Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Петрини Вікторії Іванівни
Прізвище, ім'я, по батькові студента(ки)

на здобуття ступеня вищої освіти Бакалавра

Мобільний застосунок для обліку передплат у поштовому відділенні
Назва теми

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

КПППЗ. 2101101.01.03.ПЗ

Виконала студентка III курсу, група ІПЗс-21-1 _____ Петрина В.І.
Підпис Ім'я, ПРІЗВИЩЕ

Керівник д-р фіз.-мат. наук, проф. _____ Бедратюк Л. П.
Науковий ступінь, вчене звання Підпис Ім'я, ПРІЗВИЩЕ

Нормоконтролер старший викладач _____ Бедратюк Г.І.
Посада Підпис Ім'я, ПРІЗВИЩЕ

До захисту допускаю:
Завідувач кафедри інженерії
програмного забезпечення _____ Бедратюк Л. П
Підпис Ім'я, ПРІЗВИЩЕ

6 червня 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри Л. П. Бедратюк
02 01 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Петрині Вікторії Іванівні
Прізвище, ім'я, по батькові студента

1. Тема кваліфікаційної роботи Мобільний застосунок для обліку передплат у поштовому відділенні

Керівник кваліфікаційної роботи Бедратюк Леонід Петрович, д-р фіз.-мат. наук, проф.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 08.01.2024 р. № 6-КП
2. Строк подання студентом роботи на кафедру 06.06.2024 р.
3. Вихідні дані до роботи Матеріали переддипломної практики
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
1. Дослідження предметної області та постановка задачі
2. Проєктування програмного забезпечення
3. Програмна реалізація та тестування програмного забезпечення
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Три креслення у форматі А3 (UML-діаграма варіантів використання, ER-діаграма бази даних, UML-діаграма послідовностей)

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Бедратюк Г.І., старший викладач	03.06.24	03.06.24
Антиплагіат	Форкун Ю.В., канд.тех.наук, доцент	30.05.24	06.05.24

7. Дата видачі завдання « 2 » січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1 Ознайомлення з тематикою кваліфікаційної роботи, визначення та узгодження індивідуальних тем кваліфікаційних робіт(КвР)	01.12– 31.12.2023	
2 Збір матеріалу за темою КвР; дослідження предметної області, в якій планується використання програмного забезпечення(ПЗ), визначення задач та вимог; розробка технічного завдання	01.01 – 20.02.2024	
3 Проектування програмного забезпечення	21.02 – 20.03 2024	
4 Програмна реалізація	21.03 – 30.04.2024	
5 Тестування програмного забезпечення	01.05 – 25.05.2024	
6 Написання вступу, загальних висновків, оформлення переліку джерел посилання та додатків. Оформлення пояснювальної записки КвР згідно вимог	01.05-25.05.2023	
7 Попередній захист КвР	23.05.2024	
8 Перевірка КвР на плагіат, нормоконтроль, отримання відгуків, рецензій. Брошурування(зшиття) пояснювальної записки	26.05 – 30.05.2024	
9 Здача КвР на кафедру; підготовка КвР для розміщення у репозитарії ХНУ; підготовка до захисту та захист КвР	З 01.06.2024	

Студент _____ Петрина В.І.
Підпис Ініціали, прізвище

Керівник роботи _____ Бедратюк Л.П.
Підпис Ініціали, прізвище

Анотація

Тема кваліфікаційної роботи: «Мобільний застосунок для обліку передплат у поштовому відділенні».

Автор роботи: Петрина Вікторія Іванівна

Керівник роботи: Бедратюк Леонід Петрович

Пояснювальна записка: 72с., 65 рис., 1 табл., 2 дод., 27 джерел.

Графічна частина: 3 креслення

Ключові слова: ОБЛІК ПЕРЕДПЛАТ, МОБІЛЬНИЙ ЗАСТОСУНОК, ПОШТОВЕ ВІДДІЛЕННЯ, ANDROID, FIREBASE, JAVA, ANDROID STUDIO

Метою роботи є розробка мобільного застосунку під Android ОС для обліку передплат у поштовому відділенні, що забезпечує операторів зручним та інтуїтивно зрозумілим інтерфейсом для реєстрації передплатників та оформлення передплат.

