МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА МОБІЛЬНОГО КЛІЄНТА СХОВИЩА ФАЙЛІВ РЕАЛЬНОГО ЧАСУ»**

Виконала студентка 2 курсу, групи 8.1218-з

спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

освітньої програми інженерія програмного забезпечення
(назва освітньої програми)

М. І.-В. Савка
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, к.ф.-м.н. Кудін О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної математики, доцент, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення

(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
програмної інженерії,
к.ф.-м.н., доцент

Лісняк А.О.

(підпис)

« » 2019 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Савці Марії Іванівні-Володимирівні

(прізвище, ім'я та по-батькові)

1. Тема роботи (проекту) Розробка мобільного клієнта сховища файлів реального часу

керівник роботи (проекту) Кудін Олексій Володимирович, к.ф.-м.н.

(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 29 » травня 2019 року № 812-с

2. Строк подання студентом роботи

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі, основні теоретичні відомості
2. Проектування бази даних та мобільного додатку
3. Розробка мобільного додатку та його тестування

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 29.05.2019

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	22.09.2019	Виконано
2.	Збір вихідних даних.	15.10.2019	Виконано
3.	Обробка методичних та теоретичних	30.10.2019	
	джерел.		Виконано
4.	Розробка першого розділу.	14.11.2019	Виконано
5.	Розробка другого розділу.	12.12.2019	Виконано
6.	Оформлення та нормоконтроль		
	кваліфікаційної роботи.	27.12.2019	Виконано
7.	Захист кваліфікаційної роботи.	10.01.2020	Виконано

Студент _____ М.І.-В. Савка
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.В. Кудін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.В. Кудін
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка мобільного клієнта сховища файлів реального часу»: 42 с., 16 рис., 1 табл., 7 джерел.

БАЗИ ДАНИХ РЕАЛЬНОГО ЧАСУ, АВТЕНТИФІКАЦІЯ, ХМАРНІ СХОВИЩА БАЗ ДАНИХ, КОЛЕКЦІЯ ДОКУМЕНТІВ, ІЄРАРХІЯ ДОКУМЕНТІВ, ПРОЕКТУВАННЯ БАЗ ДАНИХ

Об’єкт дослідження – процес управління великим обсягом документованої інформації в реальному часі.

Мета роботи: розробка мобільного застосунку автоматизації роботи сховища документів реального часу.

Метод дослідження – аналітичний, методи програмної інженерії.

У роботі було розглянуто та проаналізовано переваги платформи Google Firebase для розробки Android-застосунку в програмному середовищі розробки Android Studio. Для створення додатку використано вбудовані бібліотеки Android та Firebase. В додатку імплементовано систему акаунтів з використанням бібліотеки Firebase Authentication для реєстрації в додатку та отримання доступу до баз даних. Бібліотека Firebase Realtime Database дозволяє управляти документами бази даних реального часу, як сховище документів використано можливості Firebase Cloud Firestore. Створений додаток розрахований для використання в умовах компанії з метою управління документованою інформацією.

SUMMARY

Master's Qualification Thesis «Development of the Realtime File Storage Mobile Client»: 42 pages, 16 figures, 1table, 7 reference.

REALTIME DATABASE, AUTHENTICATION, CLOUD DATABASE STORAGE, COLLECTION OF THE DOCUMENTS, DOCUMENTS HIERARCHY, DATABASE ENGINEERING

The object of the study is the process of developing an Android application.

The aim of the study is to develop an application for the Android OS, allowing users to work with real-time database documents.

The methods of research are analytical, software engineering methods.

The work examined and analyzed the advantages of the Google Firebase platform for developing Android applications in the Android Studio development environment. To create the application, the built-in Android and Firebase libraries are used. The application implemented an account system using the Firebase Authentication library to register in the application and gain access to databases. The Firebase Realtime Database library allows to manage real-time database documents as a document repository using the capabilities of the Firebase Cloud Firestore. The created application is designed for use in a company environment with the goal of managing documented information.

ЗМІСТ

ВСТУП

Основою для досягнення компанією стратегічних конкурентних переваг в умовах економічної глобалізації та швидкозростаючої конкуренції є повна орієнтація його діяльності на споживача. В комплексі стратегій розвитку такої компанії є стратегія менеджменту якості.

Згідно з міжнародним стандартом ISO серії 9000:2015, система менеджменту якості – це координована діяльність з управління та керування діяльністю організації стосовно якості.

Процес управління якістю компанії, виходячи із основних положень сучасної концепції менеджменту якості, розглядається як самостійна, складна функція управління бізнес-процесами.

Результати виконання бізнес-процесів відображаються в документованій інформації – протоколах, записах, журналах, керівництвах, процедурах та т.ін. Компанія повинна контролювати та підтримувати документовану інформацію в актуальному стані. Таким чином, одним із ключових процесів системи менеджменту якості є процес управління документованою інформацією.

В умовах складної розгалуженої системи компанії, яка складається із підрозділів різного спрямування, що можуть бути розміщені не в одному офісі, працівники виконують свою роботу, перебуваючи у постійному русі між офісами, філіями, місцями базування та т. ін. Таким чином у компанії виникає необхідність віддалено контролювати користування актуальними версіями документованої інформації всіма працівниками.

Метою роботи є розробка мобільного застосунку автоматизації роботи сховища документів реального часу.

Для досягнення поставленої мети сформульовано наступні задачі:

- провести аналітичний огляд мобільних технологій реального часу;
- розробити вимоги до програмного забезпечення;

- спроектувати та реалізувати мобільний застосунок;
- виконати тестування.

Об’єктом дослідження є процес управління великим обсягом документованої інформації в реальному часі.

Предмет – розробка мобільного програмного додатку для забезпечення управління обігом документованої інформації, контролю версій документів та розподілу доступу до документів компанії.

Методи дослідження: аналітичний, методи програмної інженерії.

Структурно робота складається з трьох розділів. У першому розділі проаналізовано вимоги до додатку та розглянуто можливості баз даних реального часу. Другий розділ містить проект програмного застосунку. Особливості реалізації та тестування програмної системи викладено у третьому розділі.

1 ОГЛЯД ТЕХНОЛОГІЇ

Для створення додатку під операційну систему Android з функціями роботи з базами даних було використано платформу Google Firebase.

1.1 Аналіз вимог до додатку

Враховуючи дані, які були отримані при аналізі проблем управління документованою інформацією, можна сформулювати декілька вимог стосовно функціоналу розроблюваного додатку.

Оскільки в умовах реальної компанії додатком будуть користуватись працівники різної вікової категорії та з різним рівнем навичок роботи з мобільними приладами, виникає необхідність використання інтуїтивно зрозумілого то простого дизайну в інтерфейсі.

З тієї ж причини додаток повинен бути простим у використанні та не містити зайвої інформації.

За умов відсутності сталого інтернет-з’єднання вся інформація про змінах в документах повинна зберігатись та надаватись користувачеві для розгляду після того, як мережа знову з’явиться.

Додаток повинен забезпечувати різницю в рівні доступу до документів – користувач чи розробник. Розробник документу має право вносити правки до документу на етапі його розробки та сповіщати про зміни інших користувачів. Користувач має право лише переглядати перелік документів та документи, для яких надано право доступу. Для користувачів повинен бути доступний перелік тільки актуальних документів.

Оскільки одним мобільним пристроєм компанії може користуватись декілька користувачів з різними правами доступу та належності документів, необхідно забезпечити автентифікацію користувача при вході до додатку.

Мобільний додаток повинен використовуватись щоденно та іноді в умовах незадовільної якості інтернет-мережі, додаток повинен забезпечувати задовільну швидкість роботи та відповідний рівень відмовостійкості.

