunctionsClass();
    $dbOperationsObject = new DBOperations();
    $encryptDecryptSecurityObject = new EncryptDecryptSecurity();
    $user_id = $_POST['user_id'];
    $name = $_POST['name'];
    $email = $_POST['email'];
    $password = $encryptDecryptSecurityObject->encrypt($_POST['password']);
    $status = $_POST['status'];
    $token = $_POST['token'];
    $image_uploaded = $_POST['image_uploaded'];
    $edit = 0;

    $result_token = $dbOperationsObject->isTokenExist($user_id, $token);
    if (mysqli_num_rows($result_token) > 0) {

        $resultEmailFound = $dbOperationsObject->isEmailExist($email);
        if (mysqli_num_rows($resultEmailFound) > 0) {
            $user = $generalFunctionsObject->getUserInfo($resultEmailFound);
            if ($user["user_id"] != $user_id) {
                $edit = 0;
            } else {
                $edit = 1;
            }
        } else {
            $edit = 1;
        }

        if ($edit == 1) {
            $errorImage = 0;
            if (isset($_POST['image_new'])) {
                try {
                    $image_new = $_POST['image_new'];
                    $image_new_name = $_POST['image_new_name'];
                    $binary = base64_decode($image_new);
                    $file = fopen('../uploadsProfiles/' . $image_new_name, 'w');
                    fwrite($file, $binary);
                    fclose($file);
                    $image_new = $file_upload_url . basename($image_new_name);
                } catch (Exception $e) {
                    $errorImage = 1;
                }
            }

            if ($errorImage == 0) {
                if (isset($_POST['image_new'])) {
```

```
                $result = $dbOperationsObject->editUser($user_id, $name, $email, $password, $status, $image_new);
            } else {
                $result = $dbOperationsObject->editUser($user_id, $name, $email, $password, $status, $image_uploaded);
            }

            if (mysqli_affected_rows($result) >= 0) {
                $resultUser = $dbOperationsObject->getUser($user_id);
                if (mysqli_num_rows($resultUser) > 0) {
                    $user = $generalFunctionsObject->getUserInfo($resultUser);
                    $response["success"] = 1;
                    $response["user"] = $user;
                    echo json_encode($response);
                } else {
                    $response['success'] = 0;
                    $response["message"] = "Oops! An error occurred.";
                    echo json_encode($response);
                }
            } else {
                $response['success'] = 0;
                $response["message"] = "Oops! An error occurred.";
                echo json_encode($response);
            }
        } else {
            $response["success"] = 0;
            $response["message"] = "Oops! An error occurred.";
            echo json_encode($response);
        }
    } else {
        $response["success"] = 0;
        $response["message"] = "there is a user has this email";
        echo json_encode($response);
    }

} else {
    $response["success"] = 0;
    $response["message"] = "Invalid token";
    echo json_encode($response);
}

} else {
    $response["success"] = 0;
    $response["message"] = "Missing required fields";
    echo json_encode($response);
}
?>
```

forget_password.php

```
<?php
$response = array();
include  __DIR__."/GeneralFunctions.php";
include  __DIR__."/utilities/sendPHPMail.php";
include  __DIR__."/utilities/encrypt_decrypt.php";

if (isset($_POST['email'])) {
 $email = $_POST['email'];
 $encryptDecryptSecurityObject = new EncryptDecryptSecurity();
 $generalFunctionsObject = new GeneralFunctionsClass();
 $dbOperationsObject = new DBOperations();
```

