Державний вищий навчальний заклад
“Прикарпатський національний університет імені Василя Стефаника”
Кафедра інформаційних технологій

УДК 004

ДИПЛОМНИЙ ПРОЕКТ

Тема: Android додаток - ПНУ соціальна мережа

Спеціальність: 121 Інженерія програмного забезпечення

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДП.ІПЗ-19.ПЗ
(позначення)

Рецензент
викл. Пікуляк М.В.
(посада) (підпис) (дата) (розшифровка підпису)

Студент
ІПЗ-41 Попович В.В.
(шифр групи) (підпис) (дата) (розшифровка підпису)

Нормоконтролер
викл. Пікуляк М.В.
(посада) (підпис) (дата) (розшифровка підпису)

Керівник дипломного проекту
доц. Ткачук В.М.
(посада) (підпис) (дата) (розшифровка підпису)

Допускається до захисту

Завідувач кафедри

зав.кафедри Козленко М.І.
(посада) (підпис) (дата) (розшифровка підпису)

2020

Державний вищий навчальний заклад
«Прикарпатський національний університет імені Василя Стефаника»
Факультет математики та інформатики Кафедра інформаційних технологій
Спеціальність Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Завідувач кафедри Козленко М.І.
студенту Поповичу Вадиму Віталійовичу
„_____” _____ 20___р.

ЗАВДАННЯ
НА ВИКОНАННЯ ДИПЛОМНОГО ПРОЕКТУ

Студенту Поповичу Вадиму Віталійовичу
(прізвище, ім’я, по батькові студента)

1.Тема проекту Android додаток - ПНУ соціальна мережа
затверджена розпорядженням по факультету математики та інформатики від „25” жовтня 2019 р. №7

2. Термін здачі студентом закінченого проекту 22 травня 2020 р.

3. Вихідні дані до дипломного проекту стандарт кафедри інформаційних технологій ПНУ, технології розробки – Android, Java, Firebase , Apache software license.

4. Зміст пояснювальної записки (перелік питань, що їх належить опрацювати)
1. Аналіз предметної області, постановка задачі
2. Інструменти, ресурси та архітектура розробки програми
3. Розробка програмного забезпечення
4. Економічна ефективність хмарних сервісів

5. Перелік графічного матеріалу (з точним забезпеченням обов’язкових креслень): 1. Титульний аркуш; 2. Мета та актуальність роботи, завдання для проектування; 3. Інструменти та засоби реалізації; 4. Архітектура дерева даних програми; 5. Архітектура класів; 6. Початкова сторінка та вхід; 7. Реєстрація за допомогою студентського квитка; 8. Головна сторінка студента; 9. Список користувачів що оцінили пост, друзі та отримані заявки в друзі; 10. Надіслані заявки в друзі, пошук користувачів(всі користувачі); 11. Підписники, читачі та профіль користувача; 12. Месенджер та діалоги повідомлень; 13. Економічна ефективність хмарних сервісів; 14. Висновки

6. Дата видачі завдання 11.09.2019

Керівник _____ Ткачук В.М.
(підпис) (розшифровка підпису)

Завдання прийняв до виконання _____ Попович В.В.
(підпис) (розшифровка підпису)

КАЛЕНДАРНИЙ ПЛАН

Номер і назва етапів дипломного проекту	Термін виконання етапів проекту	Примітка
1. Аналіз предметної області, постановка задачі	02.12.2019	
2. Інструменти, ресурси та архітектура розробки програми	15.02.2020	
3. Розробка програмного забезпечення	17.04.2020	
4. Економічна ефективність хмарних сервісів	11.05.2020	
5. Оформлення пояснювальної записки	18.05.2020	

Студент _____ Попович В.В.
(підпис) (розшифровка підпису)

Керівник проекту _____ Ткачук В.М.
(підпис) (розшифровка підпису)

РЕФЕРАТ

Пояснювальна записка: 93 сторінки, 54 рисунки, 1 таблиця, 28 джерел, 6 додатків.

КЛЮЧОВІ СЛОВА: ANDROID STUDIO, JAVA, FIREBASE, REALTIME DATABASE, АНАЛІТИКА, АНДРОЇД, БАЗА ДАНИХ, БІБЛІОТЕКА, ВУЗОЛ ДАНИХ, ДЕРЕВО ДАНИХ, ДІАГРАМА, МЕСЕНДЖЕР, МОВА ПРОГРАМВАННЯ, СОЦІАЛЬНА МЕРЕЖА, ШАБЛОН.

Об`єктом дослідження є андроїд додаток - ПНУ соціальна мережа.

Мета роботи розробити андроїд додаток - ПНУ соціальна мережа, який дозволяє створювати профілі тільки для студентів ПНУ. Додаток повинен забезпечувати основними можливостями соц мереж, а саме створювати списки: друзів, підписників, читачів, відвідувати профілі інших користувачів, обмінюватись повідомленнями, створювати записи на різних профілях із можливостю їх оцінювати, залишати коментар та зберігати запис на своєму профілі, можливість створення опитувань та інше.

Стислий опис тексту пояснювальної записки: У даному дипломному проекті описані основні поняття та концепції, які необхідні для розробки повноцінного андроїд додатку з функціонуванням віддалених серверів, хмарної платформи та їх синхронізація з локальною базою. Описано засоби та інструменти розробки, зображено діаграми класів, користувальницьких інтерфейсів та їх реалізації. Наведена детальна аналітика впровадження та розвитку додатку.

ABSTRACT

Explanatory note: 93 pages, 54 figures, 1 tables, 28 sources, 6 applications.

KEYWORDS: ANDROID STUDIO, JAVA, FIREBASE, REALTIME DATABASE, ANALYTICS, ANDROID, DATABASE, LIBRARY, DATA NODE, DATA TREE, DIAGRAM, MESSENGER, PROGRAMING LANGUAGE, SOCIAL NETWORK, LAYOUT.

The object of study is android application PNU social network.

The goal of the work is to develop an android application - PNU social network, which allows you to create profiles only for students of PNU. The application should provide the main features of social networks, namely to create lists: friends, subscribers, readers, visit profiles of other users, exchange messages, create entries on various profiles with the ability to rate them, leave a comment and save the record on your profile, the ability to create surveys and other.

Brief description of the explanatory note text: This thesis describes the basic concepts and concepts that are necessary for the development of a full-fledged android application with the operation of remote servers, cloud platform, their synchronization with the local database. Described means and development tools, diagrams of classes, user interfaces and their implementation are shown. A detailed analysis of the implementation and development of the application is given.

Зміст

ВСТУП 9
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, ПОСТАНОВКА ЗАДАЧІ 11
1.1 Аналіз предметної області 11
1.2 Аналоги проекту 14
1.3 Постановка задачі 19
2 ІНСТРУМЕНТИ, РЕСУРСИ ТА АРХІТЕКТУРА РОЗРОБКИ ПРОГРАМИ .. 21
2.1 Програмні засоби та середовище розробки 21
2.2 Можливості та ресурси платформи Firebase 29
2.2.1 Firebase Analytics 31
2.2.2 Firebase Authentication 31
2.2.3 Firebase Realtime Database 33
2.2.4 Firebase Storage 40
2.3 Моделювання архітектури діаграми сторінок 40
2.4 Моделювання архітектури діаграми класів 42
3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 44
3.1 Моделювання шаблонів та дизайну сторінок 44
3.2 Реалізація та демонстрація функціоналу 57
3.2.1 Асинхронні слухачі дерева даних 76
3.2.2 Реалізація списків та стрічок подій 77
4 ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ ХМАРНИХ СЕРВІСІВ 78
4.1 Бізнес план реалізації та поширення проекту 78
ВИСНОВКИ 91
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 92
ДОДАТОК А 95
ДОДАТОК Б 96
ДОДАТОК В 97
ДОДАТОК Г 98

ДОДАТОК Д 99
ДОДАТОК Е 102

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Баг	жаргонізм, що означає помилку, ваду або дефект в програмі
Емодзі	особлива мова ідеограм і смайлів, які широко використовують в електронних повідомленнях
ІД студентського квитка	унікальний номер на особисті картці студента
ОС	операційна система
ПЗ	програмне забезпечення
ПНУ	Прикарпатський національний університет
XML	Extensible Markup Language
ORM	Object-Relational Mapping
Firebase	платформа розробки мобільних та веб- застосунків
Storage	сховище файлів

Вступ

В наші дні активно розвиваються ґаджети, зокрема, смартфони, планшети без яких важко уявити сучасне суспільство. Вони служать не тільки як засоби для розваг та комунікацій, але й беруть активну участь в повсякденній роботі та навчанні. Найбільш популярним програмним забезпеченням, яким користуються люди в своїх ґаджетах є месенджери та соціальні мережі. Вони об'єднують велику кількість людей та дозволяють швидко зв’язатись з вибраним користувачем. Робити розсилку повідомлень в спеціально створені групи, переглядати профіль користувача, слідкувати за останніми подіями, і багато інших функцій. При цьому вони не потребують значних ресурсів на девайсі, та є надзвичайно швидкими. Основними гігантами подібного програмного забезпечення для мобільних девайсів є: Facebook – понад 5 мільярдів завантажень, WhatsApp– понад 5 мільярдів завантажень ,Instagram - понад 1 мільярд завантажень. Тенденції розвитку такого роду програмного забезпечення, щороку зростають удосконалюючи старі і розробляючи нові можливості додатків.

Незважаючи на те, що існує безліч подібних додатків, жоден не може реалізовувати вузько спеціалізовані задачі і функції, які потрібні студентам та викладачам, наприклад онлайн журнал оцінок чи онлайн розклад, інформацію про навчання. Маючи додаток який містив би всіх студентів ПНУ, викладачі мали б змогу швидко надсилати студентам важливу інформацію про ті чи інші події, завдання, оголошення.

Завдання дипломного проекту – розробити та реалізувати додаток соціальна мережа для Прикарпатського національного університету, в якому будуть зареєстровані тільки студенти та викладачі даного закладу. Додаток повинен реєструвати студентів по серії та номеру студентського, оскільки кожен студент має свій студентський квиток та унікальний номер. Це дозволить ідентифікувати та реєструвати тільки студентів ПНУ. Додаток повинен мати систему входу та реєстрації, власний профіль, надавати можливість відвідувати

інші профілі, можливість надсилати повідомлення, містити інформацію про оцінки та успішність, список друзів, підписників, читачів, заявок в друзі та класифікацію по групах. Можливість змінювати дані у профілі, передбачати різні ситуації та вміння їх обробляти, зокрема, додаток повинен вміти синхронізувати дані з локальною базою і відображати їх разі відсутності підключення до інтернету.

Актуальність дипломного проекту заключається в створенні соціальної мережі для операційної системи «Android», яка є найпопулярнішою операційною системою. У березні 2017 року ОС Android стала найпопулярнішою системою, з якої заходили у інтернет. Так 37,94 % користувачів заходили в інтернет із Android. Близько на 85% смартфонів у світі поставлена дана операційна система і платформа для мобільних телефонів та планшетних комп'ютерів. Створена компанією Google на базі ядра Linux. Основним елементом цієї ОС є реалізація Dalvik, віртуальної машини “JVM”. Все програмне забезпечення ОС Android і застосування спираються на реалізацію Java. Платформа легко пристосовується для двовимірної і тривимірної графіки, розроблених на основі OpenGL специфікації, традиційних інструментаріїв для смартфонів.

Android показує себе набагато краще одного зі своїх конкурентів, Apple, iOS в ряді особливостей та можливостей, таких як веб-серфінг, інтеграція з сервісами Google та інших. Також Android на відміну від iOS, є відкритою платформою, що дозволяє реалізувати на ній більше функціоналу. Кожного року Android випускає нову версію операційної системи, що ставиться на більшість найновіших смартфонів на ринку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області

Розробка додатків для мобільних пристроїв ділиться на три основні платформи Android, Windows і iOS. Розподілення ринку нових мобільних пристроїв у 2018 році, можна визначити на прикладі США, за рік частки Android і iOS складають близько 59% і 39% відповідно. У Китаї iOS продовжує зрозстати, зараз у них близько 22% проти 12% роком раніше, а частка Android знизилася з 87% до 77%. В загальгому розподіл часток між основними ОС практично не змінилося. У Великобританії, Італії та Японії Android за рік кілька наростила позиції на 1-5 проценів iOS збільшила частку в Німеччині Італії, Іспанії, Франції та Австралії. Також можна відзначити, що найбільша частка Windows Mobile зберігається в США і Італії, де система займає 1,5% і 1,4% відповідно. Але цікавіше те, що в Китаї та Австралії частка цієї ОС зросла до 0,5% і 0,8% відповідно [1].

Слід зазначити досить цікавий факт - в 2015 році компанія Microsoft оновила середовище розробки (IDE) Visual Studio 2015, в якій підтримується можливість створення додатків з одного коду для основних мобільних платформ. А у лютому 2016 року було оголошено про придбання корпорацією Microsoft компанії Xamarin. Яка реалізувала кросс-платформену середу Mono, що користується популярною Common Language Infrastructure (CLI) і .NET від Microsoft, для створення мобільних додатків для Android і iOS. Вважаю, що рух Microsoft у напрямку розвитку інструментів для створення мобільних додатків для цих платформ буде мати досить інтенсивний вплив на ринок розробки мобільних додатків і рівень популярності інструментів розробки. Завдяки тому факту, що у перше, Microsoft безпосередньо конкурує з основними мовами розробки Java і Objective-C [2].

Якщо аналізувати зручність розробки програм для різних мобільних платформ, можна зробити висновок, що по цьому критерію одноголосно виграє платформа Android. Завдяки відносної відкритості і простоті використання та модифікації. iOS по цьому критерію посягає, мабуть, останню позицію, завдяки досить замкнутим обмеженням щодо середи розробки (XCode) і апаратних засобів лише ОС компанії Apple. Платформу Windows досить важко оцінювати у зв'язку з останніми змінами у підходах до розробки мобільних додатків і інструментах розробки. Досить важко оцінити перспективи розвитку і просування мобільної платформи Windows. Вважаю, що в часи з 2008-2009 років компанія не приділяла уваги достатньої для розвитку мобільної платформи Windows. Тому, зміни на ринку мобільних платформ в найближчому майбутньому можуть мати досить інтересні наслідки і напрямки розвитку. В цілому розробка для мобільних платформ не підійде кожному до душі, оскільки включає в себе розробку як дизайну так і програмну частину, і важко розділити ці дві складові для окремих груп розробників. Також недоліком є довгий час відгуку на результат, тобто для того щоб подивитись на результат своєї внесеної зміни в проект, необхідно зібрати проект однією з систем автоматичної збірки Gradle або Maven, запустити емулятор ОС, емулятору необхідно виділити від, як правило 512 Мб до 2Гб ОЗУ, залежно від мобільного пристрою який буде емульованим. Це займає певний час, від декількох секунд до декількох хвилин, час збірки та запуску емулятора напряму залежить від характеристик пристрою на якому запускається проект. Якщо пристрій немає достатніх ресурсів, то можуть виникнути проблеми, зокрема довгий час збірки проекту або зупинка ОС.

Найбільш популярною на цей час мобільною платформою є ОС Android. Яка, відрізняється від інших, ефективністю функціонування, простотою використання, розповсюдженістю. Головними напрямками розробки під Android як в Україні, так і у світі, є ігрова індустрія. Інша досить велика індустрія використання та розробки під Android - додатки і програмні комплекси для

компаній і бізнесу. Такі висновки зроблені на основі досвіду використання найбільших й найпопулярніших сервісів для фрілансерів - Upwork.com і Freelance.com, на яких розміщуються замовлення на створення різних видів ПЗ. Основними лідерами на ринку розробки ПЗ під мобільні пристрої є дві групи виконавців - фріланцери і компанії, що здійснюють розробку ПЗ. Якщо аналізувати різні географічні регіони представників різних країн у вищезазначених системах, то лідерами з кількості виконавців є Індія, Китай, країни Азії і арабські країни (Пакистан, наприклад). Але, виконавці з Індії, Азії і арабських країн, мають досить неоднозначну і погану репутацію у зв'язку з низькою якістю виконання роботи. Про те вони займають велику нішу в даній галузі, оскільки пропонують найнижчий рівень цін на розробку програмних продуктів [3].

У проведеному аналізу мобільних додатків під ОС Android складається враження, що більша частина об'єму розробки спрямована на зовнішній ринок, що в першу чергу пов'язане з відсутністю постійного об'єму заказів на внутрішньому ринку. Це пов'язано в свою чергу, з складностями при продажах на внутрішньому ринку України платного ПЗ. З іншої сторони корпоративний сектор країни є досить консервативним і у більшості випадків, не націлений на використання сучасних інформаційних технологій. Тому вважаю розробку програмного забезпечення для внутрішнього ринку України досить важливою і цікавою і складною задачею, що буде просувати ринок внутрішньої розробки України до світлого майбутнього.

Цифри які були наведені вище у вступі про кількість завантажень на Google Play Market найбільш популярних соц мереж, свідчать не тільки про доступність інформації користувачів та швидкість зв’язку з іншими. Але і про нові тенденції розвитку спілкування . Зокрема перевагою спілкування через месенджер, є можливість передачі різної інформації: графічної, аудіо, відео, є можливість по іншому передавати свої емоції та враження через так звані «Емодзі» – це особлива мова смайлів, знаків символів, ідеограм. Меседжери

створили новий тип спілкування, який не поступається традиційному спілкуванню. Будь-хто може відправляти смс, не відриваючись від справ і витрачаючи всього на це декілька секунд. Також перевагою є збереження повідомлення та можливість повернутися в будь-який момент до старішого повідомлення та знайти необхідну інформацію навіть без наявності інтернету завдяки синхронізації додатків з локальною базою, зберігаючи дані безпосередньо на смартфоні. Це також призводить до неактуальності телефонних дзвінків і витіснення їх з ужитку повідомленнями за допомогою месенджерів.

1.2 Аналоги проекту

Результат вивчення мобільних додатків у найбільшій системі розповсюдження Android додатків Google Play 2019-2020 роках, дає підстави стверджувати, що саме такого додатку (ПНУ соціальна мережа) або соціальних мереж для інших університетів України взагалі не існує [4]. Найбільш схожий за ідеєю додаток “Athabasca University Students” існує для Сполучених Штатів Америки, більше 5000 завантажень з Google Play [5]. Цей додаток розробила компанія "Ready Education", додаток найбільш серед інших знайдених, схожий на ідею мого дипломного проекту та відображає його основний функціонал та суть. Тільки даний проект є значно маштабнішим оскільки його розробляла ціла компанія, а не одна людина. Проект складається з інтернет-сайту і мобільного застосування для Android. Основні функції якими володіє додаток:

- classes – можливість керувати своїми групами, дозволяє створювати завдання та нагадування;
- events - події, що відбуваються на території університету;
- campus services – послуги що пропонуються в університеті;

- groups & clubs - про клуби в університеті і про те, як залучати до них активних користувачів;
- campus map - напрямки до класів, подій та департаментів;

Додаток для студентів приносить послуги і дає змогу з'єднатися з одногрупниками та друзями. Ви можете отримати доступ до подій, календарів, контактів, карт тощо.

Додаток забезпечує обмін інформацією між студентами і викладацьким складом, дозволяє відстежувати зміни в розкладі, дає можливість студентам бачити актуальні оголошення від роботодавців, мати доступ до методичних вказівок, статей, наукових робіт, що видається викладачами ВНЗ. Надає список друзів, постів, курсів. Система сповіщення забезпечує своєчасне інформування про всі події, що плануються в університеті. На наступних рисунках наведено сторінки до датку.

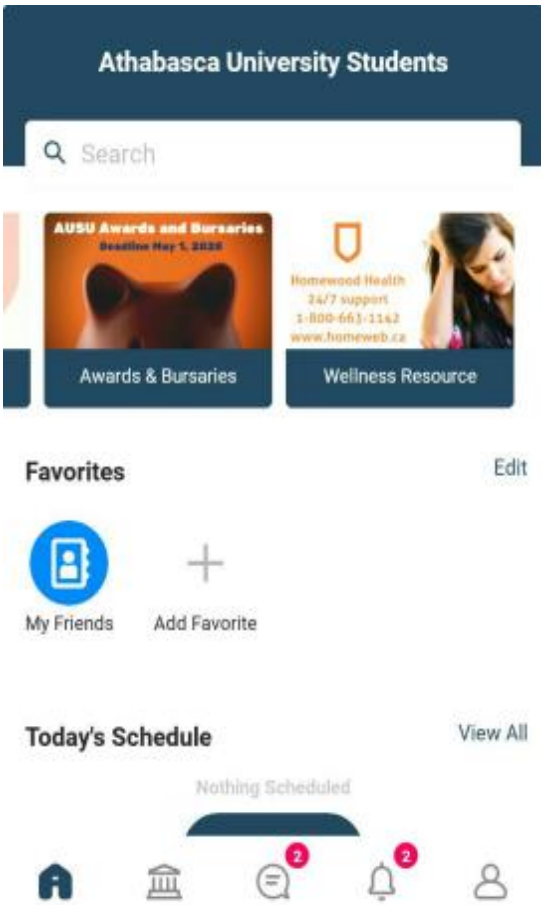


Рисунок 1.1 – Аналог дипломного проекту - ” Athabasca University Students ”, сторінка “Home page”

Сторінка наведена рис. 1.1, відображає домашню сторінку додатку, яка містить події університету, що допомомагають студентам дізнаватись останні події університету та брати активну участь у його розвитку. Списки фаворитів – можна створювати власні списки, наприклад друзів, читачів та інші. На сторінці відображається поточний розклад для студента на сьогоднішний день, огологення меню додатку та інше.

В своєму додатку планую реалізувати сторінку власного профілю, як на рис. 1.2, але з дещо зміненим виглядом та більшою інформаціює про студента. Додатково планується розміщувати записи на цій сторінкі під інформацією про корустувача.

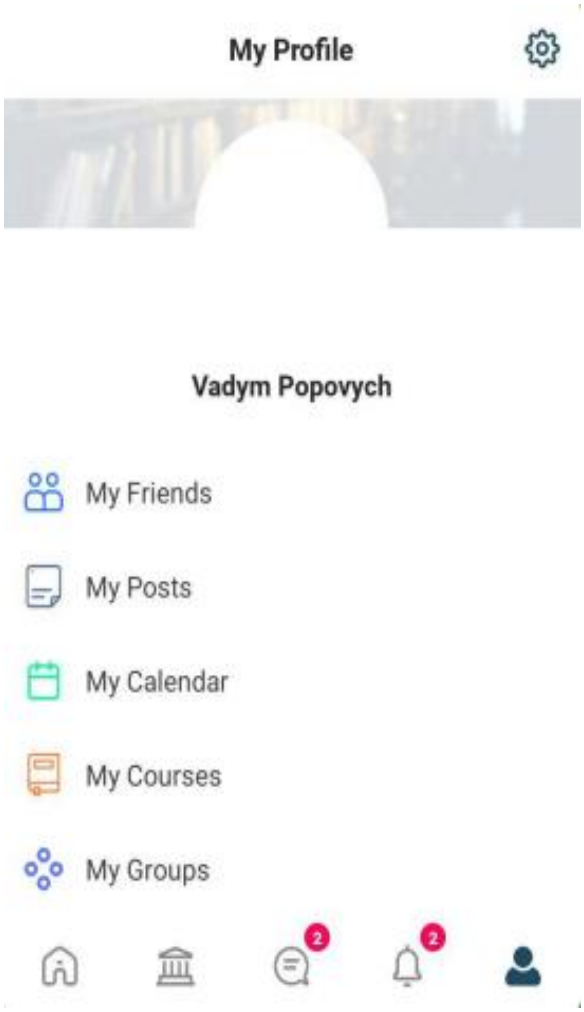


Рисунок 1.2 – Аналог дипломного проекту - ” Athabasca University Students ”, сторінка “My profile”

Як показано на рис. 1.2, додаток дозволяє як і типові соц мережі, створювати друзів, записи, а також групи , курси, тримати календар подій та інше. Приклад профілю іншого користувача зображено на рис. 1.3, він є дещо обмеженим в плані функціоналу, в додатку ПНУ – соц мережа планую розміщувати інформацію таку саму як і на сторінці “My profie” рис. 1.1, із списками друзів, підписників, читачів, записів користувача. Профіль користувача, буде заповнюватись на основі зазделегіть створеної бази даних та деяких наданих даних від студента, накриклад дані: факультет, група, прізвище, ім’я, рік вступу в університет, оцінки будуть братися з бази університету, а дані про місце проживання, фото профілю та інші студент буде надавати сам. Також будуть надаватись ще інші цікави факти з життя студента які допоможуть зробити його профіль більш цікавим та відвідуваним.

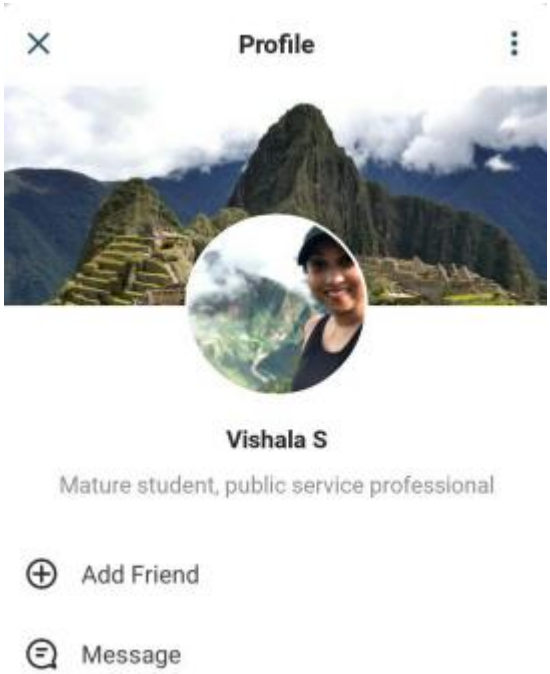


Рисунок 1.3 – Аналог дипломного проекту - ” Athabasca University Students ”, сторінка “Profile”

Також додаток Athabasca University Students володіє багатьма іншими можливостями, як наприклад месенджер - він є необхідною складовою будь-якої соц мережі та корпоративного додатку, оскільки дає змогу обмінюватись

повідомленнями між учасниками додатку в реальному часі, в дипломну проекті його теж буде реазіловано.

Велика частина додатків із галузі освіти мають свій розклад проведення занять. Аналогів онлайн розкладу є багато, в тому числі і ПНУ має свій онлайн розклад, тільки не на ОС Android, а на WEB платформі. Перевагою розміщення розкладу на Android додатку є краща його доступність, студенту не потрібно вводити назву групи, розклад на поточний день, для кожного студента відображається його власний розклад на відповідній сторінці.

Інші аналоги вже мають менше спільного з моїм дипломним проектом і вирішують спеціфічні завдання для окремих регіонів, наприклад додаток “Newcastle University”.

