МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ АВТОМАТИЗАЦІЇ БРОНЮВАННЯ КІМНАТ ДЛЯ ВІДЕО-КОНФЕРЕНЦІЙ»**

Виконала: студентка 2 курсу, групи 8.1220-з
спеціальності 122 комп'ютерні науки
освітньої програми комп'ютерні науки
(шифр і назва спеціальності)
Є.В. Бєлоус
(ініціали та прізвище)

Керівник доцент кафедри комп'ютерних наук, доцент, к.т.н. Решевська К.С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри програмної інженерії, доцент, к.ф.-м.н. Лісняк А.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедракомп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 122 комп'ютерні науки

Освітня програма комп'ютерні науки

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

комп'ютерних наук, д.т.н., проф.

_____ Чопоров С.В.

(підпис)

"_14_" __06___ 2021 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Бєлоус Євгенії Вячеславівні

(прізвище, ім'я та по– батькові)

1. Тема роботи (проекту) Розробка мобільного додатку автоматизації бронювання кімнат для відео-конференцій

керівник роботи (проекту) Решевська Катерина Сергіївна, к.т.н, доцент

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

затверджені наказом ЗНУ від « 09 » червня 2021 року № 850-с

2. Строк подання студентом роботи 19.11.2021

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

4. Зміст розрахунково– пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі, аналіз предметної області.
2. Моделювання та проектування мобільного додатку.
3. Реалізація та тестування мобільного додатку.

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**
1			
2			
3			
Додатки			

7. Дата видачі завдання 14.06.2021

К А Л Е Н Д А Р Н И Й П Л А Н

№	**Назва етапів кваліфікаційної роботи**	**Строк виконання етапів роботи**	**Примітка**
1.	Розробка плану роботи.	15.06.2021	
2.	Збір вихідних даних.	20.06.2021	
3.	Обробка методичних та теоретичних джерел.	01.07.2021	
4.	Розробка першого та другого розділу.	23.09.2021	
5.	Розробка третього розділу.	19.11.2021	
6.	Розробка четвертого розділу.	20.11.2021	
7.	Оформлення та нормоконтроль кваліфікаційної роботи бакалавра.	22.11.2021	
8.	Захист кваліфікаційної роботи бакалавра.	08.12.2021	

Студентка _____ Є.В. Бєлоус
(підпис) (ініціали та прізвище)

Керівник роботи _____ К.С. Решевська
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спица
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка мобільного додатку автоматизації бронювання кімнат для відео-конференцій»: 51 с., 24 рис., 11 джерел.

МОБІЛЬНИЙ ДОДАТОК, ANDROID, ANDROID STUDIO, FIREBASE, FLAMELINK, KOTLIN

Об’єкт дослідження – засоби розробки мобільного додатку.

Предмет дослідження - автоматизоване бронювання переговорних кімнат через мобільний додаток.

Мета роботи: розробити мобільний додаток, що допоможе автоматизувати бронювання переговорних кімнат для проведення відео-конференцій.

Метод дослідження – методи збору та аналізу вимог до програмного забезпечення, методи моделювання, проектування, конструювання та тестування програмного забезпечення.

При розробці мобільного додатку було проведено аналіз предметної області, спроектована функціональність, реалізовано додаток за допомогою платформи Android Studio, мови програмування Kotline та бази данних Firebase, здійснене тестування мобільного додатку. В результаті роботи отримано мобільний додаток для автоматизації бронювання переговорних кімнат.

SUMMARY

Master’s Qualification Thesis «Development of a Mobile Application to Booking Video Conference Room Automation»: 51 pages, 24 figures, 11 references.

MOBILE APP, ANDROID, ANDROID STUDIO, FIREBASE, FLAMELINK, KOTLIN

The object of research is the means of developing a mobile application

The subject of research - automated booking of meeting rooms through a mobile application.

The aim of the study is to develop a mobile application that will help automate the booking of meeting rooms for video conferencing.

Research method - methods of collecting and analyzing software requirements, methods of modeling, designing, designing and testing software.

During the development of the mobile application, the subject area was analyzed, the functionality was designed, the application was implemented using the Android Studio platform, the Kotline programming language and the Firebase database, and the mobile application was tested. As a result of work the mobile application for automation of reservation of meeting rooms is received.

ЗМІСТ

ВСТУП

Важко уявити сучасну людину, що не користується мобільним пристроєм. Смартфон, планшет чи інші гаджети, на мою думку, міцно та надовго закріпились у нашому світі, як незмінні помічники засобів комунікацій. З появою нових мобільних пристроїв та платформ, на яких вони розробляються, активно розвивається сфера розробки мобільних додатків для різних сфер діяльності.

Функції мобільних додатків майже не відрізняються від функцій сайтів, завдання обох надати інформацію користувачу що до послуг чи товарів, які надають відповідні компанії. Одна з відмінностей мобільних додатків в тім, що вони надають більше варіантів піднесення інформації. Користувачу нема необхідності відкривати браузер, вводити в пошуку назву сайту чи його веб-адресу, достатньо натиснути на ярлик встановленого додатку й уся інформація перед очима.

Метою кваліфікаційної роботи є розробка мобільного додатку для автоматизації бронювання переговорних кімнат. Потреба виникла у зв’язку з тим, що через плутанину з занятістю кімнат компанія може зазнавати збитків, через зірвані перемови, втрачати перспективних співробітників та псувати свою репутацію.

Кваліфікаційна робота складається з вступу, чотирьох розділів, висновків, літератури та додатків.

Перший розділ містить опис мети розробки, розглянуті існуючі мобільні платформи та обґрунтовано .

Других розділ містить опис засобів розробки мобільного додатку.

Третій розділ складається з процесу розробки мобільного додатку.

Четвертий розділ включає в себе тестування створеного додатку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Мета розробки мобільного додатку

В даний час внаслідок збільшення доступності мобільного Інтернету спостерігається тенденція до використання мобільних додатків для здійснення таких дій, як замовлення, бронювання або купівля будь-яких товарів та послуг. Це дозволяє користувачеві уникнути необхідності самостійно здійснювати телефонний дзвінок, який може обернутися тривалим очікуванням, недоступністю абонента та витратою коштів. Крім того, не завжди користувач опиняється в умовах тиші та можливості говорити по телефону, що робить використання мобільного додатка зручнішим, ніж здійснення дзвінка.

Ідея створення цього проекту виникла з метою полегшення взаємодії користувача з сервісом та фахівцями. Компанія, для якої розроблятиметься програма, часто вдається до функціоналу відео-зустріч. Для проведення відео-конференцій використовуються спеціально відведені кімнати, де встановлено термінал відео-конферец зв’язку. Зважаючи на те, що не налагоджена система бронювання цих кімнат - зрив нарад не рідкість. Ця проблема може спричинити фінансові збитки, погіршення репутації компанії, втрату кадрів.

Щоб уникнути вищевказаних наслідків, було прийнято рішення створення сервісу для бронювання через мобільний додаток. Веб-версія цього функціоналу вже є в компанії, проте його використання не практично, оскільки постійно носити з собою персональний компьютер або ноутбук не зручно. Чого не скажеш про мобільний телефон, пристрій не займає багато місця і практично скрізь зі своїм власником.

1.2 Платформа Android

Платформа “Android” є продуктом групи “Open Handset Alliance” , яка має мету створити найбільш досконалу мобільну систему. Дивлячись з точки зору розробки програмного забезпечення “Android” знаходиться в самому центрі розробки відкритого програмного забезпечення. Перший пристрій, що отримав цю систему, був випущений у 2008 році і єдиним інструментом розробки програм для нього були почергові випуски пакета розробки “SDK”.

За шириною можливостей платформа “Android” не програє операційним системам, що встановлені на персональних комп’ютерах. Це багаторівнева система на основі ядра “Linux” з широкими функціональними можливостями. В підсистему інтерфейсу користувача входять вікна, представлення та віджети. “Android” має великий перелік можливостей підключення, таких як “Wi-Fi”, “Bluetooth” та протоколи передачі даних через стільникову мережу. В стек програмного забезпечення “Android” входить і підтримка сервісів, заснованих на визначенні місцеположення та акселерометрів , хоча не всі пристрої на цій платформі устатковані необхідним обладнанням. Також наявна підтримка відеокамери. Історично двома областями мобільні системи відставали від настільних, це були графіка та способи збереження даних. “Android” вирішує проблему графіки завдяки вбудованій підтримці графіки, включаючи бібліотеку “OpenGl”. Задача збереження даних спрощується завдяки наявності в платформі “Android” популярної бази даних з відкритим кодом “Sqlite”.

Операційна система “Android” працює поверх ядра “Linux”. Додатки пишуться на мові програмування “Java” і виконуються в віртуальній машині. Важливо уточнити, що віртуальна машина це не “JVM”, що більш очікувано, а відкрита технологія “Dalvik Virtual Machine”. Кожний додаток “Android” запускається всередині екземпляра “Dalvik VM” , який в свою чергу укладений в рамках контрольованого ядром “Linux” процесу.

Компанія Google вклала багато ресурсів в скорочення платформи та зростання її ефективності. Оптимізація в першу чергу направлена на зменшення розміру, збільшення швидкості, економію заряду акамулятора та зниження витрат пам’яті. Вони провели роботу на декількох рівнях стеку. Віртуальна машина Dalvik була ретельно розроблена з метою задоволення цих вимог до продуктивності та має декілька незвичайних характеристик. На відміну від звичайного виконуваного середовища Java, Dalvik заснований на регістрі, а не на стеку і не підтримує JIT компіляцію. Кожен окремий додаток виконується в окремому екземплярі віртуальної машини Dalvik і пам’ять розподіляється між цими екземплярами для зменшення витрат. Для того щоб скоротити час запуску додатка, Dalvik має компонент під назвою Zygote, який попередньо ініціалізує екзмепляри віртуальних машин і розгалуджує їх, якщо в цьому виникає необхідність.

Кардинально інший підхід Android до розробки мобільних додатків пропонує деякі унікальні переваги, але він також створює багато проблем для сторонніх розробників. Найбільша перевага в такому підході це те, що він забезпечує високий рівень однорідності. Більша частина додатків Android зможуть безперешкодно працювати майже на будь-якому пристрої на базі Android, без потреб виконання змін у майбутньому.

“Android” включає в себе комплект базових додатків: клієнти електронної пошти, календар, різноманітні карти, браузер, програма для керування контактами тощо.

“Android” дозволяє використовувати всю потужність “Android” , що використовуються в додатках ядра. Архітектура побудована таким чином, що будь який додаток може використовувати уже реалізовані можливості іншого додатка, при умові, що останній відкриє доступ до використання своєї функціональності. Таким чином архітектура реалізує принцип багаторазового використання компонентів і додатків. Нижче, на рисунку 1.1. наведено архітектуру системи.

