Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Програмно-алгоритмічний комплекс для обміну повідомленнями користувачів Android-пристроїв

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

(підпис) Орлінський М.В.
(прізвище та ініціали)

Керівник (підпис) Матійчук Л.П.
(прізвище та ініціали)

Нормоконтроль (підпис) Шимчук Г.В.
(прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О.
(прізвище та ініціали)

Рецензент (підпис) Жаровський Р.О.
(прізвище та ініціали)

Тернопіль
2022

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Орлінському Максиму Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмно-алгоритмічний комплекс для обміну повідомленнями користувачів Android-пристроїв

Керівник роботи Матійчук Любомир Павлович, к.е.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 16 » березня 2022 року № 4/7-161.

2. Термін подання студентом завершеної роботи 13 червня 2022р.

3. Вихідні дані до роботи Наукові публікації, електронні ресурси, підручники , посібники з тематики дослідження

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. Розділ 1. Аналіз предметної області та проектування. 1.1 Коротка характеристика об'єкту та опис предметної області. 1.2 Огляд аналогів систем обміну миттєвими повідомленнями. 1.3. Розроблення архітектури програмної системи 1.4. Проектування структури бази даних. 1.5 Висновок до першого розділу. Розділ 2. Програмна реалізація, тестування та дослідна експлуатація. 2.1 Програмна реалізація проекту. 2.2. Програмна реалізація бази даних. 2.3. Тестування та досдідна експлуатація. 2.4 Розгортання програмного продукту. 2.5 Висновок до другого розділу. Розділ 3. Безпека життєдіяльності, основи хорони праці. Висновки. Перелік джерел.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Тема. 2. Мета роботи. 3.Організаційна робота системи обміну повідомленнями. 4. Бізнес процеси системи. 5.Огляд аналогів. 6. Діаграма прецендентів. 7.Діаграма класів. 8. Діаграма послідовності «Взаємодія з чатом». 9. Діаграма послідовності «Робота з писком контактів».10. ER- діаграма бази даних. 11. Інструкція користувача. 12. Форма для реєстрації. 13-14. Основні екранні форми. 15. Висновки. 16. Дякую за увагу.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Гурик О.Я. кандидат технічних наук, доцент кафедри МТ		

7. Дата видачі завдання 24 січня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	24.01.2022	*Виконано*
2.	Підбір джерел про програмно-алгоритмічний комплекс для обміну повідомленнями користувачів Android-пристроїв	04.01.2022-30.01.2022	*Виконано*
3.	Переклад та опрацювання джерел про програмно-алгоритмічний комплекс	31.01.2022-06.02.2022	*Виконано*
4.	Виконання дослідження щодо програмно-алгоритмічний комплекс для обміну повідомленнями користувачів Android-пристроїв	07.02.2022-13.02.2022	*Виконано*
5.	Оформлення розділу «Аналіз предметної області та проектування»	14.02.2022-06.03.2022	*Виконано*
6.	Оформлення розділу «Програмна реалізація, тестування та дослідна експлуатація»	07.03.2022-03.04.2022	*Виконано*
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	04.04.2022-17.04.2022	*Виконано*
8.	Виконання завдання до підрозділу «Основи хорони	18.04.2022-01.05.2022	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	02.05.2022-15.05.2022	*Виконано*
10.	Нормоконтроль	16.05.2022-22.05.2022	*Виконано*
11.	Перевірка на плагіат	01.06.2022	*Виконано*
12.	Попередній захист кваліфікаційної роботи	07.06.2022	*Виконано*
13.	Захист кваліфікаційної роботи	21.06.2022	

Студент _____ Орлінський М.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Матійчук Л.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Програмно-алгоритмічний комплекс для обміну повідомленнями користувачів Android-пристроїв // Кваліфікаційна робота освітнього рівня «Бакалавр» // Орлінський Максим Вікторович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2022 // С.57 , рис. 42– , табл.– 2, бібліогр. – 15.

***Ключові слова**:* мобільний застосунок, миттєві повідомлення, база даних, Android-пристроїв.

Метою роботи є дослідження та детальний аналіз вже існуючих на ринку систем для обміну миттєвими повідомленнями і в подальшому оцінці всіх їх позитивних та негативних рис, щоб врахувати це при розробці власної системи.

В першому розділі було розглянуто основні аспекти роботи системи для обміну миттєвими повідомленнями. Було описано основні бізнес-процеси, що відбуваються в системі. Здійснено порівняння розроблюваної системи з існуючими на ринку аналогами такими, як Viber, Telegram, Skype, WhatsApp. Описана специфікація вимог до створюваного програмного забезпечення.

В другому розділі було розглянуто програмну реалізацію системи обміну повідомленнями. Створено базу даних для мобільного застусунку. Описані тести, що здійснювались над додатком для перевірки, а саме здійснювались такі види тестування, як модульне, інтеграційне та навантажувальне.

ANNOTATION

Software and algorithmic complex for messaging users of Android devices.// Qualification work of the educational level "Bachelor"// Orlinsky Maxim Viktorovich // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Sciences, SNsn-42 group // Ternopil, 2022 // C.57, fig. 42-, table. - 2, bibliogr. -15.

Keywords: mobile application, instant messaging, database, Android devices.

The aim of the work is to study and analyze in detail the existing systems on the market for instant messaging and further evaluate all their positive and negative features to take this into account when developing your own system.

The first section discussed the main aspects of the system for instant messaging. The main business processes taking place in the system were described. A comparison of the developed system with existing analogues on the market such as Viber, Telegram, Skype, WhatsApp. The specification of requirements to the created software is described.

In the second section the software implementation of the messaging system was considered. A database for mobile applications has been created. The tests performed on the application for verification are described, namely such types of testing as modular, integration and load testing.

ЗМІСТ

ВСТУП……..6
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПРОЕКТУВАННЯ……8
1.1 Коротка характеристика об’єкту та опис предметної області ……8
1.2 Огляд аналогів систем обміну миттєвими повідомленнями ……12
1.3. Розроблення архітектури програмної системи……17
1.4 Висновок до першого розділу……..25
РОЗДІЛ 2. ПРОГРАМНА РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ……..26
2.1 Програмна реалізація проекту……26
2.2. Програмна реалізація бази даних……32
2.3. Тестування та дослідна експлуатація……33
2.4 Розгортання програмного продукту……38
2.5 Висновок до другого розділу……46
РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ
3.1 Забезпечення електробезпеки користувачів ПК……48
3.2 Вимоги щодо охорони праці при роботі з комп’ютерами……51
3.3 Висновок до третього розділу……..……..54
ВИСНОВКИ……55
ПЕРЕЛІК ДЖЕРЕЛ……..56

ВСТУП

Актуальність теми. Інтернет змінив спосіб спілкування. Електронна пошта була найшвидшою формою спілкування, яка коли-небудь була відома. Менш ніж за два десятиліття тому мало хто чув про неї. Тепер багато хто з нас користується електронною поштою замість того, щоб писати листи або навіть телефонувати по телефону. Люди в усьому світі щодня надсилають мільярди повідомлень електронної пошти.

Але іноді навіть електронна пошта недостатньо швидка. Можливо, ви не знаєте, чи є особа, якій ви хочете надіслати лист електронною поштою, в даний момент у мережі. Крім того, якщо ви надсилаєте лист електронною поштою, вам зазвичай потрібно здійснити декілька кроків. Ось чому миттєві повідомлення стали настільки популярними.

За допомогою чату можна зберігати список людей, з якими ви спілкуєтесь. Ви можете обмінюватися повідомленнями з будь-ким у списку контактів так довго, наскільки ця особа перебуває в режимі онлайн. У випадку, якщо ця особа не є в режимі онлайн, вона все одно отримає ваше повідомлення, як тільки зайде в систему для обміну повідомленнями. Ви надсилаєте повідомлення один одному в невеликому вікні, яке відображається на обох екранах пристроїв.

Коли список друзів відкритий, надіслати миттєве повідомлення не складе сильних зусиль жодній людині. Подвійне клацання по псевдоніму контакту повідомляє клієнтському програмному забезпеченню про створення вікна миттєвого обміну повідомленнями, адресованому цьому користувачеві. Вам потрібно тільки ввести своє повідомлення в текстове поле та натиснути на кнопку відправки повідомлення. Ваша робота виконана.

Мета і задачі дослідження. Метою роботи є дослідження та детальний аналіз вже існуючих на ринку систем для обміну миттєвими повідомленнями і в подальшому оцінці всіх їх позитивних та негативних рис, щоб врахувати це при розробці власної системи та уникнути тих самих негативних рис.

Можна виділити декілька основних задач, що ставляться до розроблюваного програмного забезпечення, спираючись на мету:

а) На ринку існуючих продуктів дослідити аналоги систем для обміну миттєвими повідомленнями. Проаналізувати їхні переваги та недоліки.

б) Використовуючи проаналізовані переваги та недоліки аналогів, розробити функціональні та нефункціональні вимоги до розроблюваного програмного забезпечення.

в) Створити модель бази даних.

г) Для розробки функціональні програмного забезпечення обрати потрібні технології.

д) Розробка системи для обміну миттєвими повідомленнями відповідно до виконаних задач, розкритих в попередніх пунктах.

Об’єкт дослідження – процес обміну миттєвими повідомленнями між користувачами з використанням глобальної мережі.

Предмет дослідження – використання сучасних технологій для створення додатків, що працюють на основі операційної системи Android.

Для успішного створення системи для обміну миттєвими повідомленнями використовуватимуться наступні технології:

– Java – об'єктно-орієнтована мова програмування від Oracle, незалежна від платформи.

– SQL – декларативна мова програмування для управління реляційними базами даних і маніпулювання даними. SQL використовується для запиту, вставки, оновлення та модифікації даних.

– Firebase — це платформи розробки мобільних та веб- застосунків.

Практичне значення одержаних результатів. В результаті розробки такого додатку ми отримуємо можливість миттєво обмінюватись повідомленнями з іншими користувачами, а головне повністю безкоштовно, не враховуючи плати за використання глобальної мережі Інтернет.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Коротка характеристика об’єкту та опис предметної області

Для того, щоб краще зрозуміти навіщо розроблювати дану систему, потрібно краще розкрити тему миттєвих повідомлень. Адже, напевно, переважна більшість людей знають, що таке електронна пошта і можливість листуватись з її допомогою. Та ще далеко не всі знають, що таке обмін миттєвими повідомленнями, хоча й це явище стає все більш популярним в останні роки.

Організаційна схема роботи типового додатку для обміну миттєвими повідомленнями зображена на рисунку 1.1.

Рисунок 1.1 – Організаційна схема роботи системи обміну повідомленнями

В життєвому циклі системи обміну повідомленнями існують такі бізнес-процеси, як авторизація, перегляд інформації облікового запису, перегляд списку групових діалогів, перегляд списку контактів.

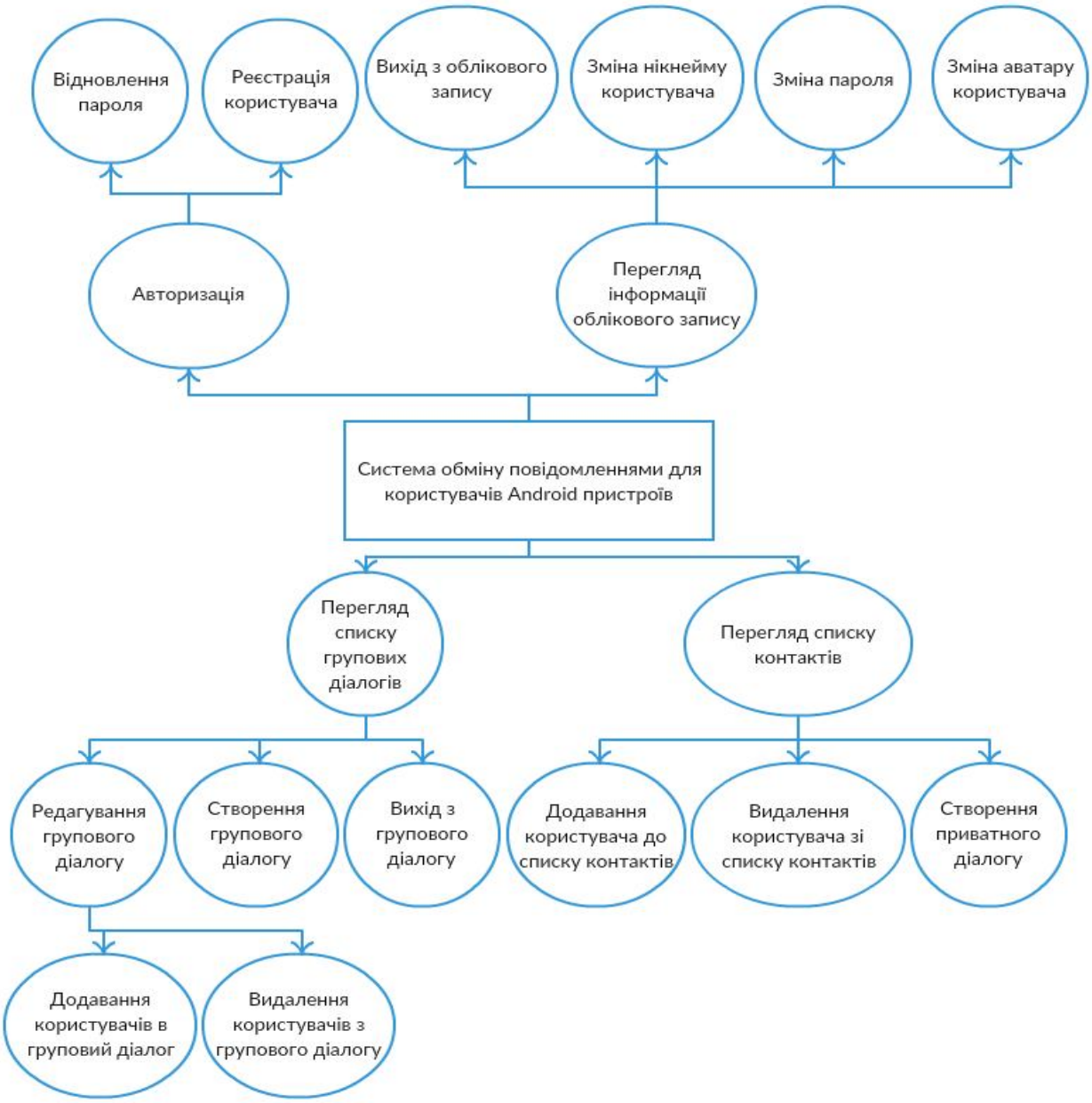

Рисунок 1.2 – Бізнес-процеси системи

Обов’язковим етапом в системі є процес авторизації. Під час його виконання користувач вказує свій адрес електронної пошти та пароль, які він використовував при реєстрації свого облікового запису. Система шукає вказаного користувача в базі даних і, у випадку, якщо найде його, надає йому доступ до його облікового запису.

Якщо був вказаний невірний адрес електронної пошти або пароль, система видає відповідне повідомлення та надає користувачу можливість вказати їх ще раз та повторити спробу входу. У випадку, якщо користувач не

зможе згадати свого пароля, він може скористуватись функцією «Забули пароль?».

При успішній авторизації, система надає доступ користувачу до його особистої інформації, списку всіх контактів, які користувач додав. Також, користувач отримує доступ до всіх своїх діалогів, групових або приватних, а також має доступ до всієї історії переписок. Також, щоб отримати доступ до вищеперерахованих функцій користувач має бути авторизований в системі, відповідно, користувач має бути зареєстрований в системі, щоб здійснити це.