У кваліфікаційній роботі проведено аналіз предметної області та існуючих програмних рішень, визначено вимоги до програмного забезпечення, спроектовано архітектуру мобільного застосунку та його основні компоненти. Для реалізації програмного продукту використано мову програмування Java, середовище розробки Android Studio, базу даних Firebase та UML для проєктування.

Мобільний застосунок призначений для використання у поштових відділеннях, де необхідно вести облік передплатників та їхніх передплат. Застосунок дозволяє користувачам реєструвати нових передплатників, оформлювати нові передплати, редагувати інформацію та переглядати каталог видань.

Практична значимість роботи полягає в автоматизації процесу обліку передплат, що значно спрощує роботу операторів поштових відділень, підвищує точність ведення даних та зменшує кількість помилок.

Отримані результати продемонстрували високу ефективність застосунку в реальних умовах експлуатації. Подальший розвиток може включати інтеграцію з додатковими сервісами для аналізу даних та розширення функціональності для більш детального обліку та прогнозування передплат.

Підпис студента _____ Дата 03.06.2024

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	КПІПЗ. 2101101.01.03.ПЗ	Пояснювальна записка	72		
2	А4		Завдання на кваліфіка-	1		
			ційну роботу			
3	А4		Анотація	1		
			Графічні документи			
5	А3	КПІПЗ. 2101101.01.03.ПЗ	UML-діаграма варіантів	1		
			використання			
6	А3	КПІПЗ. 2101101.01.03.ПЗ	ER-діаграма бази даних	1		
7	А3	КПІПЗ. 2101101.01.03.ПЗ	UML-діаграма	1		
			послідовностей			

Змн.	Арк.	№ докум.	Підпис	Дата	КПІПЗ. 2101101.01.03.ВД		
Виконала		Петрина В.І.		05.06	Мобільний застосунок для обліку передплат у поштовому відділенні Відомість документів	Літ.	Арк. 1 / Аркушів 1
Керівник		Бедратюк Л.П		05.06			
Н. контр.		Бедратюк Г.І.		05.06		ХНУ, ІПЗс-21-1	
Зав. каф.		Бедратюк Л.П.		05.06			

ЗМІСТ

					КПІПЗ. 2101101.01.03.ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Виконала		Петрина В.І.		05.06.	Мобільний застосунок для обліку передплат у поштовому відділенні Відомість документів	Літ.	Арк.	Аркушів
Керівник		Бедратюк Л.П		6.06			2	2
Н. контр.		Бедратюк Г.І.		6.06		ХНУ, ІПЗс -21-1		
Зав. каф.		Бедратюк Л.П.		6.06				

ВСТУП

У сучасному світі цифрові технології відіграють надзвичайно важливу роль у багатьох сферах життя. Мобільні пристрої та програмне забезпечення для них стали невід'ємною частиною повсякденного існування, забезпечуючи зручний доступ до інформації та послуг у будь-який час та в будь-якому місці. Однією з важливих галузей, де мобільні технології мають значний вплив, є облік і управління передплатами у поштових відділеннях.

Аналіз та спостереження за ринком мобільних застосунків протягом останніх років показали, що все більше користувачів обирають мобільні додатки для вирішення своїх повсякденних завдань. Згідно зі статистикою, українці активно використовують мобільні застосунки в різних категоріях, включаючи фінансові сервіси, комунікаційні платформи та інструменти для управління особистими фінансами. Однак, у сфері обліку передплат та управління ними, особливо в контексті поштових відділень, все ще існує певний дефіцит спеціалізованих рішень.

Мобільні пристрої та програмне забезпечення для них надають широкі можливості для автоматизації процесів, підвищення ефективності та точності обліку даних. У сучасних умовах особливу актуальність набувають мобільні застосунки, які дозволяють інтегрувати різні джерела даних, забезпечуючи централізований доступ до інформації та спрощуючи управління передплатами. Використання мобільних технологій для обліку передплат у поштових відділеннях дозволяє значно спростити роботу операторів, підвищити точність ведення даних та зменшити кількість помилок.

У зв'язку з зростаючим попитом на цифрові рішення та мобільні застосунки, розробка мобільного застосунку для обліку передплат у поштовому відділенні є актуальним завданням. Такий застосунок забезпечить операторів зручним інструментом для реєстрації передплатників та оформлення передплат, що сприятиме підвищенню ефективності роботи поштових відділень та задоволенню потреб клієнтів.