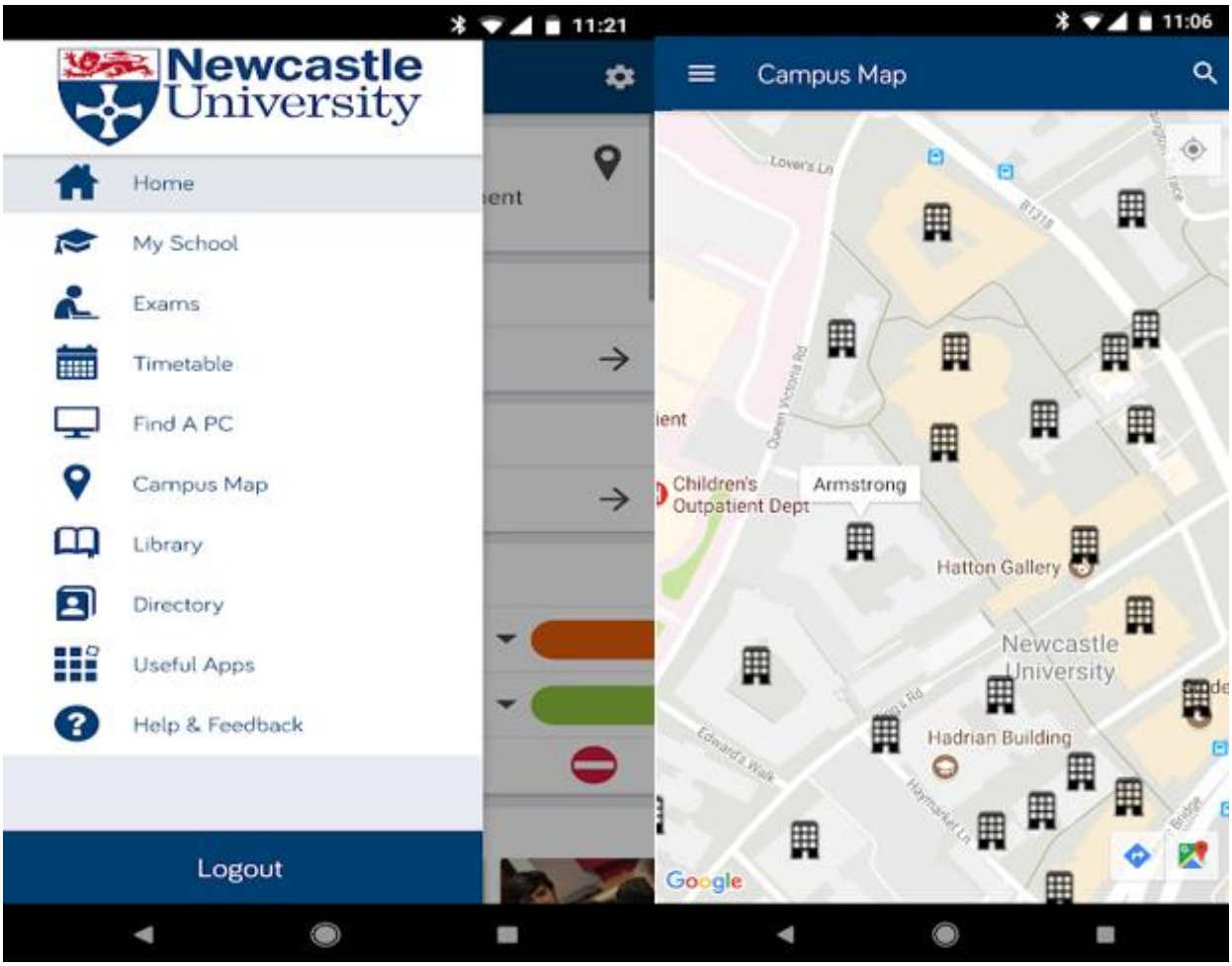


Рисунок 1.4 – Аналог дипломного проекту - ” Newcastle University”, сторінка “Home” та “Campus Map”

Даний додаток вирішує питання, які стосуються міста та університетів у Ньюкасл-апон-Тайн, що у знаходиться у Великобританії. Він відображає бібліотеки та аудиторії для занять в місті. Має більше 10 тисяч завантажень на Google Play. Його функції подібні до попереднього додатку, тому детально його розглядати не будемо.

Подібних іноземних додатків є біля кілька десятків, найбільш похожий на мою ідею це додаток - ” Athabasca University Students ”, розглянутий вище, всі додатки які тісно зв’язані з університетом це або розклад занять, як наприклад додаток “Студентський Журнал - Розклад занять” – Львівської політехніки, або інші додатки які допомагають вивчати предмети та різні дисципліни, але це вже інша галузь додатків для ОС Android.

1.3 Постановка задачі

В дипломному проекті необхідно спроектувати та реалізувати додаток соціальна мережа для Прикарпатського національного університету на базі мобільних пристроїв що підтримуються операційною системою Android. Додаток повинен містити всіх студентів та викладачів університету/факультету. Для реалізації необхідно виконати наступні пункти:

- реєстрацію студенти повинні здійснювати по студентському квитку та номеру телефона, додаток повинен містити тільки студентів та викладачів ПНУ;
- вхід в додаток по номеру телефона;
- розробити профілі для студентів;
- список друзів, підписників, читачів можливість пошуку по різним фільтрам;
- можливість відвідувати інші профілі, залишати записи, переглядати списки друзів, підписників читачів користувача;
- систему опитувань;

- спроектувати месенджер з підтримкою передачі файлів, емодзі, групових чатів, та виконання різних операцій копіювання, видалення повідомлень.
- для новин та подій університету розробити сторінку зі стрічкою новин з можливістю переходу на них по кліку для детальнішого ознайомлення.
- надати можливість вести успішність студента у вигляді слайдера або іншого компонента та виставляти бали в режимі онлайн.
- сторінку з налаштуваннями для зміни даних та настройки профілю
- меню, яке повинно давати змогу переміщатись між сторінками
- розробити систему постів для користувачів із можливість коментувати, давати відповідь на коментар(вкладений коментар)можливість ставити лайки, можливість надіслати запис на свій профіль із чужого профілю.

2 ІНСТРУМЕНТИ, РЕСУРСИ ТА АРХІТЕКТУРА РОЗРОБКИ ПРОГРАМИ

2.1 Програмні засоби та середовище розробки

Для розробки та проектування дипломного проекту були обрані наступні засоби: мова програмування Java, локальна база даних SQLite та ORM(високорівневий інтерфейс управління базою даних) – ROOM, мова розмітки XML, платформа для серверної частини та ресурсів Firebase, система керування версіями та збереження коду Git, середовище розробки Android Studio, систему автоматичної збірки Gradle. Нижче короткий опис по кожному із засобів.

Мову програмування Java було обрано для розробки дипломного проекту оскільки: по-перше вона ідеально вписується в стек технологій та засобів, які перелічені вище, оскільки на Java посилається більшість офіційної документації Google, згадана вище платформа Firebase з 2014 року теж належить компанії Google, тому вона добре працює з мовою програмування Java. По- друге Google надає безліч безкоштовних бібліотеки та керівництв підтримуваних Java. По-третє Java офіційна мова програмування, підтримувана середовищем розробки Android Studio.

Java – це об'єктно-орієнтована мова програмування, код-програми компілюється у байт-код, який інтерпретується віртуальною машиною “JVM” для конкретної платформи, саме це і є основним слоганом мови Java “Написано один раз, запускається будь-де”, власна віртуальна машина дозволяє запускати код на різних операційних системах. Java має багато запозичень з мови програмування C++, але основна мета була розробити мову програмування, яка б немала таких недолік як в C++, а саме низько рівневе управління пам’яттю в Java це доручено віртуальній машині, множинне наслідування, яке викликало конфліктні ситуації між членами класу та надлишкову важкість коду для сприйняття. За результатами опитування минулого року ресурсу “Stackoverflow”

Java ввійшов у топ 5 найпопулярніших мов програмування. Велика частина корпоративних додатків для операційної системи Android, написані на Java.

До переваг Java можна віднести:

- кроссплатформність - незалежність від операційної системи, код працює всюди однаково;
- об'єктність – будь-які дані можна перенести з реальної моделі із світу в об’єктну модель Java класу. Дані та функції ініціалізуються в класи об'єктів. Виключенням з об’єктів класів є примітиви (int, ,String, float тощо). Наявність примітивів дозволяє збільшувати швидкість виконання програми, оскільки в програмах часто виниках потреба в примітивах для збереження простих значень чи даних без створення об’єктів;
- безпека – Java надає багато засобів та рівнів безпеки, які роблять її однією з найбільш безпечних мов програмування. Java дозволяє перехоплювати ряд випадків чи ситуацій(Exception), коли програма виявляє неочікувані ситуації, наприклад: недопустимі операції над даними, вихід за межі масиву, читання з неправильної адреси, ввід некоректних даних користувачем, відсутність даних. Віртуальна машина дозволяє обробити подібні ситуації (виключення), уникнути краху системи та продовжити виконання програми за тим сценарієм який програміст описав у виключенні. До безпеки програми можна також віднести строгу типізацію даних, це дозволяє розміщувати дані в структурах даних лише певного типу, що призводить зменшення кількості помилок при виконанні операцій на даними;
- автоматичне керування пам'яттю – одним із нововведень в мову Java, порівняно із C++ є автоматичний збирач сміття (GC - Garbage Collector), він сам керує пам’яттю об’єкта, програмістові потрібно лише створити об’єкт та виділити пам’ять під нього, збирач сміття звільнить пам'ять коли об’єкт вже не буде використовуватись і стане непотрібним. Це реалізується наступним чином якщо до об'єкта вже не залишається

посилань, збирач сміття автоматично видаляє його із пам'яті. Витік пам'яті може статися, якщо код, має посилання на вже непотрібні об'єкти, але ризик цього значно менший порівняно з мовами де звільняти пам’ять від об’єктів потрібно вручну наприклад як в C++ за допомогою деструкторів [6];

На Java написано більшу кількість додатків та програм для операційної системи андроїд. Хоча ця мова і не є найновішою в цьому напрямку, вона все ще зберігає лідируючі позиції по популярності і з великою ймовірністю в наступні декілька років не зійде з ринку, вона добре пристосована до інших технологій, тому в її доцільності вибору та використання в межах дипломної роботи немає сумнівів.

Практично будь-який андроїд додаток повинен містити локальну базу даних. По-перше для зберігання та обробки інформації у випадку, якщо немає підключення до інтернету та немає сторонніх реалізацій синхронізації даних з віддаленою базою, для відображення деякої інформації без інтернету. По-друге для зберігання деякої постійної інформації користувача, у моєму випадку це унікальний згенерований ключ залогіненого користувача та його дані.

SQLite - це полегшена реляційна система керування базами даних. Реалізована у вигляді бібліотеки, де реалізовано запити та синтаксис зі стандарту SQL. SQLite не використовує клієнт-серверну архітектуру, вона не є процесом який взаємодіє з програмою, а працює як бібліотеки яка компілюється і виконує всю необхідну роботу. Таким чином, запити використовуються як виклики функцій (API) або ORM систем бібліотеки SQLite. Це зменшує витрати ресурсів, час виконання запиту і спрощує програму. База даних зберігається у файлі включаючи метадані, таблиці, індекси і дані користувача на тому пристрої, на якому виконується програма. Простота реалізована завдяки, блокуванню файлу бази даних, перед виконання транзакції. ACID-функції(властивості, що гарантують надійну роботу транзакцій бази даних: атомарність, узгодженість, довговічність, тобто певна послідовність операцій) досягаються за рахунок

створення файлу-журналу. Є підтримка багатопоточності, кілька потоків можуть без проблем читати дані з однієї бази. Запис можна здійснити тільки у випадку, якщо інших запитів у цей час не має, інакше запис закінчується помилкою, або задати автоматичне повторення спроб запису протягом заданого інтервалу часу, але зазвичай при використанні даної бази у мобільних пристроях, взаємодія відбувається тільки з одним користувачем.

У комплекті інсталяції є також функціональна частина у вигляді виконуваного файлу sqlite3, він дозволяє у командному рядку виконувати операції над базою, запити, та формування вигляду результату запиту у різних видах [7].

ORM ROOM.ORM(Object-Relational Mapping)-технологія програмування, яка зв’язує базу даних з концепціями об’єктно орієнтованої мови програмування, надає високорівневий інтерфейс управління базою даних та створює “віртуальну об’єктну базу даних”. ORM ROOM надає зручну обгортку для роботи з базою даних SQLite. ROOM має три основних компоненти: Entity, Dao і Database. Entity – це клас об’єкта який зберігається в базі даних, його позначається анотацією “@Entity”. Поля класу це поля в таблиці бази даних. Для полів є наступні анотації: “PrimaryKey”, “ColumnInfo”. Клас Entity буде використаний для створення таблиці в базі. Як ім'я таблиці буде використано ім'я класу. Анотацією PrimaryKey ми помічаємо поле, яке буде ключем в таблиці, ColumnInfo вказує ім’я для поля та їх тип.

В об'єкті Dao описуються методи для роботи з базою даних, тут виконуються всі запити в таблиці. Клас позначається анотацією Dao. Над методами пишуться анотації та SQL запит. Існують наступні анотації Query, Delete, Insert, Update Для вставки, оновлення, видалення використовуються методи insert, update, delete з відповідними анотаціями. Назви методів можуть бути будь-якими. Головне - анотації.

Анатоцією “Database” помічається основний клас по роботі з базою даних. Цей клас повинен бути успадковувати RoomDatabase. В параметрах анотації

Database вказуються, Entity класи, що будуть використовуватися, і версію бази. Для кожного Entity класу entities буде створена таблиця. В Database класі описуються абстрактні методи для отримання Dao об'єктів [8].

Макети сторінок та весь дизайн додатка реалізується за допомогою XML-розширюваної мови розмітки (англ Extensible Markup Language). Макет будує візуальну структуру користувальницького інтерфейсу, призначеного для відображення сторінок додатка .За допомогою довідника XML-елементів, який є в Android Studio, можна швидко і просто створювати макети призначеного для користувача інтерфейсу. Важливим етапом при їх побудові є розміщення в контейнерів відносно один-одного. Елементи що містяться в коді створюються подбіно до створення веб-сторінок в HTML - за допомогою вкладених елементів. XML в андроїді надає теги для компонентів, наприклад кнопка - тег <Button> і відповідні атрибути та значення для атрибутів такі як визначення дизайну кнопки, розмірів, форми, та інше.

В андроїд проектах XML файли мають наступну структуру пакетів.

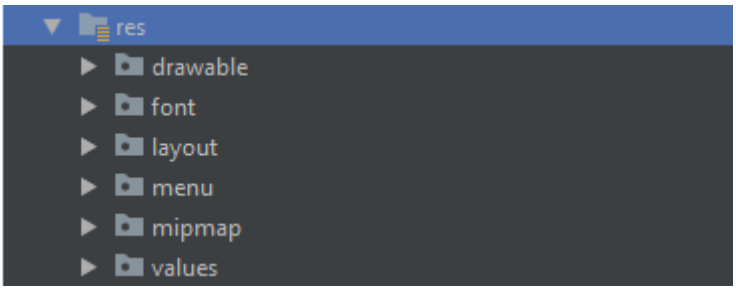


Рисунок 2.1 Структура XML пакетів

Drawable - файли растрових зображень (.png, .jpg, .gif) або файли XML шаблонів, які складаються з графічних ресурсів. Тут створюється дизайн для окремих компонентів із зміненим станом форми, анімації(наприклад заокруглення кнопки чи добавлення тіні), який потім задається атрибутом background або src, значенням яких є шлях до файлу drawable,при створенні макета.

Font – пакет з набором шрифтів, які не входять в стандартний набір та були завантажені із сторонніх джерел.

Layout- файли XML, які визначають самі макети призначені для користувача інтерфейсу.

Menu - файли XML, які визначають меню програми, такі як меню параметрів, контекстні меню або вкладені меню.

Mipmap - графічні файли для значків запуску з різними наборами розрімів.

Values – XML файли, які містять значення для певних наборів одиниць. Кожен ресурс визначається за допомогою свого власного елемента XML, йому можна призначати всласне ім'я файлу і поміщати ресурси різних типів в один файл. Тим не менше, може з'явитися необхідність помістити ресурси окремих типів в різні файли. Наприклад, нижче наведені типи для імен файлів ресурсів, які можна створити в цьому каталозі:

- arrays.xml для ресурсів-масивів (масиви із зазначенням типу);
- colors.xml для значень кольорів;
- dimens.xml для значень одиниць вимірювань;
- strings.xml для рядкових значень(назви і тд);
- styles.xml для стилів;

Такий підхід розміщення значень, дозволяє легко і швидко знаходити потрібне значення та змінювати його оскільки всі вони зберігаються в одному централізованому місці [9].

В якості сервера, віддаленої бази даних, сховища файлів, та платформи, що надає велику кількість послуг для розробки під ОС андроїд було обрано платформу Firebase. Оскільки вона має все необхідне для створення та функціонування даного дипломного проекту. Firebase — це платформа розробки мобільних та веб- застосунків, що надає власний back-end для великої кількості потреб. Більш детально про платформу Firebase та її можливості описано в пункті 2.2, можливості та ресурси платформи Firebase.

В проекті використовується система керування версіями та збереження коду git. Незважаючи на те ,що git призначений для опрацювання коду в команді, він надає ряд корисних можливостей і для користування єдинем розробником

проекту. Наприклад для сховища коду, яке винесено у дод А, через посилання, перегляд версій коду по історії добавлення комітів(коміт – останні зміни), також для аналізу коду, та метрики коду.

Середовище розробки дипломного проекту Android Studio. Середовище побудоване на базі продукту IntelliJ IDEA Community, що розроблений компанією JetBrains. Android Studio розвивається в рамках відкритої моделі розробки.

Середовище підтримується основними ОС, для Linux, Mac OS X і Windows. Середовище надає інструменти для розробки ПЗ не тільки для смартфонів і планшетів, але і для інших пристроїв на базі Android, таких як Wear, телевізорів (Android TV), окулярів Google Glass і автомобільних систем інформаційних панелів (Android Auto).

Середовище розробки Android Studio пристосовано для виконання різних завдань, що вирішуються в процесі розробки застосунків для платформи Android. Вона дозволяє проводити тестування програм на сумісність з різними версіями програми та надає інструменти для проектування макетів, що працюють на пристроях з дисплеями різної роздільності (смартфони, ноутбуки, планшети, окуляри, годинники тощо). Крім можливостей IntelliJ IDEA, в Android Studio є кілька нових функцій, таких як нова система тестування і розгортання програми, заснована на збірці проекту “Gradle”, додатково вона підтримує використання засобів безперервної інтеграції (CI).

Особливості:

- редактор макетів (layouts) дозволяє перетягнути і покласти компоненти інтерфейсу в макет, та заповнювати його атрибути за допомогою вкладки ”Attributes”, і переглядати в реальному часі без запуску проекта макети (layouts) на різних розширеннях екранів;
- збірка Gradle;
- Android-орієнтований рефакторинг та підказки для виправлень;

- вкладки для метрик продуктивності, удобності, сумісності версій та інших проблем;
- використання готових Android макетів, створених раніше або вибраних із базового набору Android Studio;
- консоль: детальний вивід логів, з по кроковими підказками по оптимізації;

Для запуску та тестування програми в реальних умовах необов'язково її завантажувати на свій девайс, Android Studio надає для цього емулятори з широким вибором різних марок та моделей реальних девайсів з різними версіями ОС Android.

Емулятор - це відтворення програмними та апаратними засобами Android Studio, роботу цілої ОС Android. Перевагою емулятора над реальним пристроєм є те, що він надає доступ до будь-якої версії ОС Android або будь-якого пристрою, на якому потрібно перевірити роботу програми. До недоліку можна віднести те, що у розробника немає можливості побачити, як додаток дійсно виглядає на пристрої з дисплеєм, відсутній доступ до деяких функцій реальних пристроїв. Дозволяє імітувати вхідні дзвінки та повідомлення, місце розташування, інтернет, повороти екрану і різні датчики, як у реального пристрою. Не підтримує: Bluetooth, NFC, роботу з SD-картою, роботу з навушниками, USB. Для емулятора необхідно виділити мінімум 512 Мб ОЗУ.Але для запуску емулятора та коректної роботи необхідно мати добрі ресурси на ПК, як мінімум 6 Гб, ОЗУ [10].

Роботу програми я тестував на наступних девайсах, віртуальні: Nexus 4 API 22 768x1280 Android 8.0, Pixel API 25 1080x1920 Android 5.1, реальні: Prestigio PSP7530DUO Android 5.1.1, Samsung Galaxy A70, Android 9.0, 2400x1080.

В Android Studio для збірки проекту використовується система автоматичної збірки Gradle. Gradle –це збирач проекту, він компілює всі залежності та бібліотеки які підключені до проекту, а в андроїд проектах таких

залежностей досить багато. Для цього використовує предметно-орієнтовану мову на основі мови Groovy, замість XML-подібної конфігурації проекту. Також Gradle займається конфігурацією проекту наприклад: версії SDK, версії коду, містить репозиторії для загрузки базових компонентів, підключає систему тестування jUnit, та багато іншого, що дозволяє керувати поведінкою проекту в тих чи інших випадках [11]. В дипломному проекті використовуються два основні файли Gradle це:

- build.gradle(Project) – для конфігурації проекта;
- build.gradle(Module) – для підключення залежностей та бібліотек;

Перший файл – це файл збірки верхнього рівня, до якого можна додати параметри конфігурації спільні для всіх модулів, код файлу наведено у дод. Б. Другий файл налаштовує конфігурації запуску проекту(див. дод. В) та підключає додаткові бібліотеки, список всіх бібліотек наведено у дод. Г.

Основні бібліотеки які підключені до дипломного проекту це: бібліотеки платформи Firebase, зокрема це: firebase-auth, firebase-database, firebase-storage, firebase-functions, бібліотеки модифікованих компонентів EditText та інші – Material-edittext, бібліотека для смайлів що використовуються при відправленні повідомлень SuperNove-Emoji, загрузка фото – Picasso, стиск зображення – Compressor, та інші [12].

2.2 Можливості та ресурси платформи Firebase

Як вже було сказано вище в якості сервера було обрано платформу Firebase, оскільки вона має все необхідне для створення та функціонування повноцінної соціальної мережі для ОС Android. Рішення які використовуються у дані роботі Firebase Analytics, Firebase Auth, Realtime Database, Firebase Storage, Firebase Cloud Functions - це не весь список служб які надає Firebase, але лише ті що використовуються в проекті.

Для роботи з платформою Firebase необхідно зареєструвати проект, ввести назву основного пакета, псевдонім і хеш сертифікат для доступу до сервісів та їх даних тільки з проекту. Хеш сетрифікат - це ключ “SHA1” в проекті, він знаходиться в налаштуваннях Android Studio за шляхом Gradle/ProjectName/app/android/signingReport. На рис. 2.2 наведено результат збірки цього файла.

```
Config: debug
Store: /home/vadym/.android/debug.keystore
Alias: AndroidDebugKey
MD5: 00:CB:32:58:1D:CB:96:88:2E:17:B5:27:14:81:49:
SHA1: 84:0D:F6:B2:40:3B:AE:04:DB:D3:F9:78:2F:EE:0A
SHA-256: 2F:A6:F4:5B:95:5E:6D:CD:99:B6:14:E4:1B:E7
Valid until: четвер, 2 вересня 2049 р.
----------
Variant: release
Config: none
----------

BUILD SUCCESSFUL in 3s
1 actionable task: 1 executed
19:45:31: Task execution finished 'signingReport'.
```

Рисунок 2.2 Збірка SigningReport ключ SHA1

Наступним кроком необхідно встановити Firebase SDK , та підключити необхідні бібліотеки в gradle.build(Module), для роботи з базою даних.

```
dependencies {
    implementation 'com.google.firebase:firebase-core:11.4.0'
    implementation 'com.google.firebase:firebase-auth:11.4.0'
    implementation 'com.google.firebase:firebase-database:11.4.0'
    implementation 'com.firebaseui:firebase-ui:0.6.2'
}
```

Управління проектом Firebase здійснюється в WEB консолі. В консолі можна створювати нові проекти, переглядати дані користувачів, керувати файлами, працювати з базою даних та багато іншого. В розділі 4 наведено та детально проаналізовано використання ресурсів їх аналітика та вартість.

2.2.1 Firebase Analytics

Firebase Analytics — безкоштовне рішення для оцінки застосунків та аналітики використання програми користувачами. Та аналітика використаних ресурсів. Вона дає змогу покращувати бізнес процеси програми, оскільки надає різну статистику про використання. Наприклад, місце розташування користувачів, статистику відвідувань кожної сторінки, розраховує доходи від користувача та груп користувачів, стабільність програми, популярність версії програми, опис аудиторії по ОС, та регіонам. Використання та можливості даного сервісу продемонстровані в розділі 4.

2.2.2 Firebase Authentication

Firebase Authentication – служба для аутентифікації користувачів , використовуючи код на стороні клієнта. Підтримує реєстрацію через такі провайдери як Google, Facebook, GitHub, Google Play Games, Twitter. Також включає систему управління користувачами, за допомогою якої можна увімкнути аутентифікацію користувача за допомогою входу з електронної пошти та пароля, або через номер телефону. В даній роботі я вибрав вхід та реєстрацію за допомогою номеру телефону. Оскільки проект розрахований під мобільні пристрої і такий підхід є кращою практикою ніж аутентифікація по емейлу. Тому, що не потрібно згадувати пароль та адресу електронної скриньки, здійснювати додатковий вхід на пошту, а потрібно всього лише ввести свій номер телефону та підтвердити його ввівши код з 6 цифр. Використовуючи будь-який тип аутентифікації, Firebase Authentication створює користувача, генеруючи йому унікальний ідентифікатор, та відстежує його час останнього входу. Також дозволяє переглядати число перевірок за номером телефону та країну з якої здійснюється аутентифікація.

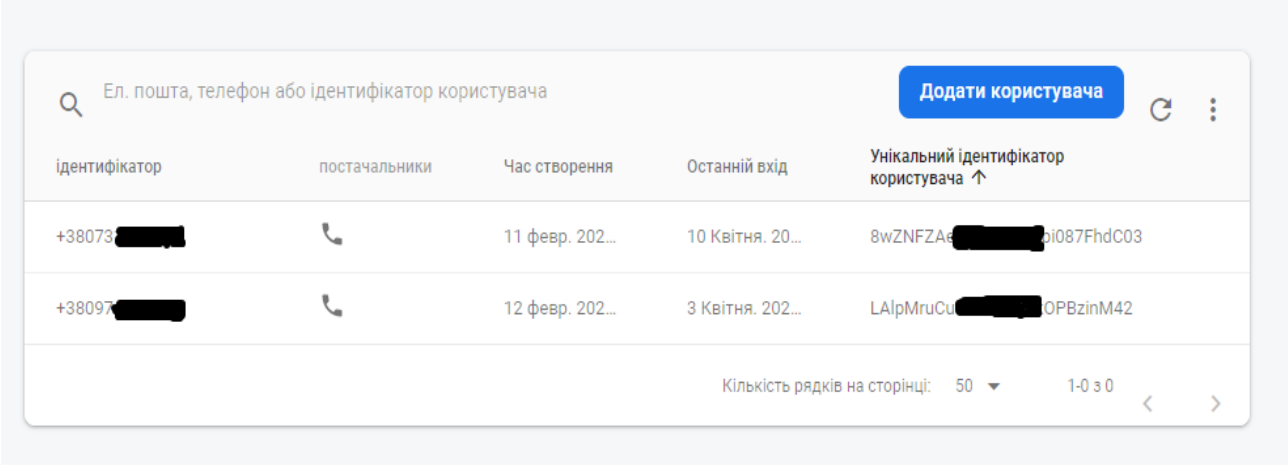


Рисунок 2.3 Firebase Authentication список залогінених користувачів в систему

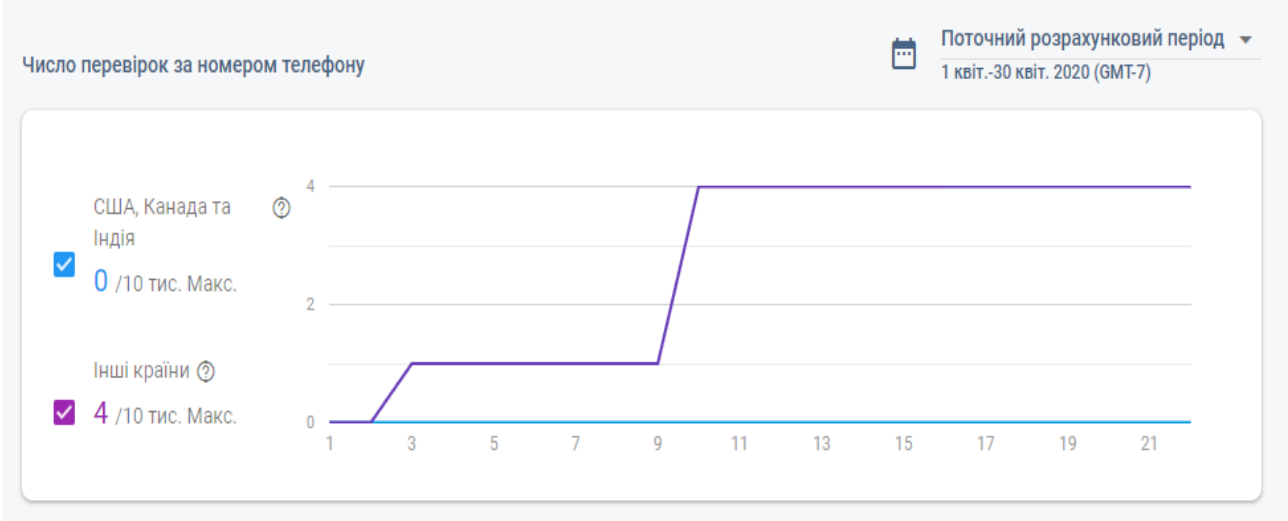


Рисунок 2.4 Firebase Authentication, кількість надісланих смс з кодом перевірки

Перевірити чи залогінений користувач, в java коді можна наступним чином FirebaseUser user = firebaseAuth.getCurrentUser(), якщо user рівний null, то користувач не залогінений, інакше вже в системі. Це дозволяє накласти додатковий рівень захисту для даних в інших службах Firebase, наприклад в Realtime Database чи Firebase Storage прописавши в інструкціях правил служб наступні правила: "rules": ".read": auth != null, "write": auth != null. Тобто доступ до даних та ресурсів буде тільки в користувачів, що є залогінені, це дозволить захистити дані від зловмисників, які спробують отримати доступ зі сторонніх систем.