```
$resultUser = $dbOperationsObject->getUserByEmail($email);
if (mysqli_num_rows($resultUser)>0)
{
  $user = $generalFunctionsObject->getUserInfoWithPassword($resultUser);
  $user_id = $user["user_id"];
  $userPassword = $user["password"];
  $userPassword=$encryptDecryptSecurityObject->decrypt($userPassword);
  $userPasswordTemp = substr(md5(uniqid(mt_rand(), true)) , 0, 8);
  $userPasswordTempEncrypted=$encryptDecryptSecurityObject->encrypt($userPasswordTemp);
  $resultUpdatePassword= $dbOperationsObject->updateUserPassword($user_id,$userPasswordTempEncrypted);

  if (mysqli_affected_rows($resultUpdatePassword)>=0) {
    $phpMailSendObject = new PHPMailSend();
    $message = $userPasswordTemp ;
    $phpMailSendResult = $phpMailSendObject->send_email($email,$message);
    if($phpMailSendResult==1)
    {
     $response["success"] = 1;
     echo json_encode($response);
   }
   else
   {
    $response["success"] = 0;
    $response["message"] = "can't send email to you";
    echo json_encode($response);
  }
 }
 else {
  $response["success"] = 0;
  $response["message"] = "Oops! An error occurred.";
  echo json_encode($response);
}
}
else
{
 $response["success"] = 0;
 $response["message"] = " this email is not registered before";
 echo json_encode($response);
}
}
else
{
 $response["success"] = 0;
 $response["message"] = "Required field(s) is missing";
 echo json_encode($response);
}
?>
```

GeneralFunctions.php

```
<?php

$DBOperations_path = __DIR__ . "/../sql/DBOperations.php";
include($DBOperations_path);

class GeneralFunctionsClass
{
   public function __constructor()
   {
```

```
}

public function getUserInfoWithPassword($resultUser)
{
   while ($rowUser = mysqli_fetch_array($resultUser)) {
      $user = $this->getUserAtRow($rowUser);
      $user["password"] = $rowUser["password"];
   }
   return $user;
}

public function getUserInfo($resultUser)
{
   while ($rowUser = mysqli_fetch_array($resultUser)) {
      $user = $this->getUserAtRow($rowUser);
   }
   return $user;
}

public function getUserInfoSafety($resultUser)
{
   while ($rowUser = mysqli_fetch_array($resultUser)) {
      $user = $this->getUserAtRowSafety($rowUser);
   }
   return $user;
}

public function getUserAtRow($rowUser)
{
   $user["user_id"] = $rowUser["user_id"];
   $user["name"] = $rowUser["name"];
   $user["email"] = $rowUser["email"];
   $user["token"] = $rowUser["token"];
   $user["status"] = $rowUser["status"];
   $user["user_date"] = $rowUser["user_date"];
   $user["image"] = $rowUser["image"];
   $user["last_seen"] = $rowUser["last_seen"];
   return $user;
}

public function getUserAtRowSafety($rowUser)
{
   $user["user_id"] = $rowUser["user_id"];
   $user["name"] = $rowUser["name"];
   $user["email"] = $rowUser["email"];
   $user["status"] = $rowUser["status"];
   $user["user_date"] = $rowUser["user_date"];
   $user["image"] = $rowUser["image"];
   $user["last_seen"] = $rowUser["last_seen"];
   return $user;
}

public function getFriendsAtRow($rowFriends)
{
   $friends["friends_id"] = $rowFriends["friends_id"];
   $friends["request_user_id"] = $rowFriends["request_user_id"];
   $friends["approved_user_id"] = $rowFriends["approved_user_id"];
   $friends["type"] = $rowFriends["type"];
   return $friends;
```