Однією з основних проблем, з якими стикаються поштові відділення, є необхідність ефективного обліку передплатників та передплат. Існуючі рішення часто не відповідають сучасним вимогам або не забезпечують достатньої інтеграції з іншими системами, що ускладнює роботу операторів та підвищує ризик помилок. Розробка спеціалізованого мобільного застосунку дозволить вирішити ці проблеми та забезпечити більш високий рівень автоматизації процесів.

Основна проблема, на вирішення якої спрямована кваліфікаційна робота, полягає у розробці мобільного застосунку для обліку передплат у поштовому відділенні, який забезпечить зручний інтерфейс для реєстрації передплатників, оформлення передплат та управління ними. Завдання роботи включають аналіз предметної області, визначення вимог до програмного забезпечення, проєктування архітектури застосунку, реалізацію програмного продукту та його тестування.

Сучасний ринок програмного забезпечення пропонує широкий вибір рішень для управління фінансами та обліку даних, однак спеціалізовані застосунки для обліку передплат у поштових відділеннях зустрічаються рідко. Існуючі рішення часто не враховують специфіку роботи поштових відділень, що знижує їх ефективність. У зв'язку з цим, розробка нового мобільного застосунку, який забезпечить інтеграцію з різними системами та підвищить ефективність роботи операторів, є актуальною задачею.

Необхідність розробки спеціалізованого мобільного застосунку обумовлена потребою у підвищенні ефективності та точності обліку передплат у поштових відділеннях. Такий застосунок дозволить автоматизувати процеси реєстрації та обліку, зменшити кількість помилок та забезпечити централізований доступ до даних. Це, в свою чергу, сприятиме підвищенню якості обслуговування клієнтів та задоволенню їхніх потреб.

Метою даної кваліфікаційної роботи є розробка мобільного застосунку під Android ОС для обліку передплат у поштовому відділенні, що забезпечить

зручний інтерфейс для реєстрації передплатників, оформлення передплат та управління ними. Застосунок повинен забезпечити синхронізацію даних з базою даних Firebase для надійного збереження та швидкого доступу до інформації.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– провести аналіз предметної області та ринку програмного забезпечення;

– визначити вимоги до програмного забезпечення та його функціональні можливості;

– спроєктувати архітектуру мобільного застосунку та його основні компоненти;

– реалізувати програмний продукт, використовуючи мову програмування Java та середовище розробки Android Studio;

– здійснити інтеграцію з базою даних Firebase для забезпечення надійного збереження даних;

– провести тестування мобільного застосунку для перевірки його функціональності та виявлення можливих помилок;

– проаналізувати результати тестування та внести необхідні корективи для покращення роботи застосунку.

Розроблений мобільний застосунок може бути впроваджений у будь-якій організації, що надає послуги передплати, зокрема у відділеннях "Укрпошти". Застосунок дозволяє централізувати облік передплатників та передплат, забезпечуючи швидкий доступ до інформації та підвищуючи якість обслуговування клієнтів.

Таким чином, розробка мобільного застосунку для обліку передплат у поштовому відділенні є актуальною та необхідною задачею, що сприятиме підвищенню ефективності роботи поштових відділень та задоволенню потреб клієнтів.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовий аналіз предметної області, її структурних та функціональних особливостей

IT-спеціаліст – це людина чиї обов'язки вкрай важко писати двома словами. Це «універсальний солдат», без якого не обійдеться практично жодна фірма. За даними на сьогоднішній день і думкою багатьох аналітиків фахівці даної галузі є популярними і будуть затребувані в найближчому майбутньому.

IT-спеціалісти потрібні в будь-якій компанії, незалежно від масштабів і сфери діяльності, адже сьогодні будь-яка сучасна установа чи організація використовує програмне забезпечення для ведення своєї роботи і пошта не є винятком.

Тенденції, що складаються в сучасному світі вказують на те, що більшість підприємств та організацій в найрізноманітніших сферах рухаються у напрямку переходу від паперового документообігу до електронного формату. Програми для ведення документації відрізняються одна від одної та мають різну специфіку, в залежності від галузі, в якій вона використовується, але всі вони мають спільну рису – попит на них росте шаленими темпами.

Такі додатки позитивно впливають на робочий процес, завдяки ним значно легше автоматизувати роботу підприємства, збільшується швидкість обробки запитів клієнтів, систематизується документообіг.