Характеристика бізнес-процесу перегляд інформації облікового запису наведена в таблиці 1.2.

Таблиця 1.2– Характеристика бізнес-процесу перегляду інформації облікового запису

Назва характеристики	Значення характеристики
І’мя бізнес-процесу	Перегляд інформації облікового запису.
Основні учасники	Користувач.
Вхідна подія	Нажата кнопка для відкриття особистої інформації облікового запису.
Вхідні документи	Немає.
Вихідна подія	Відкрита форма з особистою інформацією облікового запису.
Вихідні документи	Немає.

Для здійснення цього процесу користувач має бути авторизований. Даний процес дозволяє відкриту користувачу форму, що відображає особисту інформацію його облікового запису таку, як аватар користувача, його нікнейм, адрес електронної пошти.

Під час перегляду інформації облікового запису користувач може здійснити операцію зміни свого аватару користувача, або ж змінити свій нікнейм та пароль для входу в аккаунт. Також, при цьому користувачу дозволяється вийти з свого облікового запису і після цього повернутись до форми авторизації для входу в інший обліковий запис (за бажанням). Крім

цього, варто враховувати те, що змінювати адрес електронної пошти змінювати не можна, він завжди буде тим, що вказаний при реєстрації аккаунта.

Наступним бізнес-процесом, що ми розглянемо є перегляд списку контактів користувача зображений на рисунку 1.3.

Рисунок 1.3 – Бізнес-процес «Перегляд списку контактів»

Для здійснення даного процесу користувач повинен попередньо бути авторизований в системі.

Користувач має можливість при перегляді редагувати будь-який з групових діалогів, створений ним, тобто додати або видалити з нього користувачів, змінити назву групового діалогу. Такі можливості дозволені тільки адміністраторові групового діалогу.

Для того, щоб описати, які послуги система надає її користувачам використовується діаграма прецедентів. На діаграмі прецедентів зображуються актори, тобто користувачі, що можуть бути наділені різними правами в системі. У нашому випадку на діаграмі прецедентів зобразимо звичайного користувача, що авторизований в системі та авторизованого користувача, що створив груповий діалог, якого назвемо адміністратором. Діаграма прецедентів представлена на рисунку 1.4.

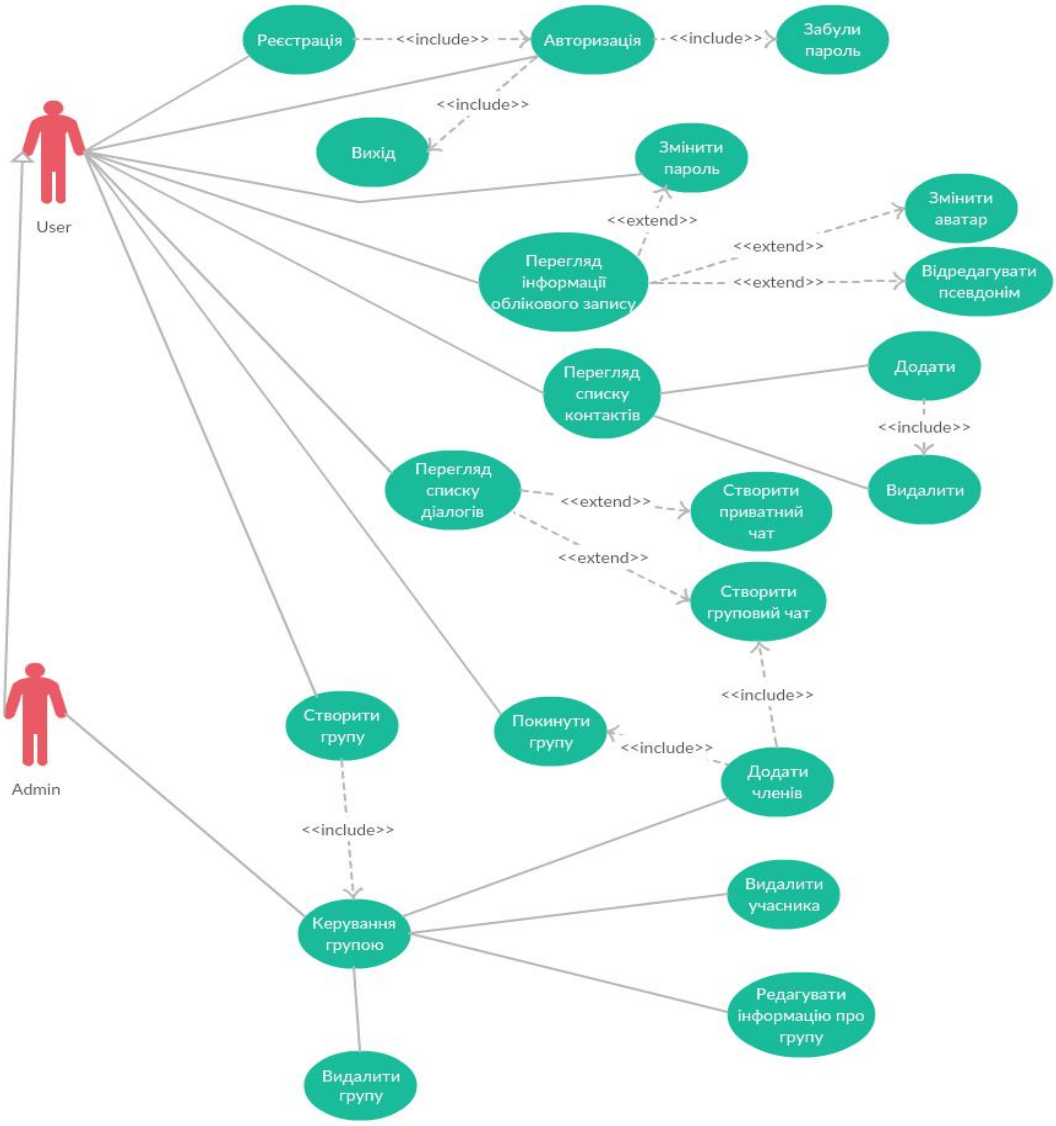

Рисунок 1.4 – Діаграма прецедентів

1.2 Огляд аналогів систем обміну миттєвими повідомленнями

Розроблюваний додаток на відміну від таких систем для обміну повідомленнями, як Viber, Telegram, WhatsApp, які прив’язуються до номеру мобільного телефону, прив’язується до вашої електронної пошти. Як і вищевказані аналоги наш додаток є безкоштовним у використанні.

Розроблювана система на відмінну від Viber, Telegram, WhatsApp призначена виключно для обміну повідомлень та не дозволяють здійснювати дзвінки. Додаток повністю напрямлений на те, щоб надати можливість користувачу відправляти текстові повідомлення.

Всі вище перераховані системи для обміну повідомленнями дозволяють встановлювати аватар користувача. Наприклад, при перегляді списку контактів, користувач зможе значно пришвидшити процес пошуку потрібного йому друга, ідентифікувавши його по аватару. Додаток автоматично перетворює зображення до потрібного йому, тому, якщо користувач вибере зображення з більшим розширенням, проблем не виникне.

Telegram, на відміну від Viberт та WhatsApp дозволяє видалити повідомлення, як у себе, так і в кінцевого отримувача. Тобто, у випадку, якщо ваш співрозмовник захоче видалити інформацію, що надіслав вам, а вона є важливою для вас і знадобиться в майбутньому, то він не зможе цього зробити, оскільки система не дозволяє даної дії.

В загальному порівнюючи Viber з розроблюваною системою, можна сказати, що основною відмінністю є те, що дана система прив'язується до номера мобільного телефону та синхронізує контакти, що зображено на рисунку 1.5.

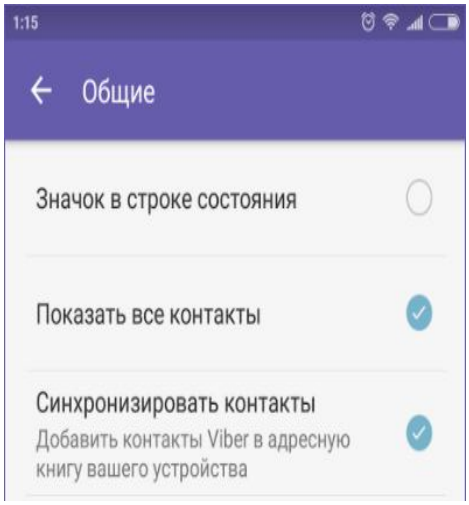

Рисунок 1.5 – Синхронізація контактів в Viber

Розроблювана ж система не використовує такий принцип реєстрації користувачів, тому не відбувається синхронізація контактів. Оскільки, при реєстрації за допомогою номера мобільного телефону, Viber автоматично зчитує контакти на вашому смартфоні та синхронізує їх з списком контактів. Також спільним з даним додатком є те, що немає веб-версії додатку. Ще однією

особливістю Viber є те, що він дозволяє синхронізувати історію повідомлень з комп'ютера на телефон, що зображено на рисунку 1.6.

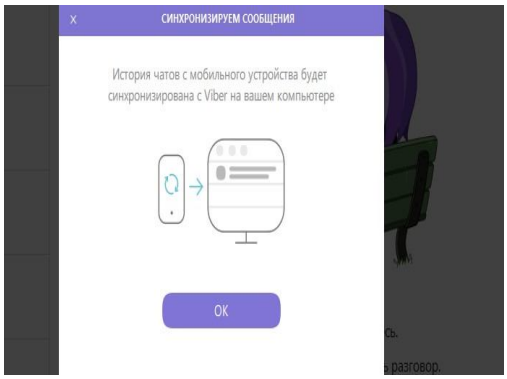

Рисунок 1.6 – Синхронізація історії чатів в Viber

Viber передбачає можливість надати іншим учасникам права для керуванням груповим діалогом. Тобто головний адміністратор, користувач, який створив груповий діалог, має можливість вибрати зі списку учасників одного або більше користувачів, які будуть наділені особливими правами в межах цього діалогу , що зображено на рисунку 1.7. При цьому варто зазначити, що видалити груповий діалог може тільки головний адміністратор, інші учасники, що були наділені правами адміністрування, не можуть зробити цього.

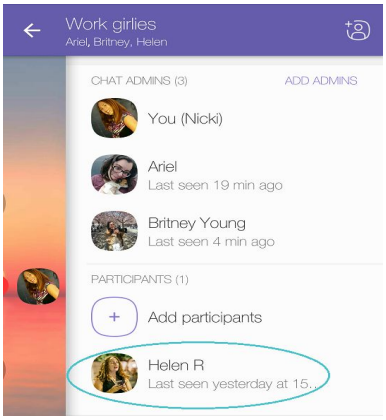

Рисунок 1.7 – Керування груповим діалогом в Viber

Інша доволі популярна система для обміну повідомленнями, та не в нашій країні, є WhatsApp. В країнах СНГ система WhatsApp не користується великою популярністю. Знову ж таки синхронізація відбувається за номером мобільного телефону, що є відмінною рисою від нашої системи. І на відмінну від даного

додатку, наша система є повністю безкоштовною і не потребує внесення коштів для її використовування. Вигляд WhatsApp зображений на рисунку 1.8.

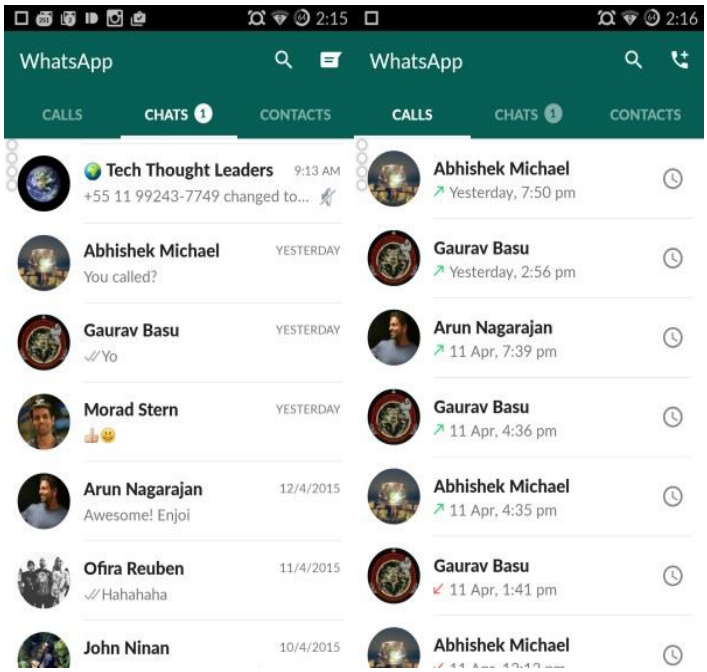

Рисунок 1.8 – Вигляд WhatsApp

WhatsApp, як і Viber дозволяє користувачам здійснювати аудіо дзвінки на безкоштовній основі, при умові підключеного пристрою до мережі Інтернет. Також, окрім аудіо дзвінків, WhatsApp дозволяє дозволяє здійснювати відео дзвінки на тій же безкоштовній основі, а також можливо здійснювати групові дзвінки групам до чотирьох людей по WhatsAppз використанням відео. Можна сказати, що це один з небагатьох мессенджерів, що потребує внесення коштів. В країнах СНГ система WhatsApp не користується великою популярністю. Здійснення відео дзвінків зображене на рисунку 1.9.

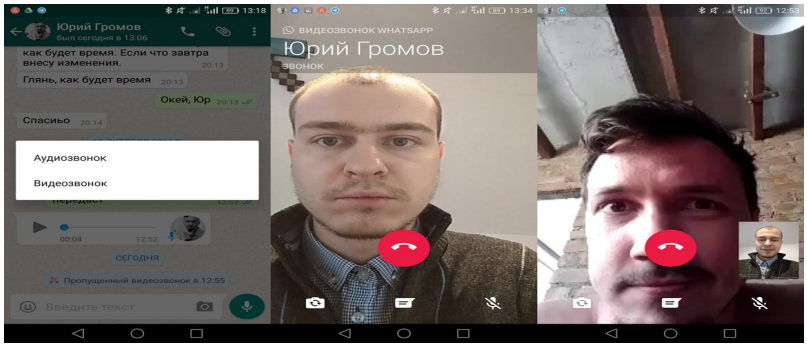

Рисунок 1.9 – Здійснення відео дзвінків в WhatApp

Telegramже в свою чергу є відносно молодим мессенджером. Його зовнішній вигляд зображено на рисунку 1.10.

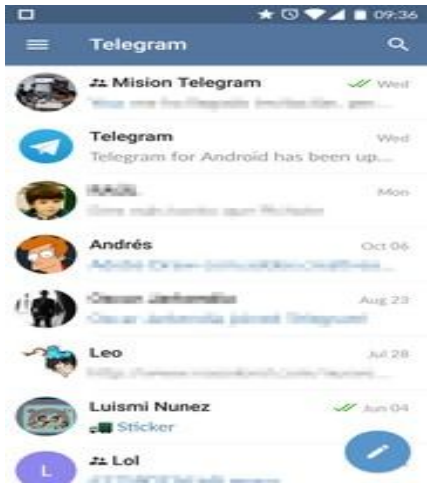

Рисунок 1.10 – Вигляд Telegram

В Telegram, як і в розроблюваному додатку є список контактів, що містить список всіх користувачів, що було добавлено. На сьогоднішній день в цій системі є дуже популярною функція приватних чатів, що забезпечують безпеку ваших даних. При даному режимі ведення чату, дані зберігаються не на сервері, як це прийнято в більшості випадків, а зберігаються тільки на кінцевих пристроях користувачів і ніхто, окрім них не зможе отримати доступу до історії їхньої переписки (див . рисунок 1.11).