На основі теми кваліфікаційної роботи та пропозицій що надійшли від майбутніх користувачів додатку було сформульовано наступні функціональні та нефункціональні вимоги до системи:

- нефункціональні вимоги:
    1) система аккаунтів для безпечного користування застосунком;
    2) отримання інформації про зміну в документах;
    3) зручний та зрозумілий інтерфейс.
- функціональні вимоги:
    1) програмний продукт для ОС Android;
    2) управлянні документами реального часу в базі даних Firestore Realtime Database;
    3) створення аккаунтів в базі Firebase Authentication;
    4) зберігання користувацьких даних у базі даних Cloud Firestore.

1.2 Огляд можливостей Google Firebase

Google Firebase – це платформа розробки програмного забезпечення для додатків Google, яка дозволяє розробляти додатки для iOS, Android та Web.

Firebase надає інструменти для відстеження аналітики, звітування та виправлення збоїв додатків, створення маркетингових та продуктових експериментів, що дозволяє розробляти високоякісні додатки, збільшувати базу користувачів та отримувати більше прибутку [1].

Послуги, що надає платформа можна розділити на три групи [2]:

1) Розробка та тестування додатку:
    - Realtime Database;

- Authentication;
- Firestore Storage;
- та т.ін.

2) Вдосконалення якості додатку:
   - Crashlytics;
   - Test Lab;
   - та т.ін.
3) Збільшення кількості користувачів:
   - Firebase Analytics;
   - Remote Config;
   - Dynamic Links;
   - Invites;
   - AdMob;
   - та т.ін.

1.3 Основні задачі, які виконує платформа Google Firebase

Realtime Database платформи Firebase – це сховище документів NoSQL у режимі реального часу . Програми зберігають дані як об’єкти JavaScript Object Notation (JSON) та взаємодіють із базою даних за допомогою API JavaScript.

Realtime Database постачається з мобільними та веб-пакетами SDK, що дозволяє створювати додатки без потреби в серверах [4].

Realtime Database забезпечує синхронізацію та оновлення баз даних для всіх пристроїв, що працюють із додатком, у разі зміни даних.

Realtime Database підтримує офлайн-операції. Якщо пристрій переходить у режим офлайн, операції записуються в локальну пам’ять та синхронізуються, коли знову встановлено мережеве з’єднання для пристрою. Цей процес відбувається автоматично, без втручання користувача.

Realtime Database надає гнучку мову правил на основі виразів, яка називається Firebase Realtime Database Security Rules, щоб визначити, як дані повинні бути структуровані, коли і ким дані можуть бути прочитані або записані. При інтеграції з Firebase Authentication розробник може визначати переваги доступу певного користувача – до яких даних і в якому порядку може бути надано доступ.

Firebase Authentication – сервіси перевірки автентичності, прості у використанні SDK та готові бібліотеки інтерфейсу для автентифікації користувачів програми. Перевірити автентифікацію користувачів можна за допомогою електронної пошти та паролю, номерів телефонів, аккаунтів Google, або Facebook, або Twitter, та т. ін. Використання Authentication дозволяє налаштувати систему автентифікації в межах 10 рядків коду, який буде обробляти всю отриману інформацію про користувачів, включаючи складні операції, такі як об’єднання облікових записів [2].

Firebase Storage забезпечує зберігання відео, фотографій чи інших типів великих бінарних файлів, використовує безпечний спосіб передачі даних у Cloud Firestore. Це рішення також працює незалежно від якості мережі.

Cloud Firestore – це база даних документів NoSQL, яка дозволяє вам легко зберігати, синхронізувати і запитувати дані для ваших мобільних і веб-додатків - в глобальному масштабі. Хоча це може звучати як щось схоже на Realtime Database, Firestore привносить в платформу багато нового, що перетворює його на щось зовсім відмінне від бази даних в реальному часі.

У той час як Realtime Database зберігає дані у вигляді гігантського дерева JSON, Cloud Firestore використовує набагато більш структурований підхід. Firestore зберігає свої дані всередині об’єктів (документів). Ці документи складаються з пар ключ-значення і можуть містити дані будь-якого типу, від рядків до бінарних даних та навіть до об’єктів, які нагадують дерева JSON (Firestore називає їх картами). Документи, в свою чергу, згруповані в колекції [5].

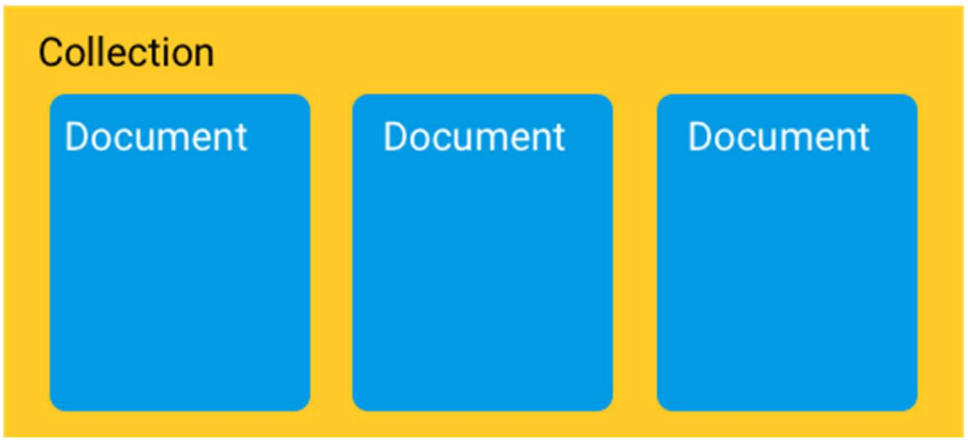

Рисунок 1.1 – Схема колекції документів

Firestore може складатися з декількох колекцій, які можуть містити документи, що вказують на вкладені колекції. Ці вкладені колекції можуть знову містити документи, що вказують на інші вкладені колекції і т.д.

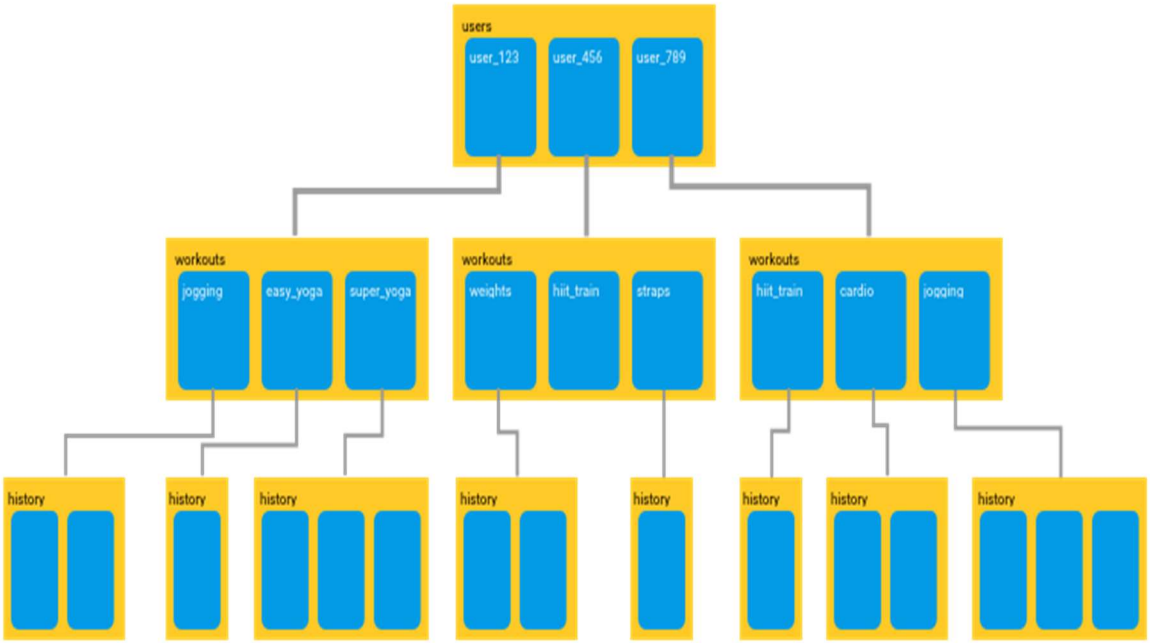

Рисунок 1.2 – Схема ієрархії документів бази даних Firebase

Firestore дає змогу створювати ієрархії для зберігання пов’язаних даних та легко діставати будь-які дані за допомогою запитів. Всі запити можуть масштабуватися в залежності від розміру набору результатів.