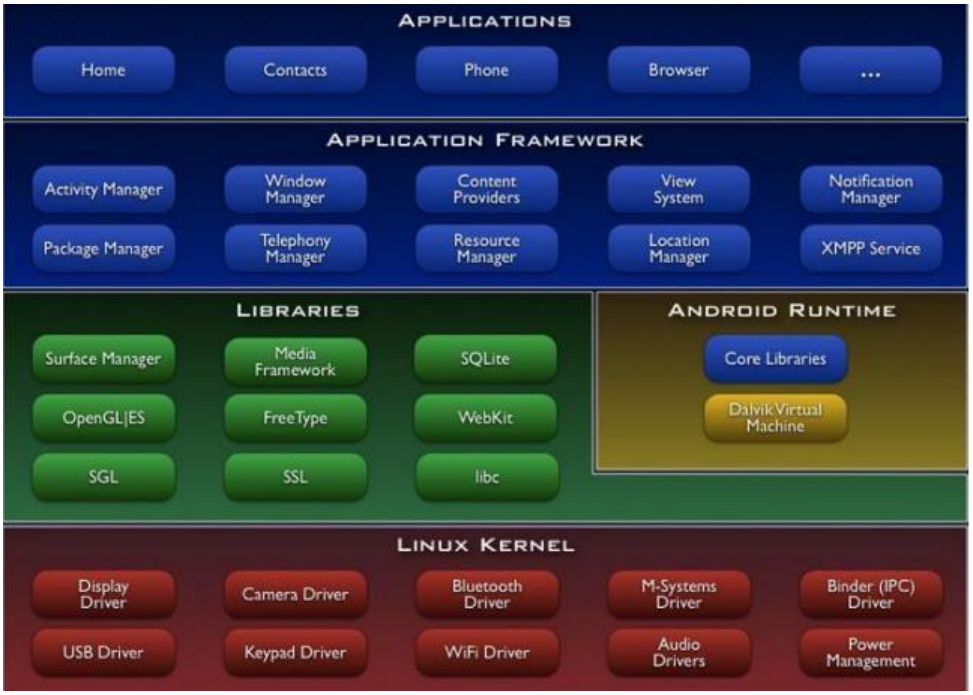

Рисунок 1.1 – Схема архітектури системи

Платформа Android надає повний та добре організований асортимент високорівневих API інтерфейсів для створення додатків та використовування основних функцій платформи. Інтерфейси API запезпечують надзвичайно високий рівень абстракції, що робить їх відносно інтуїтивно зрозумілими і простими у використанні в процесі розробки.

Сторонні додатки можуть реплікуватися або взаємодіяти практично з усіма головними компонентами операційної системи. Як приклад, Android надає методи для отримання інформації з переліку контактів користувача та для розширення системи контактів новими полями даних. Також легко зробити новий інтрефейс дзвінка або реалізувати користувацьку поведінку для системних подій, таких як вхідні повідомлення. Зокрема, API дійсно дозволяють створювати додатки, які повністю інтегруються з рештою платформи. На рисунку 1.2 зображено приклад використання API.

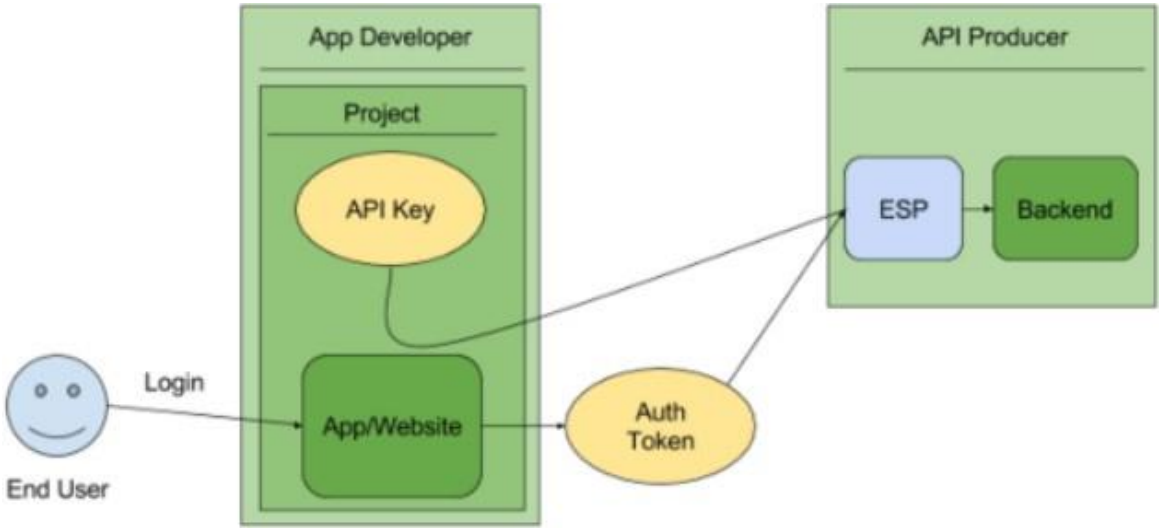

Рисунок 1.2 – Приклад використання API

Набір інструментів для віджетів Android надає велику кількість дуже корисних компонентів відразу. Окремі віджети розроблені спеціально для зручної взаємодії з пальцями, з вже вбудованими функціями, такими як кінетична прокрутка. Розробники використовують мову опису користувацького інтерфейсу на основі XML для визначення макету та атрибутів віджетів, при цьому описи XML завантажуються в програму через систему ресурсів Android.

На окремі віджети, описані в в макеті XML, можна посилатися по ідентифікатору в програмі. Також є можливість створення віджетів програмно і маніпулювати ними користувацьким інтерфейсом в процесі виконання програми. Найбільш правильним способом створення макетів є написання описів XML вручну. Пакет Android SDK надає вбудований інструмент візуального макета, проте він не підтримує всі віджети і не завжди працює здагоджено.

Додатки Android складаються з провайдерів , служб, приймачів та активностей. Всі ці компоненти мають окрему задачу в стеку додатків Android. Спосіб, яким вони можуть взаємодіями один з одним це те, що ноповнює Android його модульністю.

Провайдери контенту слугують рівнем абстракції для взаємодії з різноманітними джерелами даних і для обміну постійними даними між додатками. Вони надають потрібну інформацію через стандартизований інтерфейс запитів. Запити описуються за допомогою синтаксису URI, але розробники додатків зазвичай використовують багаторівневі класи оболонки , які генерують рядки запитів і керують ними. Коли програма надсилає запит провайдеру контента, він повертає відповідь у вигляді об’єкту-курсору, який надає програмний базисний доступ до базового контенту. Також можна підключити механізми моніторингу до провайдерів контенту, аби додаток міг отримувати повідомлення про зміну даних.

Система провайдера контенту пропонує розробникам Android додатків декілька вагомих переваг. Найбільш явною перевагою є те, що система забезпечує високий рівень взаємодії, дозволяючи додаткам обмінюватися даними досить уніфікованим методом. Декілька ключових джерел даних платформи, включаючи список контактів та системи збереження мультимедіа, за замовчуванням доступні через інтерфейси провайдерів контенту, тому їх легко використовувати з сторонніх додатків для реалізації певних функцій.

Важливою особливістю, яка відрізняє Android від інших платформ є те, що Android офіційно підтримує фонові процеси в сторонніх додатках. Вони реалізовані за дпомогою сервісного компоненту Android. Сервіси - це операції без заголовку, які виконуються в фоновому режимі протягом тривалого часу.

Система повідомлень Android приблизно схожа з аналогічною системою сигналів в Linux. Розробники вбудовують широкомовні приймачі, які можуть визначати, коли з’являються визначені системні повідомлення чи події, і винонувати певні дії у відповідь. Багато базових системних подій можна відслідковувати за допомогою системи мовлення, тому її можна використовувати для відстежування таких речей, як зміна стану акумуляторної батареї, отримання вхідних SMS повідомлень, натискання апаратних кнопок, зміна з’єднання, встановлення сховища та затемнення екрану.

Система Android Intents є ключем до розуміння того, як всі ці частини використовуються разом. Об’єкти Intents, які містять ідентифікатор дії та URI даних, використовуються для виклику дій, служб та отримувачів, Ідентифікатор дії вказує бажану поведінку, а необов’язковий URI даних надає місце розташування цих даних, над якими має виконуватися дія. На рисунку **1.3** представлено загальну схему компонентів Android додатку.

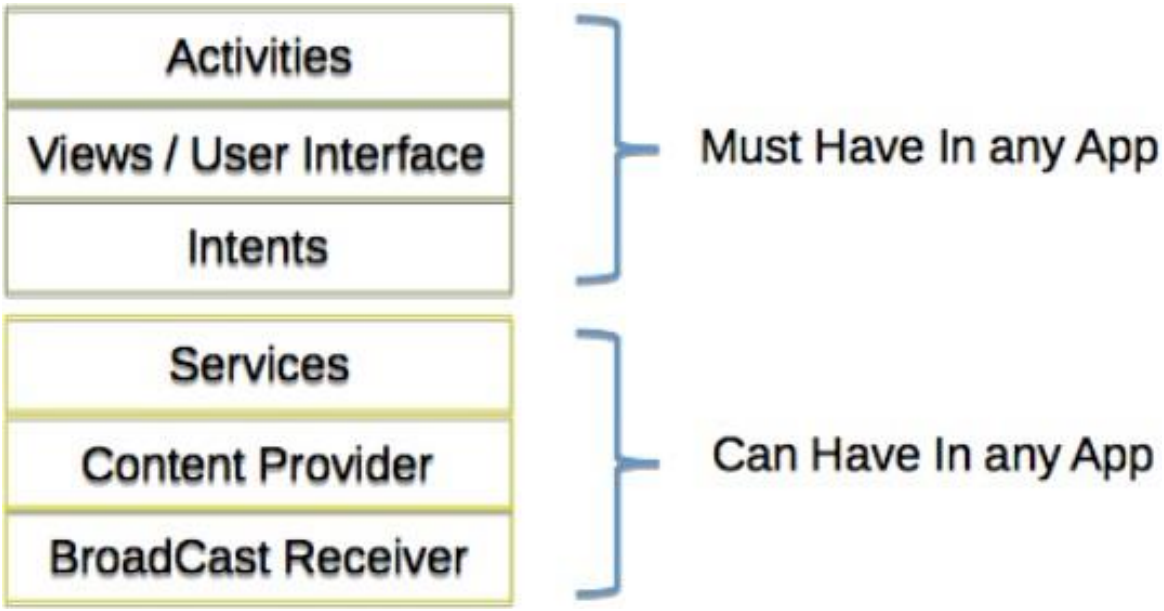

Рисунок 1.3 – Загальна схема компонентів Android додатку

Однією з найбільших переваг системи Android є її відкритість. Операційна система Android побудована на основі відкритого висхідного коду і розповсюджується на вільній основі. Це дозволяє розробникам отримати доступ до початкового коду і зрозуміти яким чином реалізовані властивості і функції додатків. Кожен користувач може прийняти участь в удосконаленні операційної системи.

2 ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ

2.1 Огляд використовуваних інструментів

2.1.1 Firebase

На конференції Google I / O 2015 була представлена хмарна база даних на основі NoSQL з палаючим назвою Firebase. Firebase - це платформа для мобільних і веб-додатків.

У Firebase є три основних сервіси які були використані при розробці:

- база даних реального часу;
- аутентифікація;
- статичний хостинг.

Firebase Realtime database - це база даних типу NoSQL, яка використовує сокети, що дозволяє клієнту отримувати інформацію в реальному часі - без необхідності відправляти запити GET на сервер.

Дана база даних забезпечить взаємодію роботи Android додатку з базою даних.

Переваги Firebase Realtime Database:

- синхронізація в реальному часі для даних JSON. База даних Firebase Realtime є хмарну базу даних NoSQL, яка дозволяє зберігати і синхронізувати дані між вашими користувачами в режимі реального часу;
- спільно працювати з пристроями. Синхронізація в реальному часі дозволяє користувачам отримувати доступ до своїх даних з будь-якого пристрою: веб-або мобільним, і це допомагає вашим користувачам співпрацювати один з одним;

- створення безсерверних додатків. База даних Realtime поставляється з мобільними і веб-SDK, тому можна створювати додатки без необхідності серверів;

- оптимізовано для автономного використання. Коли користувачі переходять в автономний режим, SDK бази даних Realtime використовує локальний кеш на пристрої для обслуговування і збереження змін. Коли пристрій підключається до мережі, локальні дані автоматично синхронізуються;

- сильна призначена для користувача безпеку. База даних Realtime інтегрується з Firebase Authentication для забезпечення простий і інтуїтивно аутентифікації для розробників. [1]

Firebase Auth - дозволяє виконувати вхід в програму за допомогою системи, з якої користувачі добре знайомі, тобто Facebook або Google. Потім додаток може зберігати дані і персональні налаштування користувача в захищеному хмарному сховище і забезпечувати доступ до них на всіх його пристроях.