```
}

public function getMessagesByUser($resultMessages)
{
   $dbOperationsObject = new DBOperations();
   $messages = array();
   while ($rowMessage = mysqli_fetch_array($resultMessages)) {
      $message = array("message_id" => $rowMessage['message_id'],
         "sender_id" => $rowMessage['sender_id'],
         "receiver_id" => $rowMessage['receiver_id'],
         "message" => $rowMessage['message'],
         "message_type_id" => $rowMessage['message_type_id'],
         "message_date" => $rowMessage['message_date'],
         "deleted_by_sender_id" => $rowMessage['deleted_by_sender_id'],
         "deleted_by_receiver_id" => $rowMessage['deleted_by_receiver_id'],
         "name" => $rowMessage['name'],
         "email" => $rowMessage['email'],
         "token" => $rowMessage['token'],
         "status" => $rowMessage['status'],
         "image" => $rowMessage['image'],
         "user_date" => $rowMessage['user_date'],
         "last_seen" => $rowMessage['last_seen']
      );
      $resultMessageType = $dbOperationsObject->getMessageType($rowMessage["message_type_id"]);
      $messageType = $this->getMessageType($resultMessageType);
      $message["message_type"] = $messageType;
      array_push($messages, $message);
   }
   return $messages;
}

public function getLastMessagesByUser($resultMessages, $user_id)
{
   $dbOperationsObject = new DBOperations();
   $messages = array();
   while ($rowMessage = mysqli_fetch_array($resultMessages)) {
      $message = array("message_id" => $rowMessage['message_id'],
         "sender_id" => $rowMessage['sender_id'],
         "receiver_id" => $rowMessage['receiver_id'],
         "message" => $rowMessage['message'],
         "message_type_id" => $rowMessage['message_type_id'],
         "message_date" => $rowMessage['message_date'],
         "deleted_by_sender_id" => $rowMessage['deleted_by_sender_id'],
         "deleted_by_receiver_id" => $rowMessage['deleted_by_receiver_id']
      );

      $resultMessageType = $dbOperationsObject->getMessageType($rowMessage["message_type_id"]);
      $messageType = $this->getMessageType($resultMessageType);
      $message["message_type"] = $messageType;
      if ($user_id != $rowMessage['sender_id']) {
         $senderUser = $this->getUserById($rowMessage['sender_id']);
         $friendInfo = $this->getFriendRelation($senderUser["user_id"], $user_id);
         $senderUser["friends_id"] = $friendInfo["friends_id"];
         $senderUser["request_user_id"] = $friendInfo["request_user_id"];
         $senderUser["approved_user_id"] = $friendInfo["approved_user_id"];
         $senderUser["type"] = $friendInfo["type"];
         $message["contact"] = $senderUser;
      } else {
         $receivedUser = $this->getUserById($rowMessage['receiver_id']);
         $friendInfo = $this->getFriendRelation($receivedUser["user_id"], $user_id);
```

```
            $receivedUser["friends_id"] = $friendInfo["friends_id"];
            $receivedUser["request_user_id"] = $friendInfo["request_user_id"];
            $receivedUser["approved_user_id"] = $friendInfo["approved_user_id"];
            $receivedUser["type"] = $friendInfo["type"];
            $message["contact"] = $receivedUser;
        }
        array_push($messages, $message);
    }
    return $messages;
}

public function getMessagesByUserWithContact($resultMessages)
{
    $dbOperationsObject = new DBOperations();
    $messages = array();
    while ($rowMessage = mysqli_fetch_array($resultMessages)) {
        $message = array("message_id" => $rowMessage['message_id'],
            "sender_id" => $rowMessage['sender_id'],
            "receiver_id" => $rowMessage['receiver_id'],
            "message" => $rowMessage['message'],
            "message_type_id" => $rowMessage['message_type_id'],
            "message_date" => $rowMessage['message_date'],
            "deleted_by_sender_id" => $rowMessage['deleted_by_sender_id'],
            "deleted_by_receiver_id" => $rowMessage['deleted_by_receiver_id']
        );