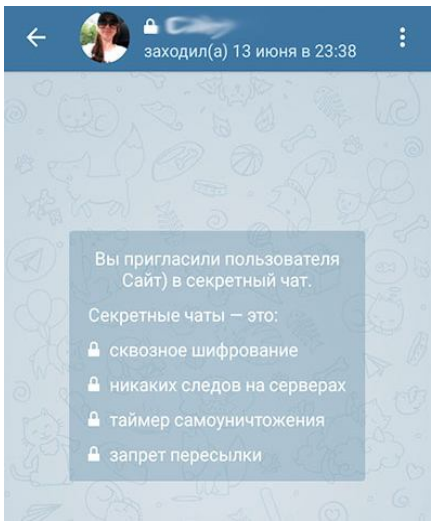

Рисунок 1.11 – Приватний чат в Telegram

Окрім приватних чатів, які не зберігають дані на сервері, система дозволяє створювати діалог між двома користувача у звичайному режимі. Також система дозволяє створювати групові діалоги, як і розглянуті аналоги WhatsApp та Viber. Крім всього система дозволяє здійснювати аудіодзвінки на безкоштовній основі.

1.3. Розроблення архітектури програмної системи

Для розробки додатку буде використовуватись архітектурний шаблон Model-View-Controller (MVC). У світі, де логіка інтерфейсу користувача має тенденцію змінюватися частіше, ніж бізнес-логіка, потрібен спосіб розділення функціональності інтерфейсу користувача. Шляхом використання архітектурного шаблону MVC було вирішено дані проблеми:

- Model - шар даних, відповідальний за управління бізнес-логікою і обробкою баз даних.
- View - шар UI - візуалізація даних з моделі.
- Controller - логічний шар, отримує сповіщення про поведінку користувача та оновлює модель за потреби.

На рисунку 1.12 зображено структуру класів MVC.

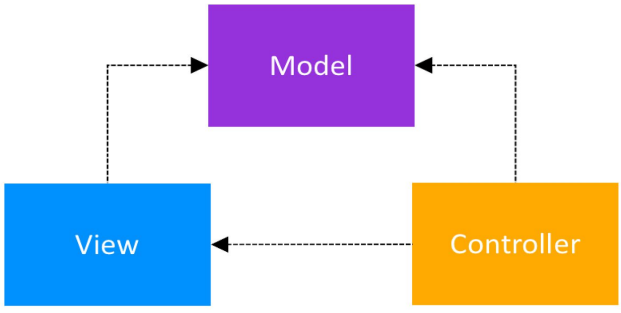

Рисунок 1.12 – Структура класів MVC

У версії пасивної моделі контролер є єдиним класом, який маніпулює моделлю. На основі дій користувача, контролер повинен змінити модель. Після оновлення моделі контролер повідомить про це, що він також потребує оновлення. У цей момент View буде запитувати дані з моделі. На рисунку 1.13 зображена поведінка пасивної моделі MVC.

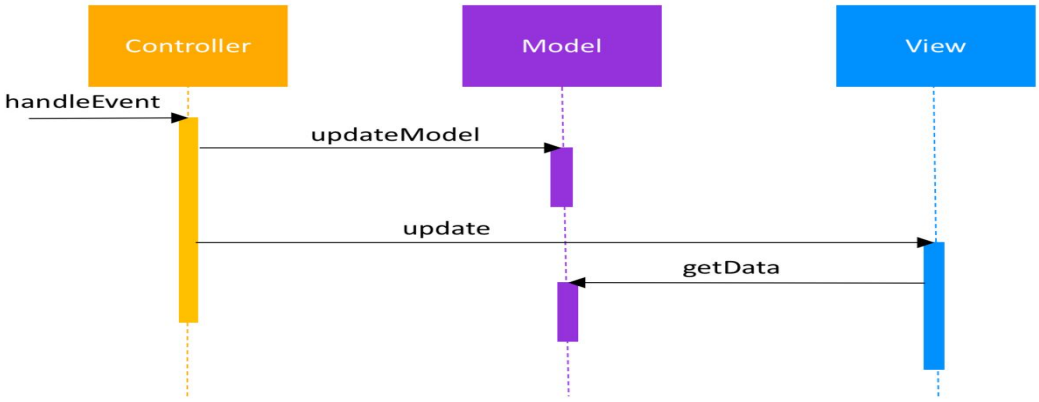

Рисунок 1.13 – Поведінка пасивної моделі MVC

Застосовуючи даний шаблон до розроблюваного додатку, можна зробити такий поділ:

- Model - все про дані. Що маніпулюється, що зберігається і як.
- View – layout, resources і віджети, такі як EditText.
- Controller – активність, фрагмент або служба.

Діаграма класів розроблюваного додатку зображена на рисунку 1.14.

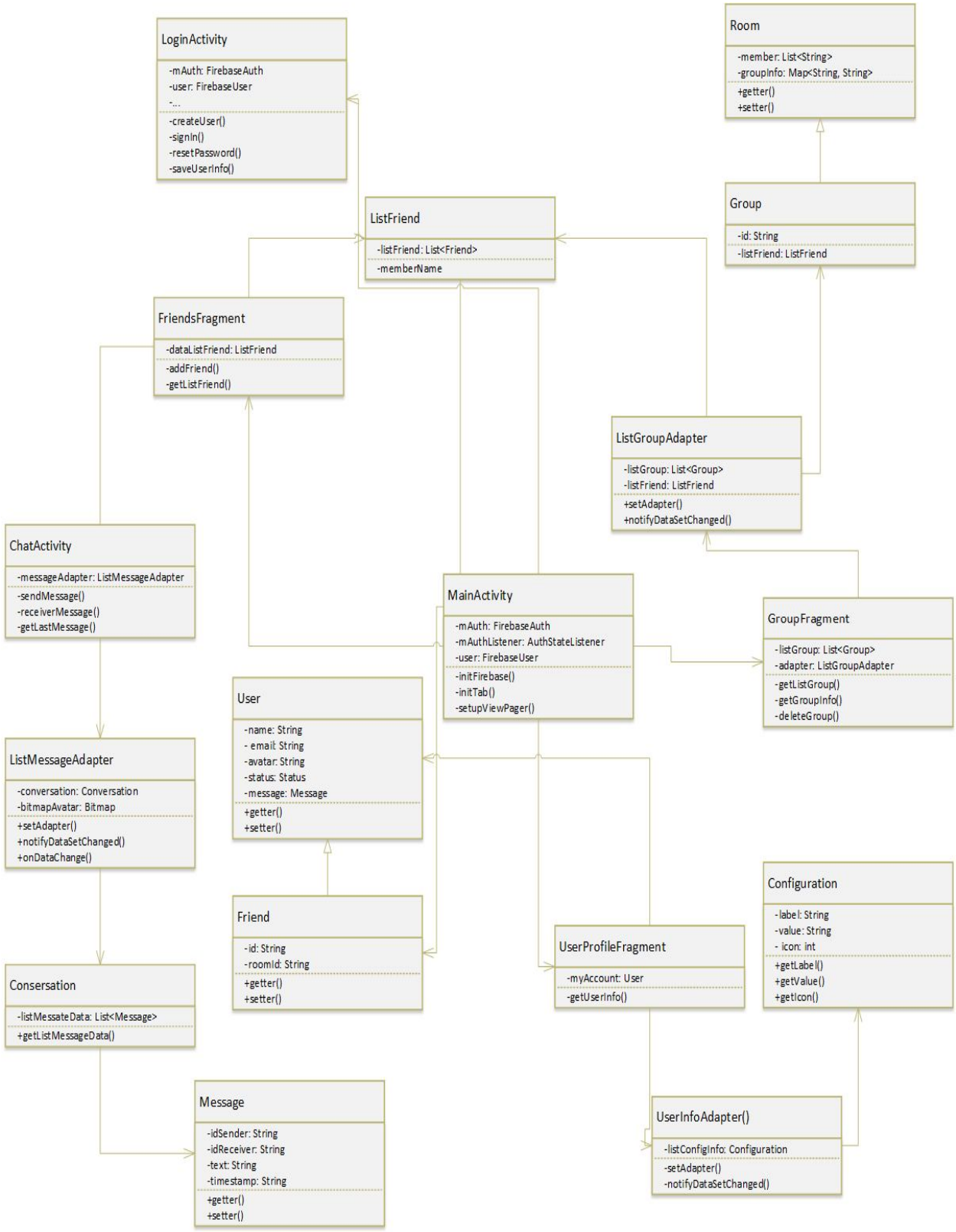

Рисунок 1.14 – Діаграма класів

Тепер перейдемо до класів, що відносяться до групи Model:

- User – відповідає за представлення користувача, містить дані про його ім’я, адрес електронної пошти, аватар, його статус.

- Friend – клас, що наслідується від Userта відповідає за представлення даних про друга, його id, idкімнати, де здійснюється переписка, крім цього містить ту ж інформацію, що і клас User.
- Consersation – відповідає за представлення даних про список повідомлень певної однієї переписки.
- Message – відповідає за представлення даних про повідомлення, тобто ким і кому було надіслане, час надсилання, текст повідомлення.
- ListFriend - відповідає за представлення даних про список друзів користувача.
- Configuration – клас, що дозволяє зберігати пари «ключ:значення». Призначений для збереження списку конфігурацій для облікового користувача.
- Room – клас, що призначений для представлення даних про переписку, містить інформацію про список учасників та інформацію про цю кімнату.
- Group – клас, що наслідується від Roomта призначений для представлення даних про групову переписку.

Розглянемо процес взаємодії користувача з чатом. На рисунку 1.15 зображена діаграма послідовності для цього процесу.

З діаграми бачимо, що контролер, яким є ChatActivity, розпочинає процес з того, що запитує в бази даних повідомлення, що було відправлене в даній бесіді. Потім, ChatActivity відображає весь список повідомлень, що були відправлені в цьому діалозі.

В процесі взаємодії користувача з чатом, контролер очікує надходження нового повідомлення і, якщо таке надійшло, отримує його від бази та відображає. У випадку, якщо користувач вводить нове повідомлення та відправляє його, контролер відправляє його в базу даних, яка зберігає його. А вже в іншого користувач по тому ж принципу контролер, який очікує нових повідомлень, отримує це повідомлення та відображає його.

З даної діаграми бачимо, що контролер ChatActivity здійснює безпосередній зв’язок з базою даних для отримання та надсилання повідомлень.

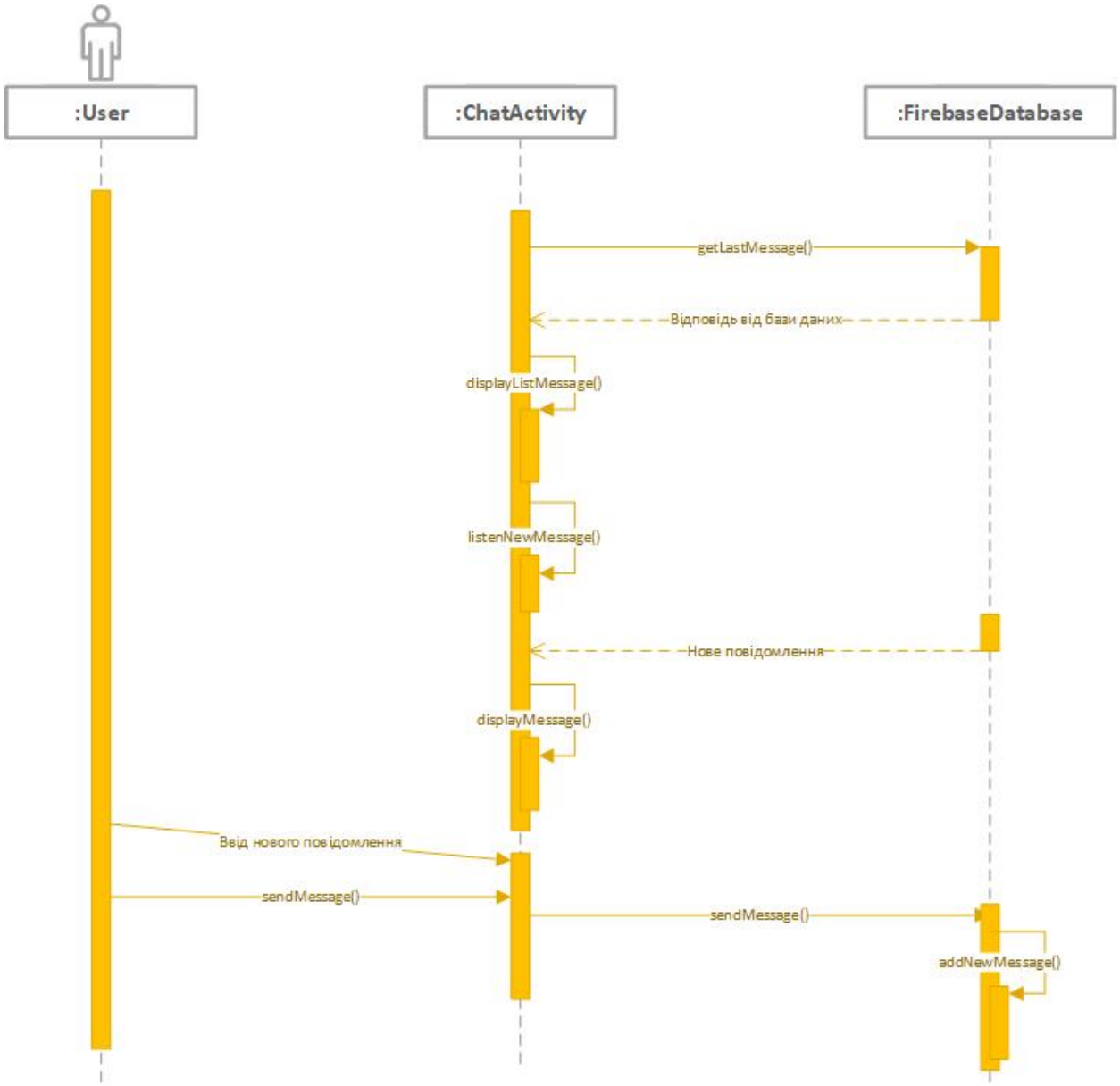

Рисунок 1.15 – Діаграма послідовності «Взаємодія з чатом»

Розглянемо діаграму активності роботи користувача з списком контактів, що зображена на рисунку 1.16. Користувач відкриває програму, після чого FriendFragment надсилає запит на отримання списку в базу даних, отримує його та відображає. Якщо користувач додає нового друга, він вводить в діалоговому вікні адрес електронної пошти іншого користувача, підтверджує додавання, після чого посилається запит в базу на пошук користувача з таким адресом. Якщо користувача було знайдено, то здійснюється перевірка, чи користувачі ще не були друзями, якщо ні, то здійснюється додавання та зміни заносяться в базу даних.