Firebase забезпечує служби серверної частини, прості пакети розробника і готові бібліотеки інтерфейсу для аутентифікації користувачів різних додатки на будь-яких платформах. Аутентифікацію можна виконувати за допомогою паролів і інтегрованих систем ідентифікації - Google, Facebook, Twitter та ін. Це значно спрощує процедуру входу в додаток і забезпечує надійний захист.

Firebase Auth допоможе мені з економити час при розробці методів для аутентифікації: замість цього можна просто зберігати інформацію про користувача після аутентифікації користувачем за допомогою Firebase. Також збережеться чимало часу, так як можна уникнути розробки методів на стороні сервера для різних видів перевірки токенів в разі, також є можливість додати соціальні логіни, такі як Facebook і Google. Все, що буде ефективно працювати з Firebase.

Переваги Firebase Auth:

- тісна інтеграція з іншими функціями Firebase;

- використання галузевих стандартів, таких як OAuth 2.0 і OpenID Connect спрощує інтеграцію з серверним кодом;
- два варіанти для розробки: FirebaseUI - повністю інтегрувальне універсальне рішення для виконання аутентифікації - або пакет Firebase Authentication SDK, який дозволяє вручну інтегрувати один або кілька методів входу в додаток;
- безпека аутентифікації і зручність забезпечуються за рахунок можливості виконувати вхід через аккаунт Google та інші інтегровані системи ідентифікації - Facebook, Twitter і т. Д. Також для входу можна використовувати пароль;
- безпечний доступ до сервісів Google. Можливість зберігати файли на Google Диск, створювати заходи в Google Календарі і ділитися новинами зі своїми контактами прямо з програми;
- оплата покупок в додатку за допомогою Google Гаманця.[2]

Дані сервіси Firebase були вибрані через їх пристосованість до Android клієнта. Це забезпечить легкість роботи з ними. Дані сервіси мають хорошу захищеність. Завдяки захищеності цих сервісів необхідність в реалізації методів захисту в даному додатку стає непотрібною. Даний сервіс бере на себе відповідальність за збереження даних. Завдяки широкому наборі інструментів в даному сервісі, необхідність реалізації додаткових компонентів стає неактуальною.

2.1.2 Админ панель flamelink

Flamelink - це проста у використанні, орієнтована на контент, безголова CMS, яка інтегрується з Google Firebase для створення мобільних та веб-додатків, цифрових кампаній та веб-сайтів. [3]

2.1.3 Android Studio

Для подальшого написання клієнтських додатків системи гнучкого управління розкладом було обрано програмне середовище Android Studio.

Android Studio - це редактор вихідних кодів, розроблений компанією Google для Windows, Linux та macOS. Вона включає в себе підтримку налагодження, вбудоване керування Git, підсвічування синтаксису, інтелектуальне завершення коду, фрагменти та рефакторинг коду. Це також налаштовується, так що користувачі можуть змінити редактор, поєднання клавіш і перевагу. Це безкоштовне та відкрите джерело, хоча офіційне завантаження знаходиться під фірмовою ліцензією [4].

Середовище розробки адаптоване для виконання завдань, що вирішуються в процесі розробки додатків для платформи Android. У тому числі у середовище включені засоби для тестування програм на сумісність з різними версіями пристроїв та інструменти для проектування застосунків, що працюють на пристроях з екранами різної роздільної здатності (планшети, смартфони, ноутбуки, годинники, окуляри тощо). Крім можливостей, присутніх в IntelliJ IDEA, в Android Studio реалізовано кілька додаткових функцій, таких як нова уніфікована підсистема складання, тестування і розгортання застосунків, заснована на складальному інструментарії Gradle і підтримуюча використання засобів безперервної інтеграції [5][6][11].

Графічний інтерфейс програмного середовища Android Studio можемо побачити на рисунку 2.1.

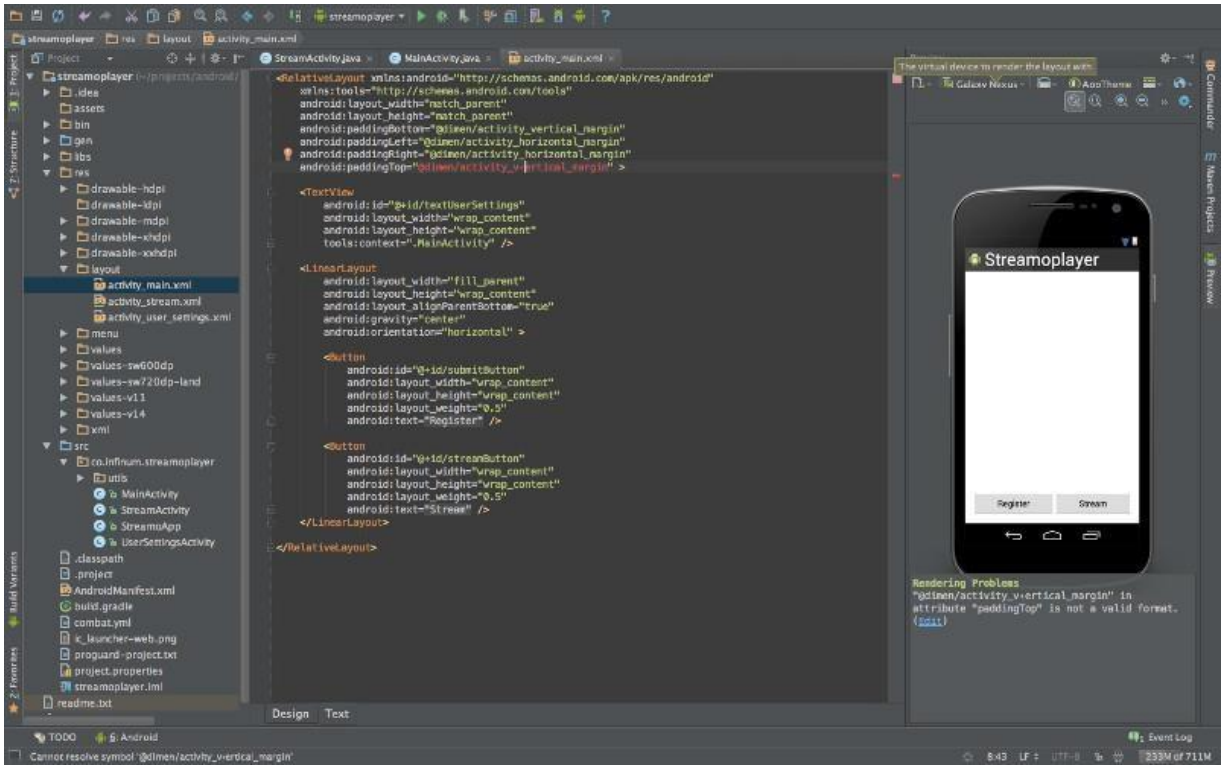

Рисунок 2.1 – Інтерфейс Android Studio

Описане вище програмне середовище використовувалось для розробки двох клієнстських додатків для адміністратора заходу і для відвідувача заходу.

Додаткові переваги обраного програмного середовища наступні:

- живі макети (layout);
- подання (rendering) програми в реальному часі;
- консоль: підказки щодо оптимізації, допомога з перекладом, стеження за напрямком, агітації та акції;
- бета релізів та покрокові релізи;
- Android-орієнтований рефакторинг коду та швидкі поправки коду;
- утиліти Lint для продуктивності, юзабіліті, сумісності версій та інших проблем;
- використання ProGuard та підписів до програм;
- шаблони для створення розширених Android компонентів;
- багатий редактор макетів, що дозволяє користувачам перетягнути і покласти компоненти користувацького інтерфейсу, як варіант, переглянути одночасно макети на різних конфігураціях екранів [7].

2.1.4 Мова програмування Kotlin

Kotlin - статично типізована мова програмування, що працює поверх JVM і розробляється компанією JetBrains. Також компілюється в JavaScript. Мову названо на честь острова Котлін у Фінській затоці, на якому розміщена частина Кронштадту. Автори ставили перед собою ціль створити лаконічнішу та типо-безпечнішу мову, ніж Java, і простішу, ніж Scala. Наслідками спрощення, порівняно з Scala стали також швидша компіляція та краща підтримка IDE.[8]

Мова розробляється з 2010 року, публічно представлена в липні 2011. Сирцевий код було відкрито в лютому 2012. В лютому було випущено milestone 1, який містив плагін для IDEA. У червні - milestone 2 з підтримкою Android. У грудні 2012 року вийшов milestone 4 та забезпечив підтримку Java 7. Станом на листопад 2015 року основні можливості мови стабілізовані, готується реліз версії 1.0. В грудні 2015 року з'явився релізкандидат версії 1.0, а 15 лютого 2016 року відбувся реліз версії 1.0. З 17 травня 2017 року входить в список офіційно підтримуваних мов для розробки додатків для платформи Android. Позиціонується розробниками як об'єктно-орієнтована мова промислового рівня, а також як мова, яка зможе замінити Java. При цьому мова повністю сумісна з Java, що дозволяє розробникам поступово перейти з Java на Kotlin. Зокрема, в Android мову вбудовується за допомогою Gradle, що дозволяє для існуючого Androidдодатку впроваджувати нові функції на Kotlin без переписування самої програми. Синтаксис мови схожий на Pascal, TypeScript, Haxe, PL / SQL, F#, Go і Scala, C ++, Java, C # і D. При оголошенні змінних і параметрів типи даних вказуються після назви (роздільник двокрапка). Крапка з комою, як роздільник операторів, є необов'язковою, так само як в Scala і Groovy, в більшості випадків перенесення рядка досить, щоб компілятор зрозумів, що вираз закінчився. Крім об'єктно-орієнтованого підходу, Kotlin також підтримує процедурний стиль з використанням функцій. Як і в мовах C/C++/D, точкою входу в програмі є функція "main", яка приймає

масив параметрів командного рядка. Програми на Kotlin також підтримують Perl і Unix / Linux shell стиль інтерполяції рядків. Kotlin також підтримує виведення типів [8].

Розробники Android пишуть, що вони спостерігали «сходження» Kotlin всі останні роки. Kotlin описують як вражаючу і лаконічну мову, яка відкриває більше можливостей і з якою приємно працювати. Вона має підвищену продуктивність: програмний код на ньому виходить в середньому на 40% коротше, ніж на інших мовах, а також Kotlin дозволяє не допускати деякі помилки в коді. Одним з визначальних чинників популярності Kotlin стало те, що він сумісний з Java, яка вже використовується при розробці додатків під Android. Тепер, коли програмісти починають створювати новий додаток в офіційному середовищі розробки Android Studio, вони відразу можуть включити плагін «підтримка Kotlin». Також можна конвертувати вже створені рядки коду на інших мовах в мову Kotlin, вставляти блоки на інших мовах в рядки коду на Kotlin. У майбутньому для мови буде розроблятися більше бібліотек і інструментів, більше навчальних матеріалів, простіше буде знайти рішення для можливих проблем. Відсутність гарантій підтримки мови з боку Google відлякувала багатьох розробників від переходу на Kotlin. Навіть якщо мова дуже подобається, програміст завжди думає про ризик, що в якийсь момент ця мова просто перестане працювати. Тепер є гарантія того, що працювати Kotlin не перестане, і ми очікуємо, що кількість користувачів мови різко зросте. Припускають, що багато компаній з часом перейдуть на Kotlin повністю, хоча технічно їх до цього нічого не змушує, це просто питання переваг. Він додав, що Kotlin дуже активно розвивається. Команда розробників зараз працює над build-системою, швидкістю компіляції, покращенням продуктивності IDE, додаванням в інструментарій нових можливостей, в тому числі пов'язаних з інтеграцією в Android Studio. Також йде робота над мультиплатформенними проектами (можливість компілювати один і той же код під кілька платформ), цілий ряд мовних поліпшень знаходиться в стадії дизайну. Варто зазначити концепція мови Kotlin, за яким

він завжди був і залишиться безкоштовним для розробників, тобто open source project. Це означає, що мова не прив'язана до якої-небудь окремої компанії, а вихідний код розповсюджується під вільною ліцензією. Завантажити продукт можна. Щоб підтримувати розвиток Kotlin, компаніями Google і JetBrains буде створено некомерційне партнерство. Також в рамках «місії» Android дуже важливо, що автори Kotlin створюють навколо свого продукту співтовариство людей, які професійно займаються розробкою на цій мові і люблять ділитися досвідом. Наприклад, проводяться конференції, де розробники можуть отримувати щоденні новини та поради про програмний продукт, зустрічатися на місцевому рівні.