        $resultMessageType = $dbOperationsObject->getMessageType($rowMessage["message_type_id"]);
        $messageType = $this->getMessageType($resultMessageType);
        $message["message_type"] = $messageType;
        array_push($messages, $message);
    }
    return $messages;
}

public function getMessageType($resultMessageType)
{
    while ($rowMessageType = mysqli_fetch_array($resultMessageType)) {
        $messageType["type_id"] = $rowMessageType["type_id"];
        $messageType["type_name"] = $rowMessageType["type_name"];
    }
    return $messageType;
}

public function getContactsByUser($resultFriends)
{
    $dbOperationsObject = new DBOperations();
    $friends = array();
    while ($rowFriend = mysqli_fetch_array($resultFriends)) {
        $friend = array("user_id" => $rowFriend['user_id'],
            "name" => $rowFriend['name'],
            "email" => $rowFriend['email'],
            "token" => $rowFriend['token'],
            "status" => $rowFriend['status'],
            "image" => $rowFriend['image'],
            "user_date" => $rowFriend['user_date'],
            "last_seen" => $rowFriend['last_seen'],
            "type" => $rowFriend['type'],
            "friends_id" => $rowFriend['friends_id'],
            "request_user_id" => $rowFriend['request_user_id'],
            "approved_user_id" => $rowFriend['approved_user_id']
```

```
        );
        array_push($friends, $friend);
    }
    return $friends;
}

public function getFriendRequestsByUser($resultFriends)
{
    $dbOperationsObject = new DBOperations();
    $friends = array();
    while ($rowFriend = mysqli_fetch_array($resultFriends)) {
        $friend = array("user_id" => $rowFriend['user_id'],
            "name" => $rowFriend['name'],
            "email" => $rowFriend['email'],
            "token" => $rowFriend['token'],
            "status" => $rowFriend['status'],
            "image" => $rowFriend['image'],
            "user_date" => $rowFriend['user_date'],
            "last_seen" => $rowFriend['last_seen'],
            "type" => $rowFriend['type'],
            "friends_id" => $rowFriend['friends_id'],
            "request_user_id" => $rowFriend['request_user_id'],
            "approved_user_id" => $rowFriend['approved_user_id']
        );
        array_push($friends, $friend);
    }
    return $friends;
}

public function getUsersByFriendsId($friends_id)
{
    $dbOperationsObject = new DBOperations();
    $result = $dbOperationsObject->getUsersByFriendsId($friends_id);
    while ($rowFriends = mysqli_fetch_array($result)) {
        $friends = $this->getFriendsAtRow($rowFriends);
    }

    return $friends;

}

public function getAllUsers($resultUsers)
{
    $users = array();
    $dbOperationsObject = new DBOperations();
    while ($rowUser = mysqli_fetch_array($resultUsers)) {
        $resultUser = $dbOperationsObject->getUser($rowUser["user_id"]);
        $user = $this->getUserInfo($resultUser);
        array_push($users, $user);
    }
    return $users;
}

public function getFriendInfo($resultFriend)
{
    $friend["friends_id"] = "-1";
    $friend["request_user_id"] = "0";
    $friend["approved_user_id"] = "0";
    $friend["type"] = "0";
```

```
        while ($rowFriend = mysqli_fetch_array($resultFriend)) {
            $friend = $this->getFriendAtRow($rowFriend);
            break;
        }
        return $friend;
    }

    public function getFriendAtRow($rowFriend)
    {
        $friend["friends_id"] = $rowFriend["friends_id"];
        $friend["request_user_id"] = $rowFriend["request_user_id"];
        $friend["approved_user_id"] = $rowFriend["approved_user_id"];
        $friend["type"] = $rowFriend["type"];
        return $friend;
    }

    public function getUserById($user_id)
    {
        $dbOperationsObject = new DBOperations();
        $result = $dbOperationsObject->getUser($user_id);
        $user = $this->getUserInfoSafety($result);
        return $user;
    }

    public function getUserRegestrationId($user_id)
    {
        $dbOperationsObject = new DBOperations();
        $result = $dbOperationsObject->getUser($user_id);
        $user = $this->getUserInfo($result);
        return $user["token"];
    }

    public function getFriendRelation($friend_user_id, $user_id)
    {
        $dbOperationsObject = new DBOperations();
        $resultFriend = $dbOperationsObject->getFriendRelation($friend_user_id, $user_id);
        $friend = $this->getFriendInfo($resultFriend);
        return $friend;
    }
}
```

getContactsByUser.php