На сьогоднішній день передплата – один з основних і найбільш зручних для споживача каналів розповсюдження періодики. Передплата періодичних видань в Україні зазвичай оформлюється у відділеннях пошти або передплатних агенціях. Поштове відділення надає послуги з організації і проведення передплати періодичних видань – вітчизняних і зарубіжних газет, журналів, видань журнального типу та книг в Україні і за її межами.

Зі стабільною тенденцією використання такої послуги, як передплата, зростає попит на спрощення реалізації її оформлення. Саме тому обрана тема дипломного проекту є актуальною і затребуваною.

Багато підприємств активно користуються спеціалізованими періодичними виданнями у своїй діяльності. Ці видання можуть містити важливу для бізнесу інформацію, новини, аналізи та інші матеріали, які сприяють розвитку компанії та підвищенню професійної компетентності співробітників.

У випадку з додатками, які спеціалізуються на електронну передплату періодичних видань можна виділити ряд переваг, що отримують їх користувачі. Перш за все слід виділити те, що в сучасних реаліях окрім звичайної передплати паперових видань з'являються також і цифрові. Завдяки розроблюваному програмному забезпеченню весь широкий спектр передплат зібраний в одну єдину базу, що значно спрощує оформлення різних передплат і дає змогу зробити підписку на різні видання в одному місці та за один раз, без необхідності здійснювати таку процедуру кожного разу вручну для кожної передплати.

До цього також можна віднести те, що за допомогою додатку значно легше керувати вибором виду підписки та способом доставки, адже варіативність підписок та способів доставки є досить великою. Існує велика кількість підписок(щомісячна, квартальна, піврічна, річна тощо), якими стає надзвичайно легко керувати, коли вони об'єднані для кожного окремого користувача в один єдиний запис в системі.

Це також стосується і керуванням способами доставки та форматом видання, адже більшість видавництв на даний момент мають змогу надавати свої видання в електронному і паперовому форматі, і вже сам користувач відповідно до своїх вимог може обрати спосіб який йому найбільше підходить відповідно до обраного формату(наприклад, цифрова версія на електронну пошту або паперовий варіант з доставкою на відділення). Завдяки

розроблюваному програмному забезпеченню в працівника, що оформлює передплату або редагує її за запитом користувача, з’являється можливість переглядати, продовжувати або скасовувати підписки в кілька кліків.

Не менш важливим є те, що програмне забезпечення спрощує збір різної інформації та статистики, за допомогою чого для кожного окремого користувача можна сформувати список видань на основі уподобань користувача та історії його читання. Також цю статистику можна використовувати для розуміння вподобань усіх користувачів загалом, що дає змогу підприємству розширювати список видавництв та видань, з якими воно співпрацює та надає послуги з передплати їх продукції.

Вивчення предметної області допомагає визначити функціональні особливості, що необхідні для розробки майбутнього програмного забезпечення.

Для того, щоб наочно зобразити та описати бізнес-процеси підприємства, можна скористатися IDEF0 діаграмою. Цей метод розроблений для того, щоб допомогти зрозуміти, аналізувати та вдосконалювати складні процеси. IDEF0 діаграма зосереджується на відображенні функцій (або діяльностей) і їх взаємозв'язків за допомогою графічних елементів.

Контекстна діаграма застосунку обліку передплат (рисунок 1.1.1) являє собою найвищий рівень абстракції, який ілюструє взаємодію між системою обліку передплат та зовнішніми сутностями, що надсилають і отримують інформацію. Ця діаграма допомагає зрозуміти загальний потік даних і основну функціональність системи.