2.2 Архітектура додатку MVVM

Model-View-ViewModel (MVVM) - шаблон проектування архітектури програми. Представлений у 2005 році Джоном Госсманом (John Gossman) як модифікація шаблону Presentation Model. Орієнтовано сучасні платформи розробки, такі як Windows Presentation Foundation, Silverlight від компанії Microsoft, ZK framework.[9]

Model - шар даних, відповідаючий за управління бізнес-логікою, взаємодію з мережею та базами даних.

Viev - UI (user interface), просто користувальницький інтерфейс.

VievModel - надає потоки даних, що стосуються Viev.

Використовується для поділу моделі та її уявлення, що необхідно їх зміни окремо друг від друга. Наприклад, розробник задає логіку роботи з даними, а дизайнер працює з інтерфейсом користувача.

MVVM зручно використовувати замість класичного MVC і йому подібних у тих випадках, коли в платформі, на якій ведеться розробка, є зв'язування даних (рис. 2.2). У шаблонах проектування MVC/MVP зміни в інтерфейсі користувача не впливають безпосередньо на Модель, а попередньо

йдуть через Контролер (Controller) або Presenter. У таких технологіях, як WPF та Silverlight, є концепція «зв'язування даних», що дозволяє зв'язувати дані з візуальними елементами в обидві сторони. Отже, при використанні цього прийому застосування моделі MVC стає вкрай незручним через те, що прив'язка даних до подання безпосередньо не укладається в концепцію MVC/MVP.[9]

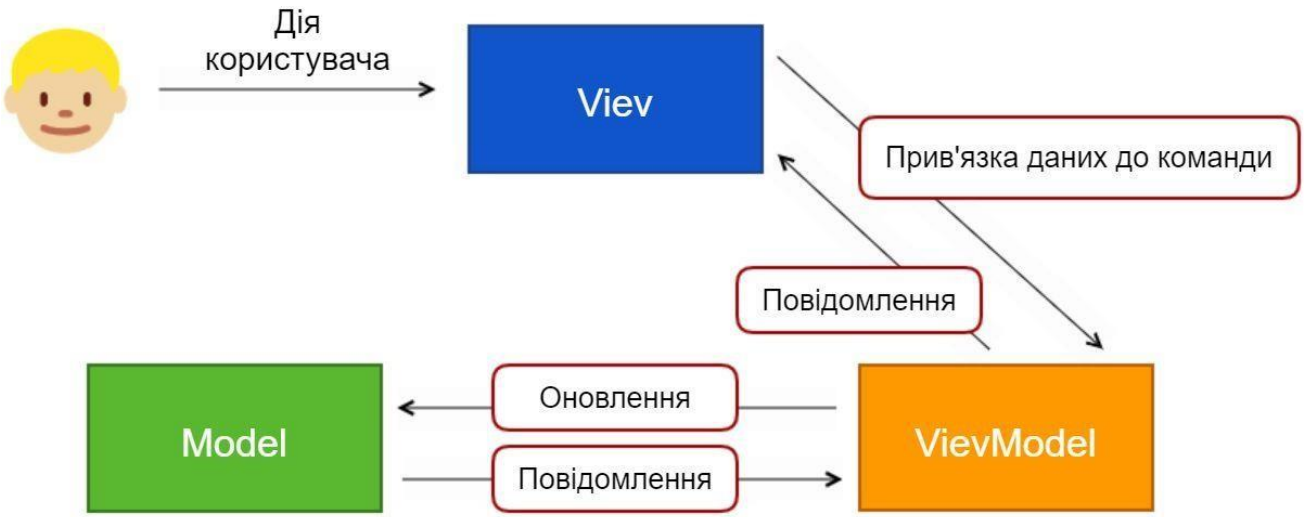

Рисунок 2.2 – Взаємодія елементів у патерні MVVM

2.3 Проектування за допомогою діаграми прецедентів

Модель бізнес-прецедентів описує бізнес-процеси з точки зору зовнішнього користувача, тобто відображає погляд на діяльність організації ззовні (рис. 2.3).

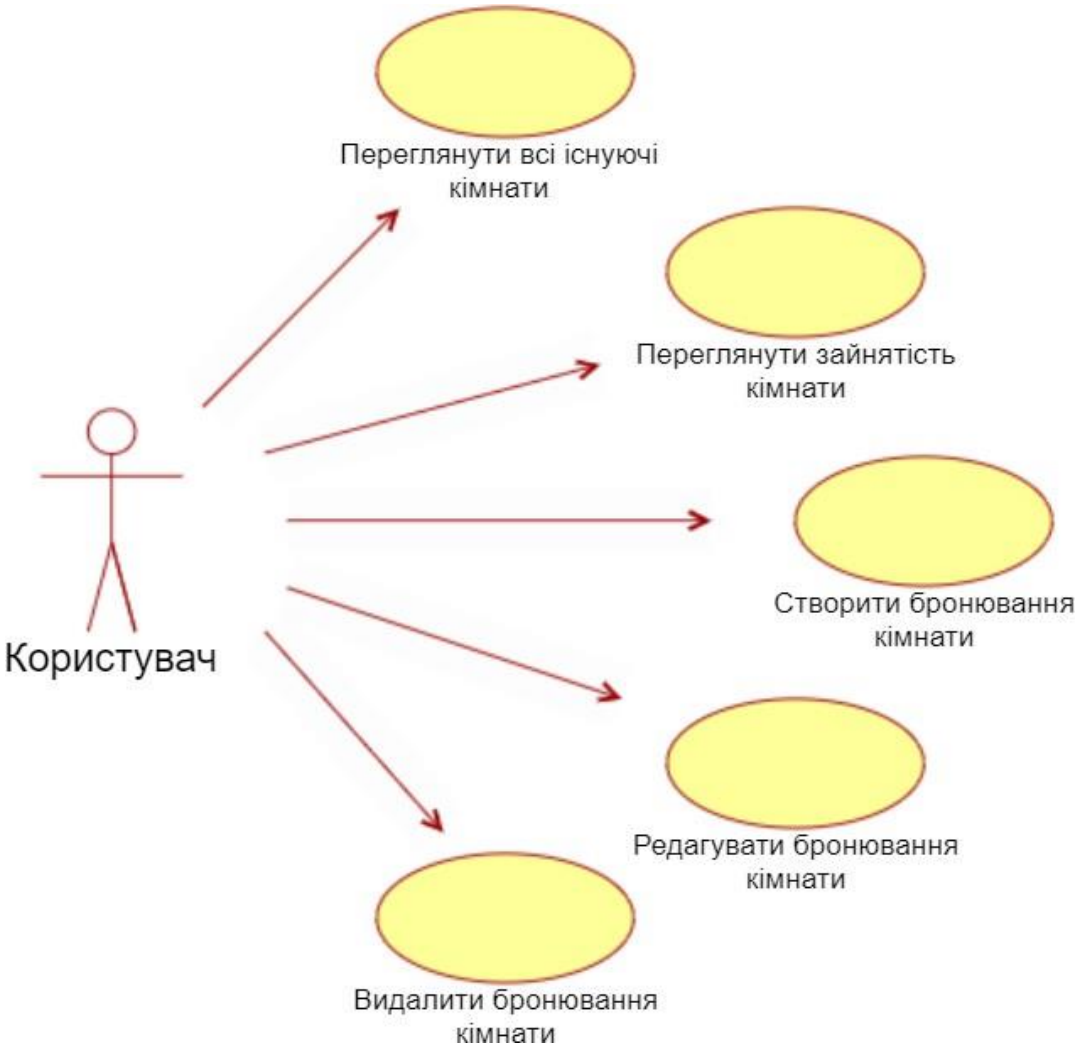

Рисунок 2.3 – Діаграма прецендентів

2.4 Проектування за допомогою діаграми діяльності

Діаграми видів діяльності (діаграми діяльностей, activity diagrams) - модель бізнес-процесу або поведінки системи в рамках прецеденту (рис. 2.4).

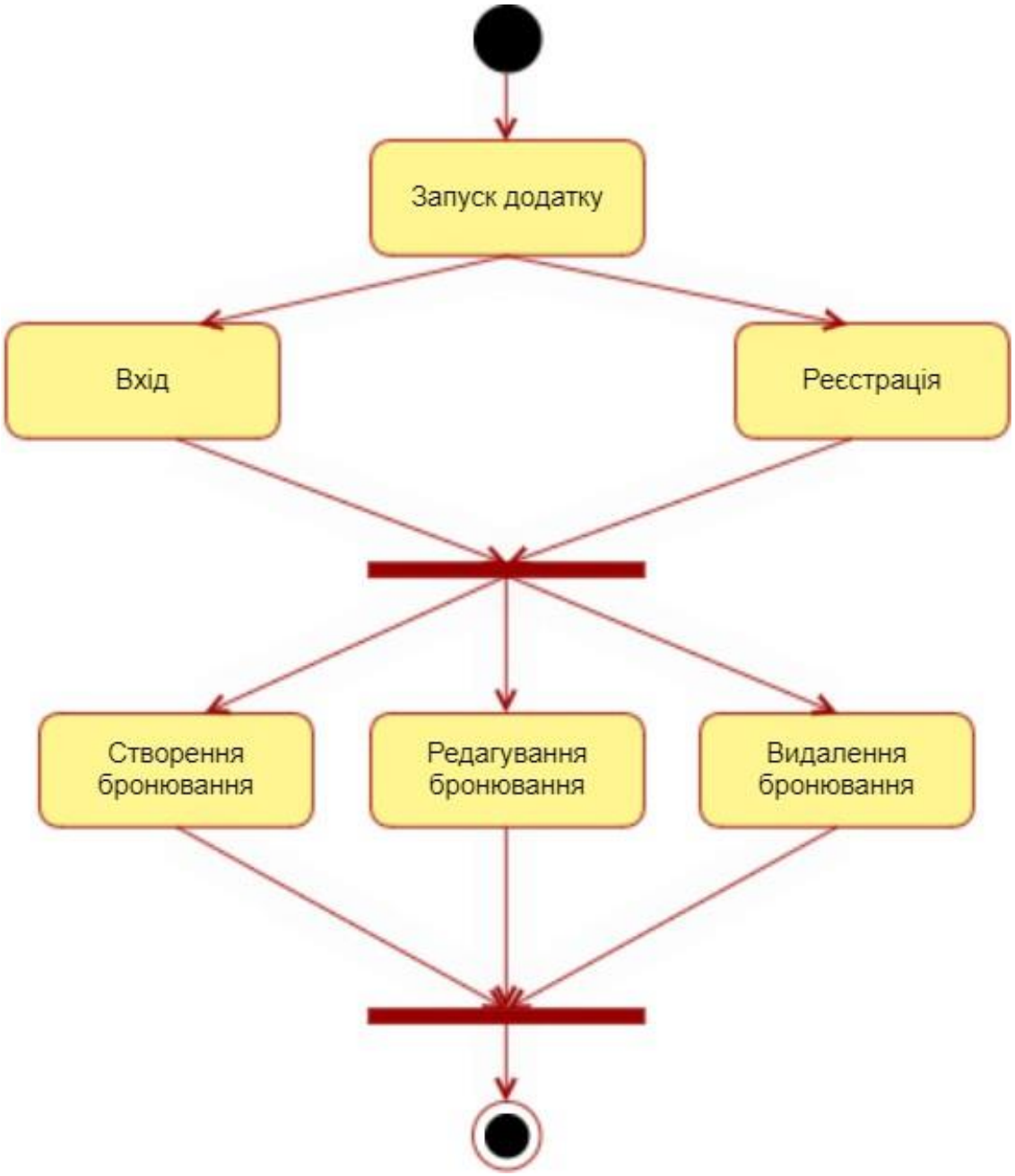

Рисунок 2.4 – Діаграмма діяльності

3 РЕАЛІЗАЦІЯ МОБІЛЬНОГО ДОДАТКУ

3.1 Опис функціональності системи

На початку створення кваліфікаційної роботи було визначено функціонал мобільного додатка, необхідний для надання послуг бронювання.