```
<?php
$response = array();

include __DIR__ . "/GeneralFunctions.php";

if (isset($_POST['user_id']) && isset($_POST['token'])) {
    $user_id = $_POST['user_id'];
    $token = $_POST['token'];
    $dbOperationsObject = new DBOperations();
    $generalFunctionsObject = new GeneralFunctionsClass();

    $result_token = $dbOperationsObject->isTokenExist($user_id, $token);
    if (mysqli_num_rows($result_token) > 0) {
```

```
        $resultFriends = $dbOperationsObject->getContactsByUser($user_id);
        $friends = $generalFunctionsObject->getContactsByUser($resultFriends);
        $response["success"] = 1;
        $response["friends"] = $friends;
        echo json_encode($response);
    } else {
        $response["success"] = 0;
        $response["message"] = "Invalid token";
        echo json_encode($response);
    }

} else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";
    echo json_encode($response);
}
?>
```

getContactWithUser.php

```
<?php
// array for JSON response
$response = array();
include __DIR__ . "/GeneralFunctions.php";

if (isset($_POST['user_id'])
    && isset($_POST['contact_user_id'])
    && isset($_POST['token'])
) {
    $user_id = $_POST['user_id'];
    $token = $_POST['token'];
    $contact_user_id = $_POST['contact_user_id'];
    $generalFunctionsObject = new GeneralFunctionsClass();
    $dbOperationsObject = new DBOperations();

    $result_token = $dbOperationsObject->isTokenExist($user_id, $token);
    if (mysqli_num_rows($result_token) > 0) {

        $contactUser = $generalFunctionsObject->getUserById($contact_user_id);
        $friendInfo = $generalFunctionsObject->getFriendRelation($contact_user_id, $user_id);
        $contactUser["friends_id"] = $friendInfo["friends_id"];
        $contactUser["request_user_id"] = $friendInfo["request_user_id"];
        $contactUser["approved_user_id"] = $friendInfo["approved_user_id"];
        $contactUser["type"] = $friendInfo["type"];
        $response['success'] = 1;
        $response['contact'] = $contactUser;
        echo json_encode($response);

    } else {
        $response["success"] = 0;
        $response["message"] = "Invalid token";
        echo json_encode($response);
    }
} else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";
    echo json_encode($response);
```

```
}
?>
```

getFriendRequestsByUser.php

```
<?php
$response = array();
include __DIR__ . "/GeneralFunctions.php";
if (isset($_POST['user_id']) && isset($_POST['token'])) {
    $user_id = $_POST['user_id'];
    $token = $_POST['token'];
    $dbOperationsObject = new DBOperations();
    $generalFunctionsObject = new GeneralFunctionsClass();

    $result_token = $dbOperationsObject->isTokenExist($user_id, $token);
    if (mysqli_num_rows($result_token) > 0) {

        $resultFriendRequests = $dbOperationsObject->getFriendRequestsByUser($user_id);
        $friend_requests = $generalFunctionsObject->getFriendRequestsByUser($resultFriendRequests);
        $response["success"] = 1;
        $response["friend_requests"] = $friend_requests;
        echo json_encode($response);

    } else {
        $response["success"] = 0;
        $response["message"] = "Invalid token";
        echo json_encode($response);
    }
} else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";
    echo json_encode($response);
}
?>
```

getLastMessagesByUser.php

```
<?php
$response = array();
include __DIR__ . "/GeneralFunctions.php";
if (isset($_POST['user_id']) && isset($_POST['token'])) {
    $user_id = $_POST['user_id'];
    $token = $_POST['token'];
    $dbOperationsObject = new DBOperations();
    $generalFunctionsObject = new GeneralFunctionsClass();