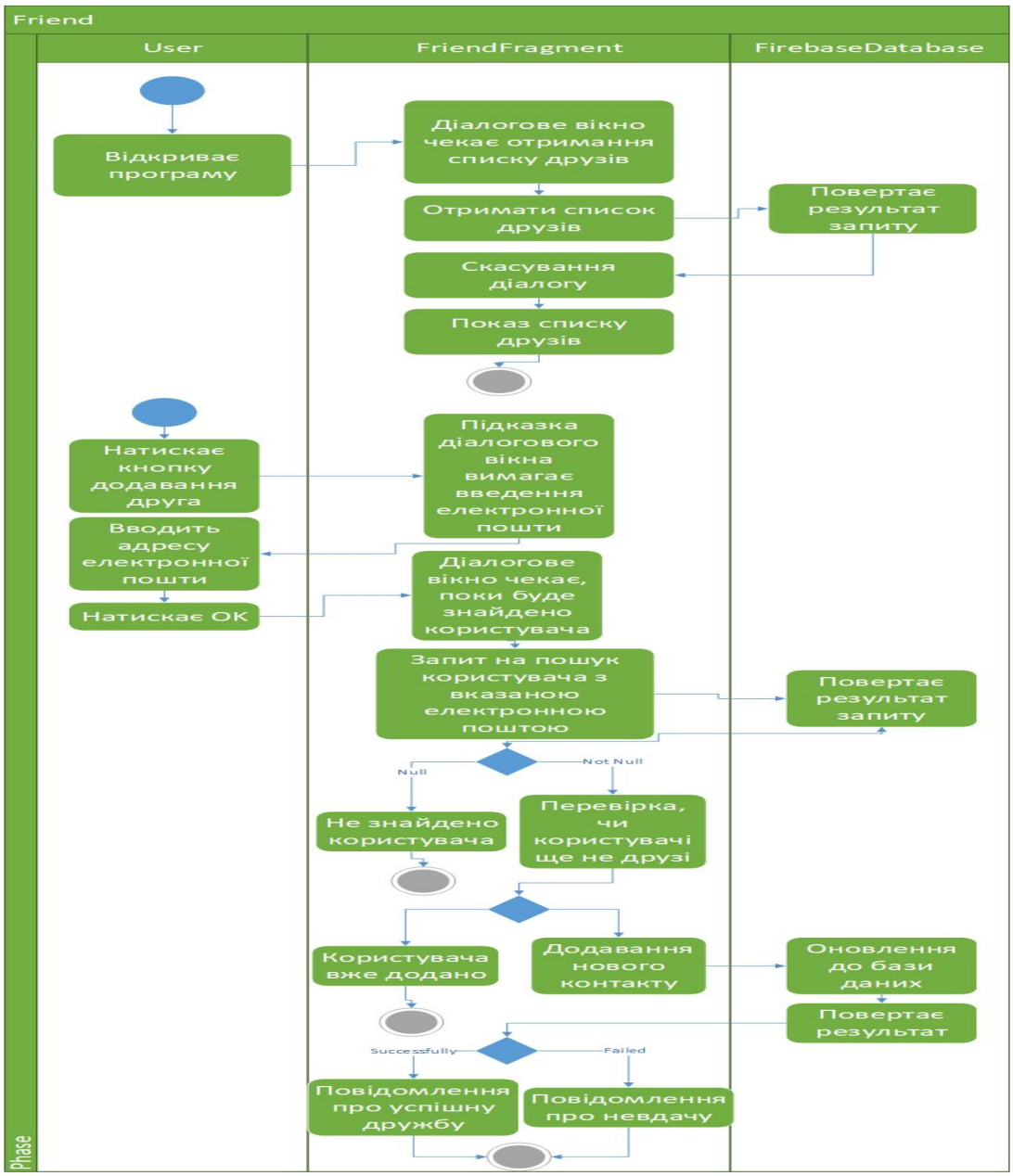

Рисунок 1.16 – Діаграма активності «Робота з списком контактів»

Якщо користувач обирає опцію створення групового діалогу, то GroupFragment відкриває форму для створення групи, де користувач вказує назву та обирає учасників і зберігає групу. GroupFragment перевіряє, чи кількість учасників групового діалогу більше, як 2 користувача. Якщо учасників більше, як 2, то здійснюється відправка запиту в базу даних на створення нової групи. Після чого база даних повертає відповідь і GroupFragment оновлює список груп і відображає створену групу або ж видає повідомлення про помилку, що сталася під час створення.

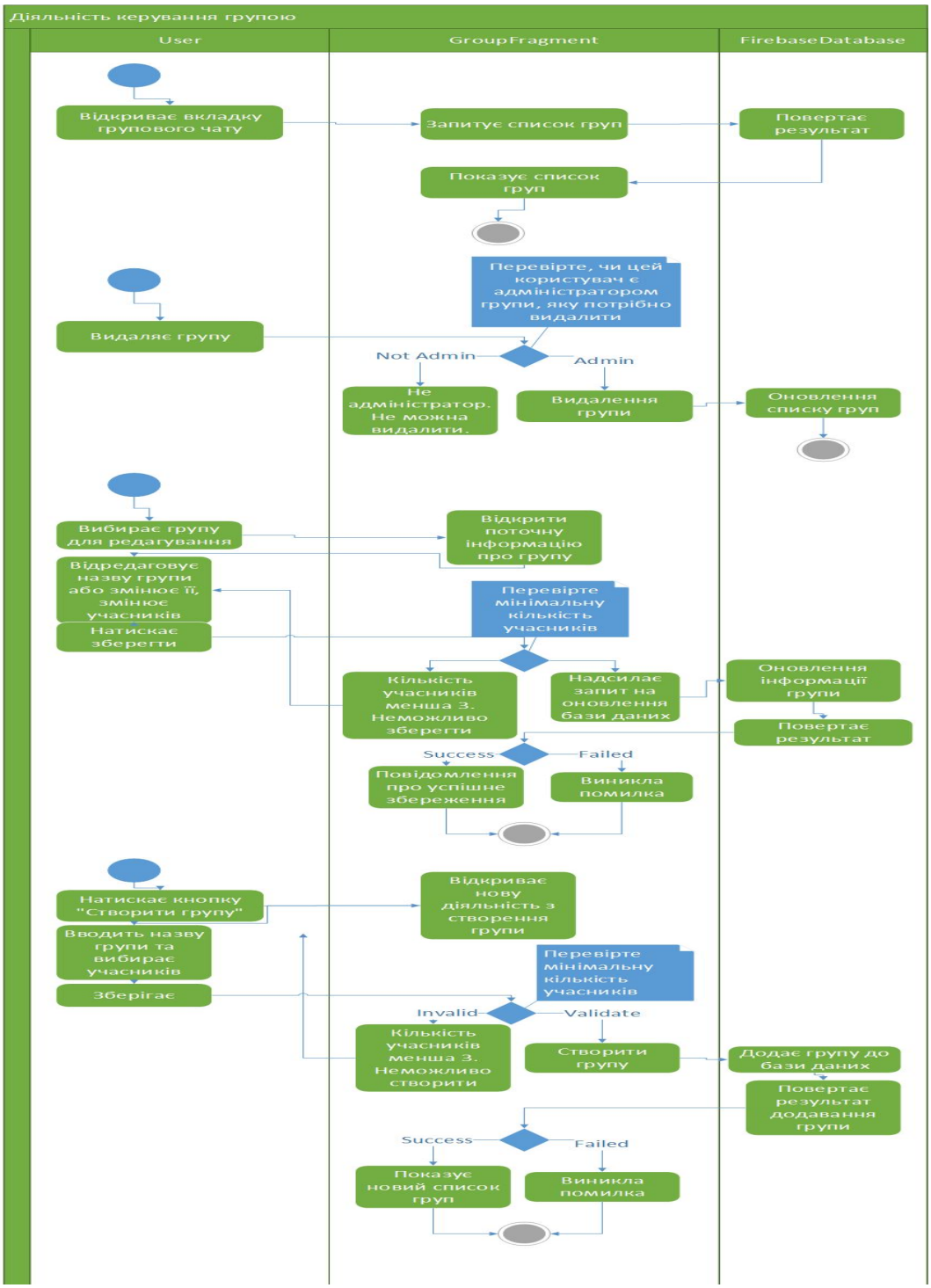

Рисунок 1.17 – Діаграма активності «Керування групою»

Для системи обміну повідомленнями буде використовуватись база даних, що складається з таких таблиць, як: User, Message, Friend, Group, GroupMember, GroupMessage. Тепер розглянемо кожну з них.

В таблиці Userбуде зберігатись інформація про користувача, а саме ім’я користувача, його зображення профіля, тобто аватар закодований методом base64, адрес електронної поштита пароль користувача.

Таблиця Friendбуде проміжної, в ній буде зберігатись два зовнішніх ключа, один буде вказувати на користувача, а інший на його друга, що входить в список його контактів.

На рисунку 1.18 зображена схема бази даних в нотацій Crow’sFoot.

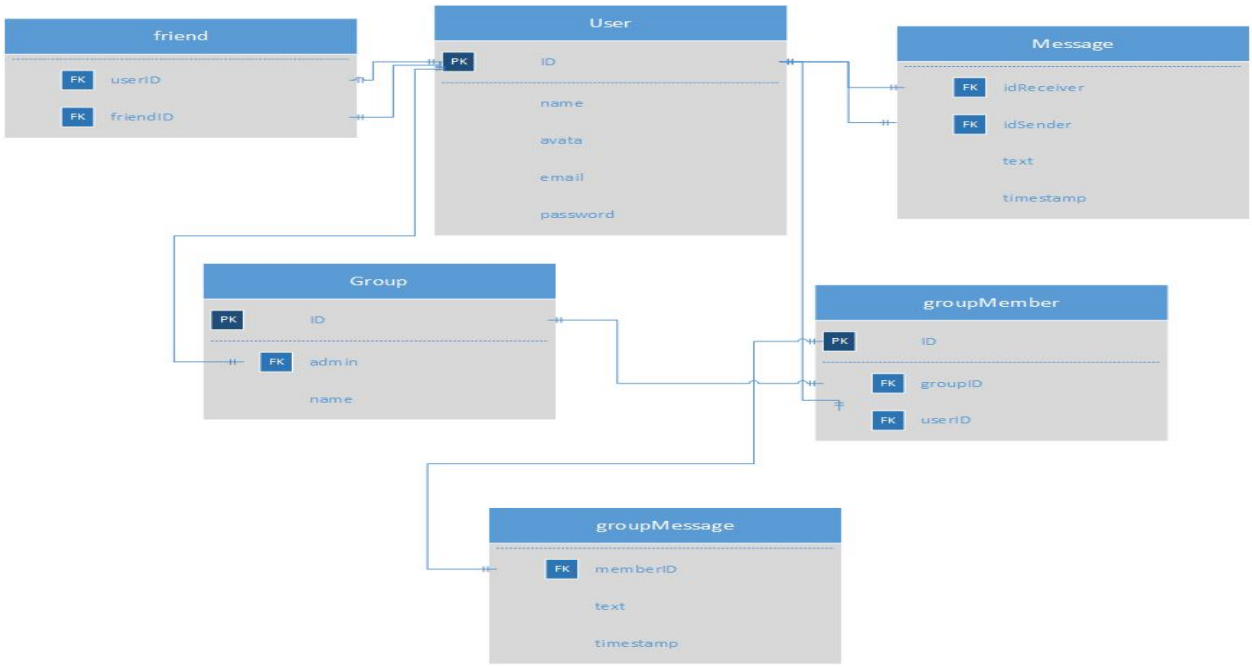

Рисунок 1.18 – ER-діаграма бази даних

Таблиця Message буде зберігати повідомлення. Вона містить два зовнішніх ключа, один з них вказує на користувача, що відправив повідомлення, а інший вказує на того користувача, якому призначається повідомлення. Крім цього, вона буде містить сам текст повідомлення та час в який було відправлено повідомлення в форматі timestamp.

Таблиця Group буде зберігати групові діалоги створені в системі. Вона містить зовнішній ключ на користувача-адміністратора, що створив груповий діалог та назву групи.

Таблиця groupMemberє проміжною таблицею що зберігатиме учасників певного групового діалогу. Вона міститиме зовнішній ключ, що посилається на первинний ключ групового діалогу та зовнішній ключ на користувача, що є учасником певного групового діалогу.

І остання таблиця GroupMessage, що буде містити повідомлення певного групового діалогу. В ній є зовнішній ключ, що посилається на таблицю

GroupMember, тобто на конкретного учасника діалогу, а також містить сам текст повідомлення та час відправки повідомлення в форматі timestamp.

1.4 Висновки до першого розділу

В першому розділі було розглянуто основні аспекти роботи системи для обміну миттєвими повідомленнями. Було описано основні бізнес-процеси, що відбуваються в системі.

Здійснено порівняння розроблюваної системи з існуючими на ринку аналогами такими, як Viber, Telegram, Skype, WhatsApp. Аналоги було проаналізовано та виділено їхні негативні та позитивні риси для покращення розроблюваної системи.

Описана специфікація вимог до створюваного програмного забезпечення. Було описано функціонал системи обміну миттєвими повідомленнями. Наведено діаграму варіантів використання та детально описано кожен з варіантів використання.

Було розглянуто ER-діаграму бази даних, оглянуто таблиці та атрибути, що будуть використовуватись, крім цього була наведена таблиця ідентифікаторів, в якій було розглянуто обмеження, що накладаються на відношення.

РОЗДІЛ 2 ПРОГРАМНА РЕАЛІЗАЦІЯ

2.1. Програмна реалізація проекту

Для реалізації системи обміну повідомленнями використовується мова програмування Java. Оскільки розроблюваний додаток призначений для платформи Android, під час розробки використовуватиметься інтегроване середовище розробки AndroidStudio. Android підтримує найбільш важливі класи в J2SESDK. За допомогою Google APIava буде написаний код, який потім буде скомпільований в файли класів. Варто враховувати, що Android не використовує JVM (віртуальну машину Java) для виконання файлів класів. Спеціально для цього використовується віртуальна машина Dalvik. Для цього файли класів компілюються в формат DalvikExecutable. Після перетворення в даний формат, файли класів та інші ресурси об'єднуються в пакети Android(APK)для послідуючої інсталяції на різних пристроях.

Також для додатку буде використовуватись платформа розробки Firebase для мобільних застосунків від Google. Дана платформа надає багато інструментів та послуг, які допомагають розробляти високоякісні програми. В додатку використовуватиметься база даних Firebase Realtime – це база даних NoSQL, що розміщена у хмарі та дозволяє зберігати та синхронізувати користувачів у реальному часі. Сама база – це великий об'єкт JSON, яким можна керувати у реальному часі.

Крім бази даних, буде використовуватись Firebase Authentication–серверні служби, прості у використанні SDKі готові бібліотеки для аутентифікації користувача у додатку. Перевірка автентичності користувача здійснюватиметься методом електронної пошти та пароля.

На рисунку 2.1 зображений скриншот з інтегрованого середовища розробки Android Studio на якому зображена структура проекту.

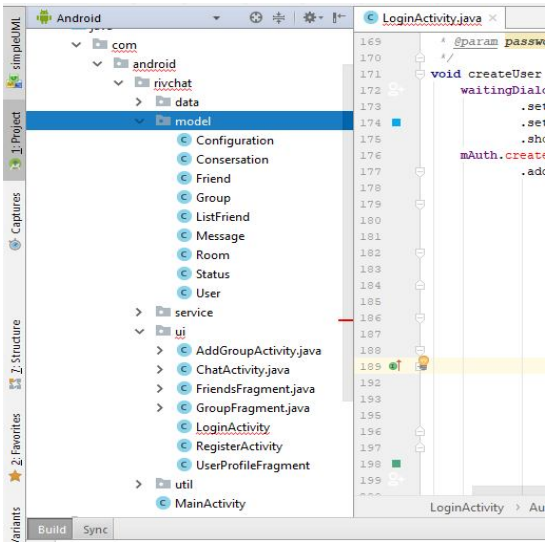

Рисунок 2.1 – Структура проекту

Перейдемо до розгляду коду основних функцій додатку.