Необхідні функції програми:

- реєстрація та авторизація у додатку;
- перегляд доступності кімнат;
- створення бронювань;
- редагування бронювань;
- видалення бронювань.

Про реалізацію цих функцій і йтиметься у наступних пунктах.

3.2 Розробка мобільного додатку

Будь-який Android - додаток складається з кількох активностей, які іноді пов'язані між собою. Активіті - це один екран інтерфейсу програми, певна комбінація XMLфайлу і Java-файлу. По суті Активіті - це контейнер, який містить як дизайн, так і кодування. XML-файл представляє нам дизайн нашого екрану. Файл Java містить код сторінки, який відповідає за те, що відбувається в активіті.

Зазвичай одна з активностей у додатку позначається як «основна», пропонована користувачеві під час першого запуску програми, у моїй кваліфікаційній роботі такою активністю є авторизація користувачів.

3.2.1 Розробка вікна авторизації

Активіті авторизації користувача є стартовою сторінкою програми (рис. 3.1). Ця сторінка складається з двох полів, необхідних для введення поштової адреси та пароля відповідно. Кнопка “Login” запускає функцію, яка виконує багато завдань. На початку функції зчитуються введені дані, які порівнюються із вимогами валідації. Наприклад, жодне з полів не може бути порожнім або недостатньо довгим, а правильність введеного email перевіряється за допомогою вбудованої бібліотеки в Android studio. Якщо введені дані не пройшли перевірку, користувач отримає сповіщення про помилки. Якщо дані пройшли валідацію, формується запит POST за допомогою бібліотеки Volley, параметрами якого є email і пароль користувача. У разі позитивної відповіді від системи користувач переноситься на сторінку вибору кімнати, в іншому випадку виходить повідомлення про те, що введені дані не вірні. Також сторінка містить кнопку: “Реєстрація”. Натиснувши на “Реєстрація”, користувач переміщується на сторінку реєстрації в системі (рис. 3.2).

Активіті реєстрація складається з 5 полів для введення даних, прізвище, ім'я, поштову адресу, пароль та повторний пароль. Вимоги валідації такі ж, як і на сторінці авторизації, тільки додається правило, що пароль та повторний пароль повинні збігатися. Також на сторінці є кнопка для повернення до функції входу для вже зареєстрованих користувачів.

Рисунок 3.1 – Вікно авторизації

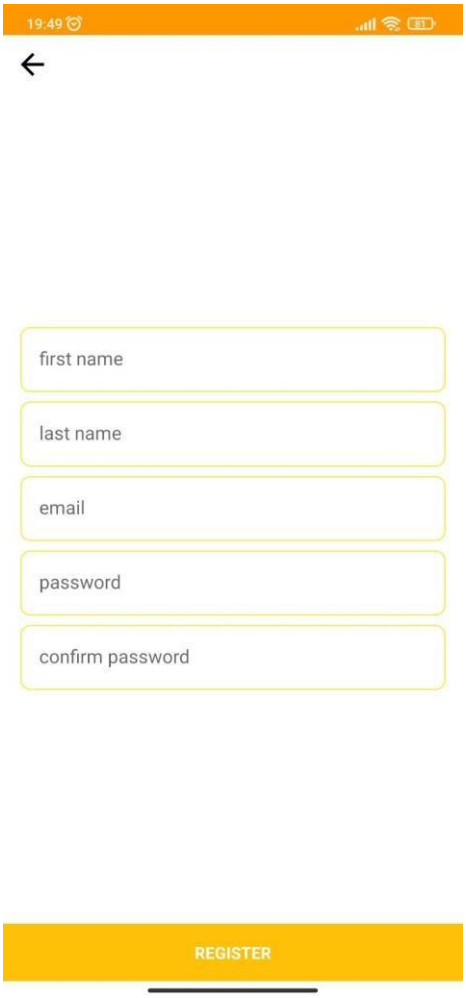

Рисунок 3.2 – Вікно реєстрації

3.2.2 Розробка вікна вибору кімнати

На цій сторінці представлені всі наявні в додатку кімнати (рис. 3.3), а також кнопка виходу з облікового запису для можливості перелогування. Якщо вибрано потрібну кімнату, програма перенаправляє користувача на календар, у якому відображаються створені бронювання.

Рисунок 3.3 – Перелік кімнат, що доступні до бронювання

3.2.3 Розробка вікна бронювання

Ця сторінка з'являється після вибору кімнати на попередньому етапі. Перше, що ми бачимо на даній сторінці при відкритті, це список бронювання на поточний день (рис. 3.4). Дані про бронювання тягнуться із Firebase. Також, до перегляду доступні дані про бронювання за попередній період та за майбутній (рис. 3.4).

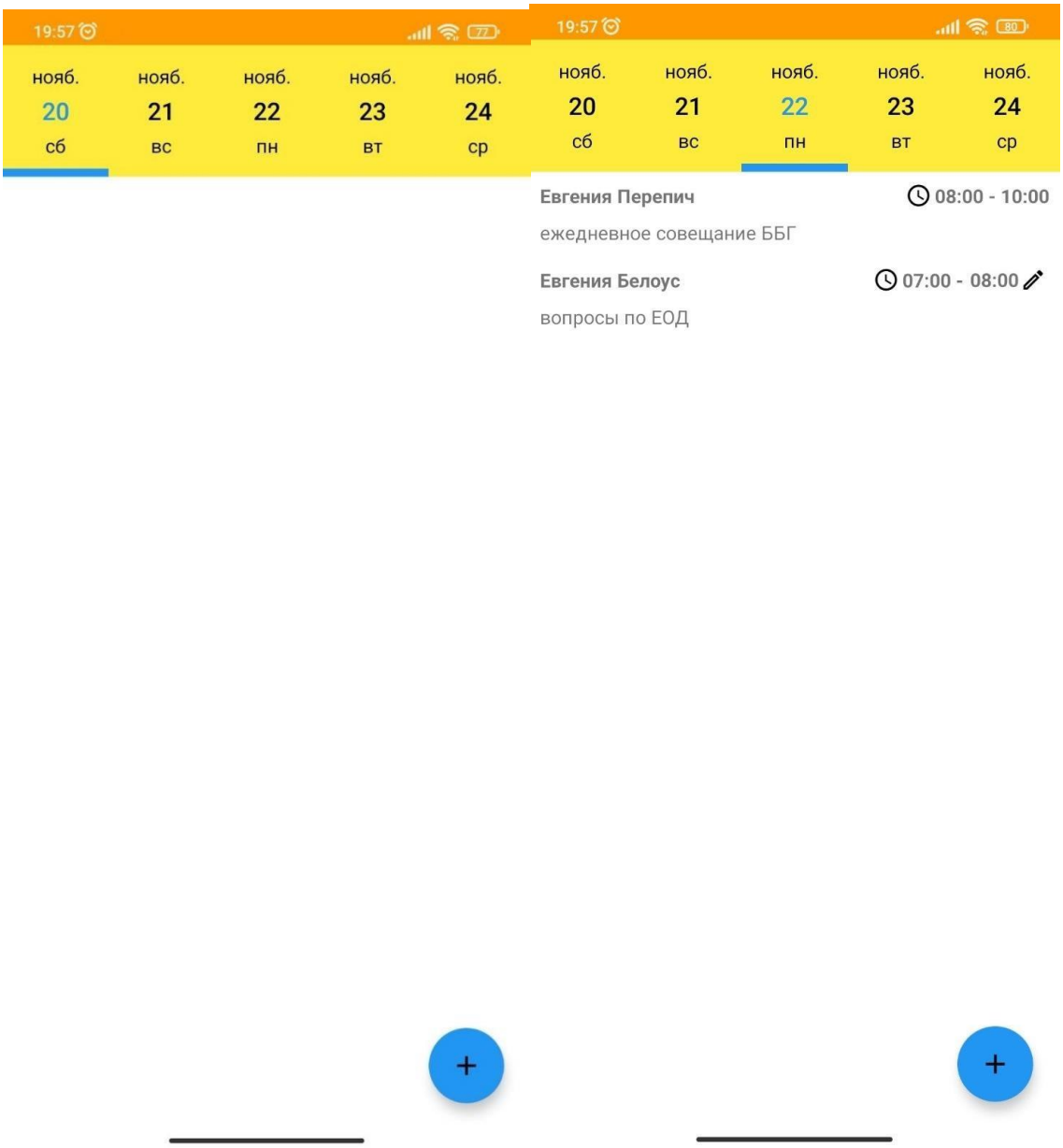

Рисунок 3.4 – Пронювання на поточну дату та майбутню відповідно

3.2.4 Розробка вікна редагування та видалення бронювання

Для редагування чи видалення бронювання необхідно натиснути на відповідному бронюванні. Далі з’явиться вікно, для внесення коректив, при натисканні кнопки «SUBMIT» або видалення бронювання, при натисканні кнопки «DELETE» (рис. 3.5).

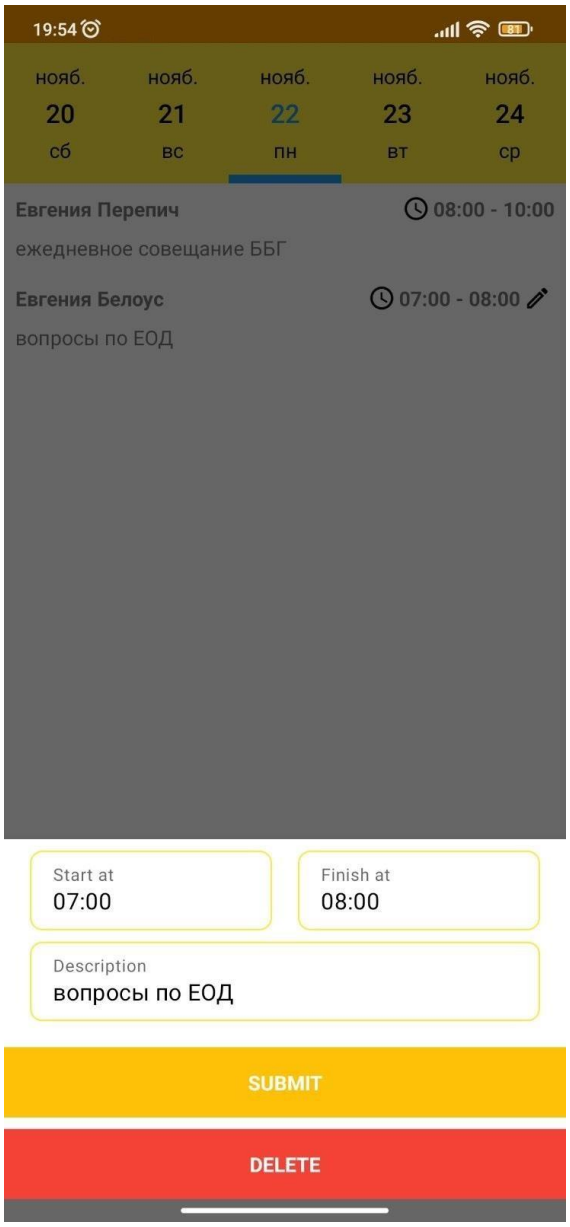

Рисунок 3.5 – Вікно внесення коректив або видалення бронювання

Виконати ці дії можливо лише з бронюванням, яке було створене тим користувачем, який вносить зміни, тобто лише своє бронювання можно редагувати чи видаляти. Перевірка відбувається через user ID під яким було створене бронювання, та тим під яким користувач увійшов наразі. Також, видалити чи відредагувати бронювання може адміністратор, через адмін панель Flamelink (рис. 3.6).