2.2.3 Firebase Realtime Database

Найбільш важливою службою в проекті від платформи Firebase є Realtime Database, тут зберігаються всі дані користувачів, їх переписки, пости та інше. Realtime Database дає можливість не тільки зберігати дані та отримувати їх за допомогою запитів, але і відображати їх в режимі реального часу без додаткових дій. Завдяки бекенду та API , що надає ця служба, які дозволяють синхронізувати дані між клієнтами та зберігати їх у хмарі Firebase. Realtime Database доступна для інтеграції із наступними застосунками: Android, iOS, Java, JavaScript, Swift, Node.js, Objective-C. Також є доступ через REST API та AngularJS, React та інші .js фреймворки. REST API використовує протокол HTTP для отримання push-повідомлень від сервера. Безпека даних в Realtime Database, надається в консолі Firebase створюючи власні правила, що застосовуються на сервері доступу до даних, наприклад такі як були сказані в кінці пункту 2.2 (це найбільш популярний спосіб захисту). Або придумати власні правил доступу, виходячи з власних потреб, додатково бажано дані шифрувати програмним способом на стороні клієнта, створюючи кращу безпеку [13]. Навіть якщо дані потраплять до зловмисників, то вони їх не зможуть розшифрувати.

В Realtime Database дані зберігаються у вигляді дерева даних, із типами зв’язку – батько-дитина, ключ-значення, якщо вітка дерева немає нащадків її називають “листком дерева”, далі будем користуватись цією термінологією. Такий підхід збереження даних дозволяє легко та продуктивно витягувати дані з бази в соціальну мережу для побудови систем друзів, підписників, месенджера та інше. Відпадає потреба у зв’язуванні даних по id, або Primary Key як в традиційних реляційних базах даних, оскільки всі дані користувача прив’язані до конкретного батька тому вони “знають” кому вони належать і звернення йде до конкретного батька дерева (користувача). Значення батька (конкретного користувача) є унікальним значенням, яке генерується як унікальний ключ при його створенні (занесенні у базу даних) [14].

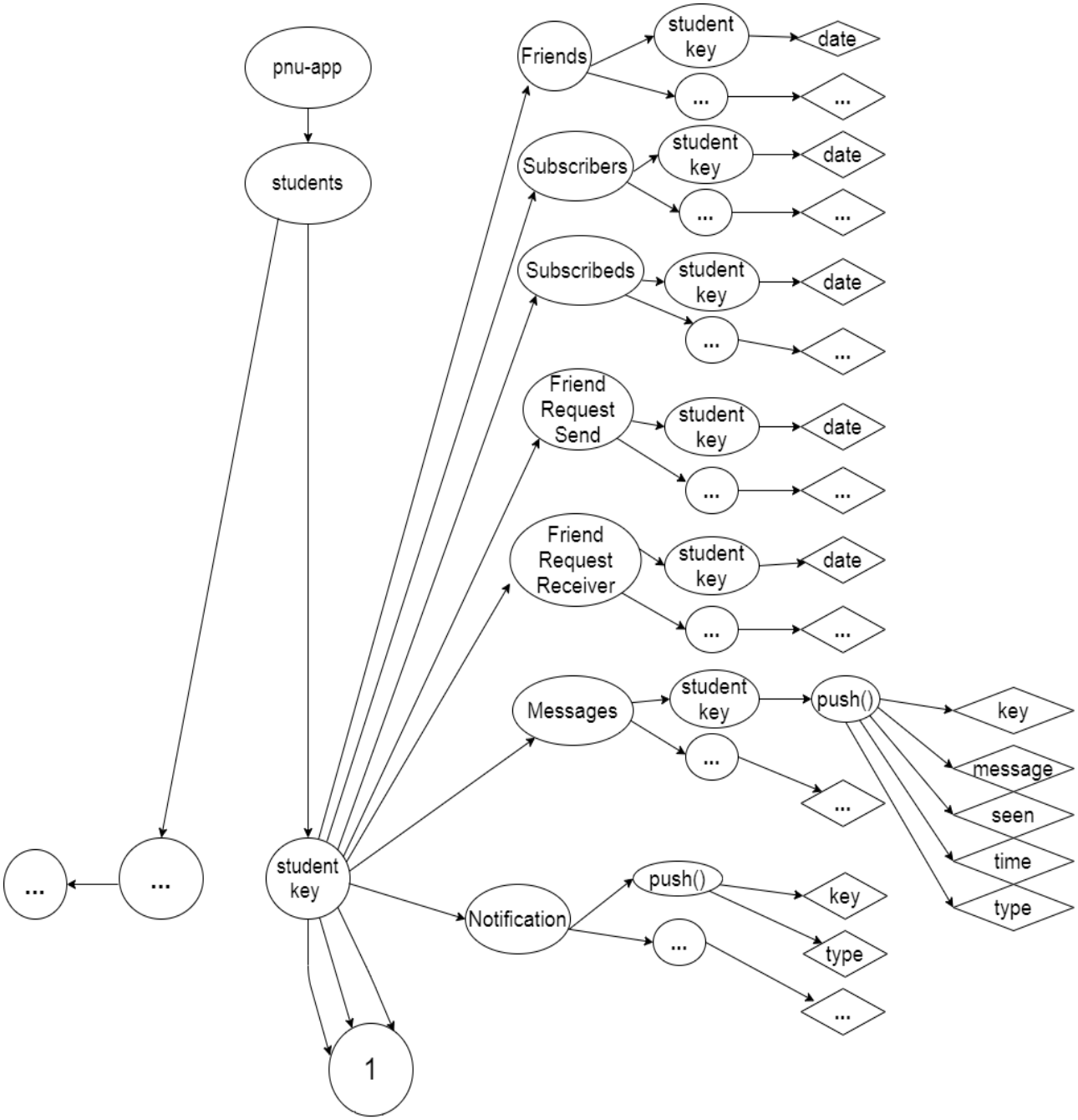


Рисунок 2.5 Діаграми бази даних у вигляді дерева, аркуш 1

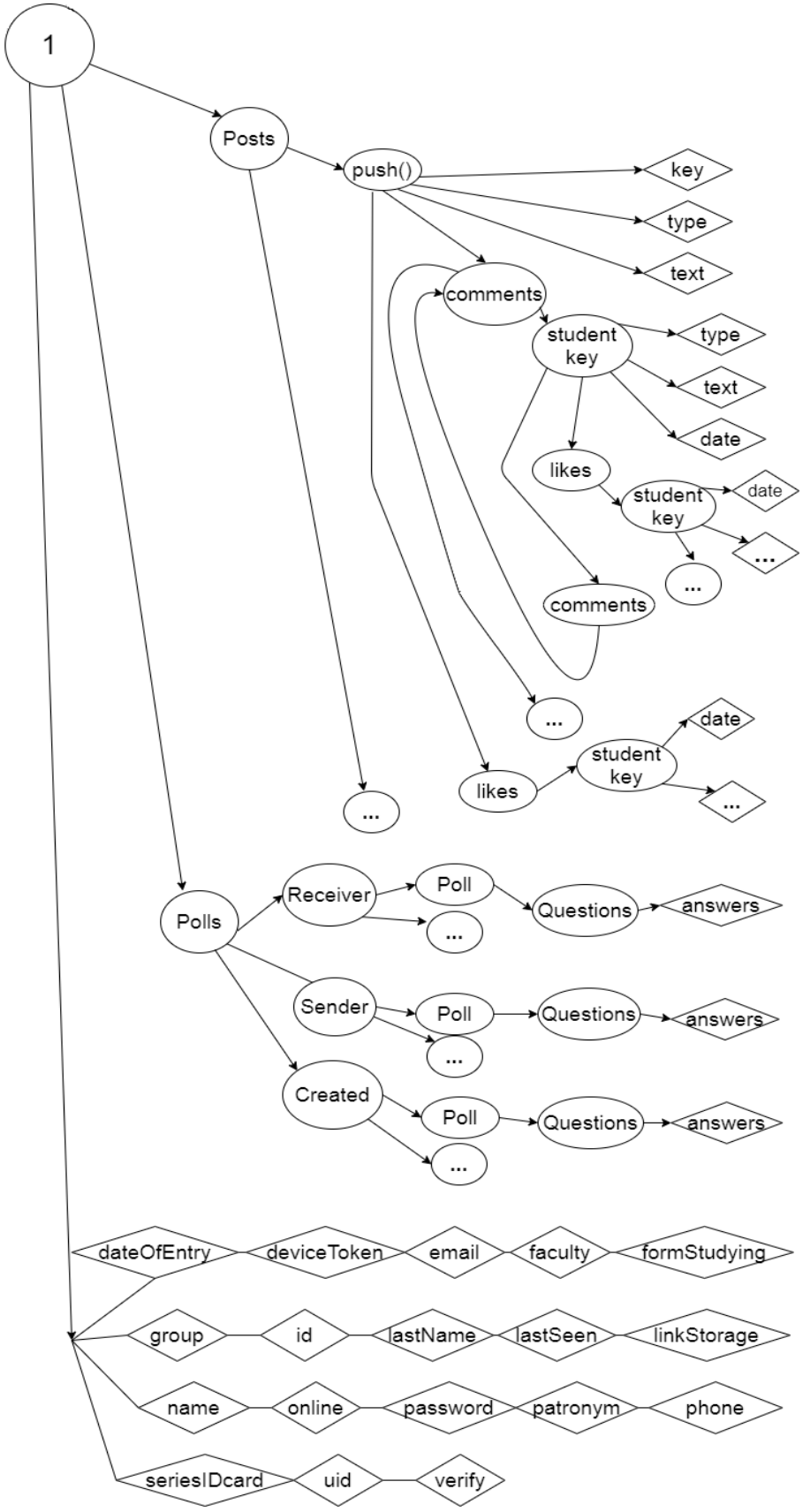


Рисунок 2.5 Діаграми бази даних у вигляді дерева, аркуш 2

Пояснення: овал - вузол(має нащадків), ромб – листок(немає нащадків), овал з трьома крапками це такі самі вузли тільки інші значення!

На найвищому рівні знаходиться корінь pnu-app, від нього наслідується вузол розгалудження students. Якщо звернути увагу, то можна помітити, що вузла students можна було б уникнути і від pnu-app переходити відразу до вузлів student key. Але цей вузол дозволяє в майбутньому змінювати структуру та добавляти нові вузли цього ж рівня для кращої структуризації, це робить дерево даних більш гнучкішим. Від вузла students наслідуються student key – це рівень конкретного студента, значенням цього вузла є унікальний ключ, який генерується при створенні. Ключ є дещо схожим до Primary key в реляційних базах та виконує подібну функцію – ідентифікацію об’єкта. Різниця суттєва тільки в значенні, в Primary key це числа починаючи від 0, а в ключа даної системи, значенням є унікальний згенерований код довжиною 15-20 символів. Який може містити букви, цифри, символ підкреслення, дефіс. Від цього унікального ключа наслідуються всі дані студента. Розглянем спочатку перші 5 вузлів студента, які наведені вище в рис. 2.5: Friends, Subscribers, Subscribeds, FriendRequestSend, FriendRequestReceiver, це списки – відповідно: друзів, підписників, читачів(на кого підписаний сам студент), заявок в друзів: надісланих та отриманих. Кожен із цих вузлів має однакову структуру – список студентів(ключів вузла students), студент має листок в яких ініціалізується часом виконання операції(добавлення в друзі, підписники і тд…) цей час потім може бути використаним для додаткових функцій, наприклад річниця дружби в соц мережі і тд. Наступний вузол – Messages в ньому міститься список надісланих та отриманих повідомлень. Структура подібна до попередніх вузлів, з різницею в тому, що в ключі студента містять вузли повідомлення, ключ повідомлення генерується методом push() (подібний до ключа студента). У вузлі повідомлень міститься 5 листків key(студента відправника), message, seen, type, time. Після вузла Messages, йде вузол Notification, подібний до попереднього вузла, тільки без рівня student key. Оскільки notification ніде не використовуються в

структурованому вигляді як повідомлення, а потрібні як список отриманих, містять лише push() ключі з 2 листками type, key(студента відправника). Аркуш 2 рис. 2.5 починається з дочірнього вузла студента Posts. Він є найбільш складним, наслідує push() вузли(унікальні ключі), унікальний ключ наслідує 3 листки: type, text,key, вузол comments – який має вузол (ключ студента)- наслідує 3 листки type, text, date, вузол likes(лайки коментарів) – наслідує вузол ключ студента, який наслідує листок з датою ставлення лайка. Вузол ключ студента, що наслідується з comments, може мати вузол comments – вкладені коментарі, цей вузол має все те саме що і верхній вузол comments він рекурсивно наслідується з батьківського вузла comments. Ну і останнім вузлом push ключів, що наслідуються з Posts є likes(лайки постів) – мають те саме що лайки коментарів.

Опитування в додатку здійснюються за рахунок структури вузла Polls. Він наслідує 3 вузли: Receiver, Sender, Created відповідно – отримані, надіслані, створені опитування. Кожен вузл списків опитувань(назви опитувань) має вузли з питаннями, які наслудють листки з відповідями. Нижня частина рис. 2.5 складається з 18 листків даних студента: рік вступу, токен девайса, емейл, факультет, форма навчання, група, ід, прізвище, час останнього відвідування додатку, ссилка на фотографію в Storage, ім’я, онлайн, пароль, по-батькові, номер телефону, серія та номер студентського, id – firebase authentication, верифікація. Вони не мають зв’язку між собою, а представлені на рис. 2.5 послідовно за для кращого візуального представлення. На основі цих даних будуються майже всі сторінки додатку. Правильно побудована архітектура збереження даних в дереві, відіграє важливу роль, оскільки варіантів як реалізувати їх розміщення в дереві є багато, потрібно вибирати такий варіант який використовує найменше ресурсів та відповідає наступним рекомендаціям:

- дані не повинні ідентично повторюватись у різних вузлах;
- не довзоляється розміщувати вузли однакових ключів на одному рівні;

- уникати багато рівневої вкладеності дочірніх вузлів(RealTime дозволяє вкладеність до 32 нащадків);
- використовувати один ідентифікатор для зв’язку між вузлами;
- будувати дерево гнучким, завдяки додатковому розділенні вузлів для різних класів збереження даних;

Нижче на рис. 2.6 наведено батьківський вузол одного користувача із дочірніми вузлами в базі даних Realtime.

Рисунок 2.6 Вузол даних користувача, аркуш 1

```
Posts
  -M73NIeD81Xh3TQRjw8I
    Comments
      -M73TD9Ns0RTJe45YMD9
        Comments
          -M73UpQKGKy9xGrs78tG
            keySender: "-LyV2v1_hoFItLn06dM
            text: "вдячний0
            time: 158921335362
            type: "text"
        keySender: "-LyV3Y30G01NKDIupKV
        likes
          -LyV2v1_hoFItLn06dM7
          -LyV3Y30G01NKDIupKVv
        text: "))))) 
        time: 158921293065
        type: "text"
      -M73UzIc9P2z6d5btR6n
    countShare: 0
    keySender: "-LyV2v1_hoFItLn06dM
    likes
      -LyV2v1_hoFItLn06dM7
        date: 158921215746
      -LyV3Y30G01NKDIupKVv
        date: 158921287809
      -LyV4EPE1ltDAkfN4jbY
        date: 158921215746
      -LzCiPYCUW6X8VpG9G5s
        date: 158921215746
    linkFirebaseStoragePostPhoto: "https://firebasestorage.googleapis.com/v0/b/pnu
    text: "Прикарпатський національний університ
    time: 158921137976
    type: "text"
  -M73LSwS4LQDvTVN9vyw
  -M69aNj7N7JjvI9JHki6
dateOfEntry: "01-07-2016
deviceToken: "dlCW4YJW-QY:APA91bHYLEEdwLn1Q3s1ALnvR8cHs1LOBic
email: ""
faculty: "математики та інформатик
formStudying: "Денна
group: "ІПЗ-41
id: 1
lastName: "Попович
lastSeen: 158954518124
linkFirebaseStorageMainPhoto: "https://firebasestorage.googleapis.com/v0/b/pnu
name: "Вадим
online: true
password: "8f4cd79c1dc61d1d1d8e4c84464e45
patronym: "Віталійович
phone: "+380731222459
seriesIDcard: "BA11582136
uid: "-LyV2v1_hoFItLn06dM
verify: true
```


Рисунок 2.6 Вузол даних користувача, аркуш 2

2.2.4 Firebase Storage

Firebase Storage – сховище для файлів різного виду фото, аудіо, відео та інші, надає надійне завантаження та вивантаження незалежно від швидкості мережі, підтримується Google Cloud Storage. Дає 5 Гб для безплатного користування. Для зберігання файлів використовується наступна структура, тип файлів/Серія та номер студентського/список файлів, наприклад images/BA12345678/mainphoto.jpg. Для економії ресурсів, велике значення відіграє стискання даних, воно може в десятки рази зменшити витрати ресурсів місця зберігання , його реалізація буде детально розглянута в розділі 3 [15].

2.3 Моделювання архітектури діаграми сторінок

В розробці додатків для операційної системи Android, замість традиційного слова - сторінка додатку, використовується – активіті додатку, надалі в цьому і наступних розділах, для коректності опису, буде використовуватись слово активіті. На рис. 2.7 продемонстровано всі активіті додатку ПНУ – соціальна мережа та зв'язки їх використання. Сірим овалом показано фрагменти. Фрагмент – представляє посилання або частину інтерфейсу в (активіті), їх можна об'єднювати для побудови багатопанельного інтерфейсу та повторного використання фрагментів у кількох активіті.

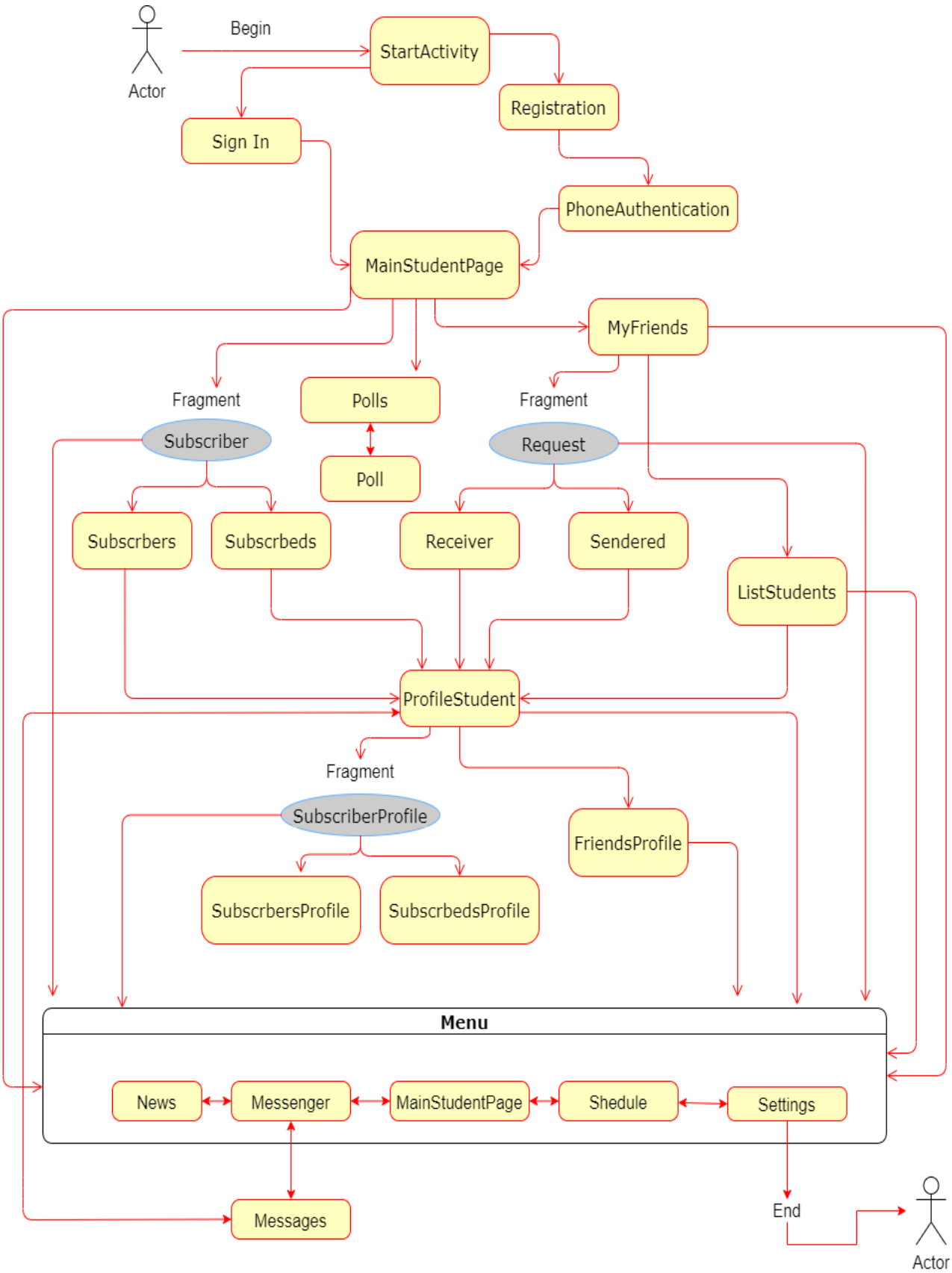


Рисунок 2.7 Діаграма сторінок(активіті) додатку

2.4 Моделювання архітектури діаграми класів

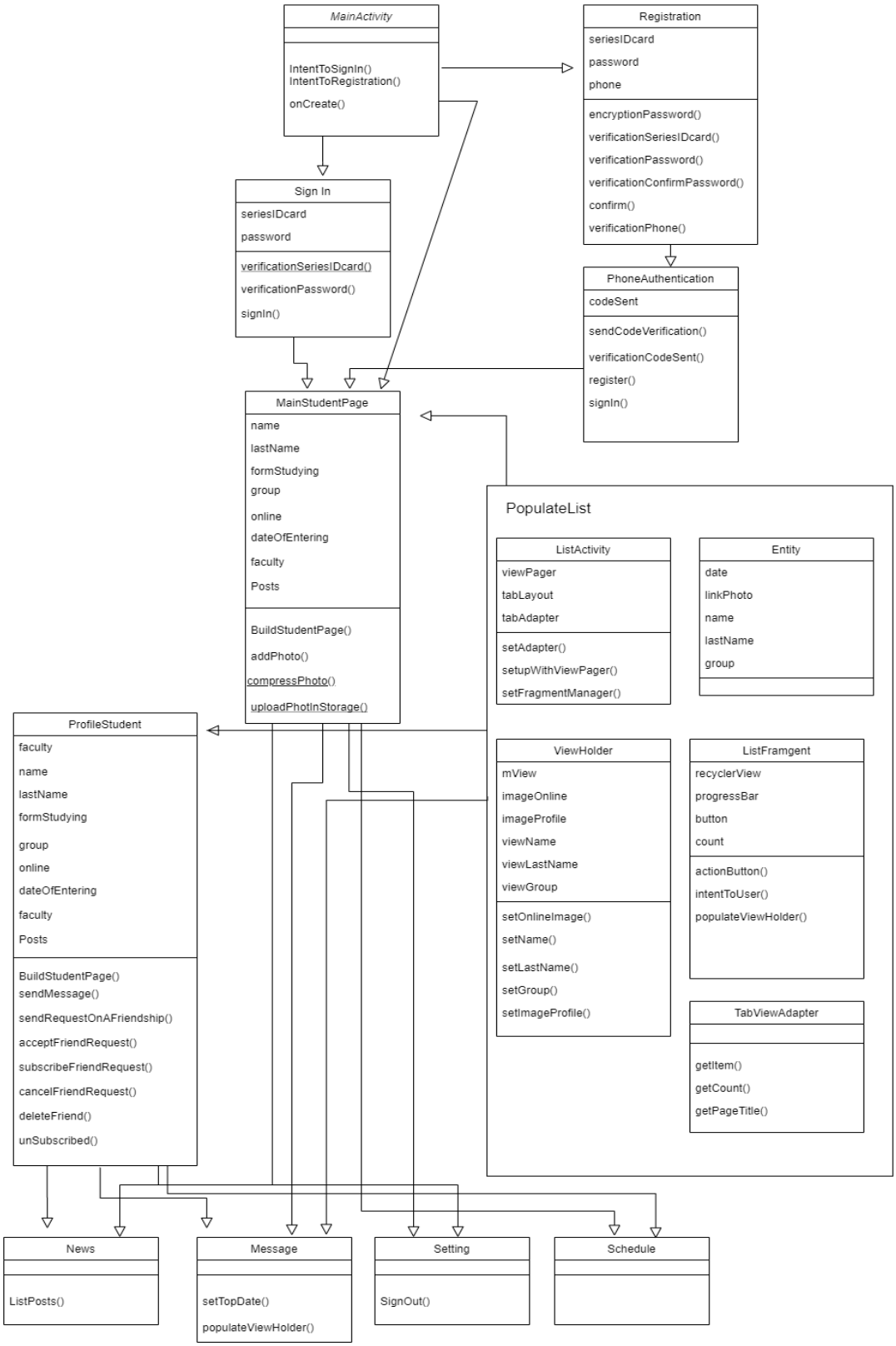


Рисунок 2.8 Діаграма класів додатку

Діаграма класів має схожу архітектуру до діаграми активіті лише з тією різницею, що один активіті може обслуговуватись декількома класами. В діаграмі класів на рис. 2.8, наведений скорочений варіант архітектрури для кращого сприйняття. Блок PopulateList це паттерн, що складається з п’яти класів навединий в діаграмі один раз. Хоча в програмі він реалізується кожен раз по новому, для реалізації нових списків студентів: друзів, підписників, читачів, отриманих та надісланих запитів в друзі, підписники друзів, месенджер, пошук друзів та інше. Цей блок завжди має однакову архітектуру, що складається з п’яти класів(у моєму випадку), різниця лише в зміні деякого внутрішнього функціоналу списку. Більш детально про реалізацію списків та методів діаграми наведено в розділі 3.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Моделювання шаблонів сторінок програми

Як вже було сказано в розділі 2, шаблони сторінок – активіті, були розроблені засобами XML. Основні компоненти:

- TextView – текстове поле;
- Button – кнопка;
- View – горизонтальна лінія;
- ProgressBar – компонент прогресу загрузки;
- EditText – текстове поле для вводу інформації;
- BottomNavigationView – меню в нижній частині екрану;
- RelativeLayout – контейнер для компонентів із розміщення по прив'язці до конкретного елементу;
- LinerLayout - контейнер для компонентів із розміщення по вертикальні або горизонтальні орієнтації;
- RecyclerView – прокручуваний список;
- ListView – прокручуваний список по вертикалі;
- ImageView – зображення;
- CircleImageView – кругле зображення;
- ScrollView – контейнер для прокрутки компонентів;

Розглянем основні шаблони активіті, що були наведені на рис. 2.11 та їх дизайн.