Запити Firestore тіньові, таким чином для отримати будь-який документу відсутня необхідність отримувати всі дані, що містяться в будь-якій з підколекцій, що пов’язані з документом [3].

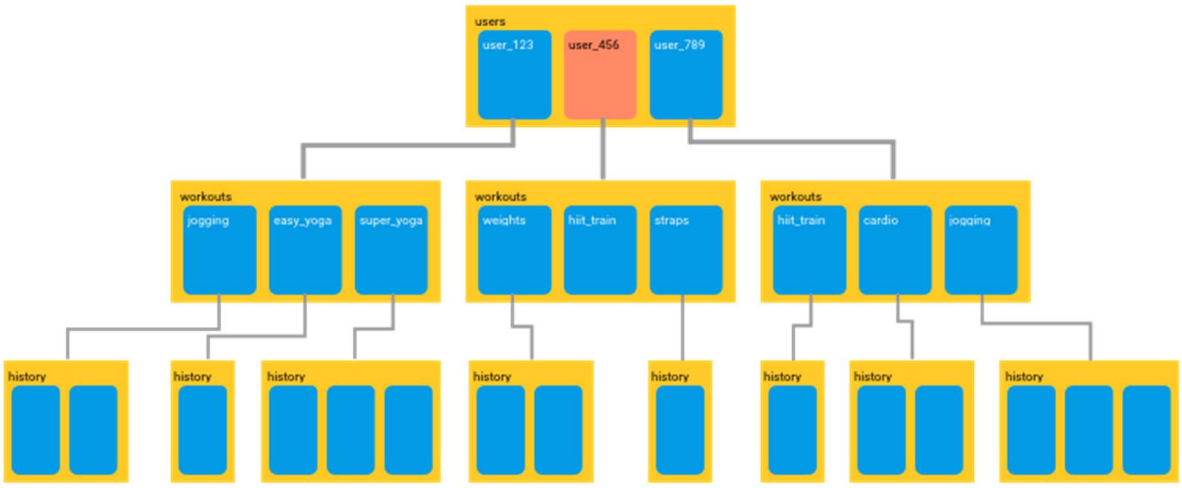

Рисунок 1.3 – Схема відповіді на запит до бази даних Firebase

Хоча Realtime Database здатна масштабуватися, все почне сходити з розуму, коли додаток стане дійсно популярним або якщо база даних стане дійсно масивної. Cloud Firestore заснований на хмарній інфраструктурі Google. Це дозволяє масштабувати його набагато простіше і з більшою ємністю, ніж база Realtime Database.

1.4 Додаткові можливості платформи

Firebase Test Lab – хмарне середовище для тестування програм Android, інтегрована в Android Studio, консоль Firebase та командний рядок G-cloud.

Тестове середовище дає розробникам можливість протестувати додатки на реальних пристроях, а не тільки на емуляторах, оскільки на ринку існує безліч різних типів пристроїв та конфігурацій Android.

Firebase Analytics для пропонує безкоштовне безлімітне звітування про 500 окремих подій. Analytics представляє дані про поведінку користувачів у додатках для iOS та Android, що дозволяє краще приймати рішення щодо підвищення продуктивності та маркетингу додатків.

Crashlytics – це репортер збоїв додатку в режимі реального часу, який допомагає розробникам відстежувати, визначати пріоритети та виправляти проблеми зі стабільністю, які знижують якість додатків. За допомогою Crashlytics розробники витрачають менше часу на організацію та усунення несправностей.

Firebase Cloud Messaging (FCM) – це інструмент для обміну повідомленнями між платформами, який дозволяє компаніям надійно отримувати та доставляти повідомлення на iOS, Android та в Інтернеті без будь-яких витрат.

Firebase Dynamic Links можна вважати «розумними» посиланнями. Це інтелектуальні URL-адреси, які дозволяють відправляти існуючих і потенційних користувачів в будь-яке місце в додатку. Користувачі отримують кращий доступ до платформи, на якій вони відкривають ваше посилання. Якщо користувач відкриває динамічне посилання, їх можна перенести безпосередньо на пов’язаний контент у вашому додатку. Якщо користувач відкриває одне і те ж динамічне посилання в браузері, його можна перевести на еквівалентний контент на необхідному веб-сайті.

Крім того, якщо користувач відкриває динамічне посилання та не має встановленої програми, користувачеві може бути запропоновано встановити його, після установки додаток запускається та може отримати доступ до посилання.

Динамічні посилання можна використовувати та передавати так само, як «звичайні» посилання. Це означає, що розробники можуть

використовувати їх в інших веб-додатках, в оновленнях соціальних медіа та в електронних повідомленнях.

Firebase Invites – кросплатформене рішення, яке дозволяє користувачам надсилати персоналізовані запрошення для встановлення програми. Ці запрошення надсилаються у вигляді електронних листів та SMS-повідомлень іншим користувачам. Firebase Invites використовує вищезазначене рішення щодо Dynamic Links, щоб полегшити користувачам перенаправлення на програми Firebase.

AdMob – спосіб монетизувати мобільні додатки, розміщуючи рекламу. Це також вже частина платформи Firebase, за допомогою якої розробники можуть демонструвати рекламу мільйонів рекламодавців Google в режимі реального часу. Платформа Firebase також надає змогу використовувати AdMob Mediation – рекламне рішення, яке пропонує рекламу з більш ніж 40 рекламних мереж для поліпшення конкуренції, спрощення рекламних операцій та в кінцевому підсумку збільшення прибутку [5].

Firebase пропонує безкоштовний план з 1 ГБ зберігання баз даних у режимі реального часу та двома платними планами підписки, ціна яких залежить від об’єму, що надається для зберігання баз даних. Всі плани включають тестування, аналітику, індексацію додатків, автентифікацію, хмарні повідомлення, аварійні зв’язки, динамічні посилання, запрошення, моніторинг продуктивності, прогнози та віддалене налаштування.

Отже, Firebase є ефективною, надійною та економічно обґрунтованою платформою для розробки додатків, що забезпечують роботу з базами даних, для мобільних операційних систем.

2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ ТА ДОДАТКУ

Розглянувши вимоги, що були сформульовані в розділі 1, розробимо моделі поведінки додатку за допомогою об’єктно-орієнтовної мови UML. Графічна мова UML призначена для візуалізації, опису параметрів, конструювання та документування різних систем, допомагає визначити та графічно представити основний функціонал додатку та принципи його роботи.

2.1 Концептуальна модель бази даних

Концептуальне моделювання дозволяє врахувати логічне уявлення структури даних у базі даних та є основою подальшого проектування бази даних та додатку для роботи з нею.

При розробці концептуальної моделі всі об’єкти бази даних позначаються прямокутником, атрибути, що характеризують об’єкт – овалом, а зв’язки між об’єктами – ромбом. Потужність зав’язків характеризуються позначками біля стрілок взаємовідношення.

Між користувачем та акаунтом встановлено зв’язок «один до багатьох». Між акаунтом та документом – «один до багатьох».