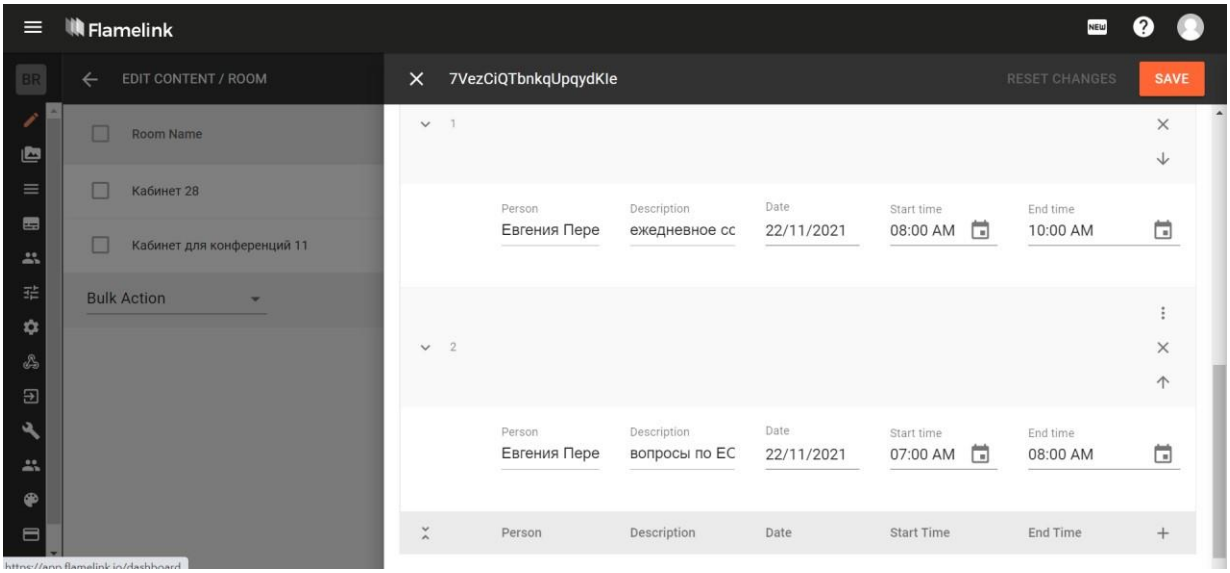

Рисунок 3.6 – Вікно редагування бронювань через адмін панель Flamelink

3.2.5 Створення нової кімнаті

Цей функціонал доступний лише для адміністратора програми. Завдяки простий у використанні, орієнтованій на контент, адмін панелі Flamelink, яка інтегрується з Firebase, виконувати роль адміністратора може навіть користувач, який не володіє навичками програмування. Для створення кімнати потрібно лише увійти в проект і натиснути кнопку «NEW ENTRY» (рис. 3.7). У вікні потрібно ввести тільки назву кімнати і додати її фото (рис. 3.8).

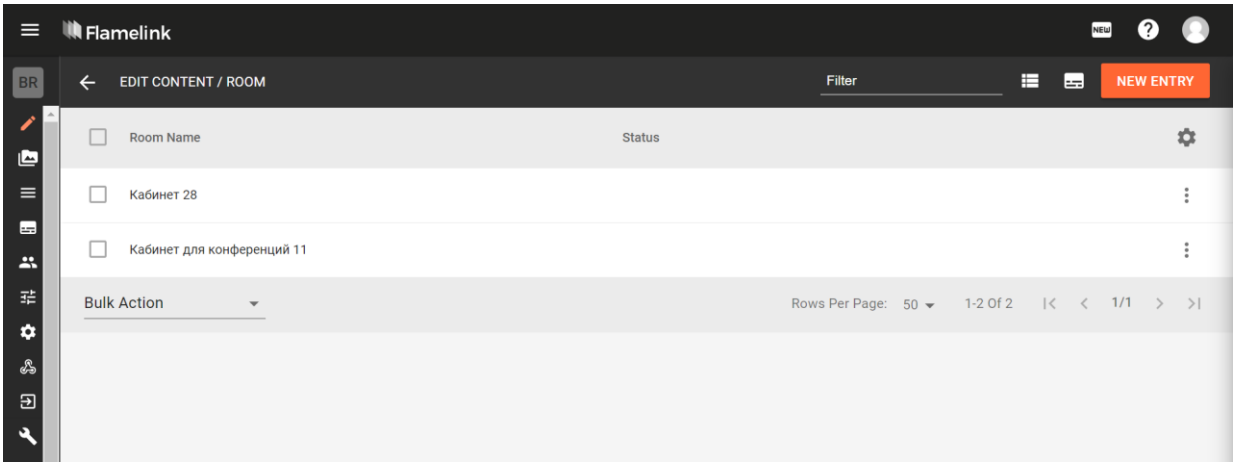

Рисунок 3.7 – Вікно додавання кімнати

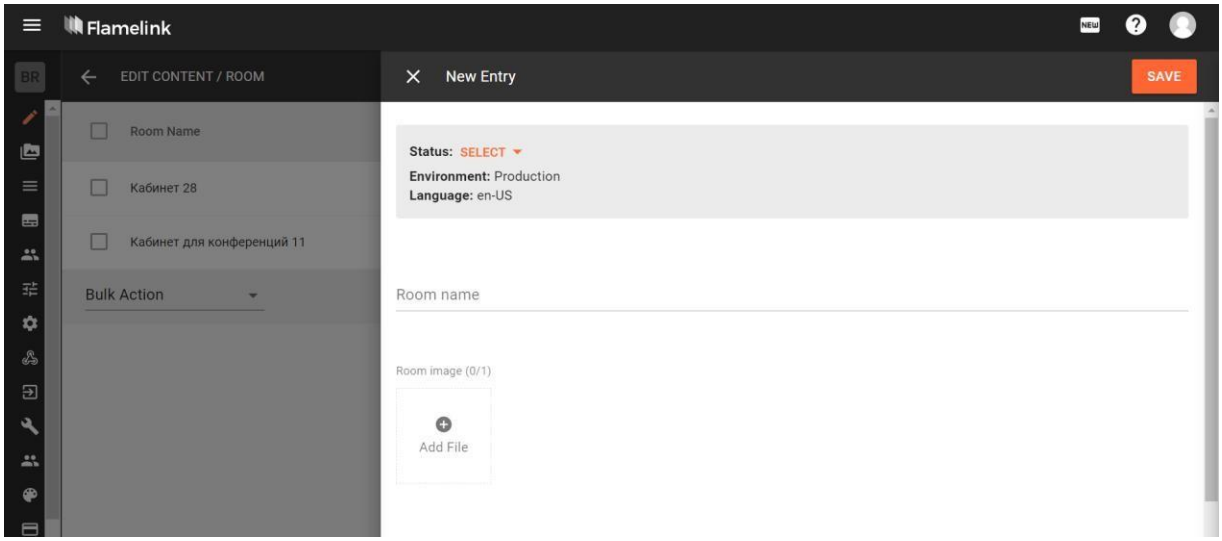

Рисунок 3.8 – Вікно вводу даних нової кімнати

4 ТЕСТУВАННЯ МОБІЛЬНОГО ДОДАТКУ

Робота будь-якого додатку починається з авторизації, тому, вважаю, розпочати тестування необхідно саме з цього функціоналу. Пропоную розглянути такі випадки для вже створених користувачів:

- вхід з коректними даними;
- введення не коректного логіну (email);
- введення не коректного паролю.

Для нових користувачів:

- реєстрація з коректними даними;
- введення не коректного логіну (email);
- введення паролю, що не підходить по параметрам;
- реєстрація без ім’я чи прізвища.

Отже, беремо випадок входу в додаток зареєстрованого користувача з коректними даними (рис. 4.1). В результаті отримаємо доступ до вікна вибору кімнат.

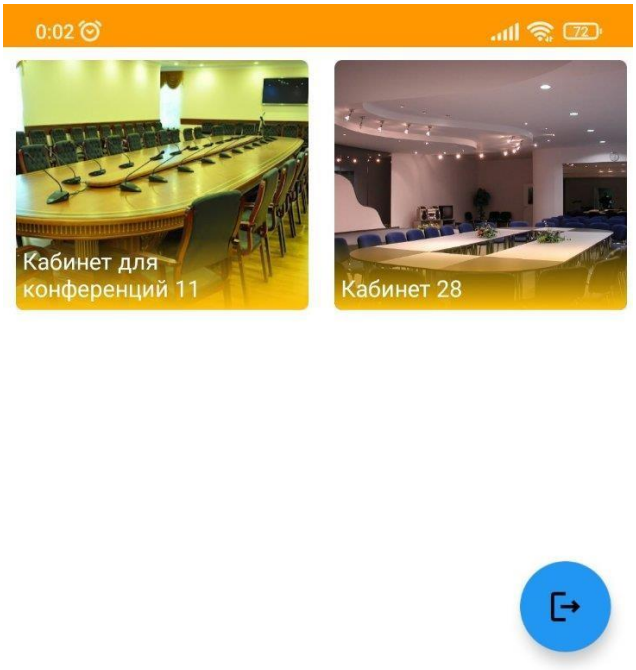

Рисунок 4.1 – Вхід з коректними даними

Випадок введення не коректного логіну (email) представлено на рисунку 4.2. Система видає помилку, що користувача з даними, що відповідають ідентифікатору, не знайдено.

Рисунок 4.2 – Введення не коректного логіну (email)

При введенні не коректного паролю (рис. 4.3) бачимо помилку «Пароль не дійсний, або користувач не має паролю», що свідчить по коректне відпрацювання скрипту перевірки пароля.

Рисунок 4.3 – Введення не коректного паролю

Тестування автентифікації для зареєстрованих користувачів завершили, тепер перейдемо до реєстрації нових користувачів.

При заповненні усіх полей коректно отримаємо той самий результат, що представлено на рисунку 4.1.

Наступним кроком, розглянемо випадок, коли користувач виконує введення не коректного логіну (email) (рис. 4.4). Отримуємо повідомлення про неправильний email, через те, що такого домену не існує.

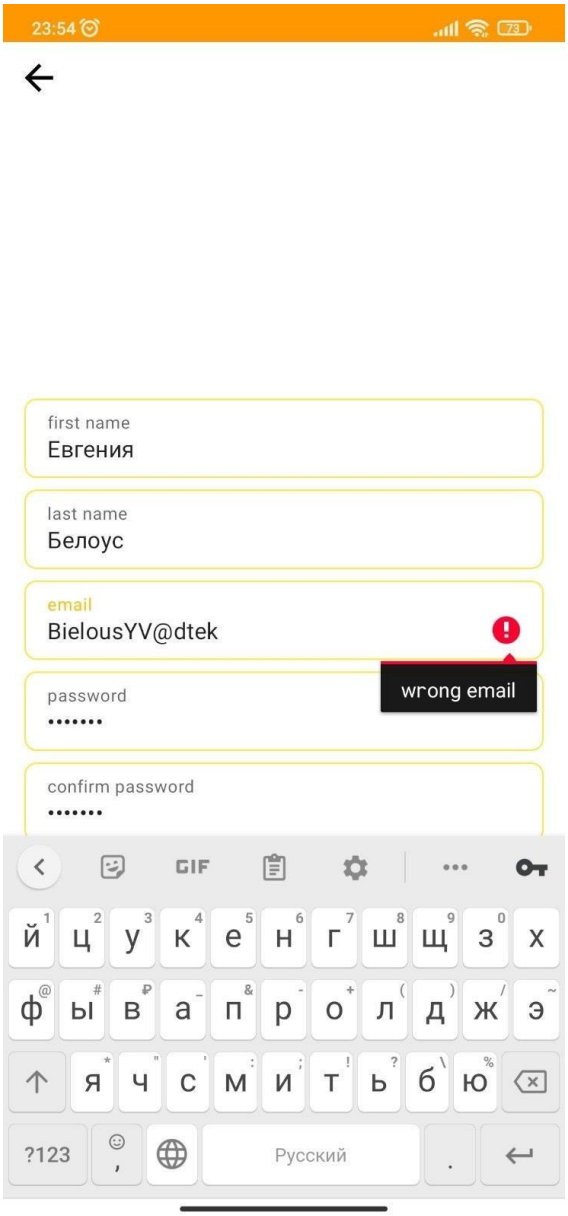

Рисунок 4.4 – Введення не коректного логіну (email)

Далі протестуємо введення паролю, що не підходить по параметрам. Надсилається запит на перевірку вимог до паролю, як бачимо по рисунку 4.5, пароль не відповідає вимогам.