Робота програми починається з даного стартового шаблону рис. 3.1, вона складається з двох кнопок, входу та реєстрації, що мають видозмінений вигляд – заокруглену форму та інші параметри. Змінити основні параметри компонента можна завдяки додаткового drawable – файлу, з такими атрибутами як: ripple, item, shape, corners, stroke, solid та інші, Drawable файл задається задопомогою атрибутів компонента: background або src. Також стартова активіті має компонент

круглого зображення з логотипом університету, та два текстові поля з різними розмірами шрифту. Посилання на файл рис. 3.1, https://github.com/Vadym-Popovych24/PNU_APP/blob/master/app/src/main/res /layout/activity_main .xml або в дод. А.

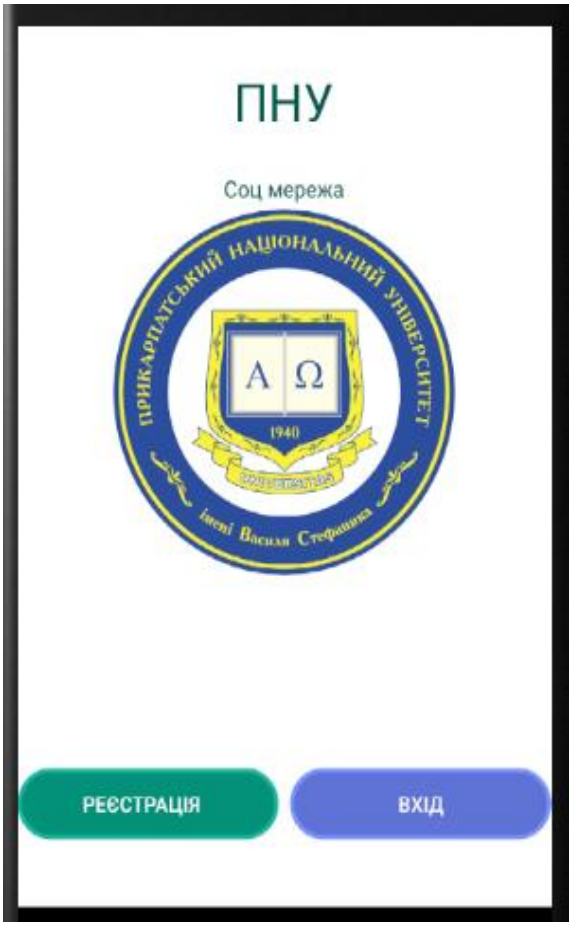


Рисунок 3.1 Стартова активіті

Коли користувач тільки починає роботу з додатком, йому потрібно пройти реєстрацію, для цього є відповідна активіті з чотирьма полями для вводу інформації, кнопкою підтвердження та текстовим полем з назвою активіті. Для кращого дизайну та простоти вводу інформації в поля, замість стандартного компонента EditText, використовується компонент MaterialEditText, ссторонньої бібліотеки Material. Ця бібліотека надає додаткові атрибути, наприклад для підказки вводу тексту в поля, як видно на рис. 3.2, активне поле підкреслюється відповідним кольором, а підказка розміщується вище поля [16]. Посилання на файл рис. 3.2, https://github.com/VadymPopovych24/

PNU_APP/blob/master/app/src/main/res/layout/activity_registration.xml або в дод. А.

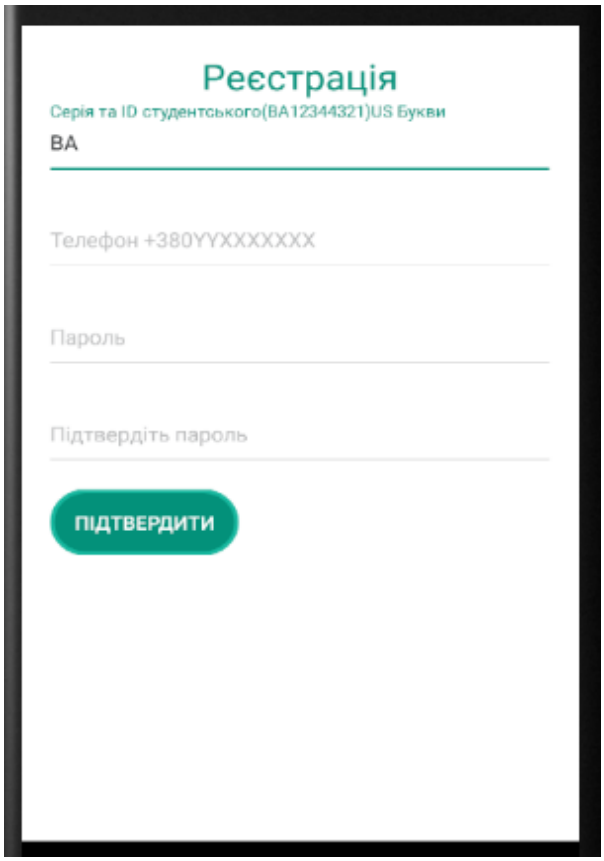


Рисунок 3.2 Реєстрація

Після успішно пройденої реєстрації слідує активіті з полем для підтвердження номеру телефону рис. 3.3. Вона містить ті ж компоненти, що і реєстрація, додатково має компонент ProgresBar для відображення того, що процес відбувається і користувачу потрібно зачекати. Для поля вводу отриманого коду, наведені відповідні підказки. Посилання на файл рис. 3.3, https://github.com/Vadym-Popovych24/PNU_APP/blob/master/app/src/main/res/layout/activity_phone_authentication.xml або в дод. А.

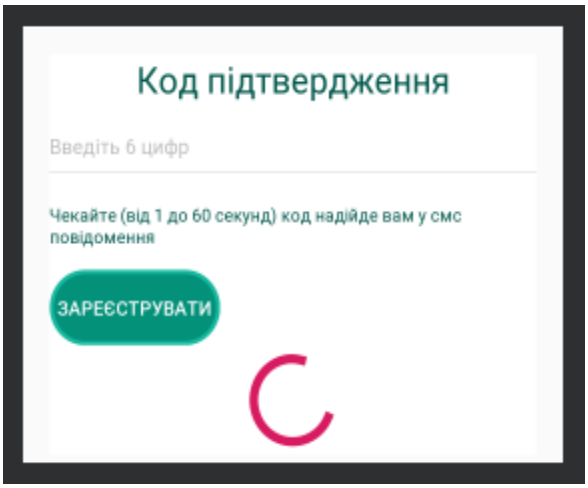


Рисунок 3.3 Код підтвердження після реєстрації

Аналогічні компоненти до попередніх двох активіті, реалізовані для входу в додаток рис. 3.4. Посилання на файл рис. 3.4, https://github.com/Vadym-Popovych24/PNU_APP/blob/master/ app/ src/ main/res/layout/activity_signin.xml або в дод. А.

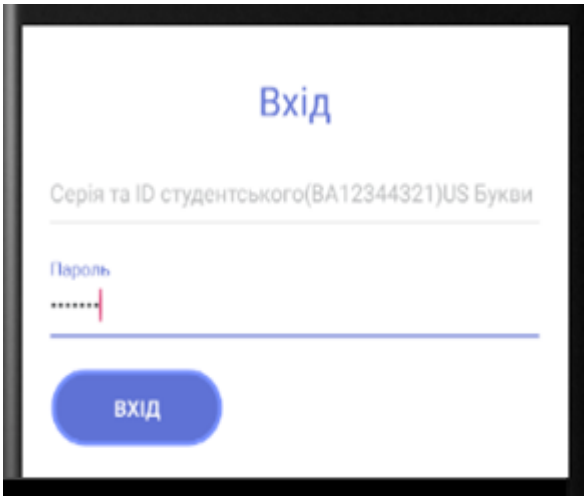


Рисунок 3.4 Вхід

Після реєстрації або входу, користувач потрапляє на головну активіті. Ця активіті містить у верхні частині зображення профілю, в круглі формі. Ідея круглих зображень взята з інших популярних соціальних мереж, оскільки кругле зображення займає менше місця на екрані та краще візуально сприймається. З правої сторони від зображення міститься ім’я та прізвище користувача, а також статус в мережі. Під ім’я, розміщуються три кнопки з кількістю друзів,

підписників та опитувань відповідно, під даними кнопками розміщується велика кнопка загрузки фото профілю, вона має подібний дизайн до кнопки входу.

Далі розміщується контейнер LinerLayout, з вертикальної орієнтацією, він розміщує блок інформації з ІД карти студента: назва групи, дата вступу, назва факультету, форма навчання, між полями розміщені лінії для кращого вигляду блоку інформації. Під блоком інформації студента розміщуються контейнер LinerLayout з горизонтальною орієнтацією який розміщує зменшене зображення профілю, кнопку завантаження зображення для запису, поле для вводу тексту запису та кнопку відправлення запису.

Нижче під контерйнером LineLayout розміщуються кількість записів та самі записи. Якщо записів у користувача нема то виводить TextView з надписом – “У вас немає записів”, якщо ж записи є то даний компонент приймає значення атрибута visibility=gone, тобто стає недоступним. Якщо записи є то вони розміщується в компоненті RecyclerView. Даний компонент передбачений для оптимізації роботи із списками, в моєму випадку список записів, на рис 3.5, цей компонент позначається списком з Items, де Item це шаблон – Layout файл. Загальною рекомендацією при роботі з цим компонентом є використання патерну ViewHolder. Суть патерну полягає в тому, що для кожного елемента списку створюється об'єкт, який зберігає посилання на окремі шаблони всередині елемента. Роботу паттерна ViewHolder описано у пункті 3.2.2. Файл шаблону запису наведений на рис. 3.6. Посилання на файл рис 3.5, https://github.com/Vadym-Popovych24/PNU_APP/blob/master/app/src/main/res/layout/activity_main_student_page.xml або в дод. А.

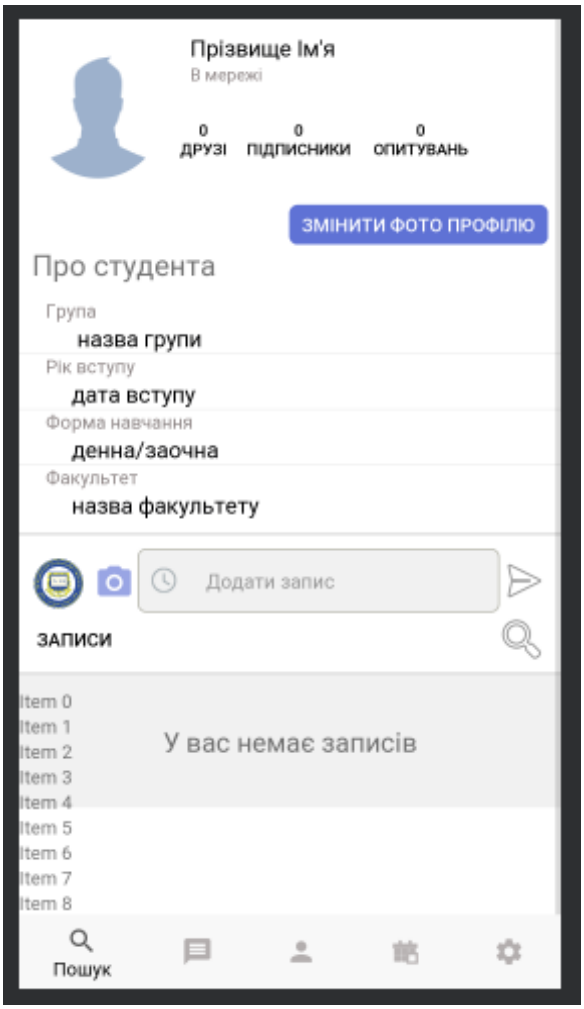


Рисунок 3.5 Головна сторінка студента

Шаблон запису на рис. 3.6 використовується в компоненті RecyclerView на рис. 3.5, для кожного Item, тобто для кожного запису. У верхній частині, зображення, ім’я та прізвище користувача що опублікував запис, час публікації поста параллельно розміщено кнопку налаштування поста. Нижче цього блоку розміщується контент запису, це текст і зображення або тільки текст, якщо тільки текст то компонент зображення приймає значення visibility=gone і стає недоступним. Далі під контентом розміщується контейнер з кнопками діями поста лайки, коментарі, репости та текстовими полями з їх кількістю. У якості фону кнопок використовуються відповідні зображення(drawable файли), які задаються атрибутом background із значення шляху до drawable файлу. Під контейнером з кнопками посту, розміщується RecyclerView – список з коментарями – це вже список, що вкладений у список постів, тобто кожен Item

на рис. 3.5 буде мати свій вкладений список. Нижче коментарів розміщується контейнер з горизонтальною орієнтацією, що містить зменшене фото користувача, зображення смайлика(емодзі), при кліку на який відкриється набір смайликів – це реалізується завдяки додатковій бібліотеці SuperNove-Emoji. Яка надає додатковий компонент EmojiIconActions [17]. Також даний контейнер містить поле вводу коментаря, кнопку надсилання та горизонтальну лінію яка розділяє пости для кращої читабельності та пошуку контенту. Посилання на файл рис. 3.6 https://github.com/Vadym-Popovych24/PNU_APP/blob/master /app /src/main/res/layout/post_layout.xml або в дод. А.

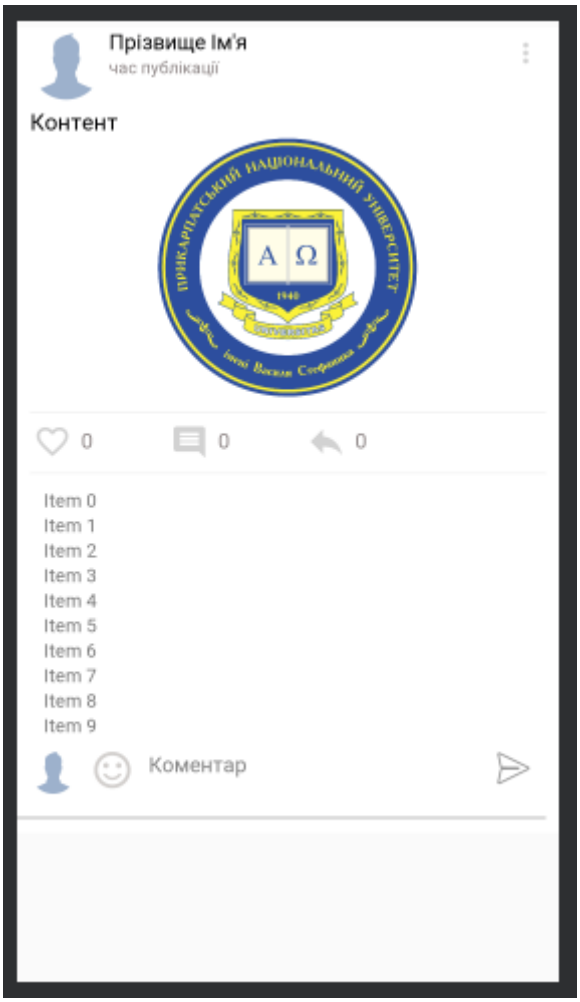


Рисунок 3.6 Шаблон запису

Оскільки шаблон на рис. 3.6 містить компонент RecyclerView і необхідно реалізація списків коментарів для запису, необхідно реалізувати додатковий шаблон для коментарів, його показано на рис. 3.7. Цей шаблон схожий до

шаблону рис 3.6 з різницею лише, що час розташовується під коментарем, а не над ним. А лайки відображаються з правого боку паралельно часові коментару і кнопки “відповісти” якої нема в попередньому шаблоні. Кнопка “Відповісти” дає можливість відповісти на коментар, тобто зробити вкладений коментар, це дозволяє коментувати будь-які коментарі що робить спілкування в додатку більш гнучким. При кліку на дану кнопку з’явиться контейнер із зображення користувача, емодзі, полем вводу та кнопкою надіслати. Так само як і попередній шаблон є кнопка налаштування, як знаходиться з правої сторони паралельно до імені та прізвища користувача, а також компонент RecyclerView для відображення вкладених коментарів. Для кожного Item на рис. 3.7 рекурсивно використовується цей же самий шаблон рис 3.7, шо дозволяє будувати вкладені коментарі будь-якої ієрархічної глубини. Посилання на файл шаблона рис 3.7 https://github.com/Vadym-Popovych24/PNU_APP/blob/master /app/src/main/res/layout/comment_layout.xml або в дод. А.

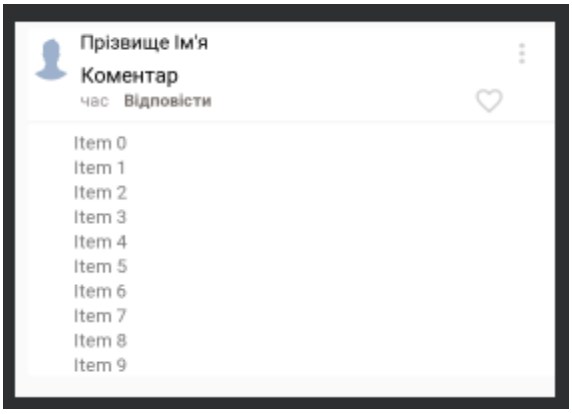


Рисунок 3.7 Шаблон коментаря

Ну і останнім компонентом рис 3.5 є меню. Меню – BottomNavigationView, використовується в багатьох активіті додатку воно завжди розміщене в низу екрана та має відступи від компонентів що знаходяться вище і це єдиний компонент що не входить в контейнер ScrollView. Це дозволяє при скролінгу сторінки тримати меню завжди непорушно у фіксованому місці на нижній частині екрана. BottomNavigationView складається з п’яти пунктівголовна сторінка, пошук, месенджер, налаштування, розклад. Пункти розташовуються в

окремому layout файлі де задаються їх додаткові параметри у вигляді іконок меню та інше. Посилання на layout файл меню https://github.com/Vadym-Popovych24/PNU_APP/blob/master/app/src/main/res/ menu/ menu_bottom_navigation.xml або в дод. А.

Повідомлення, а точніше месенджер реалізується в додатку, подібно до записів на стіні користувача на основі списку та шаблонів, активіті меседжера складається лише текстового поля із текстом – “Повідомлення”, лінії під цим полем, progresbar і меню, все інше реалізується в RecyclerView. Посилання на layout файл рис. 3.8, https://github.com/Vadym-Popovych24/PNU_APP/blob /master /app/src/main/res /layout/activity_messenger.xml або в дод. А.

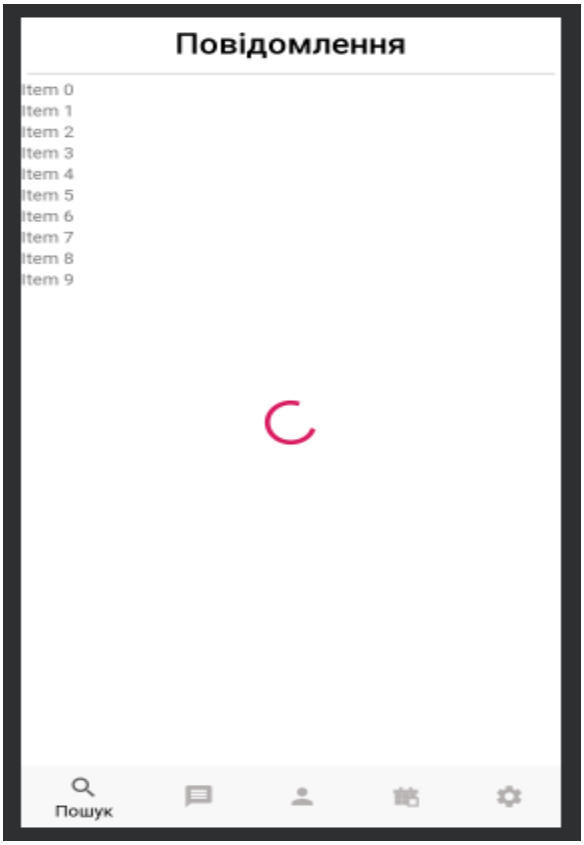


Рисунок 3.8 Месенджер

Шаблон для співбесідника складається з прізвища, ім’я, часу,останнього надісланого чи отриманого повідомлення користувача та із зображень: фото профілю користувача, онлайн позначки на зображенні – це теж додаткове окреме зображення яке стає видимим коли користувач в системі і не видимим коли

користувач не в системі (visibility = visible або gone відповідно), зображення двох галок для прочитаних повідомлень та однієї галки успішного відправлення повідомлення. Для позначки непрочитаних повідомлень служить TextView з заданням червоного круглого фону з цифрою непрочитаних повідомлень по середині [18]. Звичайно, що всі одночасно стани зображень на рис. 3.9 продемонстровані тільки для пояснення. Всі вони включаються чи виключаються динамічно в процесі виконання програми залежно від статусу користувача в мережі та даних з бази. Посилання на layout файл рис 3.9 https://github.com/Vadym-Popovych24/PNU_APP/blob/master/app/src/main/res/layout/messenger_layout.xml або в дод. А.

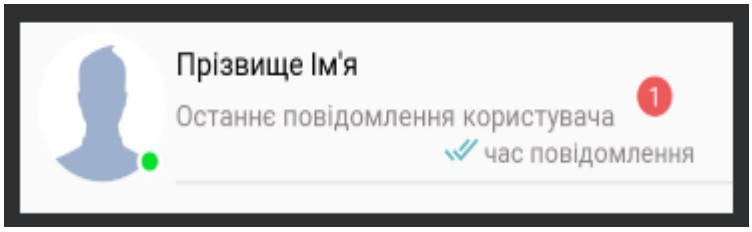


Рисунок 3.9 Шаблон співбесідника для месенджера

Активіті переписки, подібна до інших популярних месенджерів, у верхній частині розміщується інформація про співбесідника – зображення, прізвище, ім’я, час останнього візиту або онлайн статус та кнопка повернутись назад. В нижній частині емодзі, текстове поле для вводу та кнопка надіслати повідомлення. Повідомлення розміщуються в ListView – прокручуваний список елементів. В списку розміщуються повідомлення шаблону, рис. 3.11. Посилання на layout файл рис. 3.10 https://github.com/Vadym-Popovych24/PNU_APP/blob/master/app/src/main/res/layout/activity_message.xml або в дод. А.

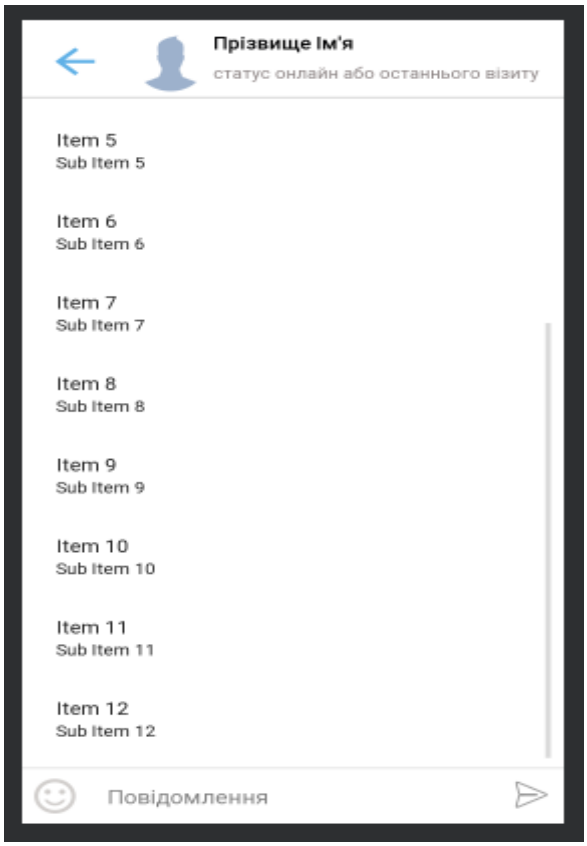


Рисунок 3.10 Сторінка з списком повідомлення користувача

Даний шаблон містить текст повідомлення, час повідомлення, та іконки однієї та двох галок, відправлених та прочитаних повідомлень відповідно. Під час виконання програми шаблон повідомлення змінюється, власні повідомлення розміщуються з правої сторони та мають колір як на рис. 3.11, повідомлення співбесідника розміщуються з лівої сторони активіті та мають сірий фон. Посилання на layout файл рис. 3.11 https://github.com/Vadym-Popovych24/PNU_APP/blob/master/app/src/main/res /layout/list_item.xml або в дод. А.

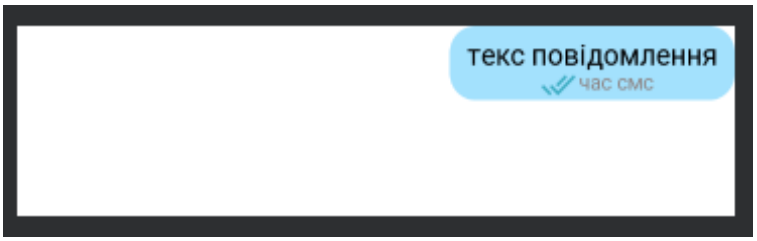


Рисунок 3.11 Шаблон повідомлення

Сторінка профілю іншого користувача, схожа на сторінку власного профіля, що на рис. 3.5, з різницею лише, що замість кнопки “змінити фото профілю”, розміщено дві кнопки – “написати повідомлення” та “додати в друзі”, тому не має потреби її розглядати детально.

Додаток містить багато активіті зі списками користувачів, які формуються по певним критеріям, друзі, підписники, читачі, отримані та надіслані заявки в друзі, пошук серед усіх користувачів, всі списки на профілі іншого користувача і тд. Всі вони мають однакову структуру та сильно схожі шаблони, які відрізняються одною або двома кнопками, тому розглянемо тільки один список для прикладу рис. 3.12, список друзів.