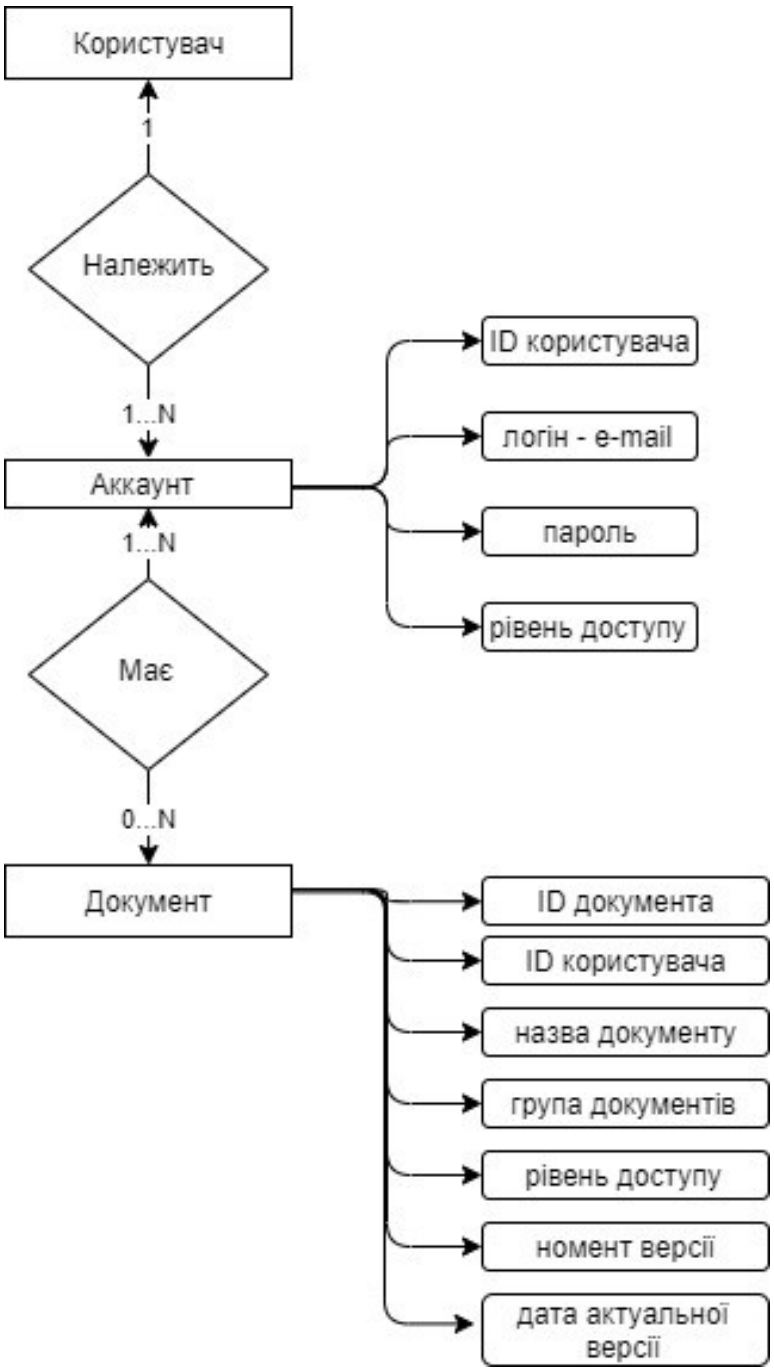

Рисунок 2.1 – Концептуальна модель бази даних

Отже, наведена концептуальна модель предметної області є основою для розробки фізичної моделі бази даних. Оскільки необхідно забезпечити роботу системи в реальному часі, найбільш перспективним є використання документо-орієнтованих систем керування базами даних.

2.2 Діаграми поведінки

Use Case Diagram – діаграма варіантів використання – призначена для моделювання функціональних вимог до системи, що представлені у вигляді сценаріїв взаємодії користувача із системою.

Рисунок 2.2 – Діаграма варіантів використання

Наведена діаграма використання є основою для розробки інтерфейсу програмного застосунку. Так, основні функціональні можливості повинні реалізовуватись засобами інтуїтивно зрозумілих графічних примітивів (поля вводу, списки, кнопки).

2.3 Діаграми взаємодії

Interaction diagrams – діаграми взаємодії – призначені для модулювання процесів обміну повідомленнями між об’єктами. Можливі варіанти діаграм взаємодії – діаграми послідовності, кооперації, діяльності, станів.

Sequence diagrams – діаграма послідовності – зображує тимчасову послідовність подій, що відбуваються в рамках одного варіанту використання, тобто встановлює порядок дій користувача, впорядковані за часом.

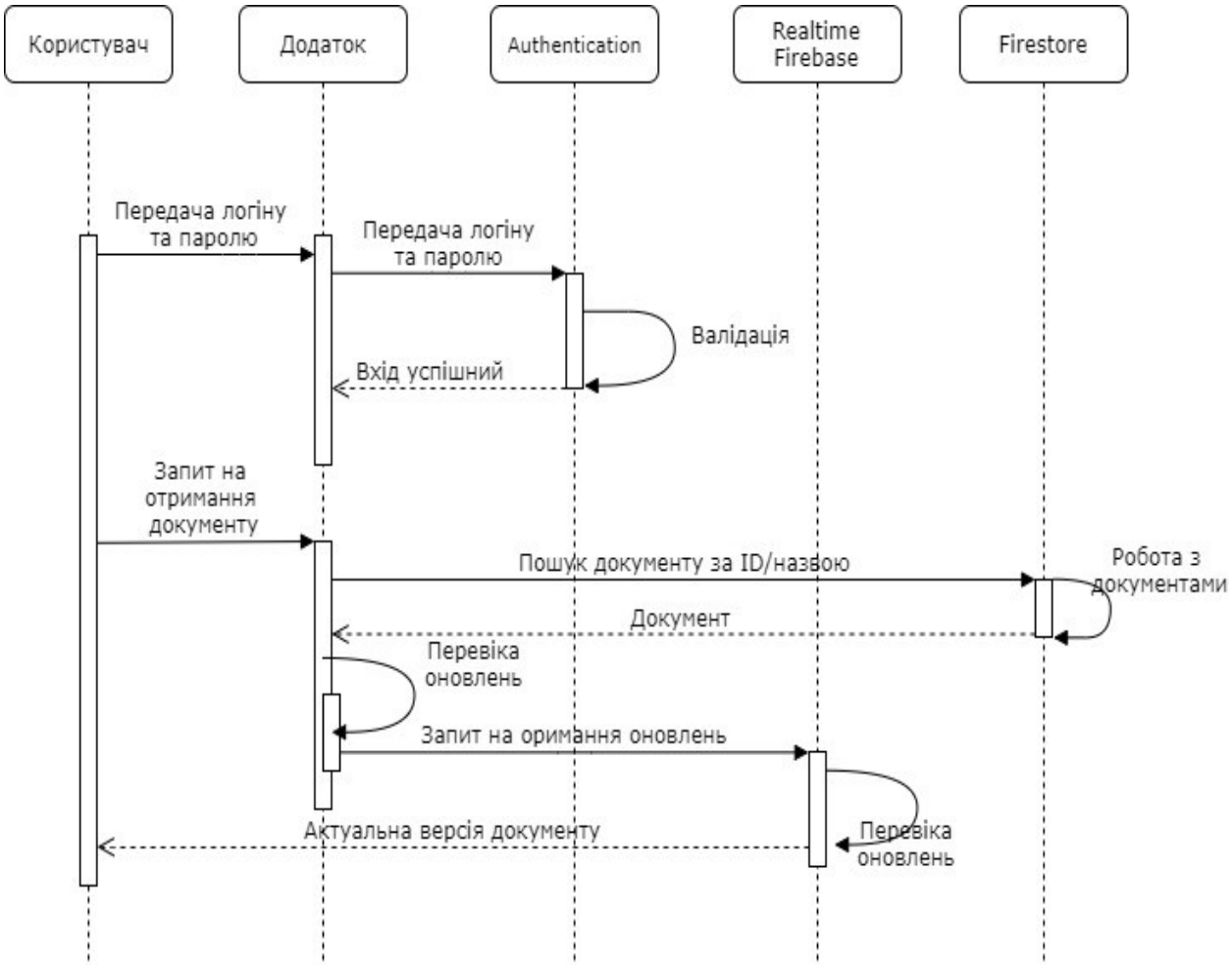

Рисунок 2.3 – Діаграма послідовності

Наведена діаграма послідовності дає змогу прослідкувати періоди активності кожного із компонентів системи – користувача, самого додатку та баз даних. Із рисунку 2.3 видно, в які періоди часу та за якими запитами користувача задіяні різні бази даних – Authentication, Realtime Database та Firestore.