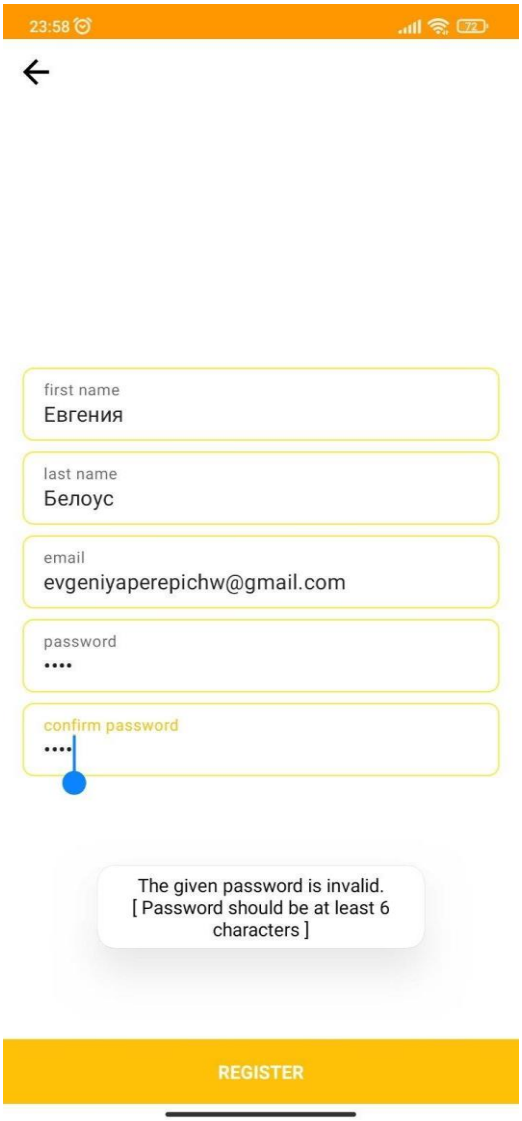

Рисунок 4.5 – Введення паролю, що не підходить по параметрам

Останній тест сторінки реєстрації це реєстрація без ім’я чи прізвища.

Перевірка йде поступово, якщо знайдено помилку в першому полу, до другого система не дійде, це видно на рисунку 4.6, як і саму помилку про незаповнене обов’язкове поле.

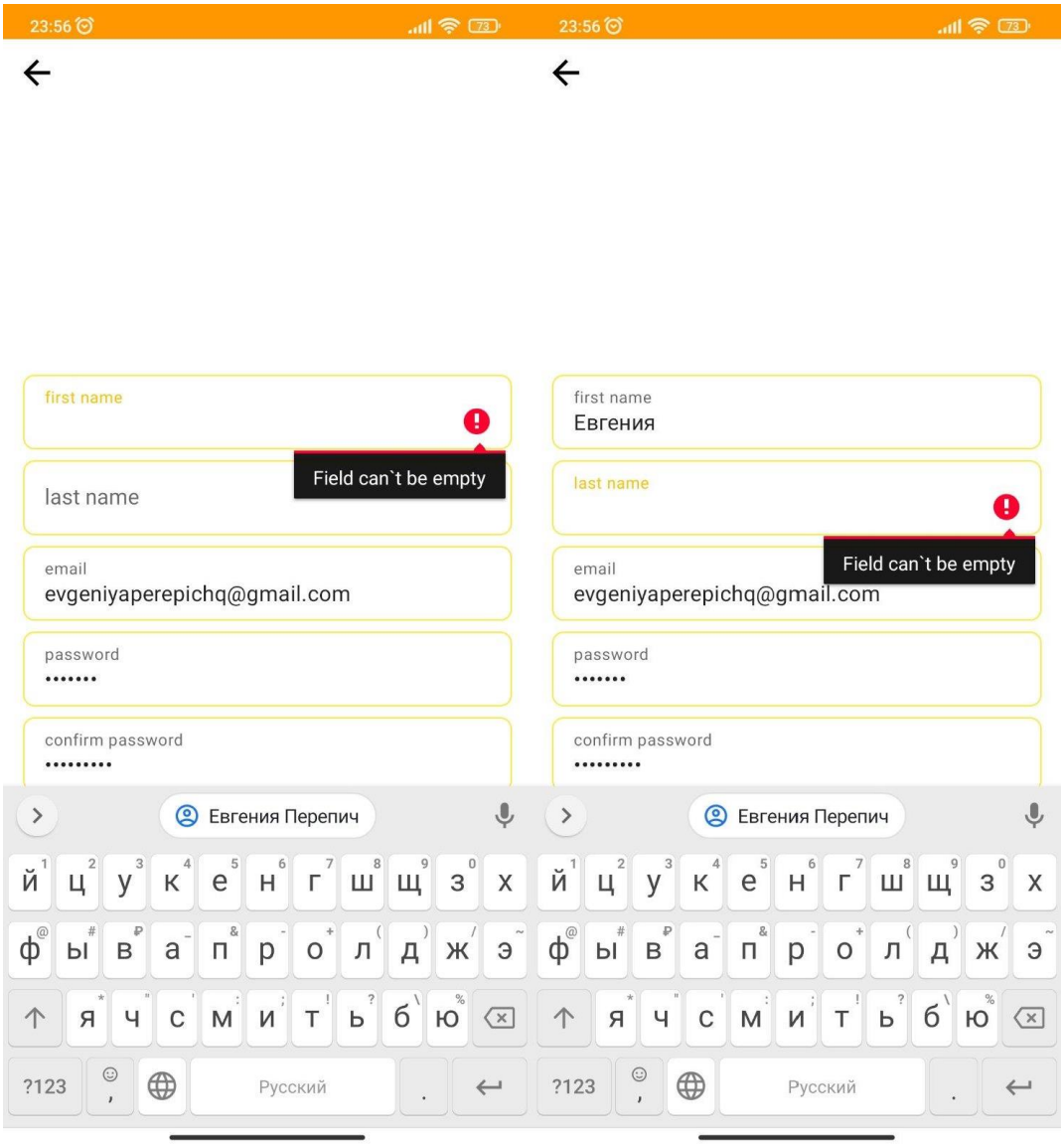

Рисунок 4.6 – Реєстрація без ім’я чи прізвища

На наступному кроці проведемо тестування етапу створення бронювання.

В цьому блоці буде розглянуто наступні ситуації:

- створення бронювання без часу закінчення;
- створення бронювання без опису;
- створення бронювання на час, який вже зайнято.

Час початку та закінчення бронювання дуже важливий, через велику завантаженість кімнати, тому ці поля є обов’язковими до заповнення (рис. 4.7). Адже, якщо користувач зробить бронювання без кінцевого часу, по-перше не

буде розуміння, чи дійсно кімната зайнята, по-друге це буде викликати негативні враження від додатку, бо такий сервіс ніяк не допомагає у запобіганні непорозумінь та зривів нарад.

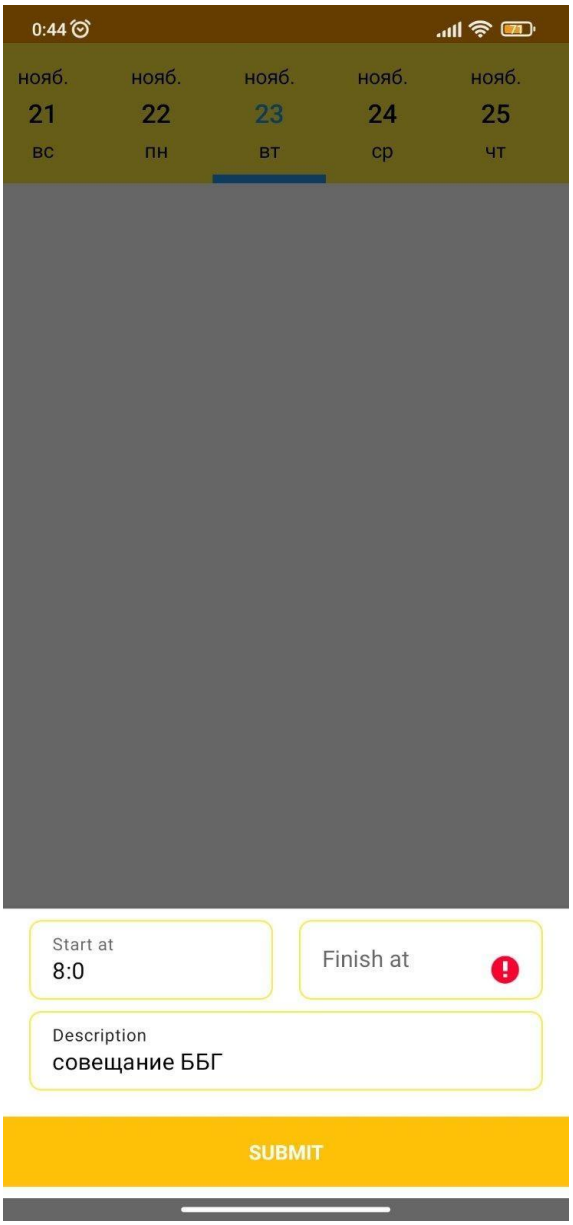

Рисунок 4.7 – Створення бронювання без часу закінчення

Поле опису додане як обов’язкове з метою розуміння цілей та критичності наради, під яку створюється бронювання, щоб у разі надзвичайних випадків була можливість домовитись по перенесення чи

видалення конфліктних бронювань. Результат незаповненого поля «опис» можно побачити на рисунку 4.8.

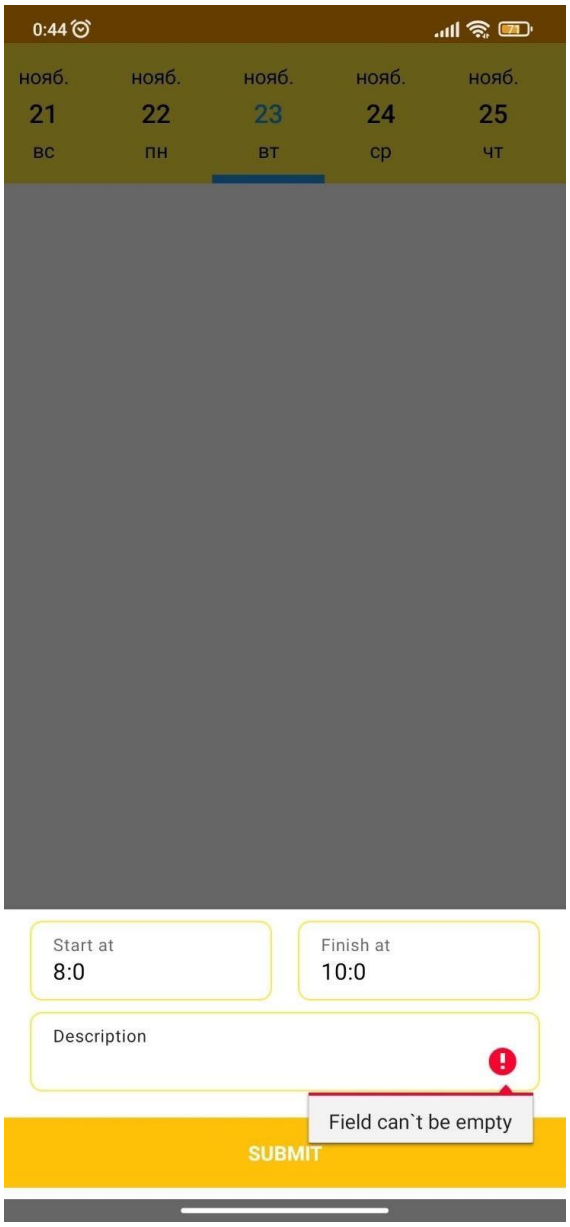

Рисунок 4.8 – Створення бронювання без опису

Одне з основних призначень додатку це приведення до 0 запланованих нарад, що конфліктують між собою. Тому наступний тест, тест на конфліктність, на мою думку найважливіший. При створенні бронювання на період с 7:00 до 11:00 система видає помилку «Цей час все зайнято» (рис. 4.9).