    $result_token = $dbOperationsObject->isTokenExist($user_id, $token);
    if (mysqli_num_rows($result_token) > 0) {

        $resultMessages = $dbOperationsObject->getLastMessagesByUser($user_id);
        $messages = $generalFunctionsObject->getLastMessagesByUser($resultMessages, $user_id);
        $response["success"] = 1;
        $response["messages"] = $messages;
        echo json_encode($response);

    } else {
        $response["success"] = 0;
        $response["message"] = "Invalid token";
```

```
        echo json_encode($response);
    }
} else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";
    echo json_encode($response);
}
?>
```

getMessagesByUser.php

```
<?php
$response = array();
include __DIR__ . "/GeneralFunctions.php";
if (isset($_POST['user_id']) && isset($_POST['token'])) {
    $user_id = $_POST['user_id'];
    $token = $_POST['token'];
    $dbOperationsObject = new DBOperations();
    $generalFunctionsObject = new GeneralFunctionsClass();

    $result_token = $dbOperationsObject->isTokenExist($user_id, $token);
    if (mysqli_num_rows($result_token) > 0) {

        $resultMessages = $dbOperationsObject->getMessagesByUser($user_id);
        $messages = $generalFunctionsObject->getMessagesByUser($resultMessages);
        $response["success"] = 1;
        $response["messages"] = $messages;
        echo json_encode($response);

    } else {
        $response["success"] = 0;
        $response["message"] = "Invalid token";
        echo json_encode($response);
    }
} else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";
    echo json_encode($response);
}
?>
```

getMessagesByUserWithContact.php

```
<?php
$response = array();
include __DIR__ . "/GeneralFunctions.php";
if (isset($_POST['user_id'])
    && isset($_POST['contact_id'])
    && isset($_POST['token'])
) {

    $user_id = $_POST['user_id'];
    $token = $_POST['token'];
    $contact_id = $_POST['contact_id'];
    $dbOperationsObject = new DBOperations();
    $generalFunctionsObject = new GeneralFunctionsClass();

    $result_token = $dbOperationsObject->isTokenExist($user_id, $token);
```

```
    if (mysqli_num_rows($result_token) > 0) {

        $resultMessages = $dbOperationsObject->getMessagesByUserWithContact($user_id, $contact_id);

        $messages = $generalFunctionsObject->getMessagesByUserWithContact($resultMessages);
        $response["success"] = 1;
        $response["messages"] = $messages;
        echo json_encode($response);

    } else {
        $response["success"] = 0;
        $response["message"] = "Invalid token";
        echo json_encode($response);
    }
} else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";
    echo json_encode($response);
}
?>
```

getUser.php

```
<?php
// array for JSON response
$response = array();
include __DIR__ . "/GeneralFunctions.php";

if (isset($_POST['user_id'])) {
    $user_id = $_POST['user_id'];
    $dbOperationsObject = new DBOperations();
    $generalFunctionsObject = new GeneralFunctionsClass();


    $result = $dbOperationsObject->getUser($user_id);
    if (mysqli_num_rows($result) > 0) {
        $user = $generalFunctionsObject->getUserInfoSafety($result);
        $response["success"] = 1;
        $response["user"] = $user;
        echo json_encode($response);
    } else {
        $response["success"] = 0;
        $response["message"] = "error in user_id";
        echo json_encode($response);
    }


} else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";
    echo json_encode($response);
}
?>
```

login.php