Activity diagrams – діаграма діяльності – для моделювання поведінки системи в рамках різних варіантів використання, або потоків управління.

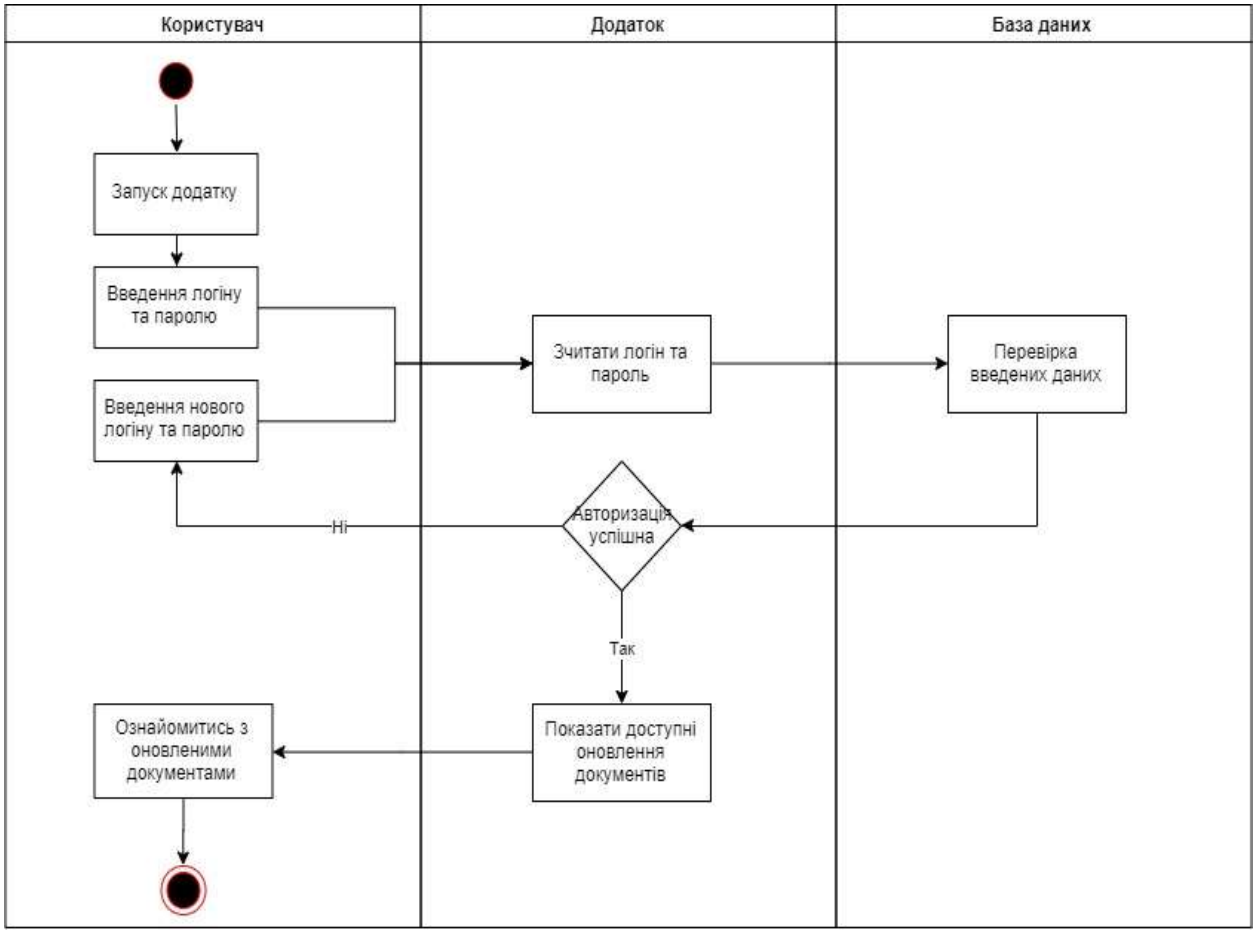

Рисунок 2.4 – Діаграма діяльності

На рисунку 2.4 наведено приклад діаграми діяльності для сценарію валідації користувача в системі. На схемі видно які елементи системи Користувач – Додаток – База даних (в даному випадку Authentication) діють на кожному кроці вказаної діяльності. Коли користувач запускає додаток,

вводить логін та пароль, додаток зчитує ці дані. Далі додаток звіряю отримані дані із записами в базі даних. У випадку незадовільної валідації користувача, додаток не надає доступу користувачеві до наявних документів та запитує нові логін та пароль. Доступні документи з’являються тільки для успішно авторизованого користувача.

Statechart diagrams – діаграма станів – моделювання поведінки об’єктів системи при переході з одного стану в інший. Діаграма станів представляє динамічне поводження сутностей, на основні специфікації їхньої реакції на сприйняття конкретних подій.

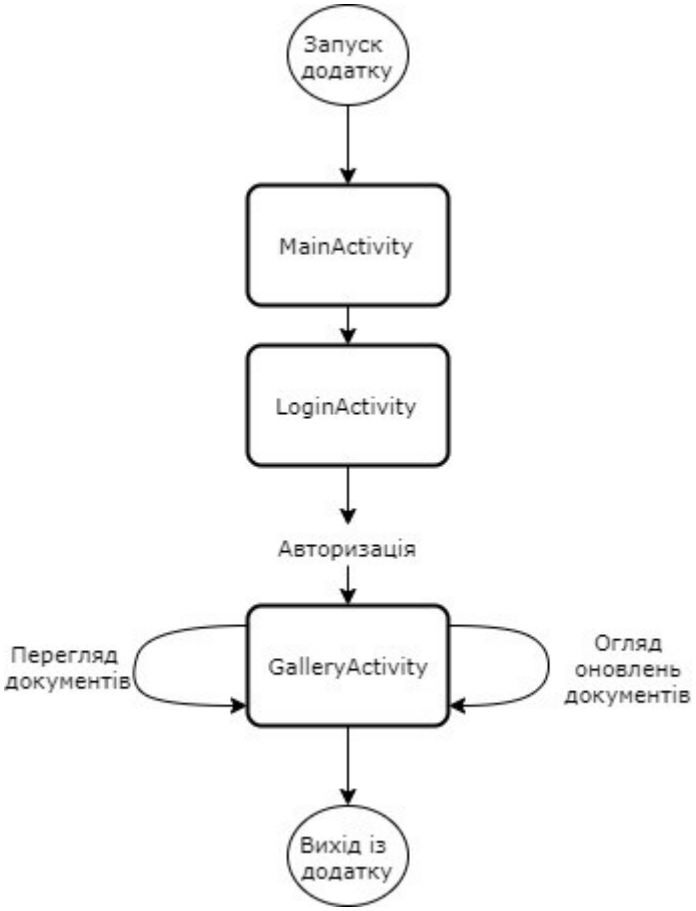

Рисунок 2.5 – Діаграма станів

Діаграма станів зображує основні активності розроблюваного додатку впродовж його життєвого циклу. Після запуску додатку користувач повинен перейти до активності авторизації з метою проходження валідації в базі

даних. Після успішної авторизації користувач має змогу бачити галерею документів, які йому доступні.

2.4 Структурні діаграми

Діаграма класів відображає співвідношення між класами додатку.