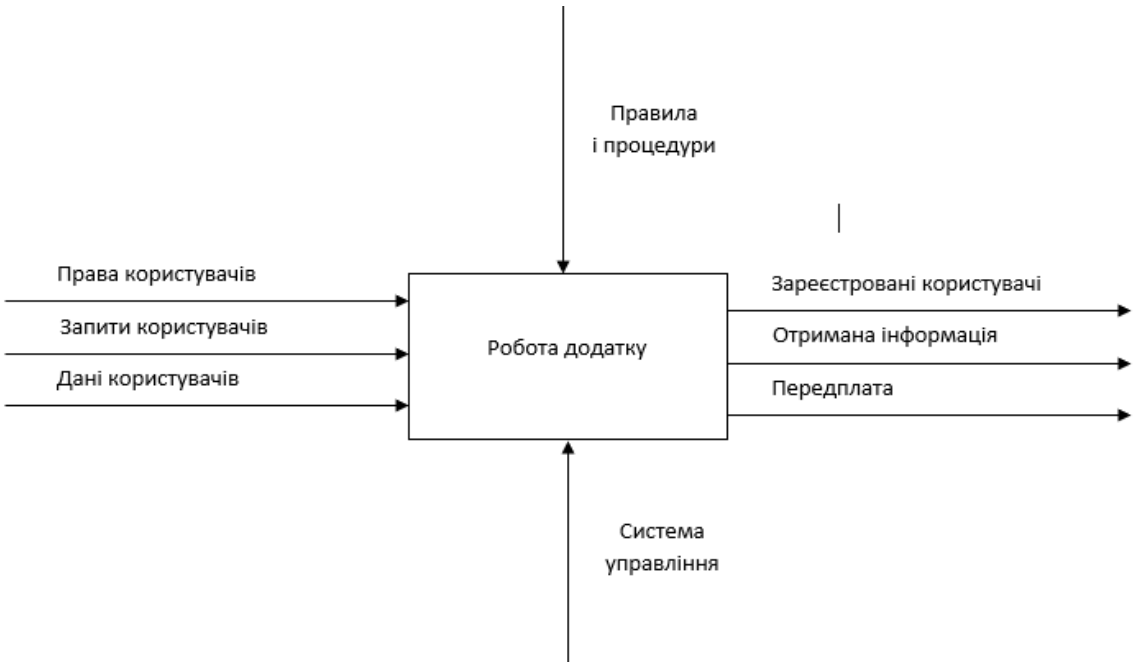


Рисунок 1.1.1 - контекстна IDEF0-діаграма роботи програмного забезпечення

Діаграма, зображена на рисунку 1.1.2 презентує, що основна функція застосунку - управління передплатами. Він включає в себе три ключові процеси: реєстрацію і авторизацію користувачів, перегляд передплат та оформлення передплат.