```
<?php
$response = array();
```

```
include __DIR__ . "/GeneralFunctions.php";
include __DIR__ . "/utilities/encrypt_decrypt.php";

if (isset($_POST['email'])
   && isset($_POST['password'])
   && isset($_POST['token'])
) {
   $email = $_POST['email'];
   $token = $_POST['token'];
   $encryptDecryptSecurityObject = new EncryptDecryptSecurity();
   $password = $encryptDecryptSecurityObject->encrypt($_POST['password']);
   $dbOperationsObject = new DBOperations();
   $generalFunctionsObject = new GeneralFunctionsClass();
   $result = $dbOperationsObject->isLoginExist($email, $password);

   if (mysqli_num_rows($result) > 0) {
      $result_token = $dbOperationsObject->updateUserTokenByEmail($email, $token);
      if (mysqli_affected_rows($result_token) >= 0) {
         $resultUser = $dbOperationsObject->getUserByEmail($email);
         $user = $generalFunctionsObject->getUserInfo($resultUser);
         $response["success"] = 1;
         $response["user"] = $user;
         echo json_encode($response);

      } else {
         $response["success"] = 0;
         $response["message"] = "Oops! An error occurred.";
         echo json_encode($response);
      }
   } else {
      $response["success"] = 0;
      $response["message"] = "error in email or password";
      echo json_encode($response);
   }
} else {
   $response["success"] = 0;
   $response["message"] = "Required field(s) is missing";
   echo json_encode($response);
}
?>
```

register.php

```
<?php
// array for JSON response
$response = array();
include(__DIR__ . "/../sql/DBOperations.php");
include __DIR__ . "/utilities/encrypt_decrypt.php";

if (isset($_POST['email'])
   && isset($_POST['name'])
   && isset($_POST['password'])
   && isset($_POST['token'])
   && isset($_POST['user_date'])) {

   $encryptDecryptSecurityObject = new EncryptDecryptSecurity();
   $dbOperationsObject = new DBOperations();
```