Код функції призначеної для авторизації користувача в системі зображено на рисунку 2.2. Метод signIn здійснює авторизації користувача за допомогою введеного адресу електронної пошти та пароля. При цьому використовується об’єкт FirebaseAuth і його метод signInWithEmailAnd Password для здійснення входу з вказаними даними. За результатом цього методу, у випадку, якщо авторизація була провалена, то виводимо користувачу діалогове вікно з помилкою. Якщо ж авторизація пройшла успішно, то зберігаємо дані про користувача в системі та запускаємо головну активність.

```
void signIn(String email, String password) {    //функція авторизації
 waitingDialog.setIcon(R.drawable.ic_person_low)   //виводимо діалогове вікно
  .setTitle("Вхід....").setTopColorRes(R.color.colorPrimary).show();
 mAuth.signInWithEmailAndPassword(email, password)    //використовуємо об'єкт FirebaseAuth для входу
  .addOnCompleteListener( activity: LoginActivity.this, new OnCompleteListener<AuthResult>() {
   @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
    //якщо авторизацію провалено, виводимо повідомлення користувачу
    //якщо успішно авторизовано - зберігаємо інформацію про користувача
    //і запускаємо головну активність.
     waitingDialog.dismiss();
     if (!task.isSuccessful()) {     //якщо авторизація провалена
      new LovelyInfoDialog( context: LoginActivity.this) {
      @Override
      public LovelyInfoDialog setConfirmButtonText(String text) {
       findView(com.yarolegovich.lovelydialog.R.id.ld_btn_confirm).setOnClickListener(new View.OnClickListener() {
       @Override
       public void onClick(View view) {
        dismiss(); }});
       return super.setConfirmButtonText(text); }}
        .setTopColorRes(R.color.colorAccent).setIcon(R.drawable.ic_person_low)    //виводимо вікно з помилкою
        .setTitle("Авторизація провалена").setMessage("Email не існує або невірний пароль")
        .setCancelable(false).setConfirmButtonText("Ok").show(); }
     else {            //якщо авторизація успішна
      saveUserInfo();   //зберігаємо інформацію про користувача
      startActivity(new Intent( packageContext: LoginActivity.this, MainActivity.class));  //запускаємо головну активність
      LoginActivity.this.finish(); } }).addOnFailureListener(new OnFailureListener() {
      @Override
      public void onFailure(@NonNull Exception e) {
       waitingDialog.dismiss(); }}); }
```

Рисунок 2.2 – Код функції авторизації

Реалізація відправлення повідомлення зображено на рисунку 2.3.

```
public void onClick(View view) {
 if (view.getId() == R.id.btnSend) {
  String content = editWriteMessage.getText().toString().trim();//вміст поля поміщаємо в стрічку
  if (content.length() > 0) {                          //перевіряємо, щоб поле не було пустим
   editWriteMessage.setText("");
   Message newMessage = new Message();        //використовуємо клас повідомлення
   newMessage.text = content;
   newMessage.idSender = StaticConfig.UID;          //вказуємо поточного користувача
   newMessage.idReceiver = roomId;                  //вказуємо кому призначається повідомлення
   newMessage.timestamp = System.currentTimeMillis();        //вказуємо час
   FirebaseDatabase.getInstance().getReference().child("message/" + roomId)
    .push().setValue(newMessage);
          //і заносимо повідомлення в базу даних
      }}}
```

Рисунок 2.3 – Код функції відправлення повідомлення

Метод onClick спрацьовує при натисканні на кнопку відправки повідомлення. В стрічку content поміщаємо вміст editText, куда вводилось повідомлення. Перевіряємо, щоб довжина стрічки була більшою нуля. Далі створюємо об'єкт класу Message, що описує повідомлення. Присвоюємо йому id відправника, отримувача, час відправки та саме повідомлення. І далі за допомогою методу push() класу FirebaseDatabasezаносимо в базу даних наше повідомлення.

Перед додаванням в друзі здійснюється пошук користувача. Код пошуку користувача представлений на рисунку 2.4.

```
private void findIDEmail(String email) {            //Метод пошуку користувача за email
 dialogWait.setCancelable(false)            //Діалогове вікно з інформацією про те, що йде пошук
  .setIcon(R.drawable.ic_add_friend).setTitle("Шукаємо друга....")
  .setTopColorRes(R.color.colorPrimary).show();
 FirebaseDatabase.getInstance().getReference().child("user").orderByChild("email").equalTo(email)
  .addListenerForSingleValueEvent(new ValueEventListener() {//Робить запит в бази на отримання
 @Override                                                  //з неї email рівний вказаній
 public void onDataChange(DataSnapshot dataSnapshot) {
  dialogWait.dismiss();
  if (dataSnapshot.getValue() == null) {  //якщо ми нічого не отримали з бази, тобто його не найдено
   new LovelyInfoDialog(context)     //виводимо діалогове вікно з інформацією про це
    .setTopColorRes(R.color.colorAccent).setIcon(R.drawable.ic_add_friend)
    .setTitle("Помилка").setMessage("Email не знайдено").show();    }
  else {            //якщо користувача знайдено
   String id = ((HashMap) dataSnapshot.getValue()).keySet().iterator().next().toString();//отримуємо id
   if (id.equals(StaticConfig.UID)) {          //якщо id рівний id поточного користувача
    new LovelyInfoDialog(context)    //то видаємо помилку, себе додати не можна
     .setTopColorRes(R.color.colorAccent).setIcon(R.drawable.ic_add_friend)
     .setTitle("Помилка").setMessage("Email не валідний").show(); }
   else {
    HashMap userMap = (HashMap) ((HashMap) dataSnapshot.getValue()).get(id);//хеш-таблиця контакту
    Friend user = new Friend();       //Створюємо та описуємо нового друга
    user.name = (String) userMap.get("name");
    user.email = (String) userMap.get("email");
    user.avata = (String) userMap.get("avata");
    user.id = id;
    user.idRoom = id.compareTo(StaticConfig.UID) > 0 ? (StaticConfig.UID + id).hashCode() + "" : ""
          + (id + StaticConfig.UID).hashCode();
    checkBeforAddFriend(id, user); }}}      //І перед тим як додати перевіримо, чи він вже не є другом
 @Override
 public void onCancelled(DatabaseError databaseError) {}});}
```

Рисунок 2.4 – Код пошуку користувача

Наведений вище метод findIDEmail(Stringemail)здійснює пошук користувача в базі даних з вказаною в editTextполі адресою електронної пошти. Спочатку здійснюємо сам пошук в базі ідентичного адресу за допомогою класу FirebaseDatabasei його методу equalTo(). Потім перевіряємо отриманий результат і, якщо він рівний null, то видаємо повідомлення про те, що користувача не було знайдено. Якщо ж користувача було знайдено, перевіряємо, чи ми не знайшли самі себе, тобто поточного користувача. Прирівнюємо id знайденого користувача до id поточного користувача. Якщо вони рівні, видаємо повідомлення з помилкою. В іншому випадку, отримуємо дані з dataSnapshotта копіюємо їх в HashMap. Після чого створюємо об'єкт класу Friend та заносимо в нього дані з HashMap. Після чого відправляємо об'єкт на перевірку факту наявності його вже в списку контактів. Код перевірки представлений на рисунку 2.5.

```
private void checkBeforAddFriend(final String idFriend, Friend userInfo) {
 dialogWait.setCancelable(false)//діалогове вікно з інформацією про очікування додавання
  .setIcon(R.drawable.ic_add_friend)
  .setTitle("Додавання друга....").setTopColorRes(R.color.colorPrimary).show();
 if (listFriendID.contains(idFriend)) {     //якщо id вже в списку id друзів
  dialogWait.dismiss();
  new LovelyInfoDialog(context)     //видаємо повідомлення про те, що ми вже друзі
   .setTopColorRes(R.color.colorPrimary).setIcon(R.drawable.ic_add_friend)
   .setTitle("Друг")
   .setMessage("Користувач "+userInfo.email + " вже в списку друзів").show();}
 else {          //якщо не є другом
  addFriend(idFriend,  isIdFriend: true);//передаємо в метод додавання друга id користувача
  listFriendID.add(idFriend);        //додаємо id користувача в список id друзів
  dataListFriend.getListFriend().add(userInfo); //попонюємо список друзів
  FriendDB.getInstance(getContext()).addFriend(userInfo);//заносимо зміни в базу даних
  adapter.notifyDataSetChanged();        //перемальовуємо список друзів
    }}
```

Рисунок 2.5 – Код перевірки наявності в друзях

Метод checkBeforAddFriend() перевіряємо,чи немає користувача в поточному списку друзів. Якщо користувач є в друзях, виводиться LovelyInfoDialog з інформацією про це. Якщо користувача немає в поточному списку друзів, то його id передається в метод addFriend() для додавання. Також додаємо id нового друга в список id друзів listFriendID. Заносимо інформацію

про нового друга в dataListFriend. За допомогою класуFriendDB вносимо зміни в базу даних, а також повідомляє adapter, що дані були змінені і він відобразив зміни в графічному представленні.

Код методу addFriend() представлений на рисунку 2.6.

```
private void addFriend(final String idFriend, boolean isIdFriend) {
 if (idFriend != null) {
  if (isIdFriend) {                    //в базі присвоюємо поточному користувачу нового друга
   FirebaseDatabase.getInstance().getReference().child("friend/" + StaticConfig.UID).push().setValue(idFriend)
    .addOnCompleteListener(new OnCompleteListener<Void>() {
    @Override
    public void onComplete(@NonNull Task<Void> task) {
     if (task.isSuccessful()) {          //якщо додано успішно
      addFriend(idFriend, isIdFriend: false);}}          //відправляємо на додавання в базі другу поточного
    })                                 //користувача в список друзів
    .addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) { //якщо виникла помилка при оновленні бази
     dialogWait.dismiss();
     new LovelyInfoDialog(context).setTopColorRes(R.color.colorAccent)
      .setIcon(R.drawable.ic_add_friend).setTitle("Помилка").setMessage("Не вдалося додати").show(); }}); }
  else {          //тепер додаємо в базі другу поточного користувача в список контактів
   FirebaseDatabase.getInstance().getReference().child("friend/" + idFriend).push()
    .setValue(StaticConfig.UID).addOnCompleteListener(new OnCompleteListener<Void>() {
   @Override
   public void onComplete(@NonNull Task<Void> task) {
    if (task.isSuccessful()) {      //якщо успішно додано
     addFriend( idFriend: null, isIdFriend: false);}}})  //переходимо до останньої умови
    .addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {          //якщо виникла помилка
     dialogWait.dismiss();
     new LovelyInfoDialog(context)      //виводимо повідомлення з помилкою
      .setTopColorRes(R.color.colorAccent).setIcon(R.drawable.ic_add_friend)
      .setTitle("Помилка").setMessage("Не вдалося додати").show(); }}); } }
 else {             //коли успішно додали в обох користувачів в базі даних
  dialogWait.dismiss();
  new LovelyInfoDialog(context) //виводимо повідомлення про успішне додання друга
  .setTopColorRes(R.color.colorPrimary).setIcon(R.drawable.ic_add_friend).setTitle("Успіх")
  .setMessage("Друга успішно додано").show(); } }
```

Рисунок 2.6 – Код методу addFriend

Метод addFriend() призначений для того, щоб оновити дані в базі даних з додаванням у список друзів один одного двох користувачів. Спочатку ми заходимо в першу умову «if(isIdFriend)», після чого додаємо в друзі поточного користувача новий контакт за допомогою класу FirebaseDatabaseта його методу push().Після чого шляхом рекурсії, передаємо методу addFriend() idпоточного користувача та вказуємо isIdFriend вfalse. Після чого заходимо в умову else, де

додаємо другу idпоточного користувача в список друзів. Після цього, знову викликаємо addFriend(), але вже з першим аргументом рівним nulli заходимо в умову else, де видаємо діалогове вікно з повідомленням про те, що користувача було успішно додано в список друзів.

2.2. Програмна реалізація бази даних

Для реалізації бази даних необхідний DDLкод для створення таблиць User, Message, Group, groupMember, friendта GroupMessage.DDLкод створення даних таблиць наведений нижче:

```
CREATETABLE `User` (
        `ID` INT NOT NULL AUTO_INCREMENT,
        `name` varchar(16) NOT NULL,
        `avata` TEXT NOT NULL,
        `email` varchar(32) NOT NULL UNIQUE,
        `password` varchar(32) NOT NULL,
        PRIMARY KEY (`ID`)
);
CREATE TABLE `Message` (
        `idReceiver` INT NOT NULL,
        `idSender` INT NOT NULL,
        `text` TEXT NOT NULL,
        `timestamp` TIMESTAMP NOT NULL
);
CREATE TABLE `Group` (
        `ID` INT NOT NULL AUTO_INCREMENT,
        `admin` INT NOT NULL,
        `name` varchar(32) NOT NULL,
        PRIMARY KEY (`ID`)
);
CREATE TABLE `groupMember` (
        `ID` INT NOT NULL AUTO_INCREMENT,
        `groupID` INT NOT NULL,
        `userID` INT NOT NULL,
        PRIMARY KEY (`ID`)
);
CREATE TABLE `friend` (
        `userID` INT NOT NULL,
        `friendID` INT NOT NULL
);
CREATE TABLE `GroupMessage` (
        `memberID` INT NOT NULL,
        `text` TEXT NOT NULL,
        `timestamp` TIMESTAMP NOT NULL
);
ALTER TABLE `Message` ADD CONSTRAINT `Message_fk0` FOREIGN KEY (`idReceiver`)
REFERENCES `User`(`ID`);
ALTER TABLE `Message` ADD CONSTRAINT `Message_fk1` FOREIGN KEY (`idSender`)
REFERENCES `User`(`ID`);
```

```
ALTER TABLE `Group` ADD CONSTRAINT `Group_fk0` FOREIGN KEY (`admin`)
REFERENCES `User`(`ID`);
ALTER TABLE `groupMember` ADD CONSTRAINT `groupMember_fk0` FOREIGN KEY
(`groupID`) REFERENCES `Group`(`ID`);
ALTER TABLE `groupMember` ADD CONSTRAINT `groupMember_fk1` FOREIGN KEY
(`userID`) REFERENCES `User`(`ID`);
ALTER TABLE `friend` ADD CONSTRAINT `friend_fk0` FOREIGN KEY (`userID`)
REFERENCES `User`(`ID`);
ALTER TABLE `friend` ADD CONSTRAINT `friend_fk1` FOREIGN KEY (`friendID`)
REFERENCES `User`(`ID`);
ALTER TABLE `GroupMessage` ADD CONSTRAINT `GroupMessage_fk0` FOREIGN KEY
(`memberID`) REFERENCES `groupMember`(`ID`);
```

Оскільки база даних для системи обміну повідомленнями є порівняно невеликою, тому в ній немає багато таблиць. База зберігає інформацію про користувачів, діалоги та повідомлення.