В розроблюваному додатку для спрощення системи управління документованою інформацією розроблено два класи: User та UserDocument. Поля класу User містять атрибути користувача. Поля класу UserDocument в свою чергу містять характеристики документованої інформації. Поля кожного із класі, окрім ідентифікаторів, не є обов'язковими. Наповнюваність полів екземпляру класу за необхідістю.

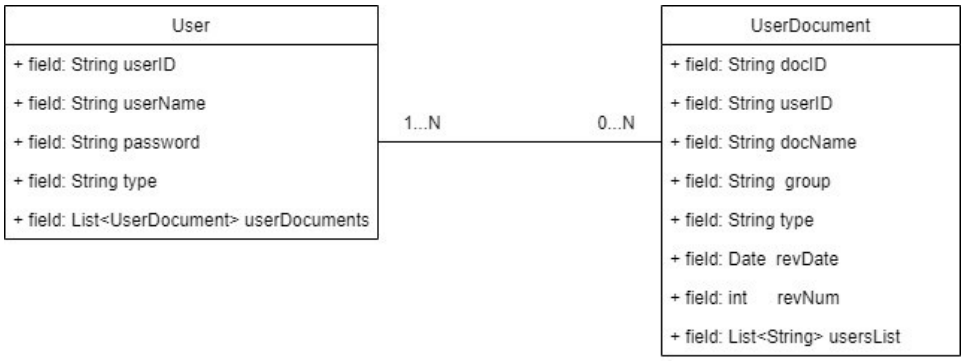

Рисунок 2.6 – Діаграма класів

Component diagrams – діаграма компонентів – використовується для візуалізації структурних компонентів та відношень між ними. На цій діаграмі справа представлені бібліотеки, які буде використано при розробці додатку.

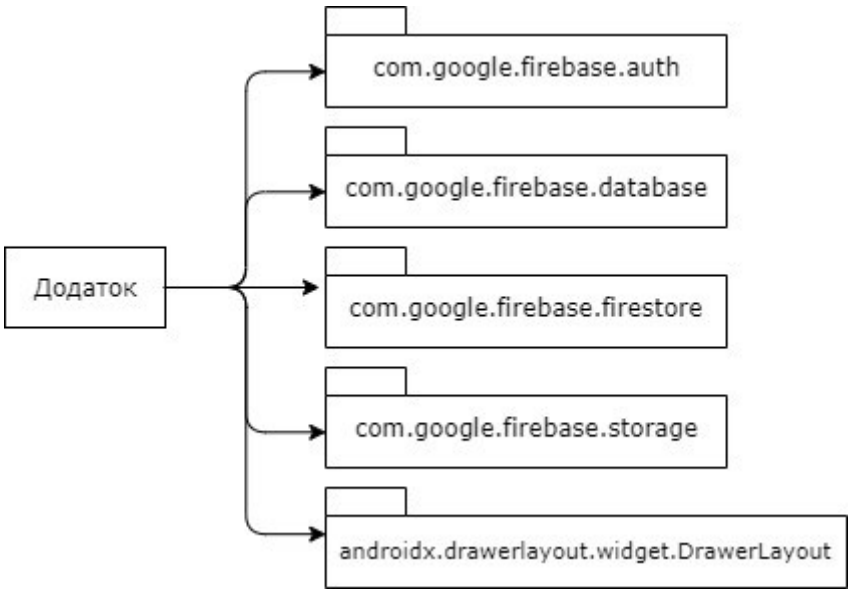

Рисунок 2.7 – Діаграма компонентів

Deployment diagrams – діаграма розгортання – використовується для моделювання фізичної архітектури системи.

Рисунок 2.8 – Діаграма розгортання

Таким чином, як можна бачити на рисунку 2.8, розгортання застосунку потребує лише наявності мобільного пристрою та баз даних.

2.5 Ескізи основних активностей

Робота додатку розпочинається із головної активності MainActivity. Основна задача цієї активності надати змогу користувачеві перейти на інші активності додатку.

Рисунок 2.9 – Ескіз MainActivity

При варіанті вибору опції Меню «Вхід/реєстрація» додаток переходить до активності LoginActivity (див. рис. 2.10), а при виборі опції «Галерея документів» – до GalleryActivity (див. рис. 2.11).

Рисунок 2.10 – Ескіз LoginActivity

Рисунок 2.11 – Ескіз GalleryActivity

Оскільки однією з вимог, що викладені у підрозділ 1.1, є доступність інтерфейсу, ескізи основних активностей виконано у мінімалістичному стилі. Планується використання Navigation Drawer menu, для зручності доступу до активностей (див. рис. 2.9). Navigation Drawer menu є стандартним типом головного меню як для мобільних застосунків, так і для Web. Перевагою цього типу меню є відповідність технології Material Design.

Material Design є стандартом дизайну мобільних та web додатків, основними принципами якого є мінімалістичність, тобто відсутність візуальних ефектів, які б відволікали користувача та плоскі графічні елементи.

3 РОЗРОБКА ТА ТЕСТУВАННЯ ДОДАТКУ

3.1 Реалізація підключення бібліотек Firebase

Після створення проекту DocViewer в середовищі розробки Android Studio, необхідно інтегрувати бібліотеки Google Firebase. Для цього додано правило в файл build.gradle рівня build.gradle, щоб включити плагін google-services:

```
uildscript {
    repositories {
        google()
        jcenter()

    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.5.0'
        classpath 'com.google.gms:google-services:4.2.0'
    }
}

allprojects {
    repositories {
        google()
        jcenter()

    }
}
```

Після цього в модулі Gradle (app/build.gradle) додано плагін applyЮ щоб ввімкнути плагін Gradle:

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

android {
…
   }
   buildTypes {
…
      }
   }
}

dependencies {
   implementation fileTree(dir: 'libs', include: ['*.jar'])
   implementation 'androidx.appcompat:appcompat:1.0.2'
   implementation 'androidx.legacy:legacy-support-v4:1.0.0'
   implementation 'com.google.android.material:material:1.0.0'
   implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
   implementation 'androidx.navigation:navigation-fragment:2.0.0'
   implementation 'androidx.navigation:navigation-ui:2.0.0'
   implementation 'androidx.lifecycle:lifecycle-extensions:2.0.0'
   implementation 'com.google.firebase:firebase-auth:16.0.5'
   implementation 'com.google.firebase:firebase-database:16.0.4'
   implementation 'com.google.firebase:firebase-storage:16.0.4'
   implementation 'androidx.annotation:annotation:1.0.2'
   testImplementation 'junit:junit:4.12'
   androidTestImplementation 'androidx.test:runner:1.1.1'
```