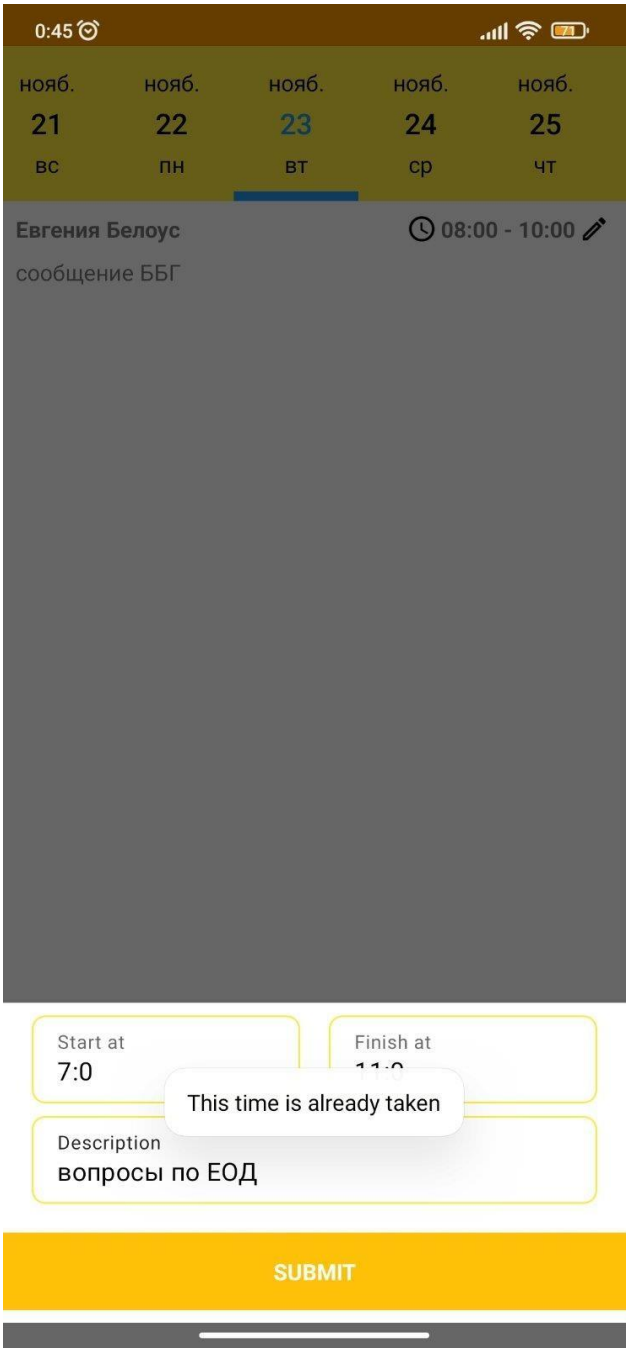

Рисунок 4.9 – Створення бронювання на час, який вже зайнято

Виходячи з отриманих результатів тесту, вважаю, що додаток виконую призначену йому мету на 100%.

ВИСНОВКИ

Метою даної кваліфікаційної роботи було полегшення взаємодії користувача з сервісом та фахівцями засобами розробки мобільного додатку для автоматизації бронювання переговорних кімнат, в яких встановлено термінал відео-конферец зв’язку.

Для досягнення мети даної роботи було використано ряд сервісів:

- програмне середовище Android Studio - інтегроване середовище розробки виробництва Google, за допомогою якого розробникам стають доступні інструменти для створення програм на платформі Android OS [10];
- хмарна база даних Firebase - це платформа для мобільних і веб-додатків, що при розробці отримала три основні сервіси: база даних реального часу, аутентифікація, статичний хостинг;
- админ панель flamelink - дозволяє швидко і легко створювати CMS для ваших проектів Firebase, вона підключається до Firebase дозволяючи керувати проектом швидко й легко;
- мова програмування Kotlin - статично типізована мова програмування, що працює поверх JVM.

Створений додаток пройшов тестування, на сьогоднішній момент недоліків не виявлено. Він має усі необхідні функції: авторизація, створення редагування та видалення бронювань. Проект було надано до безпосереднього керівника в компанії для аналізу та узгодження ідеї. Наразі додаток розроблено лише під платформу Android, у разі затвердження проекту планується його розробка під платформу IOS.

ПЕРЕЛІК ПОСИЛАНЬ

1 Firebase. Національні бібліотека ім. Н.Е. Баумана. URL: https://ru.bmstu.wiki/Firebase (дата звернення 01.07.2021)

2 Спростіть вхід за допомогою швидкої та надійної автентифікації Firebas. Developers. URL: https://developer.android.com/distribute/best-practices/develop/ firebase-authentication?hl=RU (дата звернення 01.07.2021)

3 Flamelink. URL: https://flamelink.io/ (дата звернення 09.08.2021)

4 Kotlin. URL: https://kotlinlang.org/ (дата звернення 01.07.2021)

5 Android Studio. Вікіпедія. URL: https://uk.wikipedia.org/wiki/ Android_Studio (дата звернення 09.08.2021)

6 REST. Вікіпедія. URL: https://uk.wikipedia.org/wiki/REST (дата звернення 20.09.2021)

7 Sublime Text 3. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Sublime_Text (дата звернення 20.09.2021)

8 Kotlin. URL: https://uk.google-info.org/2089990/1/kotlin.html (дата звернення 20.09.2021)

9 Model-View-ViewModel. URL: https://ru.wikipedia.org/wiki/Model-View-ViewModel (дата звернення: 03.10.2021).

10 Що таке Android Studio - Web-Proger. URL: http://web.spt42.ru/ index.php/chto-takoe-android-studio (дата звернення 19.11.2021)

11 Розробка мобільного додатку MySchool на базі операційної системи Android. Eprints Repo. URL: http://eprints.library.odeku.edu.ua/id/eprint/ 1504/1/Andrus_Rozrob_mob_B_2018.pdf (дата звернення 09.08.2021)

ДОДАТОК А

Код запиту на створення бронювання

```
package com.perepich.bookingroomapp.usecase

import com.google.firebase.firestore.FirebaseFirestore
import com.perepich.bookingroomapp.data.Book
import com.perepich.bookingroomapp.data.Room
import com.perepich.bookingroomapp.ext.toDate
import com.perepich.bookingroomapp.ext.toTime
import kotlinx.coroutines.ExperimentalCoroutinesApi
import kotlinx.coroutines.suspendCancellableCoroutine
import kotlin.collections.ArrayList
import kotlin.collections.HashMap

fun interface GetBookUseCase {
   suspend fun invoke(room: Room): List<Book>
}

class GetBookUseCaseImpl(private val firestore: FirebaseFirestore) : GetBookUseCase {

   @OptIn(ExperimentalCoroutinesApi::class)
   override suspend fun invoke(room: Room): List<Book> = suspendCancellableCoroutine { coroutine ->

firestore.collection("fl_content").document(room.id).get().addOnSuccessListener {
          val book = it["book"] as? ArrayList<*>
          coroutine.resume( book?.map {
```

```
                val item = it as HashMap<String,Any>
                Book(
                    id = item["uniqueKey"].toString(),
                    description = item["description"].toString(),
                    userId = item["userId"].toString(),
                    startTime = item["startTime"].toString().toTime()!!,
                    endTime = item["endTime"].toString().toTime()!!,
                    name = item["person"].toString(),
                    date = item["date"].toString().toDate()!!
                )
            }?: emptyList<Book>(),null)
        }.addOnFailureListener {
            }
    }
}
```

ДОДАТОК Б

Код запиту на редагування бронювання

```
package com.perepich.bookingroomapp.usecase

import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.firestore.FieldValue
import com.google.firebase.firestore.FirebaseFirestore
import com.perepich.bookingroomapp.data.Book
import com.perepich.bookingroomapp.data.Room
import com.perepich.bookingroomapp.ext.toDate
import com.perepich.bookingroomapp.ext.toTime
import kotlinx.coroutines.ExperimentalCoroutinesApi
import kotlinx.coroutines.suspendCancellableCoroutine
import java.util.*

fun interface SetBookingUseCase {
    suspend operator fun invoke(room: Room, booking: Book): Result<Unit>
}
class SetBookingUseCaseImpl(private val firestore: FirebaseFirestore,private val
auth: FirebaseAuth) : SetBookingUseCase {

    @OptIn(ExperimentalCoroutinesApi::class)
    override suspend fun invoke(room: Room, booking: Book)=
suspendCancellableCoroutine<Result<Unit>>{ coroutine->

firestore.collection("fl_content").document(room.id).update("book",FieldValue.arr
ayUnion(hashMapOf<String, Any>(
            "uniqueKey" to UUID.randomUUID().toString(),
```

```
            "person" to auth.currentUser!!.displayName!!,
            "userId" to  auth.currentUser!!.uid,
            "startTime" to booking.startTime.toTime(),
            "endTime" to booking.endTime.toTime(),
            "description" to booking.description,
            "date" to booking.date.toDate()
        )))
            .addOnSuccessListener {
                coroutine.resume(Result.success(Unit),null)
            }.addOnFailureListener {
                coroutine.resume(Result.failure(it),null)
            }
    }
}
```

ДОДАТОК В

Код запиту на видалення бронювання

```
package com.perepich.bookingroomapp.usecase

import com.google.firebase.firestore.FieldValue
import com.google.firebase.firestore.FirebaseFirestore
import com.perepich.bookingroomapp.data.Book
import com.perepich.bookingroomapp.data.Room
import com.perepich.bookingroomapp.ext.toDate
import com.perepich.bookingroomapp.ext.toTime
import kotlinx.coroutines.ExperimentalCoroutinesApi
import kotlinx.coroutines.suspendCancellableCoroutine
import java.util.*

interface DeleteBookingUseCase {
   suspend operator fun invoke(room: Room, booking: Book): Result<Unit>
}
class DeleteBookingUseCaseImpl(private val firestore: FirebaseFirestore) :
DeleteBookingUseCase {
   @OptIn(ExperimentalCoroutinesApi::class)
   override suspend fun invoke(room: Room, booking: Book) =
      suspendCancellableCoroutine<Result<Unit>> { coroutine ->
         firestore.collection("fl_content").document(room.id).update(
            "book", FieldValue.arrayRemove(
               hashMapOf<String, Any>(
                  "uniqueKey" to booking.id,
                  "person" to booking.name,
                  "userId" to booking.userId,
```

```
                    "startTime" to booking.startTime.toTime(),
                    "endTime" to booking.endTime.toTime(),
                    "description" to booking.description,
                    "date" to booking.date.toDate()
                )
            )
        ).addOnSuccessListener {
                coroutine.resume(Result.success(Unit), null)
            }.addOnFailureListener {
                coroutine.resume(
                  Result.failure(it), null
                )
            }
    }
}
```