```
    $name = $_POST['name'];
    $email = $_POST['email'];
    $date = $_POST['user_date'];
    $token = $_POST['token'];
    $password = $encryptDecryptSecurityObject->encrypt($_POST['password']);
    $result = $dbOperationsObject->isEmailExist($email);
    if (mysqli_num_rows($result) > 0) {
        $response['success'] = 0;
        $response['message'] = "email already exists";
        echo json_encode($response);
    } else {
        $result = $dbOperationsObject->addUser($name, $email, $password
            , $token, $date);
        if (mysqli_affected_rows($result) > 0) {
            $response['success'] = 1;
            echo json_encode($response);
        } else {
            $response['success'] = 0;
            $response['message'] = "can't register new user now";
            echo json_encode($response);
        }
    }
} else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";
    echo json_encode($response);
}
?>
```

ДОДАТОК В

Відгук

на кваліфікаційну роботу рівня магістра

«Розробка інформаційної технології для обміну миттєвими повідомленнями на основі сервіса «GetMessage»

студента групи 126м-20-1 Дудки Сергія Миколайовича

1 Мета даної кваліфікаційної роботи – створення інформаційної технології для обміну миттєвими повідомленнями на платформі ОС Android.

2 Обрана тема актуальна тому, що дана інформаційна технологія може використовуватись в будь-якій галузі, так як вона надає можливість користувачам встановити контакт на відстані і обмінюватися інформацією для будь-яких цілей і стосовно будь-яких сфер життя і діяльності людини. Для цього користувачу необхідно мати тільки підключення до мережі Інтернет.

3 Тема кваліфікаційної роботи відповідного рівня безпосередньо пов'язана з об'єктом діяльності магістра спеціальності 126 «Інформаційні системи та технології» галузі знань 12 «Інформаційні технології» - створення інформаційних технологій різного призначення і застосування.

4 Явища і процеси, що досліджуються в даній кваліфікаційній роботі і обрані для моделювання, оцінювання та дослідження – віднесені в освітньо-кваліфікаційній характеристиці магістрів до класу дослідних та евристичних, рішення яких заснована на знаково-понятійних уміннях.

5 Робота складається з трьох розділів. Перший розділ присвячений аналізу предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів. Оригінальність отриманих в роботі наукових результатів і їх наукова новизна полягають у наступному:

- дослідженні компонентів побудови інформаційної технології створення відповідного сервіса;

- обгрунтування дизайнерських рішень та складу компонентів побудови нового сервісу;

- виконання розробки інформаційної технології для обміну миттєвими повідомленнями на основі сервіса «GetMessage».

6 Практичне значення результатів роботи полягає у створенні додатка, що надає можливість безпечно обмінюватися миттєвими повідомленнями користувачам в режимі реального часу при наявності підключення до мережі Інтернет.

7 Практичні результати кваліфікаційної роботи отримані із застосуванням відповідних програмних систем та засобів, а також програмних продуктів MS Word і MS PowerPoint на інформаційно-технологічній платформі Windows.

8 Оформлення графічних матеріалів до кваліфікаційної роботи рівня магістр виконано на сучасному рівні і відповідає вимогам, що пред'являються до рівня виконання робіт даної кваліфікації.

9 Ступінь самостійності виконання кваліфікаційної роботи достатньо висока.

10 Деякі дискусійні положення та недоліки, які мають місце в роботі:

а) недостатньо чітко визначено структуру інформаційноної системи;

б) досить стисло описано алгоритм функціонування створеної технології.

Незважаючи на вищевказані зауваження, кваліфікаційна робота в цілому заслуговує оцінки «відмінно» та присвоєння здобувачу відповідної кваліфікації.

Керівник кваліфікаційної роботи,
проф. кафедри ІТКІ, д.т.н. Г.М. Коротенко

ДОДАТОК Г

Рецензія

на кваліфікаційну роботу рівня магістра

«Розробка інформаційної технології для обміну миттєвими повідомленнями на основі сервіса «GetMessage»

студента групи 126м-20-1 Дудки Сергія Миколайовича

Розглянута робота присвячена розробці інформаційної технології для обміну миттєвими повідомленнями на платформі ОС Android.

Завдання і зміст кваліфікаційної роботи відповідає головній цілі - перевірці знань і ступеня підготовленості студента за фахом 126 «Інформаційні системи та технології» галузі знань 12 «Інформаційні технології».

Зміст пояснювальної записки кваліфікаційної роботи відповідає необхідним критеріям та затвердженій темі.

Актуальність обраної теми обумовлена тим, що дослідження ефективності застосування інформаційних технологій і відповідних комп'ютерних засобів продовжують розвиватися на базі розширення їхнього спектру.

Повнота і глибина вирішення задач, поставлених в завданні на кваліфікаційну роботу є достатньою.

Оформлення пояснювальної записки кваліфікаційної роботи виконано в повній відповідності з діючими стандартами і нормативними вимогами.

Список літератури, наведений в роботі, налічує більше ніж 30 джерел, що свідчить про вміння автора працювати з літературою та іншими інформаційними матеріалами.

Наукова новизна результатів дипломної роботи визначається тим, що вперше розроблено оригінальну інформаційну технологію для обміну миттєвими повідомленнями на платформі ОС Android.

Практичне значення результатів роботи полягає у створенні додатка, що надає можливість безпечно обмінюватися миттєвими повідомленнями користувачам в режимі реального часу при підключенні до мережі Інтернет.

До числа загальних зауважень і недоліків роботи слід віднести:

1) неоднорідність викладення матеріалу та обгрунтування у різних частинах кваліфікаційної роботи;

2) відсутність порівняння результатів розробки дизайну з іншими відомими роботами на дану тему;

3) недостатнє обґрунтування відповідності розробленого дизайну заданим критеріям.

Однак, зазначені зауваження не здійснюють істотного впливу на підсумкові результати кваліфікаційної роботи і не знижують її безумовну практичну та наукову цінність.

Таким чином, слід зробити висновок, що кваліфікаційна робота в цілому
заслуговує оцінки «_____», а її виконавець присвоєння відповідної кваліфікації.

Рецензент, доцент кафедри безпеки
інформації та телекомунікацій, к.т.н. О.М. Галушко