З DDL коду вище ми бачимо, як створюється таблиця ‘User’, що представляє інформацію про користувачів. Створюється таблиця ‘Message’, яка представляє інформацію про повідомлення. Створюється таблиця ‘Group’, що представляє групові діалоги та таблиця ‘GroupMessage’, що представляє повідомлення в групових діалогах. Окрім цього створюється таблиця ‘groupMember’, яка представляє учасників групових діалогів. Остання таблиця, що створюється ‘friend’, призначена для зберігання інформації про друзів.

2.3 Тестування

Програми для Android працюють на різних пристроях. Також платформа Android і оболонки розробників пристроїв розвиваються на високій швидкості. Щоб переконатися, що додаток добре працює, потрібно написати тести для програмного забезпечення.

Модульне тестування (unittesting) для Android можна класифікувати на:

- Localunittests - тести, які можуть виконуватися на віртуальній машині Java.
- Instrumentedunittests – тести, які вимагають операційної системи Android.

Instrumentedunittests - це модульні тести, які виконуються на пристроях Android і емуляторах, а не на віртуальній машині Java. Ці тести мають доступ

до реального пристрою та його ресурсів і корисні для модульного тестування функціональності. З фреймворком для тестування інтерфейсу користувача Espresso, розробник рідко повинен використовувати API безпосередньо.

Espresso є фреймворком для тестування під Android, що дозволяє легко писати надійні тести для інтерфейсу користувача. Espresso автоматично синхронізує тестові дії з інтерфейсом користувача вашої програми. Структура також гарантує, що ваша діяльність буде запущена до запуску тестів. Це також дозволить перевірці чекати, поки всі спостережувані фонові дії не закінчаться.

Також під час модульного тестування буде використовуватись бібліотека для модульного тестування Junit4, а саме її метод assertEquals(). Метод assertEquals() перевіряє, чи очікуване та фактичне значення рівні.

Під час модульного тестування здійснюватиметься перевірка взаємодії користувача з інтерфейсом під час реєстрації. Для реалізації даної перевірки створимо клас ARegisterActivityTest, реалізація якого наведена на рисунку2.7.

```
@RunWith(AndroidJUnit4.class)
public class ARegisterActivityTest {
    private static final String email = "antony_zero6@gmail.com";
    private static final String pass = "Mk2okOf7jjda21dd";
    @Rule
    public ActivityTestRule<LoginActivity> mActivityTestRule =
            new ActivityTestRule<~>(LoginActivity.class);
    @Test
    public void clickRegister() {
        Espresso.onView(withId(R.id.fab)).perform(click());
        SystemClock.sleep( ms: 5000);
        Espresso.onView(withId(R.id.et_username)).perform(typeText(email));
        Espresso.closeSoftKeyboard();
        Espresso.onView(withId(R.id.et_password)).perform(typeText(pass));
        Espresso.closeSoftKeyboard();
        Espresso.onView(withId(R.id.et_repeatpassword)).perform(typeText(pass));
        Espresso.closeSoftKeyboard();
        Espresso.onView(withId(R.id.bt_go)).perform(click());
        SystemClock.sleep( ms: 5000);
        String current = SharedPreferenceHelper
                .getInstance(mActivityTestRule.getActivity())
                .getUserInfo().email;
        Assert.assertEquals(email,current); }}
```

Рисунок 2.7 – Реалізація ARegisterActivityTest

За допомогою фреймворку Espresso здійсниться авто заповнення даних під час реєстрації та підтвердження реєстрації. Результат тесту зображений на рисунку 2.8.

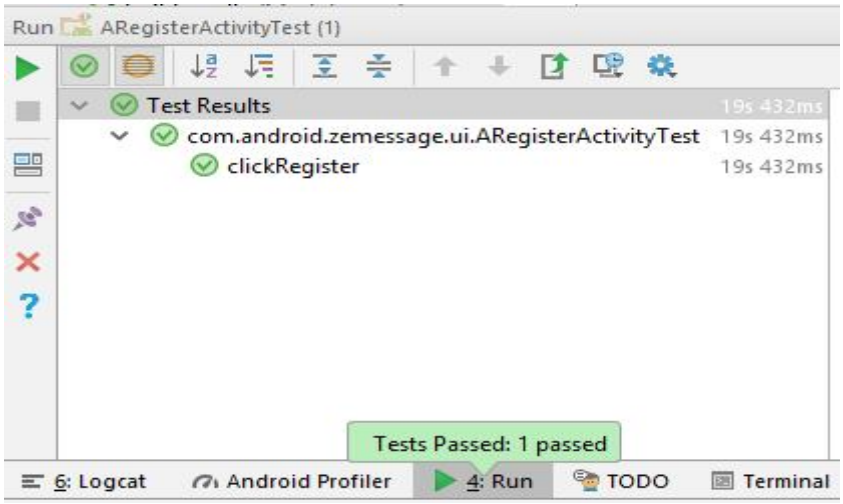

Рисунок 2.8 – Результат тесту ARegisterActivityTest

Тест виконаний успішно і email поточного користувача ідентичний email-у на який здійснювалася реєстрація.

Тепер здійснимо тестування взаємодії користувача з інтерфейсом під час авторизації в системі. Для реалізації даного тесту створимо клас BLoginActivityTest, реалізація якого наведена на рисунку 2.9.

```
@RunWith(AndroidJUnit4.class)
public class BLoginActivityTest {
    private static final String email = "antony_zero6@gmail.com";
    private static final String pass = "Mk2okOf7jjda21dd";
    @Rule
    public ActivityTestRule<LoginActivity> mActivityTestRule =
            new ActivityTestRule<~>(LoginActivity.class);
    @Test
    public void clickLogin() {
        Espresso.onView(withId(R.id.et_username)).perform(typeText(email));
        Espresso.closeSoftKeyboard();
        Espresso.onView(withId(R.id.et_password)).perform(typeText(pass));
        Espresso.closeSoftKeyboard();
        Espresso.onView(withId(R.id.bt_go)).perform(click());
        SystemClock.sleep( ms: 5000);
        String current = SharedPreferenceHelper.getInstance(mActivityTestRule
                .getActivity()).getUserInfo().email;
        Assert.assertEquals(email,current); }}
```

Рисунок 2.9 – Реалізація BLoginActivityTest

Тест відбувається по тій же логіці, що і попередній. ФреймворкEspresso здійснює автозаповнення даних та підтверджує вхід, а в кінці перевіряється email поточного користувача. Якщо він рівний тому, що використовувався при

вході, то тест пройшов успішно. Результат проходження тесту зображений на рисунку 2.10.

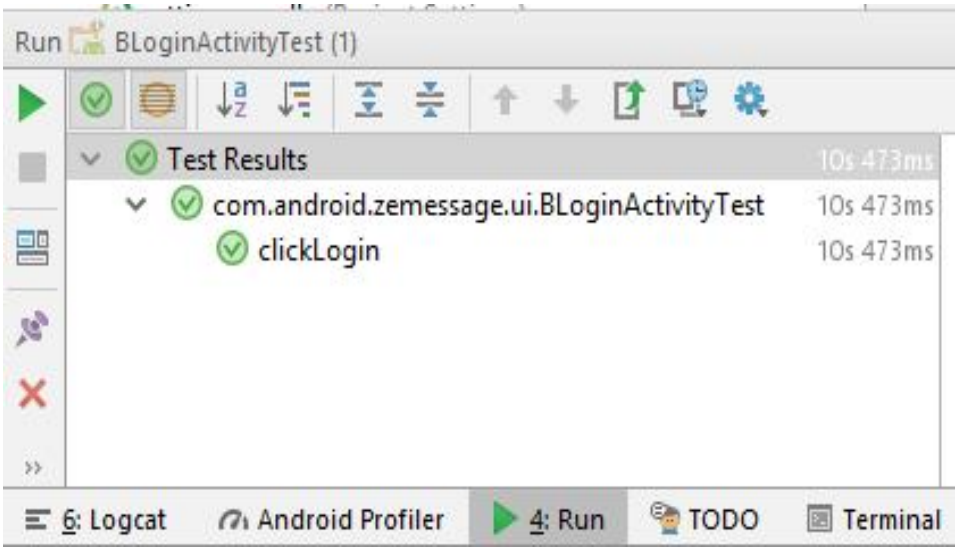

Рисунок 2.10 – Результат тесту BLoginActivityTest

Окрім взаємодії з інтерфейсом, в цих тестах перевіряється робота методів для здійснення авторизації та реєстрації. Обидва тести пройдено успішно. Тому механізми системи для реєстрації та авторизації працюють справно.

Інтеграційне тестування - це процес тестування інтерфейсу між двома програмними блоками або модулями. Воно зосереджене на визначенні правильності інтерфейсу. Метою інтеграційного тестування є виявлення несправностей у взаємодії між інтегрованими блоками. Після того, як всі модулі пройшли тестування, проводиться інтеграційне тестування. Під час такого тестування буде проводитись перевірка зв’язку між авторизацією, додаванням користувача в список друзів та здійснення відправки повідомлення цьому користувачу. Текст кейс даного тесту поданий в таблиці 2.1

Таблиця 2.1–Тестовий випадок відправки повідомлення новому другу

Опис текстового випадку	Додаємо нового користувача в список друзів. Після чого відкриваємо переписку з ним, вводимо повідомлення та відправляємо його
Кроки для	1. Відкриваємо додаток. Вводимо в полі email значення

відтворення	«gimbarr.mykytiuk@gmail.com» та в полі пароль значення «mykytiuk1998» та нажимаємо кнопку «ВВІЙТИ». 2. Відкриваємо форму додання нового друга шляхом натискання на кнопку «+» у вкладці зі списком друзів. 3. Вводимо emailцього друга, а саме «antony_zero6@gmail.com». 4. Відкриваємо переписку з користувачем «antony_zero6». 5. Набираємо повідомлення «Привіт» та відправляємо його шляхом натискання на кнопку відправки, що знаходиться в правому нижньому кутку.
Очікуваний результат	Результатом даних дій буде додана переписка у вкладці зі списком друзів, що містить доданого користувача. Переписка буде містити відправлене користувачем повідомлення.

Проводилось навантажувальне тестування, під час якого на пристрої відкривались сторонні додатки у розмірі 5 штук та використовували підключення до мережі Інтернет. При цьому тестуванні додаток поводив себе добре, під час виконання дій додавання користувачів, відправки повідомлень, створення груп та інших, не було виявлено ніяких проблем. Час затрачений на виконання цих дій не був змінений суттєво. Тест проводився на пристрої SamsungGalaxyS7, що працює під управлінням операційної системі Android 8.0.0

Для розгортання програмного продукту на пристрої, що керується операційно системою Androidпотрібно скачати apkфайл додатку. Встановити його на своєму пристрої, при цьому надати йому всі необхідні дозволи. Для запуску програмного продукту вимагається операційна система Android 5.1 (Lollipop) і вище. Рекомендовано мати на своєму пристрої 20 мегабайт вільного простору. Процесор, що використовується на пристрої та забезпечить безперебійну роботу додатку повинен бути щонайменше 2GHzdualcore. Необхідно мати на своєму пристрої, як мінімум, 1.5 гігабайта оперативної пам’яті, рекомендується мати 2 гігабайта.

Для правильної роботи додатку має бути правильно налаштована серверна частина. Оскільки в нас в якості серверної частини використовується сервіс Firebase та згенерований apk файл вже містить google-services.json, в якому містяться всі необхідні метадані, то залишається тільки налаштувати серверну частину під свої потреби в FirebaseConsole.

В FirebaseConsoleможна здійснювати налаштування бази даних, керувати користувачами та тому подібне. Це по суті серверна панель адміністратора.

Робота додатку не викликає жодних питань на пристроях, що використовують SDK 22, що використовується в Android 5.1 (Lollipop). Під час роботи з SDKменшого рівня можуть виникати збої в роботі.

2.4 Інструкція користувача

При вході в додаток перед нами відкривається форма для авторизації, що зображена на рисунку 2.11.

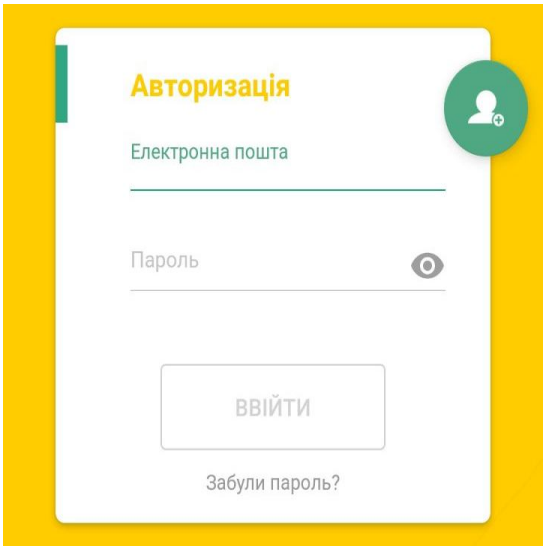

Рисунок 2.11 – Форма авторизації

На даній форма розташовані поля, необхідні для здійснення авторизації в системі, а саме поля для вводу електронної пошти та пароля. Для відновлення пароля потрібно ввести адрес електронної пошти профіля та натиснути кнопку «Забули пароль?», після чого на електронну пошту буде надісланий лист з інструкцією по відновленню пароля. Для здійснення авторизації вводимо адрес електронної пошти на який зареєстрований обліковий запис та пароль від нього. Після чого необхідно натиснути на кнопку «ВВІЙТИ», що знаходиться під полем для вводу пароля.

Також на даній формі розташована кнопка для здійснення процесу відновлення пароля та кнопка для переходу до форми реєстрації. Здійснимо перехід до форми реєстрації натиснувши на кнопку «+», що знаходиться в правому верхньому кутку. Форма для реєстрації зображена на рисунку 2.12.

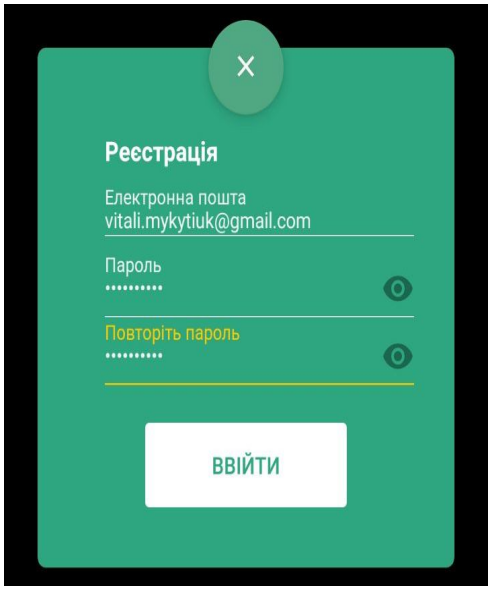

Рисунок 2.12 – Форма для реєстрації

На формі для реєстрації знаходять поля для вводу адресу електронної пошти, пароля та підтвердження пароля, які на рисунку вже попередньо заповнені. Також тут знаходиться кнопка «ВВІЙТИ», необхідна для підтвердження реєстрації. Заповнивши поля відповідними даними натискаємо на кнопку підтвердження та очікуємо завершення процесу реєстрації.

Після завершення процесу реєстрації, автоматично здійснюється авторизація в системі і ми попадаємо в головне вікно додатку, авторизовані в системі під обліковим записом, що був створений раніше. Головне вікно зображене на рисунку 2.13. Перед нами відкривається вікно, що містить 3 вкладки.