```
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.1'
}
```

Таким чином можна бачити, що до проекту DocViewer підключено три бібліотеки Firebase Authentication, Firebase Database та Firebase Storage.

3.2 Реєстрація та авторизація в системі користувача

Авторизації в системі виконується з використання бібліотеки Firebase Authentication. Реєстрація та подальший вхід до акаунту виконується за допомогою електронної пошти. Електронна адреса в цьому випадку виступає як ім’я користувача, але система автоматично призначає користувачам унікальний ідентифікатор, який потім використовується в базі даних як ключ.

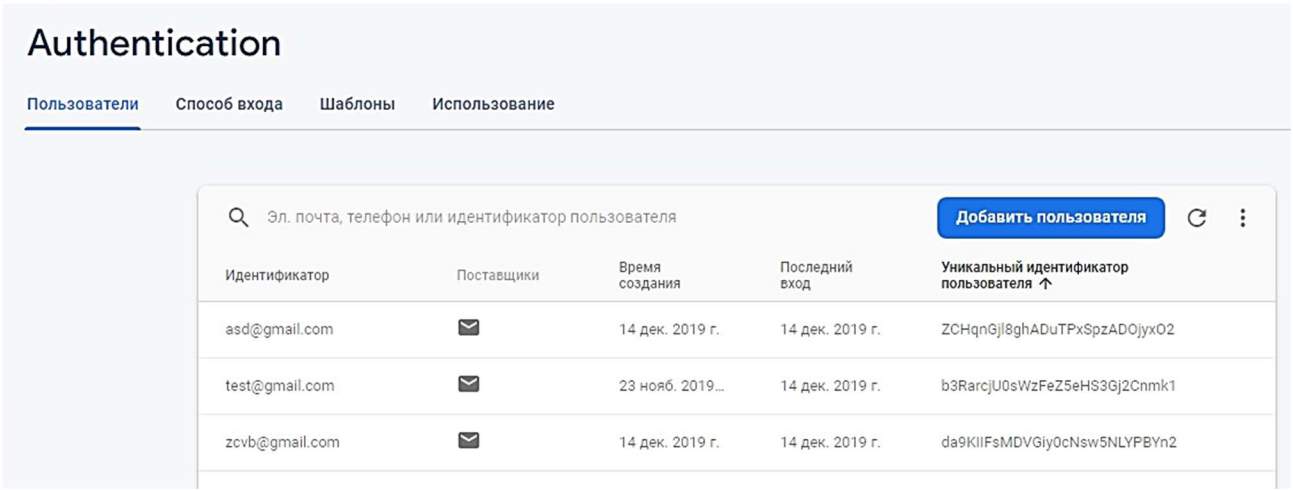

Рисунок 3.1 – Перелік акаунтів Firebase Authentication

На рівні додатку вхід та автентифікація відбувається за кодом, наведеним нижче. Система визначає, чи здійснює вхід зареєстрований користувач чи є необхідність реєстрації. При реєстрації автоматично

вноситься запис до бази зареєстрованих користувачів та відбувається автентифікація.

```
…
   FirebaseAuth                    firebaseAuth;
   FirebaseAuth.AuthStateListener  mAuthListener;
   DatabaseReference               FirebaseRef;

   @Override
   public void onCreate (Bundle savedInstanceState) {
…
      firebaseAuth        = FirebaseAuth.getInstance();
      usernameEditText    = findViewById(R.id.username);
      passwordEditText    = findViewById(R.id.password);
      loginButton         = findViewById(R.id.login);
      FirebaseRef         = FirebaseDatabase.getInstance().getReference();
      mAuthListener       = new FirebaseAuth.AuthStateListener() {
         @Override
         public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
            FirebaseUser usr       = firebaseAuth.getCurrentUser();
….};
      FirebaseRef.addValueEventListener( new ValueEventListener() {

         @Override
         public void onDataChange(DataSnapshot dataSnapshot) {
               for (DataSnapshot data : dataSnapshot.getChildren()) {
               User user     = data.getValue(User.class);
               custList.add(user.userName); }
         }
      }); }
```

3.3 Фізичний рівень баз даних

Для реалізації бази даних додатку використовуються бібліотека Firebase Realtime Database та Firebase Cloud Storage.

Додавання документу до бази відбувається із застосуванням коду, що наведено нижче.

```
myRef.addValueEventListener(new ValueEventListener() {

   @Override
   public void onDataChange(DataSnapshot dataSnapshot) {
      String value = dataSnapshot.getValue(UserDocument.class);}

   @Override
   public void onCancelled(DatabaseError error) {
      Log.w(TAG, "Failed to read value.", error.toException()); }
});
```

Робота з файлам в Firebase Cloud Storage відбувається завдяки виконання наступних методів.

```
…
private StorageReference        mStorageRef; getReference();
mStorageRef       = FirebaseStorage.getInstance().getReference();
…

Uri                 file          = Uri.fromFile(new File("readme.txt"));
StorageReference  riversRef     = storageRef.child("file.txt");

riversRef.putFile(file)
```

```
   .addOnSuccessListener(new    OnSuccessListener<UploadTask.TaskSnapshot>()
{
      @Override
      public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
         Uri downloadUrl = taskSnapshot.getDownloadUrl();  }
   })
   .addOnFailureListener(new OnFailureListener() {
      @Override
      public void onFailure(@NonNull Exception exception) {…});
…
File    localFile      = File.createTempFile("texts", "txt");
riversRef.getFile(localFile)
   .addOnSuccessListener(new
OnSuccessListener<FileDownloadTask.TaskSnapshot>() {
      @Override
      public void onSuccess(FileDownloadTask.TaskSnapshot taskSnapshot) {…}
   }).addOnFailureListener(new OnFailureListener() {
      @Override
      public void onFailure(@NonNull Exception exception) {…}
   });
```

Картка документу бази даних виглядає, як показано на рисунку 3.2.

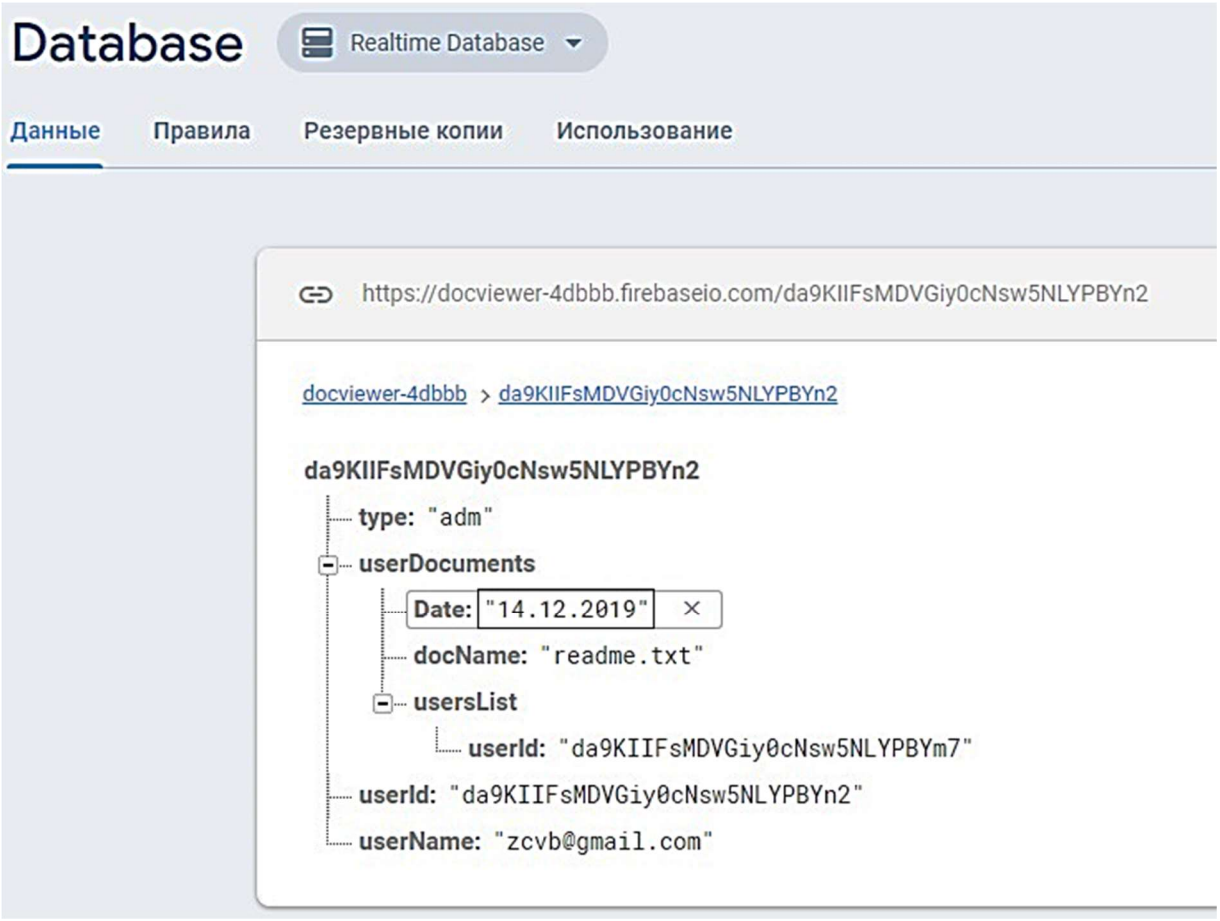

Рисунок 3.2 – Картка документу із бази даних Firebase Realtime Database

3.4 Додавання віджетів

Для зручності навігації додатком використано стандартну бібліотеку Android Drawer Layout.

Підключення бібліотеки відбувається додаванням коду:

```
import androidx.drawerlayout.widget.DrawerLayout;
```

Бібліотека забезпечує функціонування віджета меню та спрощує навігацію між активностями додатку.

Для обробки стандартних кнопок віджета меню відповідно до вимог додатку в коді основної активності додано наступний запис:

```
@Override
   public boolean onNavigationItemSelected(MenuItem item) {
      // Handle navigation view item clicks here.
      int id = item.getItemId();

      if (id == R.id.nav_login) {
         Intent  intent = new Intent(this, LoginActivity.class);
         startActivity(intent);
      }
      else if (id == R.id.nav_gallery)
      {
         Intent  intent = new Intent (this, GalleryActivity.class);
         startActivity(intent);
      }
      else if (id ==  R.id.nav_exit) {
         super.finishAffinity();
         System.exit(0);
      }
      DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
      drawer.closeDrawer(GravityCompat.START);
      return true;
   }
```

Таким чином, неавторизований користувач має вибір – пройти авторизацію, перейшовши за пунктом меню nav_login або переглянути перелік документованої інформації загального призначення за пунктом

nav_gallery. Користувач, який перейшов за пунктом nav_login, ввід логін та пароль та успішно пройшов валідацію в базі Firebase Authentication, при переході до галереї документів буде мати змогу бачити перелік тієї документованої інформації, для управління якою йому надано право.

3.5 Тест-план

Важливим етапом створення програмного забезпечення є тестування якості програмного забезпечення. Метою тестування є виявлення та усунення помилок – відхилень фактичного результату роботи програми від очікуваного [7]. При використанні каскадної моделі життєвого циклу програмного забезпечення тестування розглядається як один з завершальних етапів розробки. Однак, сучасні стратегії розробки, передбачають виконання тестування одночасно з розробкою. В цьому випадку важливим є створення тест-плану – документу, що описує весь обсяг робіт з тестування програмного забезпечення та опис тестових випадків [7]. Тестові випадки наведено в таблиці 3.1

Таблиця 3.1 – Тестові випадки

№З\п	Опис	Кроки відтворення	Очікуваний результат
1	Коректність вводу пошти	1) викликати меню Login; 2) ввести некоректну поштову адресу	З’являється toast повідомлення про некоректну пошту. Кнопка Login блокується

Продовження таблиці 3.1

2	Коректність вводу пароля	1) викликати меню Login; 2) ввести зареєстровану поштову адресу; 3) ввести некоректний пароль	З’являється toast повідомлення про некоректний пароль. Кнопка Login блокується
3	Вхід в додаток	1) викликати меню Login; 2) ввести зареєстровану поштову адресу; 3) ввести коректний пароль	Кнопка Login доступна. Після натискання з’являється toast повідомлення про успішний вхід. Перехід на активність Gallery
4	Реєстрація в додатку	1) викликати меню Login; 2) ввести нову поштову адресу; 3) ввести пароль	Кнопка Login доступна. Після натискання з’являється діалогове вікно з пропозицією реєстрації нового користувача. Після підтвердження з’являється toast повідомлення про успішну реєстрацію. Перехід на активність Gallery

Продовження таблиці 3.1

5	Відображення документів користувача	1) зареєструватися з існуючим обліковим записом; 2) звернути увагу на вміст активності Gallery	Відображаються документи зареєстрованого користувача та їх статистика
6	Надання доступу іншим користувачам	1) зареєструватися з існуючим обліковим записом; 2) перейти на активність Gallery; 3) у віджеті спадаючий список обрати користувачів, для яких слід відкрити доступ до обраних документів; 4) зареєструватися з обліковим записом одного з користувачів, обраних на попередньому кроці	Відображаються документи, відкриті для спільного доступу

Отже, після створення переліку тестових випадків, слід виконати тестування, згідно з описаними пунктами. Тест-план мобільного додатку наведено нижче.

Мета тест-плану – опис процесу тестування мобільного застосунку DocViewer, призначеного для автоматизації роботи сховища документів реального часу.

Елементи системи – мобільний застосунок та база даних Firebase.

Вимоги до апаратного та програмного забезпечення. Мінімальна версія ОС Android для коректної роботи мобільного застосунку – 7.0. Це пов’язано з необхідністю використання хмарних сервісів Firebase.

Стратегія тестування. Тестування планується виконувати методом чорної скриньки. Основний вид тестування – функціональне тестування, тобто перевірка відповідності вимогам, які викладено у підрозділі 1.1. Також планується використання навантажувального тестування.

Функціональне тестування. Мета: виділення функціональних помилок, невідповіностей очікуваній поведінці програми за допомогою тестових сценаріїв.

Классификація функцій:

А) реєстрація та авторизація;

1) реєстрація користувача

2) авторизація користувача

3) анонімний користувач

4) відновлення пароля

5) редагування облікового запису

Б) робота з документами

1) додавання нового документу користувачем

2) видалення документу

3) асоціація одного документу з декількома користувачами

Навантажувальне тестування. Мета: виділення можливої граничної кількості користувачів, що можуть одночасно працювати у додатку.

ВИСНОВКИ

В ході роботи проаналізовано можливості роботи із застосуванням платформи Google Firebase Android.

Підсумовуючи матеріали, викладені у кваліфікаційній роботі можна зробити висновок, що для роботи з класами та методами бібліотек Android, Java та інтерфейсами прикладного програмування Google існує широкий спектр документації, яку можна отримати як у Android Studio, Google Firebase так і з інтернет ресурсів.

За результатами роботи був спроектований, реалізований та протестований додаток DocViewer, який застосовується для забезпечення управління документованою інформацією в умовах однієї компанії. Цей додаток був підключений до хмарної бази даних Firebase Authentication, де зберігаються реєстраційні дані користувачів. Для створення системи аккаунтів потрібно задіяти класи FirebaseAuth з бібліотеки Firebase API. Firebase Cloud Firestore, де зберігаються файли користувачів, та до бази даних реального часу Firebase Realtime Database, для цього використані класи FirebaseFirestore.

ПЕРЕЛІК ПОСИЛАНЬ

1. Google Firebase Documentation. [Електронний ресурс] URL: https://firebase.google.com/docs (дата звернення: 17.10.2019)

2. Google Firebase Guides. [Електронний ресурс] URL: https://firebase.google.com/docs/guides (дата звернення: 17.10.2019)

3. Doug Stevenson. What is Firebase? The complete story, abridged. https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0, Sep 25, 2018 (дата звернення: 18.10.2019)

4. Doug Stevenson. The top 10 things to know about Firestore when choosing a database for your app. [Електронний ресурс] Дата публікації: Jan 2, 2019. URL: https://medium.com/@CodingDoug/ the-top-10-things-to-know-about-firestore-when-choosing-a-database-for-your-app-a3b71b80d979 (дата звернення: 22.10.2019)

5. John E Lincoln, Everything You Need to Know About Google Firebase. [Електронний ресурс] Дата публікації: June 2, 2016. URL: https://ignitevisibility.com/everything-need-know-google-firebase/ (дата звернення: 22.10.2019)

6. DEFINITION. Google Firebase. [Електронний ресурс]. URL: https://searchmobilecomputing.techtarget.com/definition/Google-Firebase (дата звернення: 22.10.2019)

7. Навчальний посібник «Методи тестування та оцінки якості програмного забезпечення із застосуванням Pairwise тестування» для студентів денної та заочної форми навчання / Гаврилей Н. В. [та ін.]. Полтава: ПолтНТУ, 2016. 391 с.