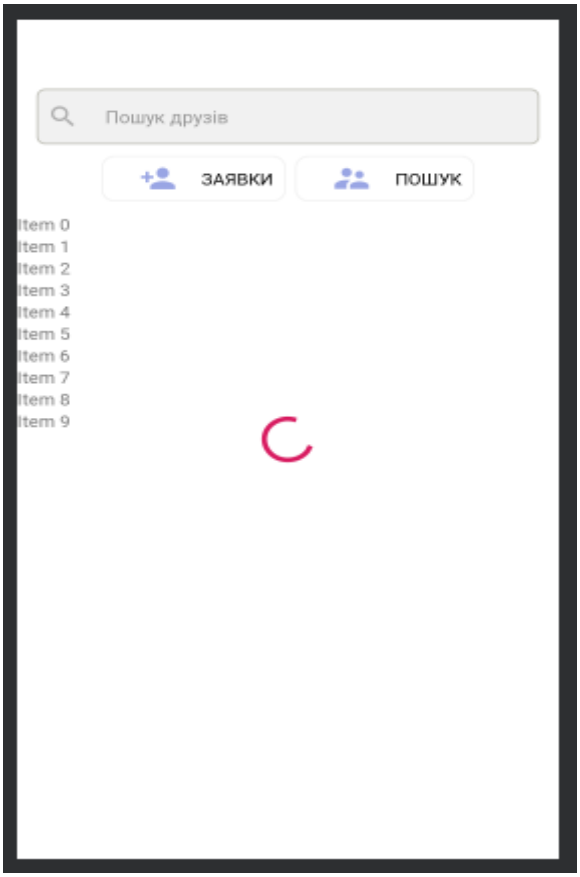


Рисунок 3.12 Список друзів

Всі списки формуються вже по описаному шляху це компонент RecyclerView і шаблони. Шаблони відрізняються як правило однією або двома кнопками, наприклад: друзі шаблон рис. 3.13 – кнопка надіслати повідомлення, шаблон отриманих запитів в друзі рис. 3.14 – кнопки додати в друзі та відхилити,

всі інші шаблони підписників, читачів та інші є аналогічними до наведених тому розглядати їх не будемо. Посилання на layout файл рис. 3.12 https://github.com/Vadym-Popovych24/PNU_APP /blob /master /app/src/main/res/layout/fragment_friends.xml або в дод. А.

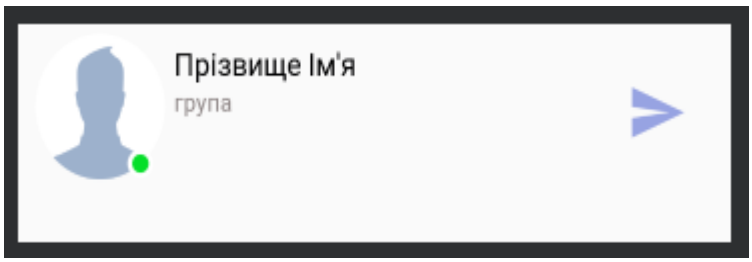


Рисунок 3.13 Шаблон друга в списку

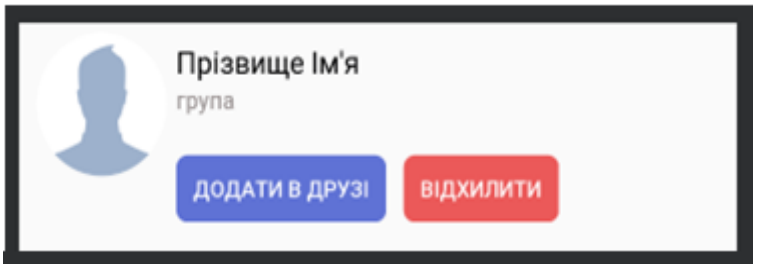


Рисунок 3.14 Шаблон отриманих запитів

Всі шаблони, фрагменти, та активіті додатку можна знайти за посилання https://github.com/Vadym-Popovych24/PNU_APP/tree/master/app/src/main/res/ layout або в дод. А.

Списки користувачів розміщуються в фрагментах. Фрагмент - представляє інтерфейс або частини призначеного для користувача активіті. Надає можливість об'єднати декілька списків фрагмента в одну операцію та побудувати багатопанельний інтерфейс і повторного використовувати фрагмент в декількох операціях. Багатопанельний інтефейс дозволяє переключатись між різними списками в одному активіті, наприклад між списками отриманих та надісланих заявок в друзі можна переключатись промотавши список по горизонталі.

3.2 Реалізація java класів

Діаграма класів наведена на рис. 2.12 містить основні методи та класи, що реалізують проект, структуру та список всіх класів наведено у дод. Е . Робота програми починається з точки входу “launcher”. В якості точки входу служить клас MainActivity, він виконує перевірку на авторизацію в методі onCreate(). Якщо користувач вже авторизувався в додатку раніше і не виходив з нього то буде здійснено миттєвий перехід на сторінку профіля користувача без загрузки інтерфейсу MainActivity. Якщо користувач не авторизований то виконується загрузка стартового активіті рис. 3.1, з якої користувач, має змогу здійснити перехід для реєстрації чи входу в додаток. Реалізація класу доступна за посиланням https://github.com/Vadym-Popovych24/PNU_APP/blob/master/app/src /main/java/com/social_network/ pnu_app/ pages/MainActivity.java або в дод. А.

При реєстрації виконується ряд перевірок та верифікацій для даних, що вводяться в поля користувачем. Перевірка здійснюється за допомогою власних регулярних виразів та порівнянь із вже існуючими даними у базі даних. Верифікація допомагає користувачеві в ситуації невірно введених даних, проаналізувавши дані про ті чи інші помилки та виводить відповідне повідомлення. Також верифікація обмежує вводити дані будь-якої довжини, щоб уникнути великих даних в базі даних. Здійснюється перевірка на існуючий ІД студентського квитка та номеру телефону в базі даних. Для безпеки даних реалізовано власний метод шифрування даних на основі метода “MD5”. Реалізація класу зображена на рис. 3.2 і доступна за посиланням https://github.com/Vadym-Popovych24/PNU_APP/blob/master/app/src/main/java/ com /social_network/pnu_app/registration/Registration.java або в дод. А.

Після класу реєстрації слідує клас з авторизацією по номеру телефона. Як тільки користувач після успішно пройденої сторінки з реєстрацією з усіма вірно введеними даними, він потрапляє на сторінку з авторизацією, йому автоматично надсилається код з шести цифр на номер телефону який він ввів при реєстрації.

Надсилання коду та авторизація здійснюється за допомогою наданих класів та методів платформи Firebase сервісу FirebaseAuthentication. А саме класу PhoneAuthProvider, методу sendCodeVerification() та перевизначених методів onVerificationCompleted(), onVerificationFailed(), onCodeSent(), signInWithPhoneAuthCredential(). Останній реєструє користувача безпосередньо в в самій платформі Firebase, надаючи йому унікальний ключ, поставника у вигляді номеру телефона користувача, час створення та час останнього входу. Після того як користувач отримав та ввід код, здійснюється перевірка на правильність коду, у разі успішної перевірки, дані введені користувачем вносяться в у віддалену базу даних Firebase RealTimeDatabase та в локальну SQLite, яку було описано в пункті 2.1 - програмні засоби та середовище розробки. Програма постійно працює з віддаленою базою даних, але бувають випадки коли на смартфоні немає доступу до мережі. Щоб не виводити просте повідомлення про відсутність мережі, а показувати сторінки користувачеві з даними, використовується локальна база даних. Таким чином будуть відображатися і зображення ссилки на яких є в локальні базі даних, оскільки інструменти завантаження зображень кешують дані та дозволяють завантажувати їх без доступу до інтернету. Також локальна база даних є корисною коли потрібно отримати унікальний ключ даного користувача чи якісь інші його дані, оскільки операція отримання даних з локальної бази є значно швидшою, та економнішою в плані ресурсів. Після успішного завантаження даних з бази даних здійснюється перехід на сторінку користувача, у разі невірного коду виводиться відповідне повідомлення. Смс може приходити від 1 секунди в до декількох хвилин. Тому користувачеві надається елемент ProgresBar(кільце загрузки), елемент буде крутитись, якщо всі процеси здійснюються успішно і зупиниться у разі їх закінчення. Це допоможе користувачеві розуміти, що відбувається в даний момент і що необхідно трохи зачекати. Під час тестування мною, код приходив за декілька секунд на вказаний номер. Сервіс FirebaseAuthentication надає додатковий захист від зловмисників,

користувач може отримувати код перевірки тільки чотири рази протягом восьми годин, якщо кількість перевищить чотири рази то код не буде приходити протягом наступних восьми годин. Реалізація класу зображена на рис. 3.3 і доступна за посиланням https://github.com/Vadym-Popovych24/PNU_APP/blob/master/app/src/main/java/com/social_network/pnu_app/registration/PhoneAuthentication.java або в дод. А.

Вхід в додаток здійснюється за допомогою полів пароля, ІД студентського та номеру телефону. Явно вказувати потрібно тільки перші два поля, перевірка по номеру телефону здійснюється автоматично по раніше зареєстрованому, що відповідає введеному ІД студентського. На номер телефону приходить смс, як і у випадку реєстрації, отриманий код вводити не потрібно, як тільки користувачу на номер телефону надійде код, система автоматично здійснить вхід в систему. Перевірка по номеру телефона потрібна для сервісу FirebaseAuthentication, щоб здійснити в нього вхід і отримати доступ до інших сервісів та ресурсів. Для входу на платформу Firebase, використовуються ті ж методи та класи, що і при реєстрації. З різницею лише в тому, що метод signInWithPhoneAuthCredential() – який відповідає за вхід на платформу, викликається в середині метода onVerificationCompleted() – викликається при успішному надсиланні коду авторизації на номер телефону користувача. За допомогою такої послідовності викликів методів і досягається автоматична перевірка по номеру телефона, без додаткового введення коду авторизації. Реалізація класу зображена на рис. 3.4 і доступна за посиланням https://github.com/Vadym-Popovych24/PNU_APP/blob/master/app/src/main/java/com/social_network/pnu_app/signin/SignIn.java або в дод. А.

Головна сторінка, як було сказано в пункті 3.1 містить інформацію про користувача, списки друзів, підписників, опитувань, пости, меню і кнопку зміни фото профілю. Дані про користувача завантажуються з RealTime Database, якщо немає підключення до інтернету то дані беруться з локальної SQLite бази, всі дії які пов’язані із завантаженням даних в компоненти інтерфейсу виконуються в

методі BuildStudentPage() . Зміна фото профілю – це процес який складається із декількох методів. Спочатку здійснюється виклик методу addPhoto() –перевіряє доступ до файлів сховища девайсу та камери. Якщо є доступ, будує нову сторінку та здійснює на неї перехід. Новоутворена сторінка містить список програм системи, за допомогою яких можна перейти на фотографії девайсу – зазвичай це галерея, диспетчер файлів та камера. Можуть бути і інші залежно від встановлених на операційну систему та її версії. Відкривши зі списку камеру можна зробити миттєве фото і воно буде вибране як фото для профілю, відкривши інший засіб для завантаження фото здійснюється його пошук та вибір. Для вибраного фото робиться стиск зображення[19]. Бібліотека “Compressor” - дозволяє стискати великі фотографії на фотографії меншого розміру з дуже малими або незначними втратами в якості зображення. Компресор дозволяє стискати зображення по різним параметрам: максимальна/мінімальна ширина/висота, якість стиску та інше. Стиск зображення в середньому дозволяє економити пам’ять у віддаленому сховищі у 5-100 раз, залежно від вказаних параметрів якості стиску, ширини і висоти зображення [20]. Стиснуте зображення, завантажується в компонент CircleImageView, за допомогою бібліотеки Picasso.

Picasso – дозволяє завантажувати зображення по url, або по шляху до локального файла - Uri. Перед завантаження зображення в компонент, бібліотека надає різні можливості для обробки зображення та випадків у разі невдалої загрузки. Нижче наведено лістинг завантаження зображення засобами класу Picasso.

```
Picasso.with(getBaseContext())
                .load(data.getData())
                .placeholder(R.drawable.logo_pnu)
                .error(R.drawable.com_facebook_close)
                .centerInside()
                .into(imStudentMainPhoto);
```

Метод with() – вказує контекст для завантаження. Метод load() – вказує шлях до зображення. Placeholder() – вказується шлях до файлу чи шаблону який буде відображатися на місці зображення під час його загрузки. Error() - вказується шлях до файлу чи шаблону який буде відображатися на місці зображення у разі невдалого завантаження. CenterInside () – обрізає та масштабує зображення так, щоб обидва розміри були рівними або меншими, ніж запитувані межі CircleImageView. Зображення буде відображено повністю, але може не заповнити весь CircleImageView, into() – вказується компонент в який потрібно завантажити зображення. Обрізати зображення потрібно для того, щоб уникнути помилки при завантаженні, оскільки зображення розміри яких є більші ніж 3225x4800, не підримуються для завантаження в компоненти зображень. Picasso також реалізовує кешування зображень та їх загрузку в окремому потоці, для швидшого опрацьовування. Дана бібліотека використовується не тільки для завантаження фото профілю користувача, але і для всіх інших зображень які є в додатку . Паралельно роботі бібліотеки Picasso, зображення завантажується в сховище Firebase Storage. Після чого отримується url зображення та встановлюється його в базу даних Firebase RealTime Database. Таким чином здійснюється подальший доступу до зображення для бібліотеки Picasso через url, що стягується з бази даних [21]. Реалізація головної сторінки студента наведена нижче на рис. 3.15, 3.16, 3.17. Реалізація класу доступна за посиланням https://github.com/Vadym-Popovych24/PNU_APP/blob/master/app/src/main/java/com/social_network/pnu_app/pages/MainStudentPage.java або в дод. А.

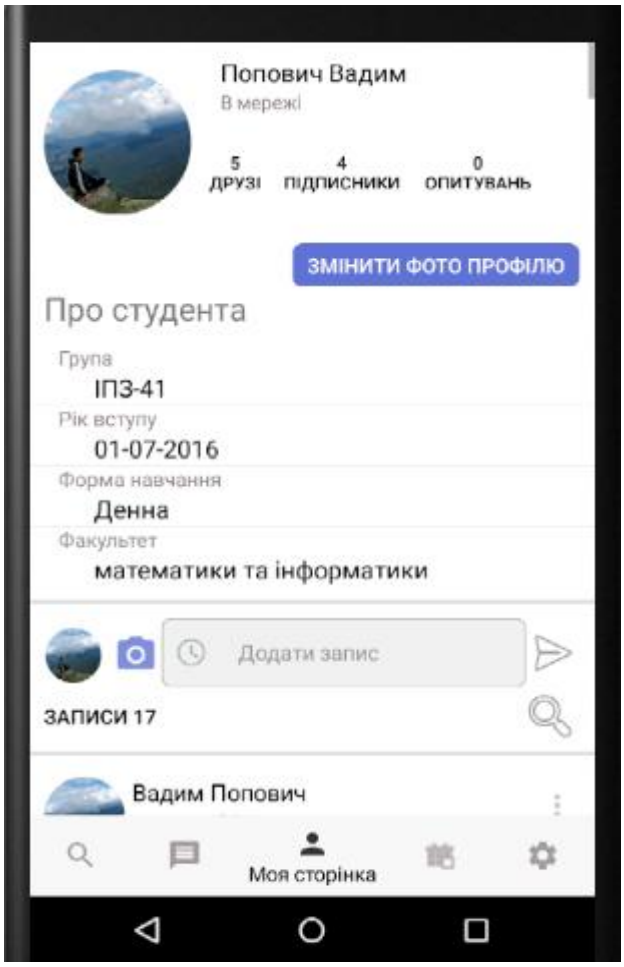


Рисунок 3.15 Головна сторінка студента

Нижче даних про користувача розміщується горизонтальний контейнер з полем для вводу тексту запису, кнопкою для завантаження фото у запис у вигляді фотокамери збоку поля для вводу тексту. Зліва від цієї кнопки зменшене фото профілю користувача для кращого дизайну та візуального сприйняття, та кнопка надіслати запит. Під цим контейнером розміщується поле з кількістю записів користувача та самі записи.

Запис складається з прізвища, ім’я, та фото автора запису, дати публікації, та контенту. Дані запису встановлюються та загружаються з RealTime Database так як показоно в розділі 2 на рис. 2.10 діаграм бази даних у вигляді дерева. Записи зберігаються у вузлі під назвою ”Posts” - цей вузол має кожен користувач окремо, із записами що розміщені на його стіні. Кожен запис генерується новим вузлом дерева у вузлі “Posts” з унікальним іменем для вузла(ключем). Вузол запису містить поле ключ вузла користувача за допомогою якого витягуються

ім’я, прізвище, ссилка на фото профілю, поле з датою та часом створення запису, поле з текстом запису, поле з ссилкою на фото запису, поле з кількістю репостів, та вузли з лайками та коментарями.

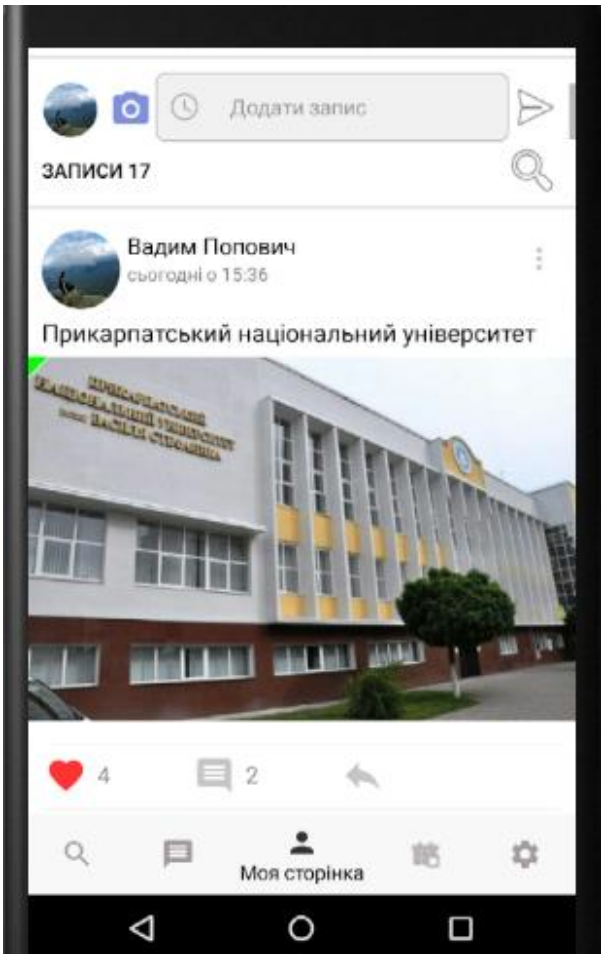


Рисунок 3.16 Записи на сторінці користувача

Лайки працюють так як і всіх інших соц мережах, при кліку зображення “лайка” змінює білий фон на червоний, якщо лайк вже поставлений під даним записом то його фон червоний і якщо навпаки то фон білий. Вузол “likes” кожного запису містить ключі користувачів хто оцінив запис, таким чином здійснюється перевірка на наявність даного користувача у вузлі “likes” та поставленого лайка, кількість лайків це кількість ключів у цьому вузлі.

Коментарі до запису містять таку саму інформацію про автора, як і про автора запису, тільки компоненти дизайну, та розміри шрифтів є трохи меншими. На відміну від записів – в яких послідовність зберігається за правилом: новий запис добавляється в початок списку записів і спочатку відображаються ті

записи, що були додані останніми. Коментарі мають зворотню послідовність: новий коментар добавляється в кінець списку коментарів і спочатку відображаються старіші коментарі. Як і записи коментарі можуть містити лайки, вони працюють так само як і для постів для вузол likes створюється у вузлі comments, а не у Posts. Також до коментарів є можливість робити відповідь(вкладений коментар), який використовує всі можливості початкового коментаря, та рекурсивно створює такий самий дочірній вузол comments, у батьківському вузлі comments.

Вкладені коментарі розміщуються під батьківським коментарем та трохи зміщені вправо відносно нього. Це дозволяє швидко їх розрізняти, також текст ім’я та прізвище відправника вкладеного коментаря мають сірий колір та додатково супроводжуються словом Відповідь. Для того щоб зробити вкладений коментар необхідно натиснути кнопку Відповісти, після чого з’явиться поле для вводу тексту. Збоку поля для вводу розміщується кнопка у вигляді смайлика, при кліку на неї з’являється клавіатура для вибору різних смайликів.

Коментарі та записи містять кнопку налаштувань, яка розміщена з правої сторони екрану у вигляді трьох крапок, вона дозволяє видаляти коментарі та записи. Реалізацію смайликів, звичайних та вкладених коментарі та наведено нище на рис. 3.17 [22].

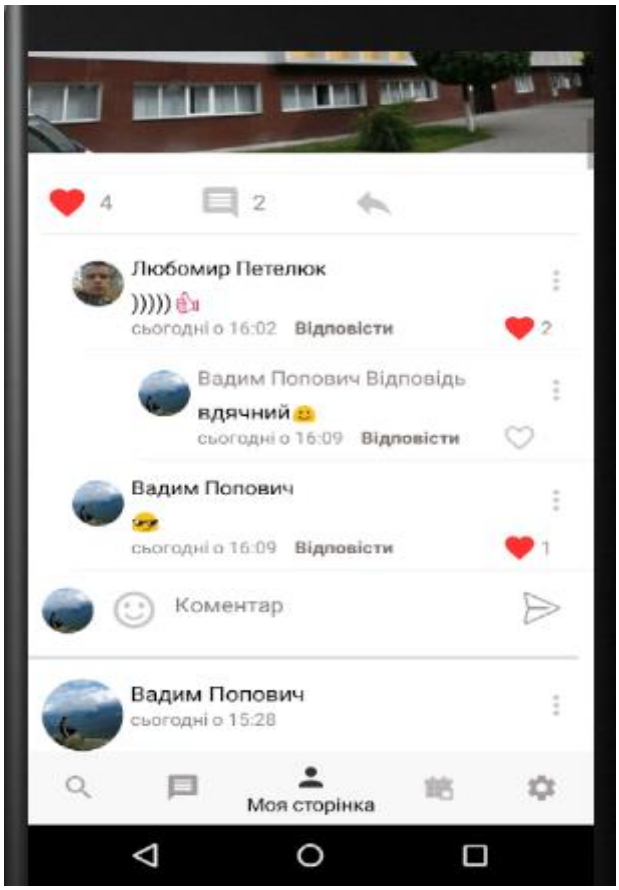


Рисунок 3.17 Коментарі до запису на сторінці користувача

Будь-який запис, можна скопіювати та добавити(репост) собі на сторінку, це можна зробити за допомогою клавіші, що розміщена най правіше під записом. Клавіша, що розміщена по центру під записом просто показує кількість коментарів(без вкладених). Якщо зробити клік на цифру, що розміщена біля клавіші лайків, на рис. 3.17 це цифра 4, то відкриється список людей, що оцінили(поставили лайк) даний запис. Із списку можна здійснити перехід на сторінку користувача просто здійснивши клік по відповідному користувачеві. Реалізацію списка користувачів, що оцінили запис наведено на рис. 3.18. Реалізація класу доступна за посиланням https://github.com/Vadym-Popovych24/PNU_APP/blob/master/app/src/main/java/com/social_network/pnu_app/pages/ActivityListLikesPost.java або в дод. А.

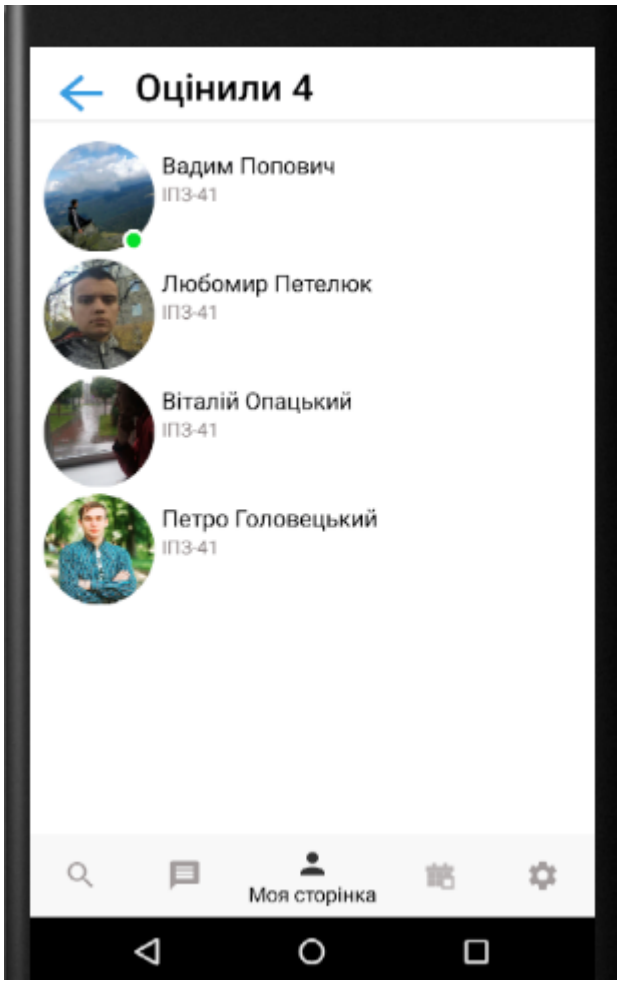


Рисунок 3.18 Список користувачів що оцінили запис

З головної сторінки можна перейти до списку друзів, натиснувши на кнопку друзі, також дана кнопка показує кількість ваших друзів. В базі даних всі друзі користувача відображаються у вузлі користувача під назвою Friends [23]. Кожен новий користувач це новий вузол з назвою ключа користувача. Кількість друзів це кількість вузлів у Friends. Реалізацію списка друзів наведено на рис. 3.19. Реалізація класу друзів доступна за посиланням https://github.com/Vadym-Popovych24/PNU_APP/blob/master/app/src/main/java/com/social_network/pnu_app/pages/FriendsFragment.java або в дод. А .

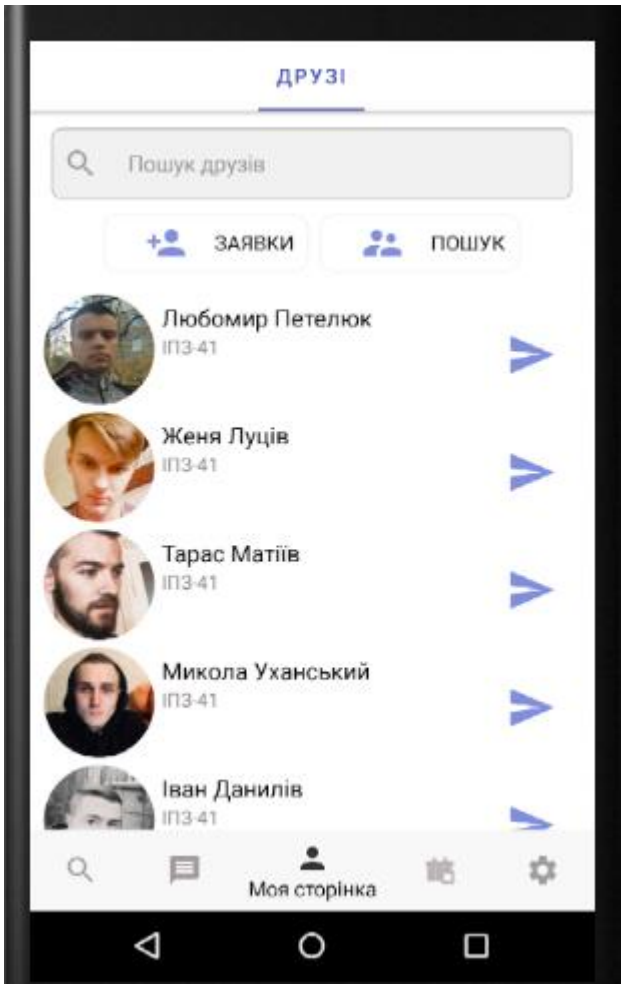


Рисунок 3.19 Список друзів

Як і на рис. 3.18, для кожного користувача відображається ім’я, прізвище, група в якій він навчається та фото профілю. Натиснувши на друга здійснить перехід на його сторінку, додатково реалізована кнопка – надіслати повідомлення, натиснувши на неї можна перейти в діалог з другом. Також реалізовано пошук по імені серед друзів, та перехід на сторінку всіх користувачів кнопкою пошук. Переглянути список отриманих та надісланих заявок можна натиснувши кнопку заявки. Реалізацію списка отриманих заявок наведено на рис. 3.20. Реалізація класу отриманих заявок доступна за посиланням https://github.com/Vadym-Popovych24/PNU_APP/blob/master/app/src/main/java /com/social _network/pnu_app/pages/RequestsFriendsFragment.java або в дод. А .