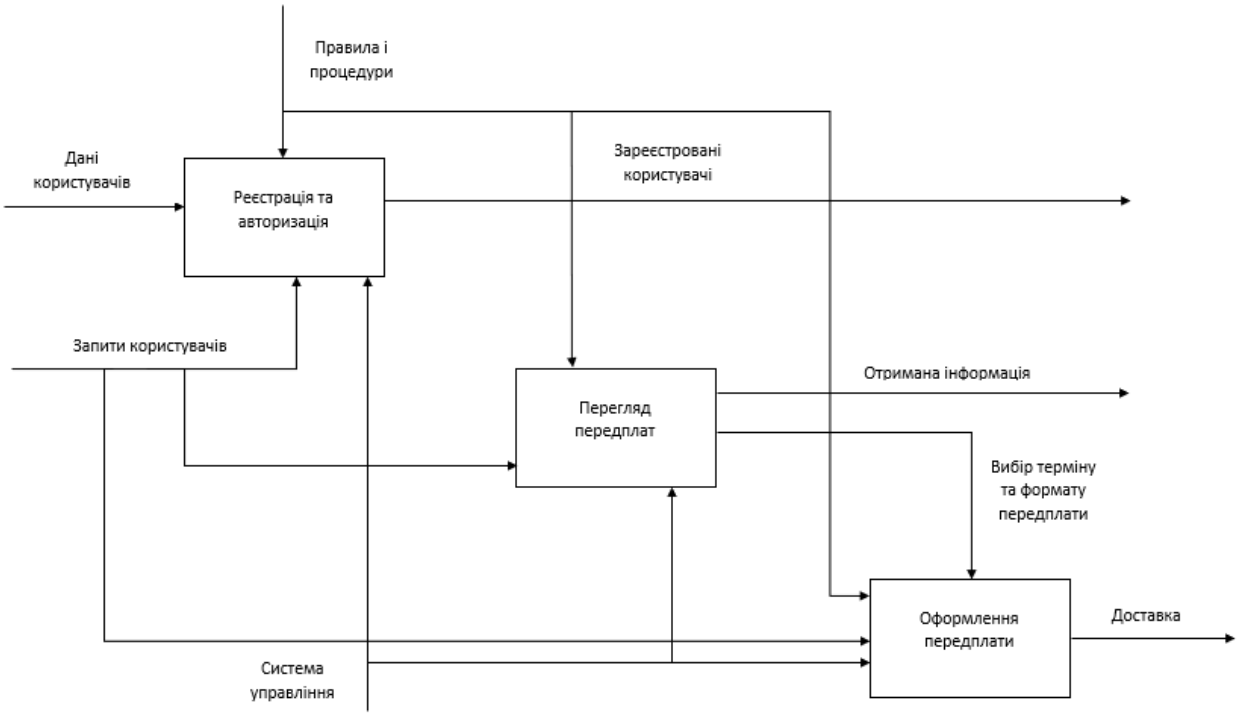


Рисунок 1.1.2 – IDEF0 діаграма

Перший процес - реєстрація і авторизація користувачів. На цьому етапі нові користувачі вводять особисту інформацію, таку як ім'я, прізвище, номер відділення, електронна пошта та пароль, для створення нового облікового запису. Для існуючих користувачів, авторизація передбачає введення своїх облікових даних для отримання доступу до системи. Після успішної реєстрації або авторизації користувач отримує доступ до інших функцій системи.

Другий процес - перегляд передплат. Після успішної авторизації користувачі можуть переглядати інформацію про передплати.

Третій процес - оформлення передплат. Це включає вибір бажаного видання та його кількість, дані передплатника, терміну передплати. Система обробляє інформацію и та оновлює інформацію у базі даних передплат. Також користувач може редагувати та видаляти вже оформлені передплати.

Таким чином, IDEF0 діаграма зображає логічну послідовність дій та взаємодій між процесами реєстрації і авторизації користувачів, перегляду передплат та оформлення передплат, забезпечуючи ефективне управління системою передплат.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

Необхідним пунктом при розробці програмного забезпечення є аналіз існуючих методів та рішень. Це дозволяє зробити проєкт максимально зручним, а його функціонал найбільш відповідним поставленій меті.

Технології застосування традиційних паперових документів про угоду передплати створювалися вже дуже давно з урахуванням можливостей застосовувалися в той час технологій зберігання, передачі і обробки інформації. Зокрема, для зберігання інформації застосовувалися паперові носії, для передачі - телеграфні та телефонні голосові канали, для пошуку інформації великі сховища паперових носіїв проглядалися очима.

Використовувані технології не дозволяли проводити в масовому порядку за досить короткий час навіть вибіркову перевірку угод. Таким чином, виписаний документ повинен був бути сам по собі носієм повної інформації, яка необхідна для надання послуг та відповідних грошових розрахунків. Разом з тим, на різних стадіях процесу оформлення передплати теоретично можливі навмисні злочинні дії по фальсифікації інформації про укладені договори з метою незаконного отримання будь-якої фінансової вигоди.

Для електронного договору не потрібен спеціальний бланк - вся інформація зберігається в пам'яті комп'ютерів, недоступних для зловмисників, і при необхідності швидко передається з одного комп'ютера на інший. Відповідно, знижуються витрати пошти на виготовлення бланків і їх звернення. Далі, з розвитком засобів електронної комерції і електронних платежів з'являється можливість оплати передплати за допомогою автоматизованих пристроїв (банкомати, платіжні термінали, банківські комп'ютерні системи). Таким чином, скорочуються витрати пошти на організацію грошового обороту.

У більш сучасних реаліях, на великих підприємствах для виконання задач такого типу може використовуватися програма АС«АРМ ВЗ» (рисунок 1.2.1). Опитавши користувачів цієї програми, виявлено, що для користування таким програмним забезпеченням необхідний довгий період часу для навчання, адаптації та ознайомлення з програмою. Тобто інтерфейс програми не є зручним та зрозумілим для нових непідготовлених користувачів. В зв’язку з цим виникають проблеми під час роботи, що значно знижує рівень продуктивності працівників.