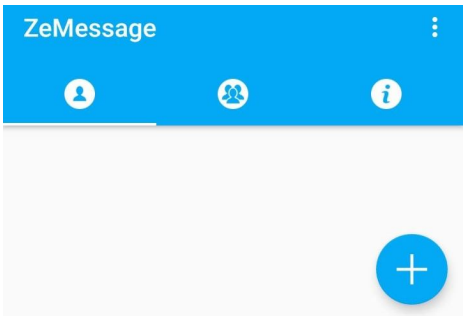

Рисунок 2.13 – Вкладка списку друзів головного вікна додатку

Перша вкладка, що містить список переписок з доданими до списку друзів користувачами. Оскільки у нас поки що немає друзів, список переписок пустий. В цій же вкладці знаходиться кнопка «+» для додання користувача в список друзів. Якщо здійснити натискання на неї, перед нами відкриється вікно для додання нового друга зображене на рисунку 2.14. В даному вікні знаходиться форма для вводу email-у користувача, якого ми хочемо додати в список друзів. Також тут знаходиться кнопка «Ок» для підтвердження. Спочатку вводимо адрес електронної пошти друга під яким зареєстрований його обліковий запис та натискаємо кнопку «Ок». Після цього здійснюється пошук друга та, у випадку його знаходження, він додається в список друзів. Якщо користувач, з вказаною адресою електронної пошти не зареєстрований в системі, то появиться повідомлення з помилкою, що сповістить користувача про необхідність перевірки введеного адресу.

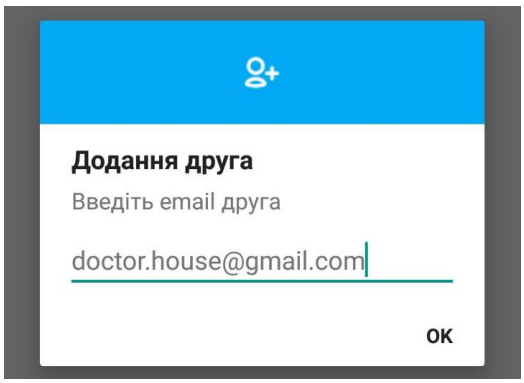

Рисунок 2.14 – Вікно для додавання друга

Після того як друга було добавлено, переписка з ним відображається в списку друзів, що зображено на рисунку 2.15.

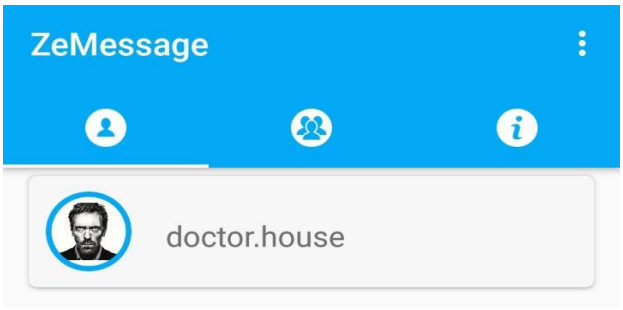

Рисунок 2.15 – Переписка з доданим другом

Для того щоб відкрити переписку з доданим другом, натискаємо на переписку з ним. Перед нами появиться вікно з історією переписки , що зображене на рисунку 2.16.

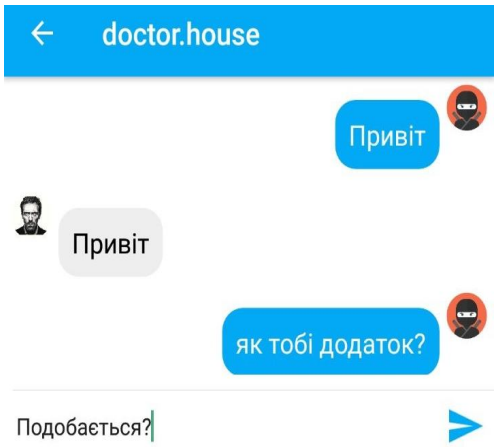

Рисунок 2.16 – Переписка з другом

На даній формі відображається історія переписки з користувачем, а також форма для вводу повідомлення та кнопка для відправки повідомлення. Для відправки повідомлення вводимо саме повідомлення в відповідне поле та натискаємо на кнопку відправки повідомлення, що знаходиться в правому нижньому кутку.

Перейдемо до головного вікна та виберемо вкладку групових діалогів, що зображена на рисунку 2.17.

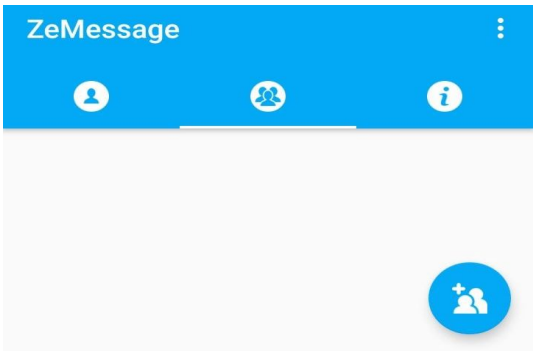

Рисунок 2.17 – Вкладка групових повідомлень

В цій вкладці відображається список групових діалогів в якому користувач бере участь і ті, що він створив сам. В правому нижньому кутку знаходиться кнопка для додавання нового групового діалогу. Здійснивши натискання на кнопку створення нового групового діалогу попадаємо в форму створення групового діалогу, що зображена на рисунку 2.18.

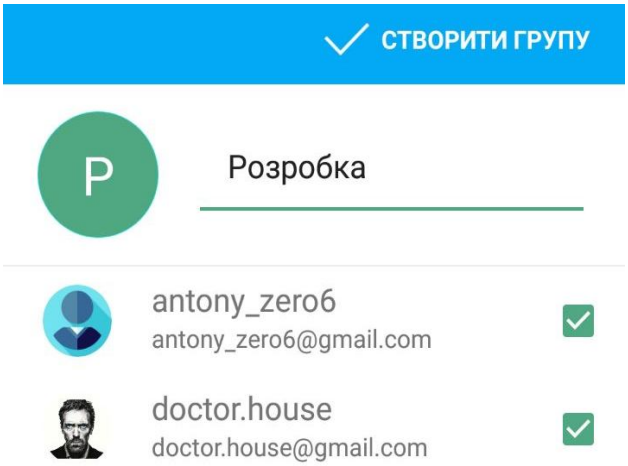

Рисунок 2.18 – Створення групового діалогу

На формі знаходиться поле для вводу назви діалогу та список користувачів, що знаходяться в списку ваших друзів. В правому верхньому кутку знаходиться кнопка «Створити групу» для підтвердження створення. Потрібно відмітити користувачів, що будуть входити в груповий діалог, їхня кількість має бути не менше 3, враховуючи користувача, що створює групу.

Після створення групового діалогу, він буде відображатись в списку групових діалогів, що зображено на рисунку 2.19. В користувачів, яких ви добавили група теж буде відображатись в списку групових діалогів. Для

відправки повідомлень в груповий діалог необхідно відкрити діалог шляхом натискання на створений діалог в списку та провести дії аналогічні тим, що проводились під час відправки повідомлення одному користувачу, що були описані раніше.

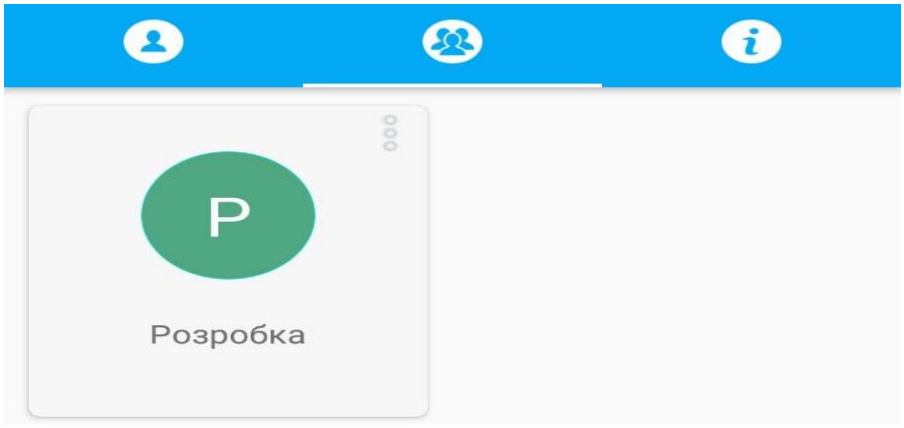

Рисунок 2.19 – Створений груповий діалог

Відкривши створений діалог отримуємо вікно для здійснення переписки, що зображене на рисунку 2.20.

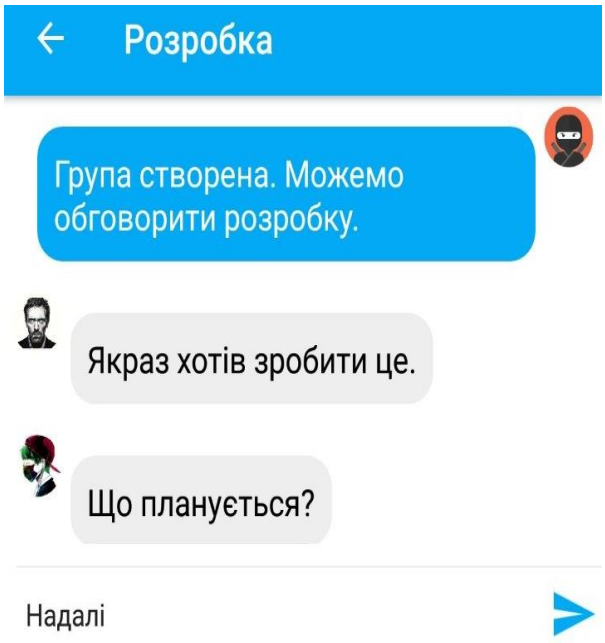

Рисунок 2.20 – Вікно групового діалогу

В даному прикладі продемонстровано груповий діалог, що складається з користувачів в кількості 3 людей. При цьому користувач, що створив груповий

діалог може здійснювати його редагування та видалення. Видалити груповий діалог може тільки користувач, що створив його. Для керування груповим діалогом потрібно натиснути кнопку трьох вертикальних крапок. І перед нами відкриється вікно, що зображене на рисунку 2.21.

Рисунок 2.21 – Керування груповим діалогом

Адміністратор, тобто користувач, що створив груповий діалог не може покинути групу, а може тільки видалити. Звичайний учасник може покинути групу. Адміністратор може здійснити редагування натиснувши кнопку «Редагувати групу». Після чого перед ним відкриється вікно редагування групи, що зображене на рисунку 2.22.

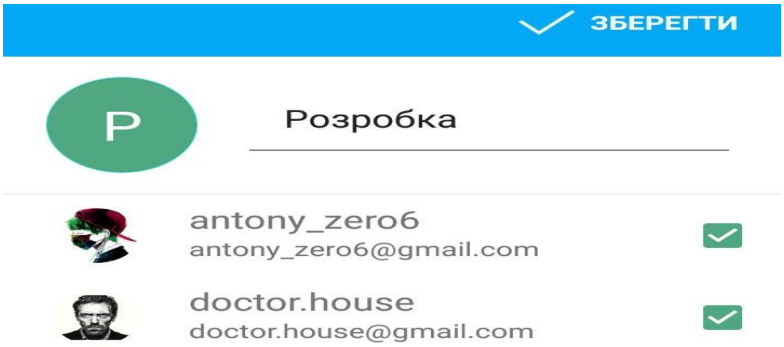

Рисунок 2.22 – Вікно редагування групового діалогу

Дане вікно, практично, аналогічне вікну створення групи, окрім кнопки «Зберегти». Адміністратор може змінити назву групи, редагувати учасників групи шляхом відмітивши нових учасників або зняти помітку з доданого учасника для видалення користувача з групи. Після здійснення потрібних змін, адміністратор повинен натиснути на кнопку «Зберегти» і групу буде редаговано і зміни будуть внесені.

Перейдемо до 3 вкладки головного вікна додатку, яка зображена на рисунку 2.23, що містить інформацію про обліковий запис користувача.

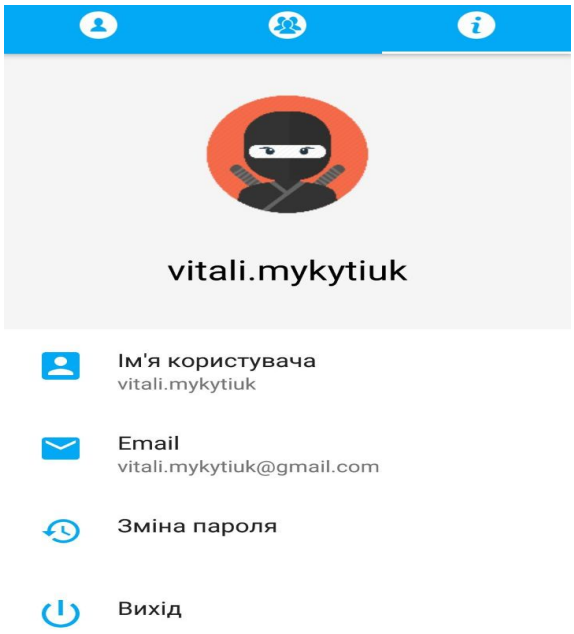

Рисунок 2.23 – Вкладка інформації про обліковий запис

З рисунку бачимо, що дана форма містить зображення профіля, ім’я користувача, email, кнопку зміни пароля та виходу. При натисканні на кнопку «Вихід», система виходить з облікового запису та повертає користувача на форму авторизації. При натисканні на кнопку «Зміна пароля», на електронну пошту користувача відправляється лист з інструкцією по зміні пароля.Emailзмінити не можна. Для зміни зображення профіля потрібно натиснути на саме зображення профіля та підтвердити зміну, після чого обрати нове зображення з галереї і підтвердити зміну. Після даних дій зображення профіля буде змінене на вибране. При натисканні на кнопку «Ім’я користувача», відкривається вікно для зміни імені користувача зображене на рисунку 2.24.

Рисунок 2.24 – Редагування імені користувача

Форма містить поле для редагування, кнопки для відміни редагування та підтвердження редагування. Користувач робить потрібні зміни у полі для редагування та підтверджує зміни шляхом натискання на кнопку «Зберегти».

2.5 Висновок до другого розділу

В цьому розділі було розглянуто програмну реалізацію системи обміну повідомленнями. Було розглянуто фрагменти коду основних функцій необхідних для системи, таких як код авторизації, відправки повідомлення, пошуку користувача в системі за вказаним адресом електронної пошти, перевірки наявності користувача в списку друзів та код для додання користувача в список друзів. Оскільки додаток в своїй роботі використовує базу даних, то в даному розділі було представлено DDL код для створення бази даних та її таблиць, таких, як ‘User’, ‘Message’, ‘Group’, ‘groupMember’, ‘GroupMessage’ та ‘friend’.

Для правильної роботи додатку потрібно не тільки написати код реалізації, але також потрібно протестувати додаток на відповідність вимогами. В цьому розділі були описані тести, що здійснювались над додатком для перевірки, а саме здійснювались такі види тестування, як модульне, інтеграційне та навантажувальне. Були описані результати цих тестів. Окрім цього було описано, як правильно розгорнути програмний продукт. Описані вимоги, які необхідні для коректного функціонування додатку. Також надано пояснення про те, як правильно розгорнути програмний продукт на Android пристроєві та, як здійснювати керування серверною частиною додатку. Окрім

того, для правильного використовування була наведена детальна інструкція користувача, що описує послідовність дій виконання кожної з операцій можливих в додатку.

РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1. Забезпечення електробезпеки користувачів ПК.

Приміщення із робочими місцями користувачів комп'ютерів для забезпечення електробезпеки обладнання, а також для захисту від ураження електричним струмом самих користувачів ПК повинні мати достатні технічні засоби захисту відповідно до ГОСТ 12.1.009-76, НПАОП 40.1-1.07-01 "Правила експлуатації електрозахисних засобів", НПАОП 40.1-1.21-98 "Правила безпечної експлуатації електроустановок споживачів", НПАОП 40.1-1.32-01 "Правила будови електроустановок. Електрообладнання спеціальних установок".

З метою запобігання ушкодженням, що можуть статися через ураження електричним струмом, загоряння, коротке замикання тощо, розроблено загальний стандарт безпеки IEC 950. Загальним стандартом електробезпечності для країн Європейської співдружності є Cemark.

Під час проектування систем електропостачання, монтажу силового електрообладнання та електричного освітлення будівель та приміщень для ПЕОМ необхідно дотримуватись вимог вищеназваних нормативно-правових актів, а також СН 357-77 "Инструкция по проектированию силового осветительного оборудования промышленных предприятий", затверджених Держбудом СРСР, ГОСТу 12.1.006, ГОСТу 12.1.030 "ССБТ. Электробезопасность. Защитное заземление, зануление", ГОСТу 12.1.019 "ССБТ. Электробезопасность. Общие требования и номенклатура видов защиты", ГОСТу 12.1.045, ВСН 59-88 Держкомархітектури СРСР "Электрооборудование жилых и общественных зданий. Нормы проектирования", Правил пожежної безпеки в Україні, ДСанПіН 3.3.2.007-98, розділів СНиП, що стосуються штучного освітлення і електротехнічних пристроїв, та вимог нормативно-технічної і експлуатаційної документації заводу-виробника ПЕОМ.

ЕОМ, периферійні пристрої ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ, інше устаткування (апарати управління, контрольно-вимірювальні прилади, світильники тощо), електропроводи та кабелі за виконанням та ступенем захисту мають відповідати класу зони за ПУЕ, мати апаратуру захисту від струму короткого замикання та інших аварійних режимів.

Під час монтажу та експлуатації ліній електромережі необхідно повністю унеможливити виникнення електричного джерела загоряння внаслідок короткого замикання та перевантаження проводів, обмежувати застосування проводів з легкозаймистою ізоляцією і, за можливості, перейти на негорючу ізоляцію.

Лінія електромережі для живлення ЕОМ, периферійних пристроїв ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ виконується як окрема групова трипровідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів.

Використання нульового робочого провідника як нульового захисного провідника забороняється. Нульовий захисний провід прокладається від стійки групового розподільчого щита, розподільчого пункту до розеток живлення. Не допускається підключення на щиті до одного контактного затискача нульового робочого та нульового захисного провідників. Площа перерізу нульового робочого та нульового захисного провідника в груповій трипровідній мережі повинна бути не менше площі перерізу фазового провідника.

Усі провідники повинні відповідати номінальним параметрам мережі та навантаження, умовам навколишнього середовища, умовам розподілу провідників, температурному режиму та типам апаратури захисту, вимогам ПУЕ.

У приміщенні, де одночасно експлуатується або обслуговується більше п'яти персональних ЕОМ, на помітному та доступному місці встановлюється

аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення.

ПЕОМ, периферійні пристрої ПЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ повинні підключатися до електромережі тільки з допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення. Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників повинні мати спеціальні контакти для підключення нульового захисного провідника. Конструкція їх має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше ніж приєднання фазового та нульового робочого провідників. Порядок роз'єднання при відключенні має бути зворотним. Необхідно унеможливити з'єднання контактів фазових провідників з контактами нульового захисного провідника.

Неприпустимим є підключення ПЕОМ та периферійних пристроїв ПЕОМ до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв.

Електромережі штепсельних з'єднань та електророзеток для живлення ПЕОМ, периферійних пристроїв слід виконувати за магістральною схемою, по 3…6 з'єднань або електророзеток в одному колі. Штепсельні з'єднання та електророзетки для напруги 12 В та 36 В за своєю конструкцією повинні відрізнятися від штепсельних з'єднань для напруги 127 В та 220 В і мають бути пофарбовані в колір, який візуально значно відрізняється від кольору штепсельних з'єднань, розрахованих на напругу 127 В та 220 В.

Індивідуальні та групові штепсельні з'єднання та електророзетки необхідно монтувати на негорючих або важкогорючих пластинах з урахуванням вимог ПУЕ та Правил пожежної безпеки в Україні.

Електромережу штепсельних розеток для живлення ПЕОМ, периферійних пристроїв ПЕОМ при розташуванні їх уздовж стін приміщення прокладають по підлозі поряд зі стінами приміщення, як правило, в металевих трубах і гнучких металевих рукавах з відводами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання.

При розташуванні в приміщенні за його периметром до 5 ПЕОМ, використанні трипровідникового захищеного проводу або кабелю в оболонці з негорючого або важкогорючого матеріалу дозволяється прокладання їх без металевих труб та гнучких металевих рукавів.

3.2 Вимоги щодо охорони праці при роботі з комп'ютерами.

Через масовий характер робіт, що виконуються працівниками за допомогою комп'ютера, законодавством України чітко врегульовано норми та вимоги до використання комп'ютерної техніки на підприємстві, безпосередньо й охорона праці при роботі з комп'ютером.

Вимоги до приміщення

Приміщення, в яких планується установка та подальша робота з комп'ютером, повинні відповідати проектній документації будинку, погодженій з уповноваженими державними органами. Крім того, роботодавець повинен враховувати санітарні нормативи освітлення, вимоги до параметрів мікроклімату (температура, відносна вологість), ступеня і сили вібрації, звукового шуму і вогнестійкості приміщення, а також характеристики електромагнітного, ультрафіолетового та інфрачервоного полів. Конкретні показники зазначених санітарних норм див. в Державних санітарих правилах і нормах роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98, затверджених Постановою Головного державного санітарного лікаря України №7 від 10 грудня 1998 року.

Правила поширюються на умови й організацію праці при роботі з візуальними дисплейними терміналами (ВДТ) усіх типів вітчизняного та зарубіжного виробництва на основі електронно-променевих трубок (ЕПТ), що використовуються в електронно-обчислювальних машинах (ЕОМ) колективного використання та персональних ЕОМ (ПЕОМ). Так, наприклад, роботодавцю заборонено установлювати комп'ютери в приміщеннях, розташованих у підвалах будинків.

Для уникнення можливих аварій та замикань, поряд з приміщеннями, де вестиметься робота з комп'ютером (над чи під ними), також не дозволяється проведення робіт, що потребують здійснення надмірно вологих технологічних процесів. Відповідне приміщення повинно бути укомплектоване системами центрального або індивідуального опалення, кондиціонування чи вентиляції повітря. Але при установці зазначених систем, необхідно переконатись, що батареї опалення, водопровідні труби, вентиляційні кабелі тощо, надійно сховані під захисними щитками, які перешкоджатимуть можливому потраплянню робітника під напругу.

У кожній кімнаті, де обладнуватимуться робочі місця співробітників, що працюватимуть на комп'ютері, повинні бути наявні елементи природного та штучного освітлення. При цьому, на вікнах слід встановити легко регульовані жалюзі чи штори, які дозволять працівникам коригувати рівень освітлення в приміщенні. Бажано розмістити комп'ютери в кімнаті таким чином, щоб світло потрапляло на екрани моніторів з півдня чи північного сходу. З метою досягнення максимального рівня безпечності і охорони праці при роботі з комп'ютером, виробничі приміщення необхідно обладнати аптечками першої медичної допомоги, системами автоматичної пожежної сигналізації і вогнегасниками. В приміщенні, в якому разом працюють 5 або більше комп'ютерів, на видимому місці установлюється службовий вимикач, який у разі потреби дозволить повністю відключити електричне живлення кімнати.

Вимоги до особистого робочого місця програміста

Роботодавець, який використовує найману працю робітників, повинен забезпечити відповідність їхніх робочих місць комфортним та безпечним умовам. Розмір одного робочого місця має становити не менше 6 квадратних метрів. При необхідності, суміжні робочі місця співробітників, що працюють з комп'ютером, слід розділити перегородками висотою до 2 метрів. При визначенні достатнього розміру приміщення і робочого місця на одну особу необхідно додатково враховувати шафи, сейфи, тумби або інші предмети меблів чи обладнання, які знаходяться в кімнаті. На столі працівника можливо розмістити допоміжні для роботи пристрої (принтери, колонки, сканери), а

також місця для зберігання документів, за умови, що це не обмежуватиме видимість екрану і не заважатиме працівнику. У разі надмірного шуму чи вібрації технічного обладнання, роботодавець повинен забезпечити працівників антивібраційними килимками. Робочий стілець співробітника має бути підйомно-поворотним, легко регульованим за висотою та забезпечувати належну підтримку та зручне положення спини і хребта особи. Щодня необхідно проводити вологе прибирання приміщення, та очищати робоче місце та безпосередньо монітор комп'ютера від запиленості.

На підприємстві забороняється: проводити ремонт та технічне обслуговування комп'ютера за робочим місцем працівника; самочинно ремонтувати або намагатись здійснити технічне налагодження комп'ютера без залучення компетентних спеціалістів; складувати на робочому місці зайві документи, деталі та предмети, що не потрібні для роботи; використовувати монітори з нечітким зображенням та монітори, у яких наявні поламки екрану; працювати з матричним принтером без антивібраційного покриття та зі знятою кришкою. Допускати до роботи осіб, які не пройшли затверджений на підприємстві курс охорони праці для роботи з комп'ютером, не дозволяється.

Законодавство:

– Наказ Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду «Про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин» від 26.03.2010 № 65;

– Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПіН 3.3.2.007-98, затверджені постановою Головного державного санітарного лікаря України від 10.12.1998 № 7;

– Примірну інструкцію з охорони праці під час експлуатації електронно-обчислювальних машин, затверджену наказом Міністерства доходів і зборів України від 05.09.2013 № 443.

3.3 Висновок до третього розділу

В даному розділі розглянуто питання щодо безпеки життєдіяльності.

Сучасний розвиток технічного та технологічного стану виробництва передбачає постійну автоматизацію та оптимізацію виробничих процесів. Сьогодні, напевно, важко уявити компанію, господарська діяльність в якій здійснювалась би без використання комп'ютерної техніки тому дотримання правил електробезпеки користувачів ПК є дуже важливим.

Розглянуто вимоги щодо охорони праці при роботі з комп'ютерами, які впливають на роботу персоналу, їхню безпеку під час роботи. Тому потрібно приділяти надзвичайно багато уваги безпеці персоналу, щоб таких факторів було якнайменше. Розглянуто умови при яких працівник досягає високої ефективності і при цьому отримує якнайменше шкоди для свого здоров'я. Отже робимо висновок, що слід дотримуватись всіх правил охорони праці, щоб зберегти своє здоров'я як для себе, так і для свої колег.

ВИСНОВКИ

В ході виконання роботи було розглянуто основні аспекти роботи системи обміну повідомленнями. Здійснено огляд предметної області та описано основні бізнес-процеси, що здійснюються в додатку. Було проведено огляд трьох найпопулярніших систем обміну повідомленнями існуючих на ринку. Проведено їхній аналіз та виявлено їхні позитивні та негативні риси. Була розроблена специфікація вимог, наведена діаграма варіантів використання та описано функціональність програмного забезпечення.

Було розроблено архітектуру програмної системи та описано застосування архітектурного шаблону MVC для розроблюваної системи. Наведено діаграми активності та послідовності для процесів, що відбуваються під час використання програмного забезпечення. Під час проектування бази даних для системи обміну повідомленнями була розроблена та наведена ER-діаграма, де були продемонстровані таблиці та зв’язки між ними.

Описані засоби, що будуть використовуватись під час розробки програмної системи, а саме мова програмування Java та мобільна платформа Firebase, що допомагає швидко розробляти якісні додатки.

Було здійснене тестування для перевірки коректної роботи окремих модулів програми з використанням бібліотеки JUnit. За допомогою фреймворку Espresso для тестування здійснено тестування інтерфейсу користувача додатку.

Здійснено написання інструкції користувача та інструкції по розгортанню програмного продукту на операційній системі Android.

ПЕРЕЛІК ДЖЕРЕЛ

1. Android (operating system) [Електронний ресурс] // Wikipedia – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Android_(operating_system).

2. Створення графічного інтерфейсу: [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/java/android/1.3.php

3. Створення Firebase: [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/ 9. Діаграма варіантів використання : [Електронний ресурс] – Режим доступу до ресурсу: http://khpiiip.mipk.kharkiv.edu/library/case/leon/gl4/gl4.html

4. Основи UML: [Електронний ресурс] – Режим доступу до ресурсу: https://pro-prof.com/archives/2594 5. Системное программирование [Електронний ресурс] // Подкаст "Podlodka". – Режим доступу до ресурсу: https://soundcloud.com/podlodka/podlodka-86-sistemnoe-programmirovanie.

5. Application Sandbox [Електронний ресурс] // Source Android – Режим доступу до ресурсу: https://source.android.com/security/app-sandbox.

6. Data and file storage overview [Електронний ресурс] // Android Developer – Режим доступу до ресурсу: https://developer.android.com/guide/topics/data/data-storage#filesInternal.

7. Man-in-the-Disk: Android Apps Exposed via External Storage [Електронний ресурс] // CHECK POINT RESEARCH. . – Режим доступу до ресурсу: https://research.checkpoint.com/androids-man-in-the-disk/.

8. Man-in-the-Disk: A New Attack Surface for Android Apps [Електронний ресурс] // Check Point Blog. – Режим доступу до ресурсу: 71 https://blog.checkpoint.com/2018/08/12/man-in-the-disk-a-new-attacksurface-for-android-apps/.

9. DEF CON 26 - Slava Makkaveev - Man In The Disk [Електронний ресурс] // DEFCONConference YouTube channel. – Режим доступу до ресурсу: https://www.youtube.com/watch?v=vvfs0u1or3M.

10. Конспект лекцій з дисципліни «Програмування для мобільних пристроїв» для студентів денної форми навчання спеціальності 126

«Інформаційні системи та технології» / Укладачі: Готович В.А., Михайлович Т.В. – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2020. – 216 с.

11.Fortnite Installer downloads are vulnerable to hijacking [Електронний ресурс] // Google Issue Tracker. – 2018. – Режим доступу до ресурсу: https://issuetracker.google.com/issues/112630336

12.MacOS X GateKeeper Bypass [Електронний ресурс] // Filippo Cavallarin. – 2019. – Режим доступу до ресурсу: https://www.fcvl.net/vulnerabilities/macosx-gatekeeper-bypass.

13.Android Q privacy change: Scoped storage [Електронний ресурс] // Developers Android – Режим доступу до ресурсу: https://developer.android.com/preview/privacy/scoped-storage.

14. Створення месенджерів для навчального закладу: [Електронний ресурс] – Режим доступу до ресурсу: https://vrc.rv.ua/case_study/vetmarketing/

15. Пошук цільової аудиторії: [Електронний ресурс] – Режим доступу до ресурсу: https://creativesmm.com.ua/jak-znajjtu-svoju-cilovuauditoriju/