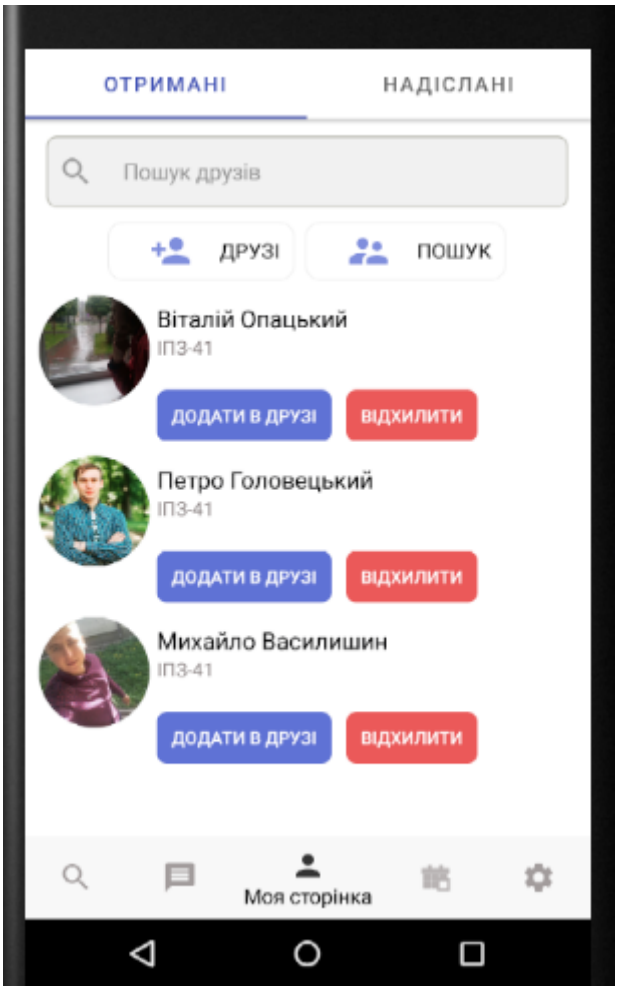


Рисунок 3.20 Список отриманих заявок в друзі

Для отриманих та надісланих заявок створюються у дереві відповідні вузли по аналогії до вузла Friends. Дана сторінка аналогічна до сторінки із друзями, додатково реалізовано в списку дві кнопки: додати в друзі та відхилити. Перша кнопка добавляє користувача в друзі, в базі даних користувач переміщуються із вузла заявок(FriendRequestReceiver) у вузол друзів(Friends). Друга кнопка добавляє користувача в підписники, в базі даних користувач переміщується із вузла заявок у вузол підписників(Subscribers) [24]. Натиснувши на кнопку надіслані або протягнувши пальцем по екрану з права на ліво здійниться перехід на список надісланих заявок в друзі. Реалізацію списка надісланих заявок наведено на рис. 3.21, реалізація класу доступна за посиланням https://github.com/VadymPopovych24/PNU_APP/blob/master/app/src/main/java/co

m/social_network/pnu_app/pages/SenderedRequesrsFriendsFragment.java або в дод. А.

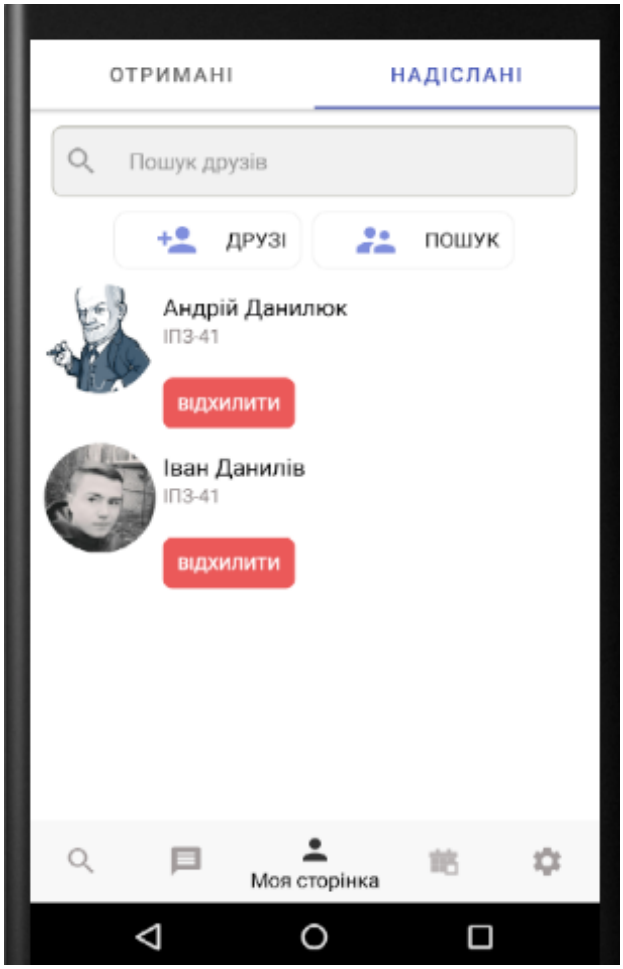


Рисунок 3.21 Список надісланих заявок в друзі

Дана сторінка аналогічна до попередньої, з різницею лише в кнопці відхилити заявку. При натиску на неї в базі даних з вузла надісланих вами заявок, видаляється вузол з ключем користувача, якому була наділсана заявка та з вузла отриманих заявок іншого користувача, видаляється користувач з вашим ключем. Так як із двох попередніх сторінок, можна здійснити перехід для пошуку друзів серед усіх інших користувачів. Реалізацію списка всіх користувчів наведено на рис. 3.22, реалізація класу доступна за посиланням https://github.com/Vadym-Popovych24/PNU_APP/blob/master/app/src/main/java/ com/social_network/pnu_app/pages/FindNewFriends.java .

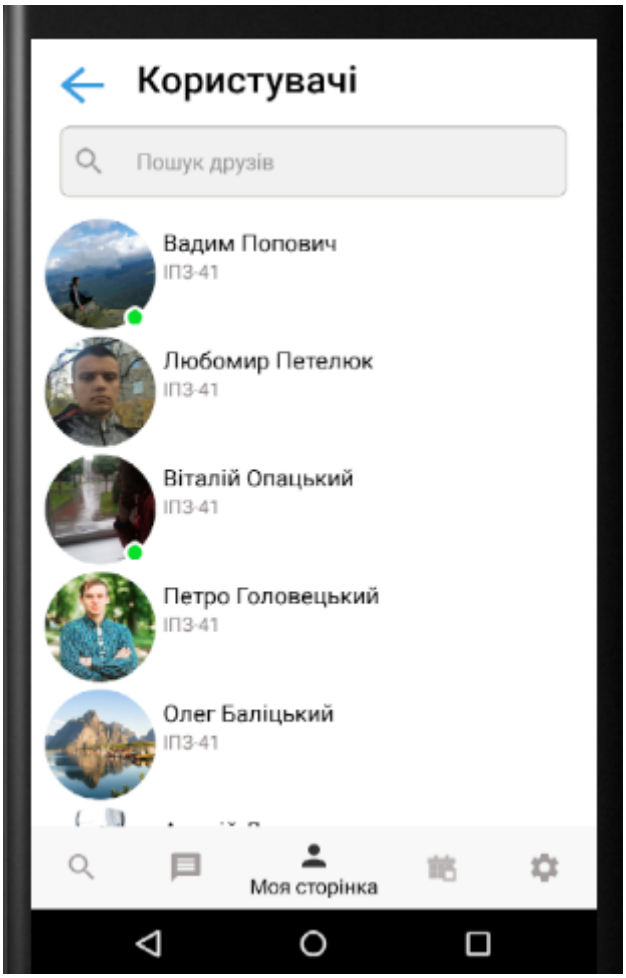


Рисунок 3.22 Список всіх користувачів

Даний список призначений для пошуку користувачів та переходу на їх профілі для подальших дій. Кожен список тих чи інших користувачів знизу на фото профілю користувача має мітку у вигляді зеленого круга, яка означає що користувач зараз онлайн, якщо мітка відсутня то користувач не в мережі. Мітка це звичай компонент круглого зображення із зеленим фоном та білою межею, який має задані параметри розміщення такі що частково перетинаються із зображенням фото профілю користувача. Мітка змінює видимість залежно від значення поля online користувача в базі даних. Поле online може приймати два значення true та false відповідно. Значення для поля online встановлюються в методах: onPause(), onResume(). Ці методи є перевизначеними від батьківського класу Activity. Метод onPause() викликається у випадку коли сторінка(активіті) не є у фокусі і користувач не може з ним взаємодіяти, наприклад коли сторінка

згорнута чи закрита, при виклику даного метода, поле online приймає значення false. В поле lastSeen встановлюється поточний час визначений сервером Firebase це поле використовуються для відображення часу останнього візиту користувача в додаток .Метод onResume() викликається у випадку коли сторінка є у фокусі і користувач може з нею взаємодіяти при виклику даного метода, поле online приймає значення true. Таким чином ці два методи викликаються для кожної сторінки і за допомогою засобів RealTime Database дозволяють відображати статус онлайн чи офлайн в реальному часі. Додаткова перевірка на наявність користувача у мережі здійснюється за допомогою метода onDisconnet(), що надається платформою Firebase.

По аналогії до кнопки друзів, реалізовано кнопку підписники. В списку підписників містяться користувачі, які були видалені з друзів або були відхилені заявки в друзі кнопкою що на рис. 3.20 або на сторінці профілю користувача.

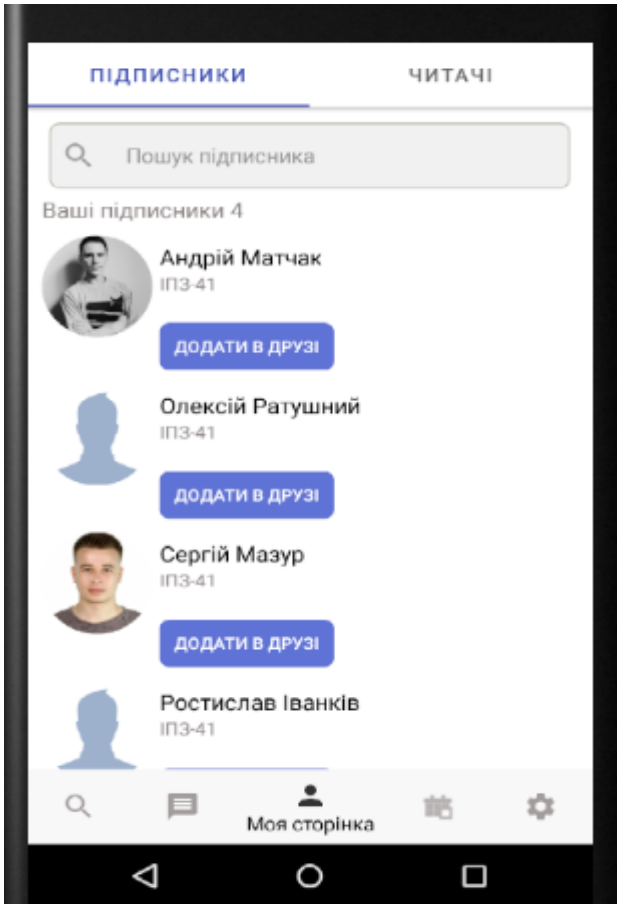


Рисунок 3.23 Список підписників

Кнопка добавити в друзі, переміщує користувача з вузла підписників у вузол друзів і у вузлі користувача з читачів у друзі. Якщо користувач не встановив фото для профілю то для нього буде відображатися зображення по замовчуванню, яке продемонстровано на рис. 3.23 для двох користувачів. Реалізація класу підписників доступна за посиланням https://github.com/Vadym-Popovych24/PNU_APP/blob/master/app/src/main/java/com/social_network/pnu_app/pages/MySubscribersFragment.java або в дод. А.

Другим фрагментом сторінки підписників є читачі. Це користувачі на яких ви є підписані, тобто ті що видалили вас із друзів або відхилили вашу заявку в друзі. У даному списку можна відписатися від користувача, таким чином у базі даних ви видалите себе у своєму вузлі із вузла читачів, а у вузлі користувача видалитись із вузла підписників. Реалізацію списка читачів наведено на рис. 3.24, реалізація класу підписників доступна за посиланням https://github.com/Vadym-Popovych24/PNU_APP/blob/master/app/src/main/java/com/social_network/pnu_app/pages/MySubscribedsFragment.java або в дод. А.

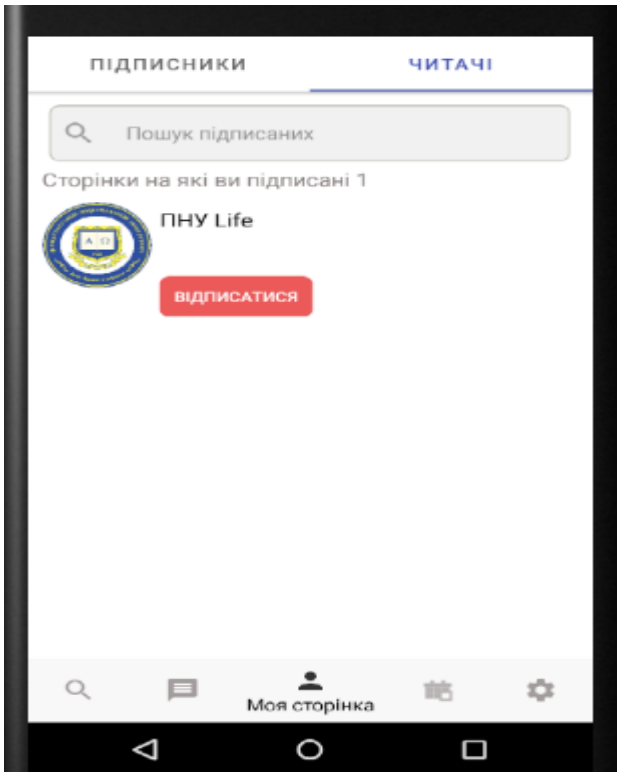


Рисунок 3.24 Список читачів

З списків, що були продемонстровані вище, можна здійснити перехід на сторонку профілю обраного користувача. Вона аналогічна до головної сторінки, різниця в тому, що замість кнопки змінити фото профілю, розміщені дві кнопки: написати та додати в друзі. Перша кнопка здійснює перехід на сторінку переписки з користувачем. Кнопка добавити в друзі, може мати різні стани залежно від того чи є користувач вашим другом, підписником, читачем, чи надіслана або отримана заявка в друзі, або жоден з цих варіантів. Відповідно кнопка може містити такий стан та текст: видалити з друзів, ваш підписник(добавити в друзі), ви підписані(відписатися), скасувати заявку, відповісти на заявку в друзі (додати чи відхилити). В душках наведені варіанти, які будуть відображені у додатковому діалоговому вікні. Перевірка станів здійснюється по вузлах вашого користувача: друзів, читачів, підписників, заявок і встановлюється відповідний стан і текст кнопки. Якщо користувач, сторінку якого ви відвідали не є в жодному із вузлів вашого користувача, то встановлюється текст додати в друзі, фон кнопки встановлюється синім, а текст білим, в усіх інших випадках фон сірий а текст синій. Реалізацію сторінки іншого користувача наведено на рис. 3.25.

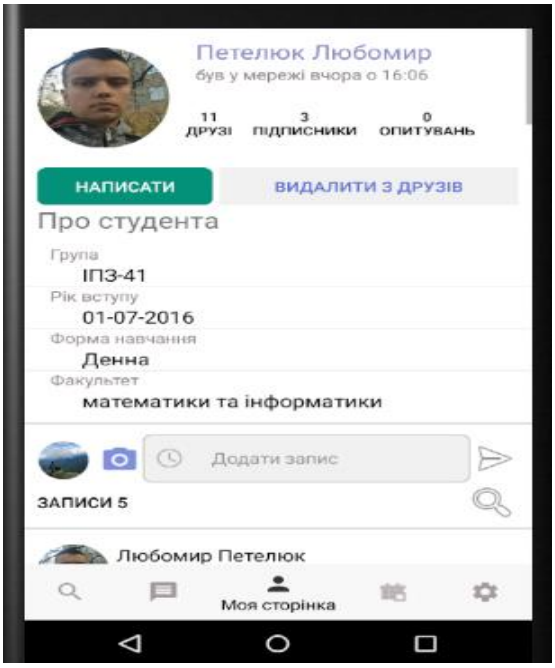


Рисунок 3.25 Сторінка профіля друга

Реалізація класу профіля іншого користувача доступна за посиланням https://github.com/Vadym-Popovych24/PNU_APP/blob/master/app/src/main/java/com/social_network/pnu_app/pages/ProfileStudent.java або в дод. А.

Обмін повідомленнями між користувачами реалізовано у вигляді традиційного месенджера. Сторінка месенджера відображає список співбесідників посортованих за часом останнього повідомлення від кожного користувача. Крім традиційних даних про користувача в списках, месенджер містить останнє повідомлення від користувача, його час та додаткові позначки у вигляді: червоного кола з кількістю непрочитаних повідомлень, та зображень галок. Одна означає успішно доставлене повідомлення до співбесідника, дві галки показують те, що користувач прочитав повідомлення. Повідомлення та їх позначки встановлюються та змінюються динамічно без перезавантаження сторінки. Реалізацію месенджера наведено на рис. 3.26. Реалізація месенджера доступна за посиланням https://github.com/Vadym-Popovych24/PNU_APP/blob/master/app/src/main/java/com/social_network/pnu_app/pages/Messenger.java або в дод. А.

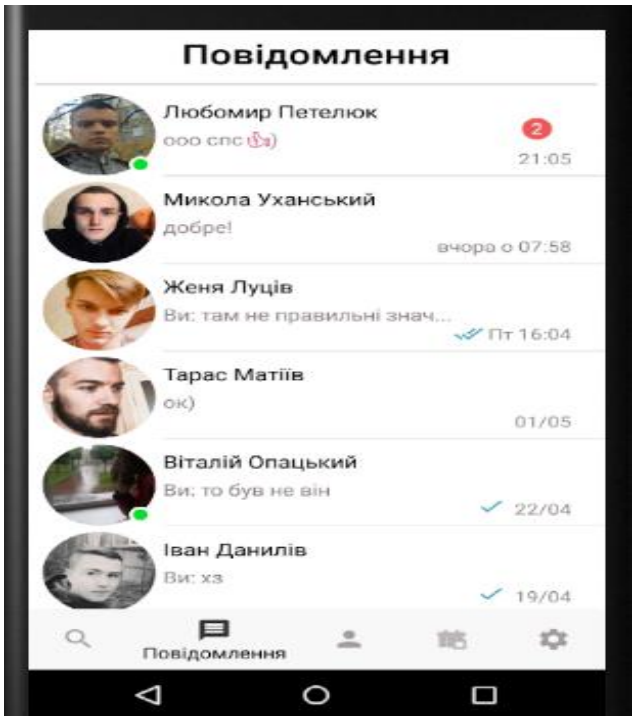


Рисунок 3.26 Месенджер

Сторінку з перепискою реалізовано з урахуванням деталей подібно до інших соц мереж. Зверху сторінки відображається інформація про співбесідника клікнувши на неї можна перейти на його профіль. Особисті повідомлення розміщуються з правої сторони екрану на синьому фоні, повідомлення співбесідника з лівої сторони на сірому фоні. Нижче кожного повідомлення розміщуюється час надісланого повідомлення. Нижче особистого повідомлення розміщуються зображення однієї та двох галок: успішно надісланого, та прочитаного повідомлення іншим користувачем відповідно. В повідомлення реалізовано можливість вставляти емодзі. Реалізацію повідомлень наведено на рис. 3.27. Реалізація повідомлень доступна за посиланням https://github.com/VadymPopovych24/PNU_APP/blob/master/app/src /main/java /com/social_network/pnu_app/ pages/Message.java або в дод. А.

Рисунок 3.27 Сторінка переписки користувача

3.2.1 Асинхронні слухачі дерева даних.

Відображати деякі дані в соц мережі необхідно в режимі реального часу без перезавантаження сторінки, наприклад відображення статусу онлайн, отримання повідомлень та інше. Для цього звичайних запитів у базі даних недостатньо, тому в даному додатку реалізовано асинхронні слухачі для даних дерева. Дані, які зберігаються в базі даних у реальному часі Firebase, витягуються за допомогою виклику асинхронного слухача до посилання на будь-який вузол бази даних . Слухач спрацьовує один раз для початкового стану даних і знову будь-коли, коли дані змінюються у вказаному в посиланні вузлі або його нащадках, залежно від типу слухача який прикліплений до посилання . Існує 3 типи слухачів для вузлів дерева. AddValueEventListener (слухач), спрацьовує при звертанні до будь-якого значенння в казаному дереві чи під дереві. AddChildEventListener(слухач) - спрацьовує при звертанні до вузлів нащадків. AddListenerForSingleValueEvent (слухач) - зворотний виклик буде викликаний один раз, а потім негайно видалений, цей тип виклику корисно використовувати для економії ресурсів для статичних даних , цей виклик працює подібно до звичайного SQL запиту.

Дані, що зберігаються в базі даних Realtime базі даних Firebase, витягуються шляхом виклику методу блокування вузла бази даних, яка повертає дані, що зберігаються у посиланні. Кожен виклик методу - це разова операція. У Java функція зворотного виклику отримує об’єкт типу DataSnapshot, який є знімком даних. Знімок - це зображення даних у конкретному вузлі бази даних за один момент часу. Тип DataSnapshot зберігає дані у вигляді батько – нащадок, ключ – значення, при необхідності їх легко зберігати та використовувати у колекції HashMap() [25]. Всі дані бази даних Realtime зберігаються як об'єкти JSON, можна вважати що це хмарне дерево. На відміну від бази даних SQL, немає ні таблиць, ні записів. Коли додаються дані до дерева JSON, вони стають вузлом у існуючій структурі JSON із пов'язаним ключем. Ключ можна створити

з власним ім’ям, такі як ідентифікатори користувачів чи семантичні імена, або можна це доручити push()методу, який створить унікальний ключ для даного вузла [26].

3.2.2 Реалізація списків та стрічок подій

Реалізація списків та стрічок у будь-якій соціальній мережі є ключовою оскільки, практично кожна сторінка містить стрічку з записами чи списки користувачів. В розділі 2 на рис. 2.8 діаграми класу додатку, в блоці PopulateList, наведено загальну структуру реалізації списків даних. В загальному випадку ця структура складається з п’яти наступних класів. ListActivity – клас сторінки додатку де розміщується список. Entity – клас об’єкта який використовується для витягування даних з бази, містить лише змінні які необхідно витягнути з бази, важливим зауваженням є те, що імена змінних мають співпадати з іменами полів у базі даних для того щоб адаптер розумів які поля витягувати.ViewHolder – клас який відповідає за встановлення даних в компоненти та їх відображення. ListFragment – клас контроллера для кожного нового фрагменту. TabViewAdapter – клас який відповідає за фрагменти.

Список створюється класом FirebaseRecyclerView, в конструкторі він приймає чотири параметри, Entity клас – об’єкт запису, шаблон– компонент списку, ViewHolder – клас ініціалізації даних в компоненти та їх обробка подій, посилання на вузол бази даних в якому містить необхідні для нас дані. Клас обов’язково наслідує метод populateViewHolder (див. дод. Д) з параметрами класу ViewHolder, Entity та номер позиції у списку. Метод викликається для всіх дочірніх вузлів даних, відносно батьківського вузла, що вказаний у параметрі конструктора FirebaseRecyclerView класу. Кожен новий вузол, це новий шаблон (layout файл). Після проходження методом всіх дочірніх вузлів у обраному нами вузлі базі даних, необхідно здійснити ініціалізацію адаптера в компонент RecyclerView(список) за допомогою метода recyclerViewName.setAdapter() [27].

4 ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ ХМАРНИХ СЕРВІСІВ

4.1 Бізнес план реалізації та поширення проекту

Проект ПНУ соціальна мережа надає різні послуги для користувачів, зокрема: власний профіль із різною інформацією, відвідувати інші профілі, можливість ведення записів на власному профілі та залишати їх у інших користувачів, можливість оцінювати, коментувати, та поширювати записи. Дізнаватись нову інформацію та обмінюватись нею з іншими, миттєвий обмін повідомлень, можливість створювати друзів, підписників, читачів, опитування та багато іншого. Цільовою аудиторією проекту є студенти та викладачі Прикарпатського національного університету.

Проект розгортається на платформі Firebase. Вона надає безплатні ресурси в обмеженій кількості для запуску та тестування програми, їх буде достатньо для старту даного проекту. Нижче на рис 4.1 наведено скриншоти використання та обмеження безплатного плану використання ресурсів.

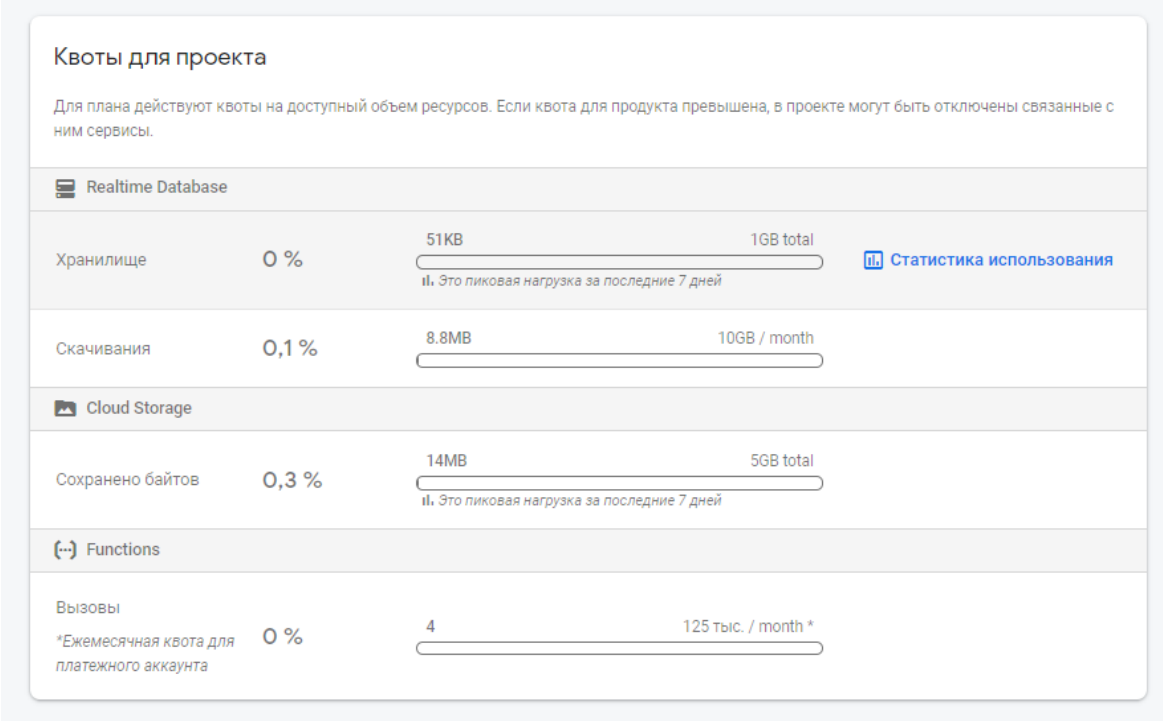


Рисунок 4.1 Квоти для бази даних, сховища файлів та нотифікацій

Phone Auth			
Проверки по номеру телефона (США, Канада и Индия)	0 %	0	10 тыс. / month
Проверки по номеру телефона (другие страны)	0 %	4	10 тыс. / month

Рисунок 4.2 Квоти для авторизацій по номеру телефона

A/B Testing	Free	
Analytics	Free	
App Distribution	Free	
App Indexing	Free	
Authentication		
Phone Auth - US, Canada, and India	10k/month	$0.01/verification
Phone Auth - All other countries	10k/month	$0.06/verification
Other Authentication services	✓	✓
Realtime Database		
Simultaneous connections	100	200k/database
GB stored	1 GB	$5/GB
GB downloaded	10 GB/month	$1/GB
Multiple databases per project	✗	✓
Remote Config	Free	
Storage		
GB stored	5 GB	$0.026/GB
GB downloaded	1 GB/day	$0.12/GB
Upload operations	20K/day	$0.05/10k
Download operations	50K/day	$0.004/10k
Multiple buckets per project	✗	✓

Рисунок 4.3 Вартість оплати ресурсів

Обрахунки на основі використання ресурсів одним користувачем, може бути визначено в приблизній кількості для використання додатку в середній активності, всіма студентами Прикарпатського національного університету(близько 15.5 тисяч). Потрібні ресурси у щомісячній оплаті в наступних об’ємах:

- база даних RealTime - 15 Гігабайт для збереження даних в базі, вартість 70 доларів, 150 Гігабайт трафіка бази даних, вартість 140 доларів;
- сховище для файлів Firebase Storage – 500 Гігабайт для збереження файлів, при використанні стиску зображень це майже 5 мільйонів зображень при середній стиснутій вазі зображення - 100 Кб. В середньому для користувача 322 зображення, вартість 12.87 доларів, 500 Гігабайт трафіка, вартість 56.40 доларів та 10.000.000 – операцій для скачувань та завантажень, вартість 12 доларів;
- сервіс авторизації по номеру телефона Firebase Phone Authentication, на даний сервіс найбільше навантаження буде здійснено в перший місяць використання додатку користувачами, при реєстрації. Сервіс надає 10 000 безплатних аутентифікацій (надісланих смс на номери телефонів), тому реєстрацію всіх студентів можна спробувати здійснити в два етапи протягом двох місяців по половині студетів ПНУ у кожний місяць або витратити 600 доларів для 20 000 аутентифікацій;

Сума використання всіх ресурсів що місяця, складає 892 долари. Суму витрат можна значно заощадити, якщо використовувати безплатні 10 тисяч аутентифікацій по номеру телефона на місяць, їх цілком може вистачати для 15,5 тисяч користувачів, якщо провести реєстрацію у декілька етапів. Сума витрат на ресурси без аутентифікацій буде становити 292 долари. Найбільші затрати потрібні для аутентифікації через телефон, ця функція є важливої оскільки вона захищає від створення багатьох акаунтів студентами з одного телефону. Також дозволяє відновити акаунт після того, як він опинився в чужих руках та надає надійний рівень доступу інших сервісів та додатку в цілому.