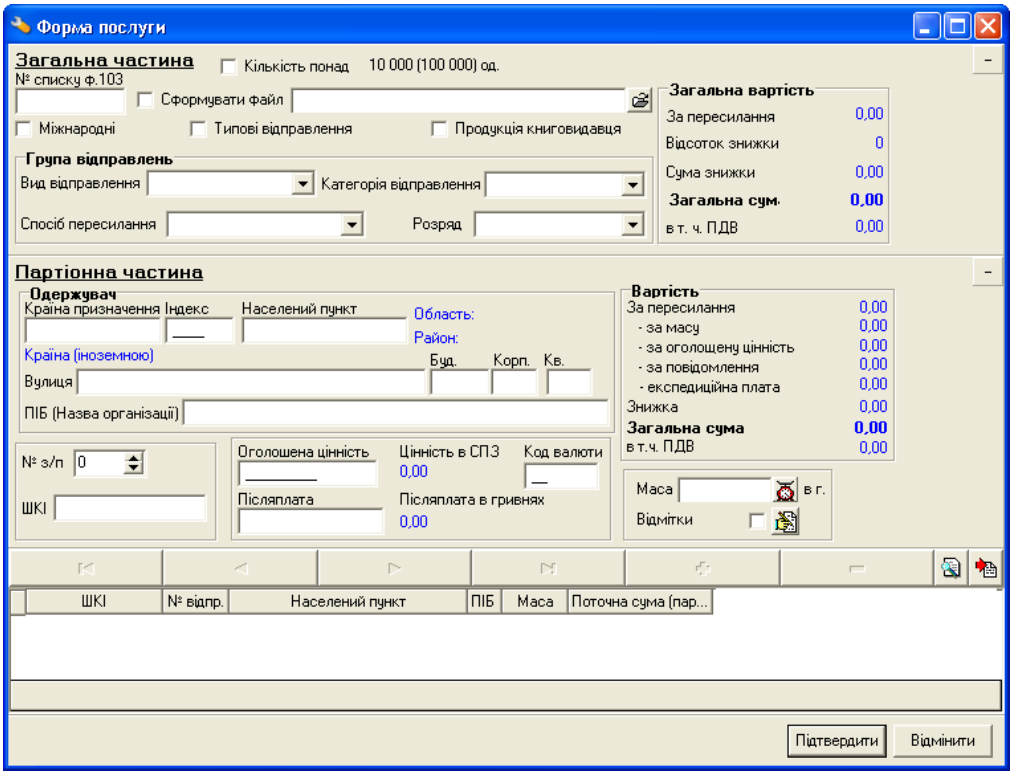


Рисунок 1.2.1 – Інтерфейс програми

Також неможливо не згадати ще один програмний засіб для обліку передплати на відділенні. Більш сучасним способом оформити передплату є користування сайтом пошти (рисунок 1.2.2).