Платформа Firebase надає безплатний сервіс Analytics, налаштування якого сильно допомагає в налагодженні бізнес процесів проекту, оскільки надає різну статистику про використання ресурсів:

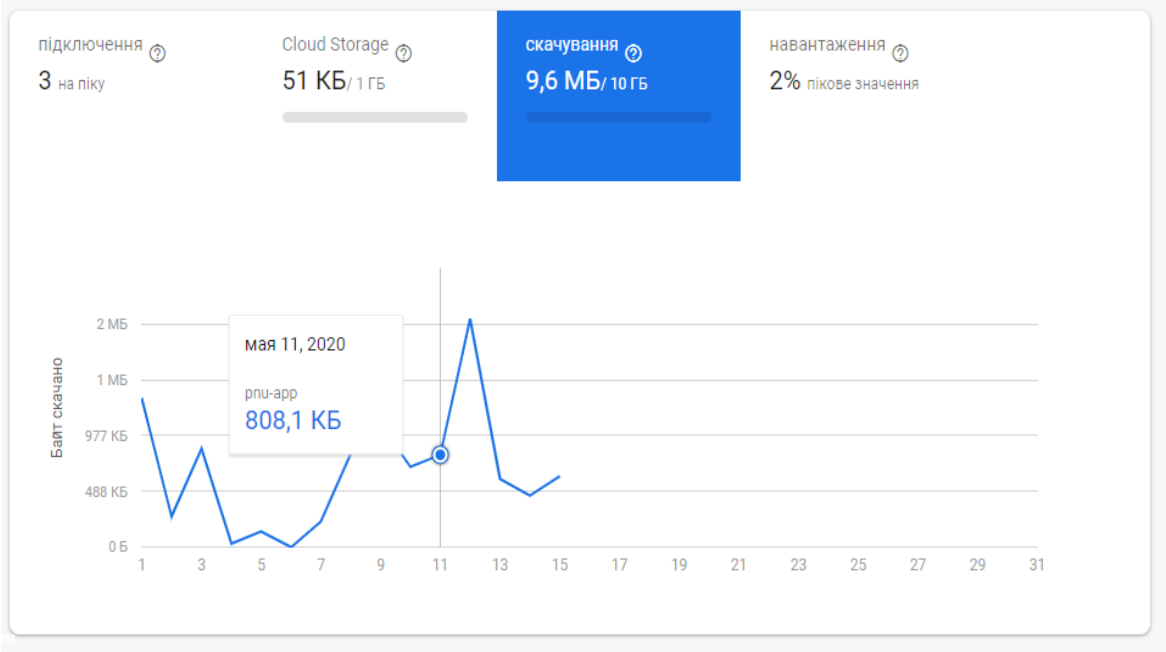


Рисунок 4.4 Використання трафіку скачування даних в RealTime Database

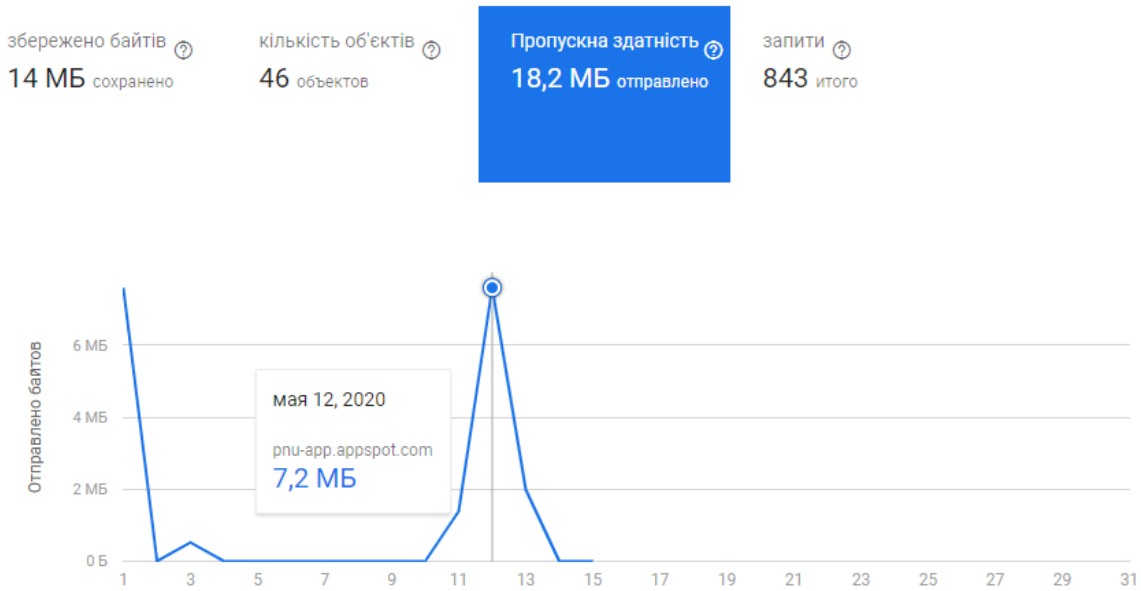


Рисунок 4.5 Використання вихідного трафіку в Firebase Storage

Платформа Firebase дозволяє налаштовувати аналітику, яка надає статистику про кількість користувачів у додатку, кількість подій з щоденним графіком, розраховує доходи від користувачів та конверсії.

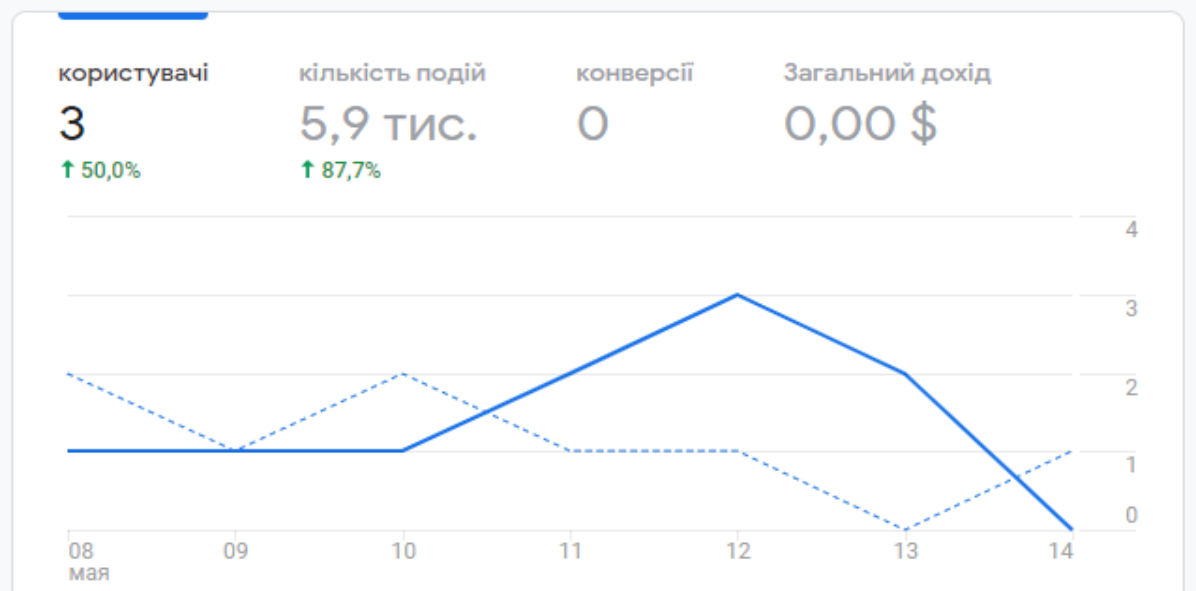


Рисунок 4.6 Аналітика додатку за останні 7 днів та його доходи

Для аналізу наскільки добре розробленим є функціонал кожного класу та які сторінки є найбільш часто відвідуваними, а які не користуються популярністю, в проект впроваджено аналітику переглядів та відвідувань сторінок.

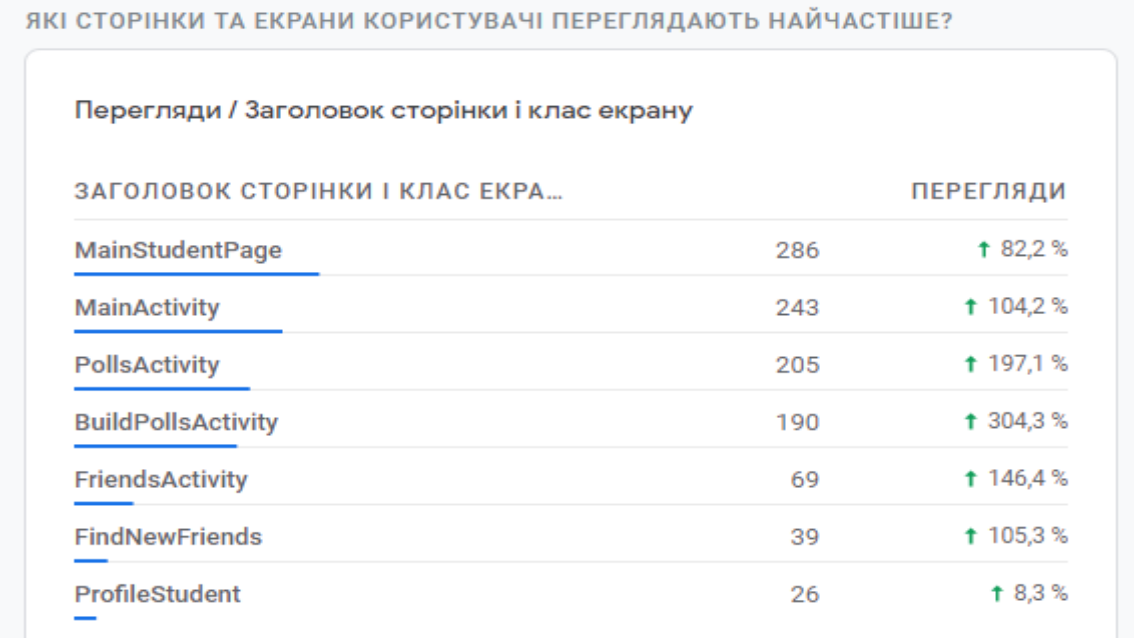


Рисунок 4.7 Аналітика кількості відвідувань найпопулярніших сторінок

Як показано на рис. 4.7, що найпопулярнішою сторінкою є головна сторінка студента, це і не дивно оскільки залогінений користувач автоматично попадає на неї після включення додатку.

	Заголовок страницы и класс экрана	Просмотров экрана на пользователя	↓Время взаимодействия	Количество событий Все события
	Итого	386,333 Средн. +0%	4 ч. 38 мин. Средн. +0%	5 897 100 % от общего числа
1	BuildPollsActivity	190	3 ч. 28 мин.	667
2	PollsActivity	205	1 ч. 50 мин.	897
3	MainStudentPage	95,333	1 ч. 13 мин.	1 737
4	Messenger	14	35 мин. 34 сек.	221
5	Message	3,333	32 мин. 08 сек.	200
6	FriendsActivity	23	14 мин. 29 сек.	524
7	ProfileStudent	8,667	10 мин. 28 сек.	259
8	FriendsProfileActivity	4	9 мин. 52 сек.	36
9	FindNewFriends	19,5	8 мин. 45 сек.	300
10	ActivityListLikesPost	3	3 мин. 10 сек.	31
11	RequestsFriendsActivity	8,5	2 мин. 28 сек.	166
12	Settings	3	2 мин. 13 сек.	54
13	MySubscribersActivity	6	2 мин. 11 сек.	139
14	SignIn	4	1 мин. 58 сек.	15
15	MainActivity	81	0 мин. 09 сек.	253

Рисунок 4.8 Аналітика екранів відвідувань, кількість подій у них та їх час взаємодії

Рис 4.8 показує, що найбільше часу користувач витрачає в сторінках створення опитувань, оскільки для опитувань необхідно створювати питання та відповіді, а не тільки перегляд сторінки. Головна сторінка посідає третє місце так як в будь-якому випадку користувач цю сторінку потрапляє в усіх випадках після завантаження додатку. Четверте та п’яте місце посідає месенджер та переписки повідомлень, наступними є список друзів, профілі інших користувачів і тд. Надана вище статистика говорить, що сторінки додатку використовуються

правильно та послідовною популярністю, але в майбутньому можуть виникнути несподівані ситуації які аналітика допоможе швидко виявити та прийняти відповідні міри для їх усунення.

Розміщуючи платні послуги чи рекламу в додатку, за допомогою аналітики можна визначати, які події в додатку є найбільш ефективними для продаж, а які потребують вдосконалення. За рахунок кількості тих чи інших подій і здійснюється продаж та обчислення доходу.

1	select_content	3 955	4	0	988,75	0,00 $
2	screen_view	1 159	3	0	386,333	0,00 $
3	user_engagement	753	3	0	251	0,00 $
4	session_start	30	3	0	10	0,00 $

Рисунок 4.9 Метрики дії користувачів

Перший стовпчик на рис. 4.9 це назва події на екрані, другий кількість подій зареєстрованих діях користувачів, третій кількість користувачі що взаємодіяли з подією, четвертий кількість нових користувачів, п’ятий середня кількість подій на одного користувача і останній стовпчик це дохід від кожної подій. За допомогою наведених даних можна робити відповідні висновки і здійснювати зміни в додатку для наступних версій, наприклад змінювати розміщення компонентів на екрані для кращої зручності чи взагалі видаляти зайві або розробляти нові.

Найбільше навантаження на додаток буде здійснюватись у період сесій, за допомогою нижче наведених графіків на рис. 4.10 тенденцій відвідувань додатку, можна буде дізнатись у які місяці студенти є найбільш пасивним у навчанні, а у які найбільш активними та продуктивними.

Рисунок 4.11 Тенденції відвідувань користувачів з плином часу

Практично всі користувачі будуть розміщені в межах України. Ринок збуту по географічному розташування користувачів наведено нижче на рис. 4.11.

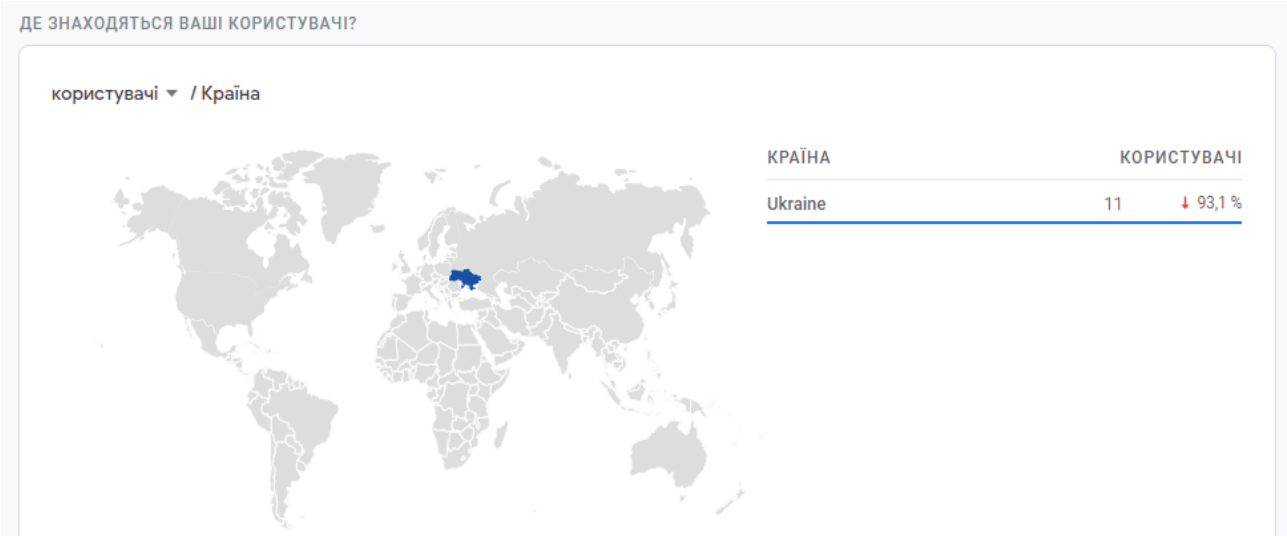


Рисунок 4.11 Розміщення користувачів по регіонам

	Итого	399 100 % от общего числа	84,36 % Средн. +0%	36,273 Средн. +0%	15 ч. 45 мин. Средн. +0%	60 879 100 % от общего числа
1	Ukraine	399	84,36 %	36,273	15 ч. 45 мин.	60 879

Рисунок 4.12 Кількісні характеристики використання додатку в межах країни

Стовпчики відповідають за: перший - назва країни розміщення користувачів, другий - кількість сеансів, що тривали більше 10 секунд, що

включають конверсію або складаються з перегляду не менше двох сторінок / екранів, третій – відсоток сеансів із взаємодією(відношення числа сеансів із взаємодією до загального кількості сеансів), четвертий – кількість сеансів із взаємодією на користувача, п’ятий – середній період часу протягом якого додаток знаходився в активному режимі і останній стовпчик – це кількість подій, зареєстрованих при діях користувачів.

Запущений додаток необхідно постійно монітори на стабільність та на виникнення багів у системі, які обов’язково з’являться з плином часу користування багатьма користувачами, оскільки неможливо все врахувати в 1 версії додатку. Для того щоб користувачі не покидали додаток необхідно вчасно реагувати, розробляти та випускати нову версію додатку. Нижче на рис. 4.13 наведено графік стабільності версії 1.0 за останній місяць.

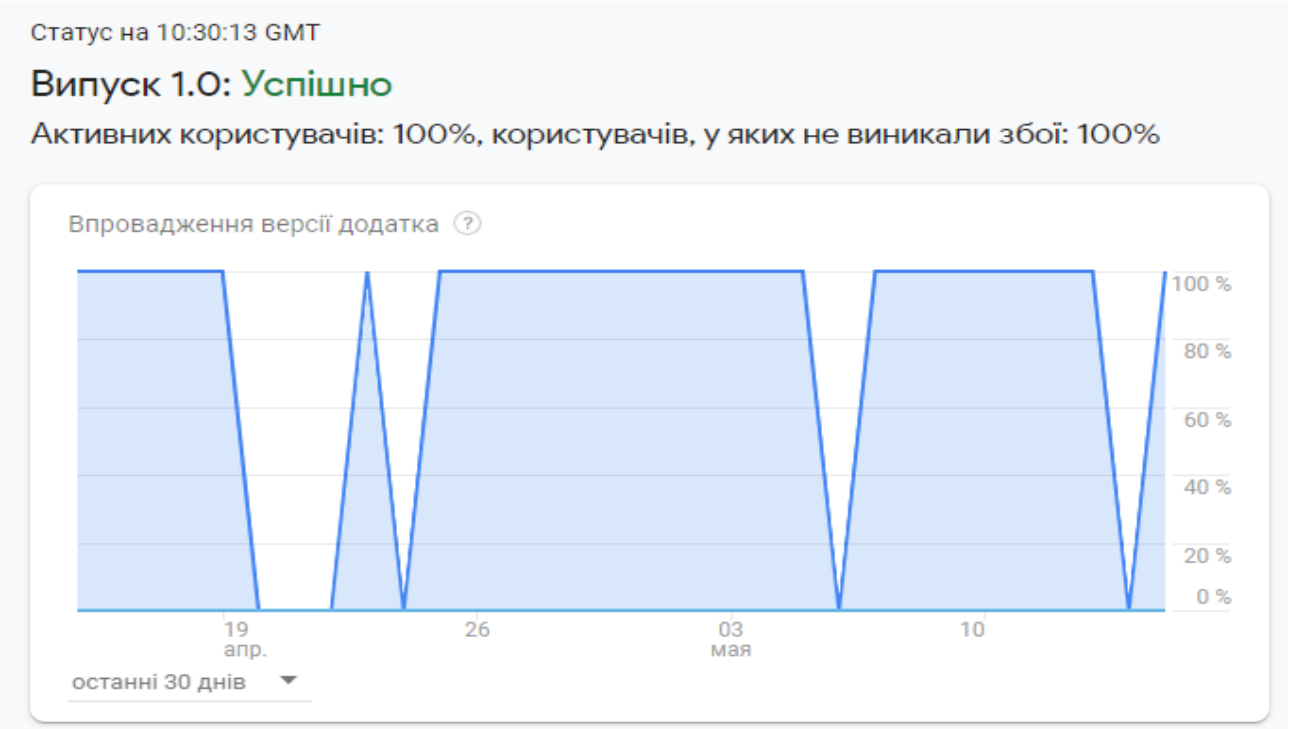


Рисунок 4.13 Стабільність додатку версії 1.0

Ефективність утримування користувачів у додатку на пряму залежить від якості розробленого функціоналу версії та деталей дизайну. Нижче на рис. 4.14 наведено таблицю утримування трьох тестових користувачів протягом двох місяців.

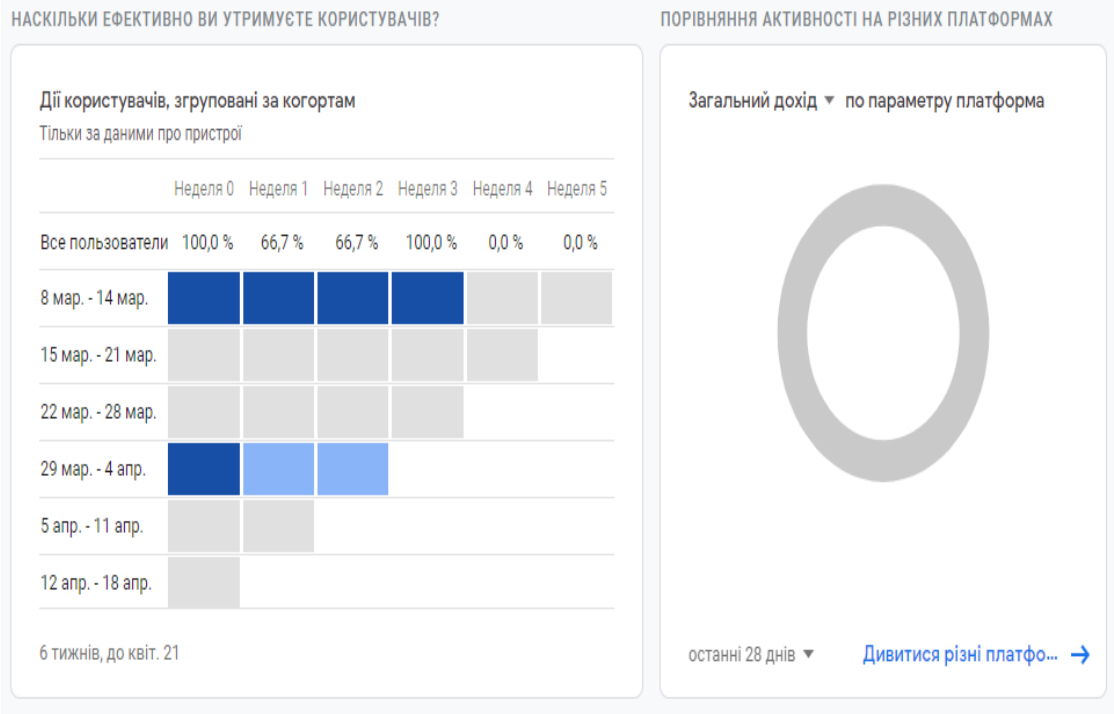


Рисунок 4.14 Ефективність утримування користувачів

Даний проект розробляється виключно для аудиторії операційної системи Android.

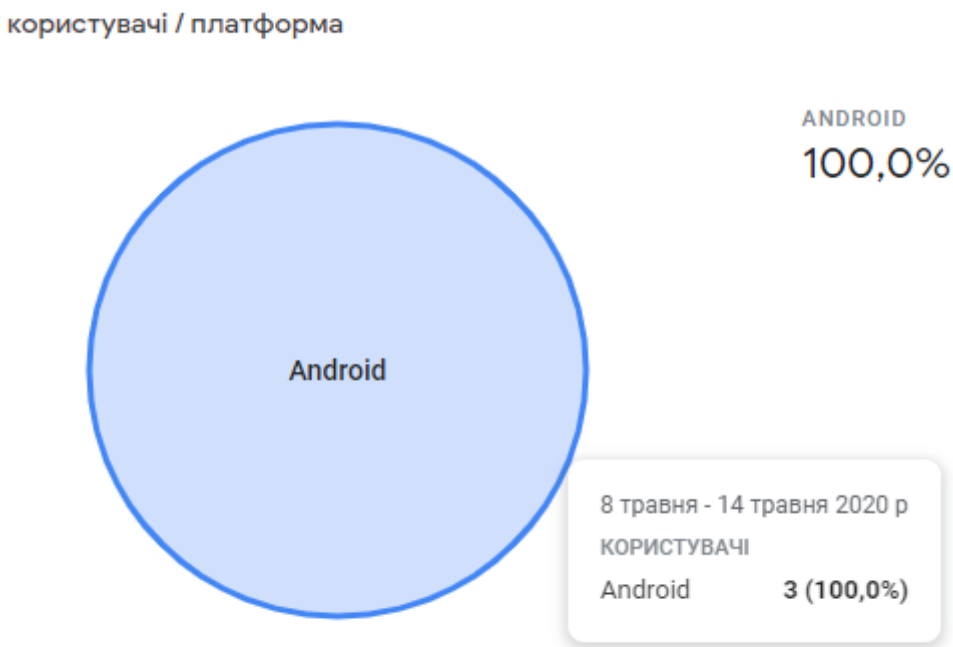


Рисунок 4.15 Групування користувачі за операційною системою

Фінансування на проект буде здійснюватися із університету або Міністерства освіти і науки, якщо додаток набере велику популярність. Даний

проект буде безплатним і оскільки він діє в межах державного університету накладати рекламу на нього є недоречним, хоча і можливи, тому єдиним отримання заробітку із додатку може стати створення унікальних платних можливостей та інструментів впроваджених в середині додатку(наприклад пробний тест/іспит).

Прогнозований об’єм реалізацій проекта складається з трьох варіантів розвитку:

- **песимістичний:** в цьому варіанті розвиток подій є найгіршим, він заключається в тому, що проект не буде випущений в масове використання, а так і залишиться в тестовій версії для декількох користувачів;
- **оптимістичний:** проект запущено в повному обсязі для цілого університету, 15,5 тис. користувачів. Проект має необхідне фінансування та підтримується декількома розробниками, які розробляють нові версії додатку. Додатково в проект інтегровано систему автоматизованої побудови індивідуалізованого навчання для кожного студента [28];
- **реалістичний:** проект запущено в межах факультету математики та інформатики ПНУ, для проекту не потрібно фінансування, оскільки безплатних ресурсів наданих платформою Firebase достатньо для функціонування додатку в межах факультету. Підтримується 1-2 розробниками на безплатній основі;

Дана програма містить наступні нормативно-правові нюанси: працює на основі даних (бази даних), доступ до якої надається Прикарпатським національним університетом імені Василя Стефаника. Для їх отримання потрібний документ з проханням від керівника та розробника проекту до університету про надання бази даних з даними студентів що містяться на персональні картці студента(студентського). В документі повинні міститись угоди що до нерозповсюдження персональних даних та конфіденційної

інформації відповідно до закону України “Про захист персональних даних”. Цей Закон поширюється на діяльність з обробки персональних даних, яка здійснюється повністю або частково із застосуванням автоматизованих засобів, а також на обробку персональних даних, що містяться у студентському чи призначені до внесення у додаток, із застосуванням неавтоматизованих засобі.

Усі компоненти які використовується в програмі, а до них відносяться: стилі текстів, зображення, аудіо, готові шаблони дизайну, ресурси та інші компоненти що покращують візуальне сприйняття програми та його використання захищені “Apache Software License” до версії 2.0. Умови ліцензування подібно до всякої іншої ліцензії на вільне програмне забезпечення, ліцензія Apache дає користувачеві право використовувати програмне забезпечення для будь-яких цілей, вільно поширювати, змінювати, і поширювати змінені копії.

Організаційно-правова форма підприємництва даного проекту: державна фірма. На даному етапі додаток розробляється єдинем розробником. У майбутньому може бути залучено від 1 до 5 співробітників. Поточна версія додатку 1.0, дана версія ще триває в процесі розробки, дата запуску версії 1.0 на платформу Firebase: 18 листопада. 2019 рік.

Початок розробки додатку було здійснено у вересні 2019 року. Календар розробки наведено , у табл 4.1. Детальну історію розробки можна переглянути за посиланням на github у вкладці історія комітів https://github.com/Vadym-Popovych24/PNU_APP/commits/master .

Таблиця 4.1 – Календар реалізації проекту

Місяць	Етапи	Витрати
№ 1 вересень 2019 рік.	Поставновка основних задач, моделювання архітектури, початок розробки - розробка локального входу та реєстрації на основі SQLite ROOM.	0 грн.

Кінець таблиці 4.1

№ 2 жовтень 2019 рік.	Інтеграція з платформою Firebase	0 грн.
№ 3 листопад 2019 рік.	Синхронізація між локальною SQLite базою та віддаленими сервісами Firebase	0 грн.
№ 4 грудень 2019 рік.	Ауторизація за допомогою номеру телефона, створення користувачів на платфомі Firebase	0 грн.
№ 5 січень 2020 рік.	Вихід на 25% планової реалізації, розробка мінімального головного профілю користувача.	0 грн.
№ 6 лютий 2020 рік.	Робота із зображеннями, їх завантаженя в профілі, стиск, збереження в Storage, зв'язування зображень із базою RealTime.	0 грн.
№ 7 березень 2020 рік.	Вихід на 50% планової реалізації, розробка профілів користувачі, меседжера.	0 грн.
№ 8 квітень 2020 рік.	Вихід на 75% планової реалізації, розробка системи друзів, підписників, читачів, заявок, записів.	0 грн.
№ 9 травень 2020 рік.	Вихід на 100% планової реалізації, розробка опитувань.	0 грн.
№ 10.. червень 2020 рік.	Підтримка , вдосконалення та розширення додатку.	0 – 23 тис. грн.

ВИСНОВКИ

В дипломній роботі було розроблено додаток соціальна мережа Прикарпатський національний університет. Проведено аналіз предметної області розробки програмного забезпечення під операційну систему Android засобами мови програмування Java, мови розмітки XML, локальної бази даних SQLite та хмарної платформи для розгортання додатку Firebase та інші.

Проведено аналіз використаних технологій, та описано основний принцип їх роботи. По етапах описано створення дизайну додатку та функціональну частину, результати роботи продемонстровано у відповідних рисунках дипломної роботи.

Розроблений додаток має аналоги, якщо говорити про його загальні можливості і всім відомі соціальні мережі та месенджери. Проте подібного додатку немає для жодного університету України, який би працював на основі даних ІД картки студента у межах певного університету чи груп університетів і надавав доступ та унікальний функціонал тільки для студентів певного закладу чи груп закладів.

Проаналізовано різні варіанти розвитку та реалізації проекту в різних кількостях використання певних ресурсів користувачами. Надано детальну аналітику додатку та його користувачів.

Додаток розроблено та запущено у тестовому режимі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

REFERENCES

1. Частки операційних систем Android і iOS. URL: https://www.ixbt.com/news/2018/05/03/doli-operacionnyh-sistem-android-i-ios-za-poslednij-god-v-mire-osobo-ne-izmenilis.html (дата звернення: 12.09.2019)
2. Матеріал із вікіпедії Xamarin. URL: https://ru.wikipedia.org/wiki/Xamarin (дата звернення: 18.09.2019)
3. Світовий ринок смартфонів. URL: http://www.tadviser.ru/index.php/Статья:Смартфоны_(мировой_рынок) (дата звернення: 22.09.2019)
4. Google Play Market. URL: https://play.google.com/store (дата звернення: 27.09.2019)
5. Додаток “Athabasca University Students” в Google Play Market. URL: https://play.google.com/store/apps/details?id=com.oohlala.athabasca (дата звернення: 28.09.2019)
6. Матеріал із вікіпедії Java. URL: https://uk.wikipedia.org/wiki/Java (дата звернення: 30.09.2019)
7. Матеріал із вікіпедії SQLite. URL: https://uk.wikipedia.org/wiki/SQLite (дата звернення: 02.09.2019)
8. Основи роботи з ORM ROOM для бази даних SQLite. URL: https://startandroid.ru/ru/courses/architecture-components/27-course/architecture-components/529-urok-5-room-osnovy.html (дата звернення: 03.10.2019)
9. Bray, Tim; Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, Franзois Yergeau (September 2006). Extensible Markup Language (XML) 1.0 (Fourth Edition) -- Origin and Goals. World Wide Web Consortium. Архів оригіналу за 2013-06-23.

10. Матеріал із вікіпедії Android Studio. URL: https://uk.wikipedia.org/wiki/Android_Studio (дата звернення: 12.10.2019)
11. Матеріал із вікіпедії Gradle. URL: https://uk.wikipedia.org/wiki/Gradle (дата звернення: 25.10.2019)
12. Build dependencies. URL: https://developer.android.com/studio/build/dependencies (дата звернення: 26.10.2019).
13. M. Kozlenko, V. Tkachuk, and M. Dutchak, "Software implementation of microcomputer based intrusion detection and prevention system with binary neural network," in Proc. 2nd International Scientific-Practical Conference "Problems of Cyber Security of Information and Telecommunication Systems" (PCSITS), O. Oksiiuk et al, Eds. Taras Shevchenko National University of Kyiv, Kyiv, Ukraine, Apr. 11-12, 2019, pp. 371-373.
14. ДСТУ 2874-94. Системи оброблення інформації. Бази даних. Терміни та визначення. — К. : Держстандарт України, 1995. — 29 с. (Інформація та документація).
15. Документація Firebase для розробників. URL: https://firebase.google.com/docs (дата звернення: 27.10.2019)
16. Бібліотека модифікованого компоненту EditText. URL: https://github.com/rengwuxian/MaterialEditText (дата звернення: 30.10.2019)
17. Бібліотека для реалізації та надання емодзі. URL: https://github.com/hani-momanii/SuperNova-Emoji (дата звернення: 02.11.2019)
18. Jeff Friesen. Learn Java for Android Development [Текст] / Jeff Friesen; Third Edition, 2014. - 1190 с.

19. I. Lazarovich and Y. Nikolaychuk, "Method of randomization and its application for adaptive data compression," Second IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2003. Proceedings, Lviv, 2003, pp. 362-364, doi: 10.1109/IDAACS.2003.1249587.
20. Бібліотека Compressor. URL: https://github.com/zetbaitsu/Compressor (дата звернення: 18.12.2019)
21. Бібліотека Picasso. URL: https://github.com/square/picasso (дата звернення: 11.12.2019)
22. Эккель Б. Философия Java. [Текст] / Б. Эккель; СПб. : Питер, 2015.- 1168 с.
23. Бази даних у питаннях і відповідях : нав. посіб. / В. В. Чубук, Р. М. Чен, Л. А. Павленко, В. І. Клименко. — Харків : Вид. ХНУ, 2004. — 287 с.
24. Коннолі Т., Бегг К., Страчан А. Бази даних: проектування і супровід. Теорія і практика : навч.-метод. посіб. 2-е вид .: Пер. з англ. - М .: Видавничий дім «Вільямс», 2001. - 1120 с.
25. Лафоре Р. Структури даних та алгоритми в Java. [Текст] / Р. Лафоре; СПб. : Питер, 2013.- 704 с.
26. Retrieving Data and listeners datasnapshot. URL: https://firebase.google.com/docs/database/admin/retrieve-data (дата звернення: 06.03.2020)
27. Data Persistence Firebase: RecyclerAdapter. URL:https://www.learnhowtoprogram.com/android/data-persistence/firebase-recycleradapter (дата звернення: 28.03.2020)
28. М. Дутчак, "Моделювання процесу автоматизованої побудови індивідуалізованого навчання в інтелектуальних освітніх онлайн системах," Прикладні науково-технічні дослідження: матеріали IV міжнар. наук.-практ. конф., Івано-Франківськ, 2020, т.1., с. 47-51.

ДОДАТОК А

Посилання на розміщення всіх вихідних кодів програми.

https://github.com/Vadym-Popovych24/PNU_APP

ДОДАТОК Б

Базові налаштування Gradle.

```
buildscript {
    repositories {
        google()
        jcenter()

        maven {
            url 'https://maven.google.com'
        }
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.5.0'
        classpath 'com.google.gms:google-services:4.3.2'

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()

        maven {
            url 'https://jitpack.io'
        }
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

ДОДАТОК В

Базові налаштування збірки проекту.

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 29
    buildToolsVersion "29.0.2"
    defaultConfig {
        applicationId "com.social_network.pnu_app"
        minSdkVersion 16
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
        multiDexEnabled true
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-
optimize.txt'), 'proguard-rules.pro'
        }
    }
}
repositories {
    maven { url "https://oss.sonatype.org/content/repositories/snapshots" }
}

apply plugin: 'com.google.gms.google-services'
```

ДОДАТОК Г

Всі бібліотеки та модулі, що підключені до проекту.

```
dependencies {
    implementation "android.arch.persistence.room:runtime:1.0.0"
    implementation 'com.google.android.material:material:1.0.0'
    implementation 'com.google.firebase:firebase-storage:19.1.1'
    implementation 'com.google.firebase:firebase-functions:16.1.3'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    annotationProcessor "android.arch.persistence.room:compiler:1.0.0"
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.0.2'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'com.android.support:cardview-v7:29.+'
    implementation 'com.rengwuxian.materialedittext:library:2.1.4'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test:runner:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.1'
    implementation 'com.google.firebase:firebase-analytics:17.2.2'
    implementation 'com.google.firebase:firebase-core:17.2.2'
    implementation 'com.google.firebase:firebase-auth:19.2.0'
    implementation 'com.google.firebase:firebase-database:19.2.0'
    implementation 'com.firebaseui:firebase-ui:0.6.2'
    implementation 'com.squareup.picasso:picasso:2.6.0-SNAPSHOT'
    implementation 'com.squareup.okhttp:okhttp:2.5.0'
    implementation 'de.hdodenhof:circleimageview:3.1.0'
    implementation 'com.github.lguipeng:BubbleView:1.0.1'
    implementation 'com.github.hani-momanii:SuperNova-Emoji:1.1'
    implementation 'com.android.support:appcompat-v7:28.0.0-alpha1'
    implementation 'com.android.support:design:28.0.0' // Design lib
    implementation 'com.google.android.gms:play-services-auth:17.0.0'
    implementation 'id.zelory:compressor:2.1.0'
    implementation 'com.android.support:multidex:1.0.3'
}
```

ДОДАТОК Д

Реалізація фрагменту списку друзів.

```
FirebaseRecyclerAdapter<Friends, FriendsViewHolder> firebaseRecyclerAdapter
        = new FirebaseRecyclerAdapter<Friends, FriendsViewHolder>
        (
                Friends.class,
                R.layout.friends layout,
                FriendsViewHolder.class,
                myFriendsReference

        ) {
    @Override
    protected void populateViewHolder(final FriendsViewHolder friendsViewHolder,
final Friends friends, final int i) {
        setTextViewForEmptyList();
        final String currentFriend = getRef(i).getKey();
        students.child(currentFriend).addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                String name = dataSnapshot.child("name").getValue().toString();
                String lastName =
dataSnapshot.child("lastName").getValue().toString();
                String grop = dataSnapshot.child("group").getValue().toString();
                final String seriesIDcard =
dataSnapshot.child("seriesIDcard").getValue().toString();
                boolean online;
                try {
                     online = (boolean) dataSnapshot.child("online").getValue();
                }catch (Exception e){
                    online = false;
                }
                if (online){
                    friendsViewHolder.setOnlineImage();
                }
                String linkFirebaseStorageMainPhoto;
                try {
                    linkFirebaseStorageMainPhoto =
dataSnapshot.child("linkFirebaseStorageMainPhoto").getValue().toString();
                }
                catch (NullPointerException nullPointerException){
                    linkFirebaseStorageMainPhoto ="";
                }
                friendsViewHolder.setStudentName(name, lastName);
                friendsViewHolder.setStudentGroup(grop);
                if (linkFirebaseStorageMainPhoto != "" && myContex != null) {
                    friendsViewHolder.setStudentImage(myContex,
linkFirebaseStorageMainPhoto);
                }



friendsViewHolder.mView.findViewById(R.id.btnSendMessageLayout).setOnClickListen
er(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        Intent intentSendMessageFriends= new
```

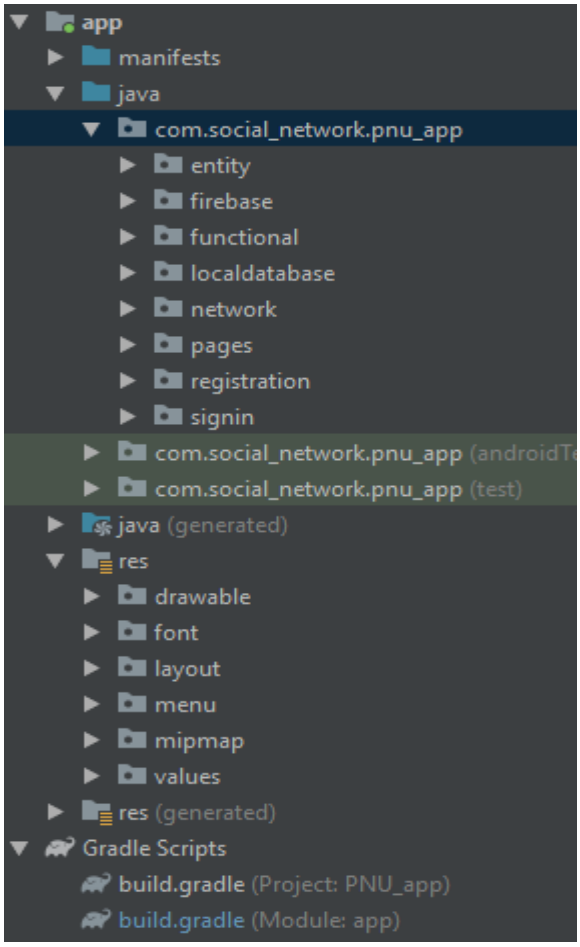
```
Intent("com.social_network.pnu_app.pages.Message");
                        intentSendMessageFriends.putExtra("VisitedStudentKey",
currentFriend);
                        startActivity(intentSendMessageFriends);
                    }
                });
                friendsViewHolder.mView.setOnClickListener(new
View.OnClickListener() {
                    @Override
                    public void onClick(View v) {

                        if (!seriesIDcard.equals(SerieIDCard)) {
                            String VisitedStudentKey = getRef(i).getKey();
                            Intent profileIntent = new
Intent("com.social_network.pnu_app.pages.ProfileStudent");
                            profileIntent.putExtra("VisitedStudentKey",
VisitedStudentKey);
                            startActivity(profileIntent);
                        } else {
                            Intent myProfileIntent = new
Intent("com.social_network.pnu_app.pages.MainStudentPage");
                            startActivity(myProfileIntent);
                        }
                    }
                });
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {
                if (!network.isOnline()) {
                    //        progressBar.setVisibility(View.GONE);
                    Toast.makeText(getContext(), " Please Connect to Internet",
                            Toast.LENGTH_LONG).show();
                }
            }
        });


    }
};
myFriendsList.setAdapter(firebaseRecyclerAdapter);

class FriendsViewHolder extends RecyclerView.ViewHolder {
    View mView;

    public FriendsViewHolder(View itemView){
        super(itemView);

        mView= itemView;
    }

    public void setOnlineImage(){

        ImageView imageOnline = mView.findViewById(R.id.img_online_friends);
        imageOnline.setVisibility(View.VISIBLE);
    }
    public void setStudentName(String studentName, String studentLastName){
        TextView nameAndLastName = mView.findViewById(R.id.friends_username);
```

```
        nameAndLastName.setText(studentName + " " + studentLastName);
    }
    public void setStudentGroup(String studentGroup){
        TextView group = mView.findViewById(R.id.friends_status);
        group.setText(studentGroup);
    }

    public void setStudentImage(final Context context, final String
studentImage) {
        final CircleImageView image =
mView.findViewById(R.id.friends_profile_image);
        Picasso.with(context)
                .load(studentImage)
                .networkPolicy(NetworkPolicy.OFFLINE)

.placeholder(R.drawable.com_facebook_profile_picture_blank_square)
                .error(R.drawable.com_facebook_close)
                .centerCrop()
                .fit()
                //.resize(1920,2560)
                .into(image, new Callback() {
                    @Override
                    public void onSuccess() {

                    }

                    @Override
                    public void onError() {
                        if (studentImage != null) {
                            if (!studentImage.isEmpty()) {
                                Picasso.with(context)
                                        .load(studentImage)
                                        .placeholder(R.drawable.logo_pnu)
                                        .error(R.drawable.com_facebook_close)
                                        .centerCrop()
                                        .fit()
                                        //.resize(1920,2560)
                                        .into(image);
                            }
                        } else {
                            Picasso.with(context)

.load(R.drawable.com_facebook_profile_picture_blank_square)
                                    .placeholder(R.drawable.logo_pnu)
                                    .error(R.drawable.com_facebook_close)
                                    .centerCrop()
                                    .fit()
                                    //.resize(1920,2560)
                                    .into(image);
                        }

                    }
                });

    }
}
```

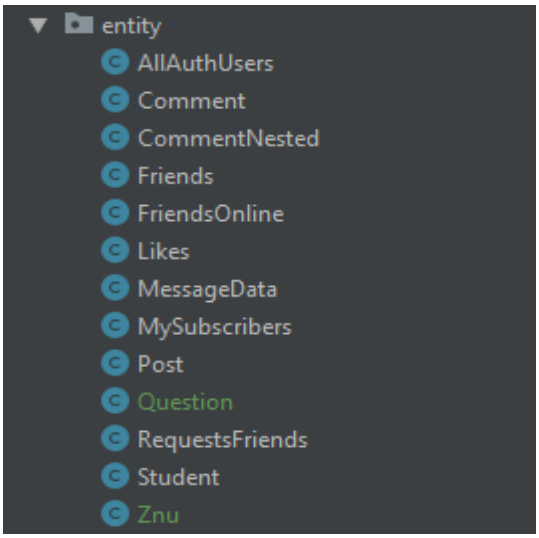
ДОДАТОК Е

Структура проекту.

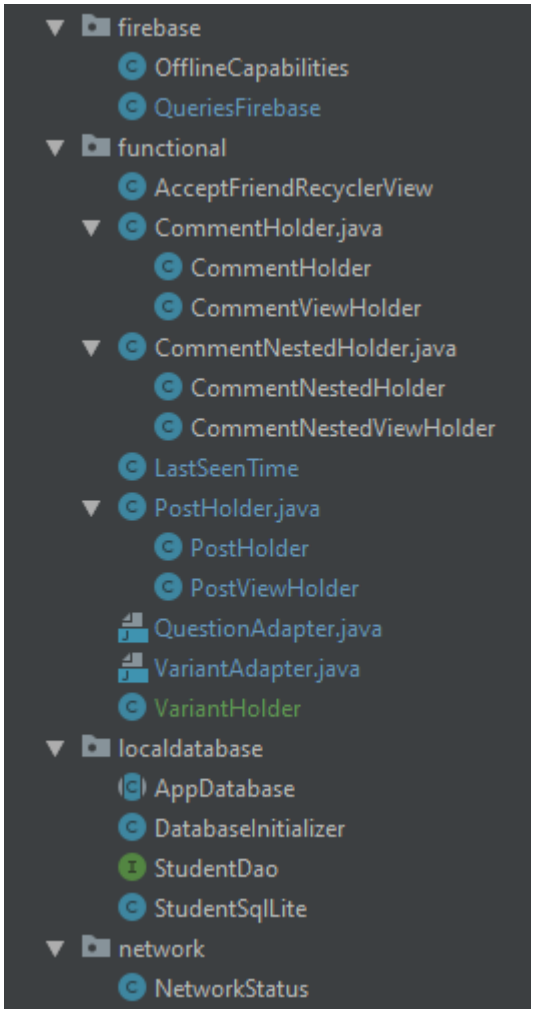
Загальна структура програми.



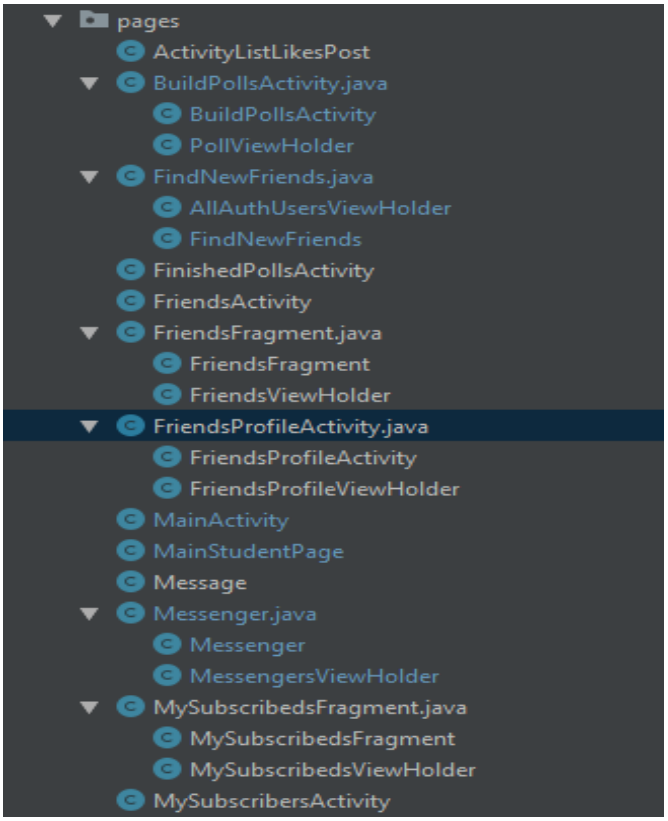
Структура java пакету Entity.

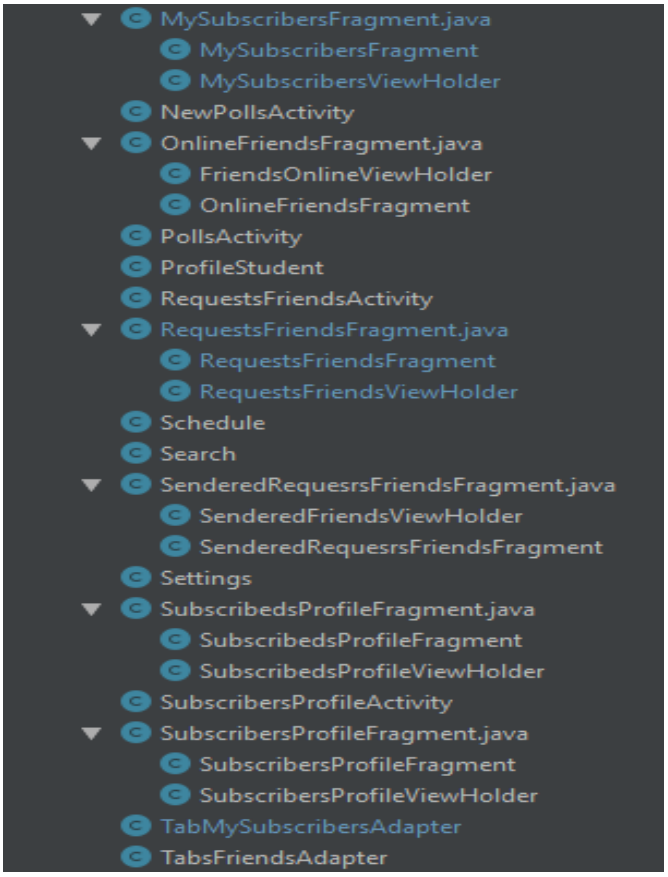


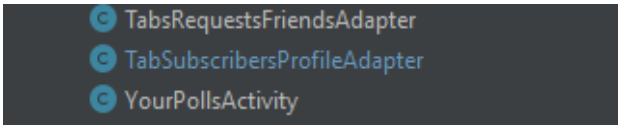
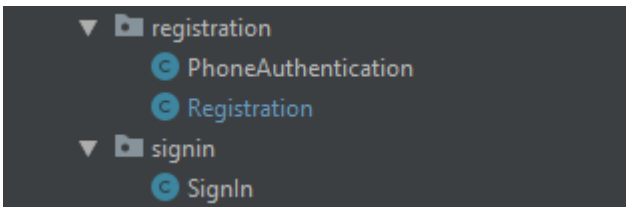
Структура java пакетів: firebase, functional, localdatabase, network.



Структура java пакету: pages.







Структура java пакетів: registration, signin.