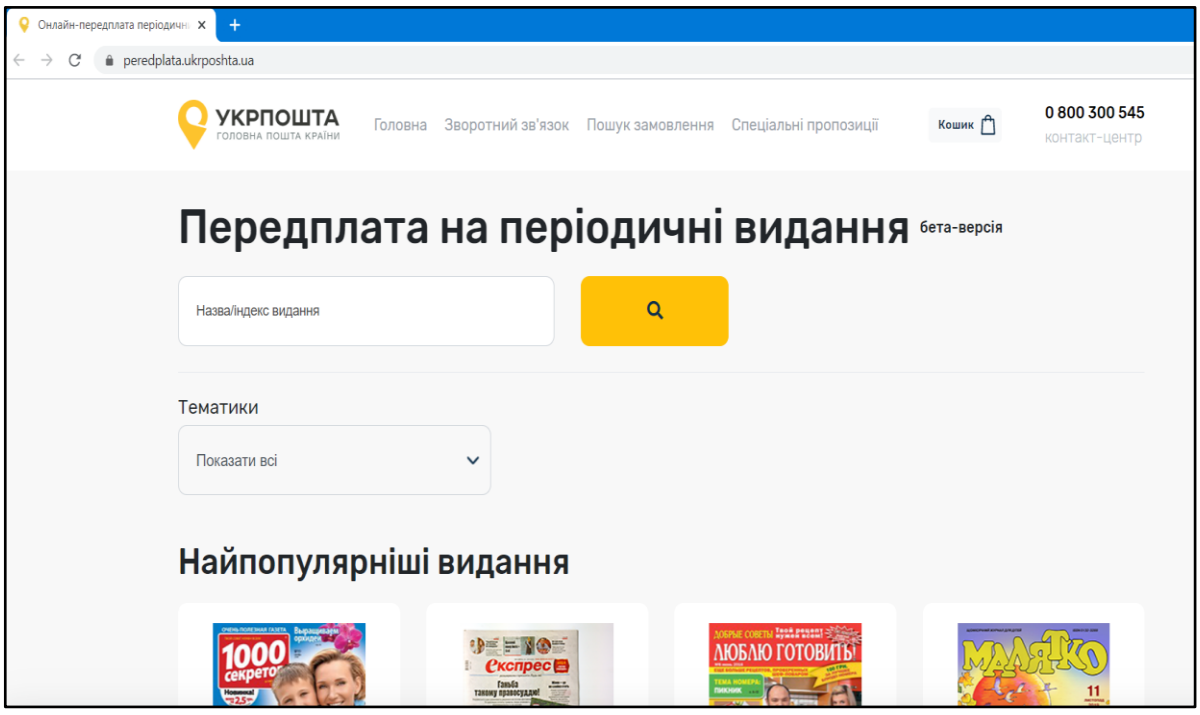


Рисунок 1.2.2 – Сайт для оформлення передплати онлайн

Сайт Укрпошти як програмно-технічне забезпечення для оформлення передплати видань є комплексним і багатофункціональним веб-ресурсом, що дозволяє користувачам швидко і зручно підписуватися на різні друковані видання. На цьому сайті представлений каталог видань, де користувачі можуть знайти журнали та газети, що їх цікавлять. Для зручності, каталог оснащений функціями сортування та фільтрації за різними категоріями, такими як жанр, частота виходу, ціна тощо. Це допомагає користувачам швидко орієнтуватися у великій кількості доступних видань і знайти те, що їм потрібно.

Процес оформлення передплати є простим і інтуїтивно зрозумілим. Користувачі можуть вибрати видання, вказати період передплати, заповнити необхідні дані отримувача та обрати спосіб доставки і оплати. Сайт підтримує різні платіжні системи, що забезпечує зручність і безпеку транзакцій.

Особистий кабінет користувача є ще однією важливою складовою сайту, де користувачі можуть керувати своїми передплатами. В особистому кабінеті зберігається історія замовлень, активні передплати, і користувачі можуть поновлювати свої підписки, змінювати налаштування профілю, а також оновлювати контактну інформацію.

Сайт Укрпошти також надає розширену підтримку клієнтів через різні канали. Користувачі можуть отримати консультацію або допомогу через онлайн-чат, скористатися формою зворотного зв’язку, знайти контактні номери телефонів або ознайомитися з розділом частих запитань (FAQ), де можна знайти відповіді на найбільш поширені питання.

Технічно, сайт побудований на сучасних веб-технологіях. Веб-інтерфейс створений з використанням HTML, CSS та JavaScript, з можливим використанням сучасних фреймворків, таких як React, Angular або Vue.js, що забезпечує швидкість і зручність користування. Серверна частина сайту може бути реалізована на таких мовах програмування, як Python, PHP або Node.js. Дані користувачів, інформація про видання та передплати зберігаються в базі

даних, яка може бути побудована на основі MySQL, PostgreSQL або інших реляційних СУБД.

Основною відмінністю сайту Укрпошти від теми кваліфікаційної роботи є те, що він орієнтований на клієнтів пошти, а не на її працівників.

Попри свою корисність і широкі можливості, він має кілька недоліків. По-перше, інтерфейс сайту може бути не завжди інтуїтивно зрозумілим, особливо для людей старшого віку, які не мають значного досвіду використання інтернет-ресурсів. Це може ускладнити процес оформлення передплати та викликати труднощі у користувачів.

По-друге, швидкість роботи сайту може залишати бажати кращого. Високе навантаження або недостатня оптимізація можуть призводити до тривалого завантаження сторінок, що негативно впливає на користувацький досвід. Крім того, сайт може бути недостатньо адаптований для мобільних пристроїв. Незважаючи на те, що більшість користувачів сьогодні віддають перевагу мобільним платформам, мобільна версія сайту може мати обмежений функціонал або працювати повільно.

Ще одним суттєвим недоліком є недостатня інтеграція з іншими системами та сервісами. Це може ускладнити процес оплати та доставку видань, особливо якщо користувачі бажають використовувати різні платіжні системи або обирають різні способи доставки. Також можливі проблеми з оновленням даних в реальному часі, що може призводити до затримок у відображенні актуальної інформації про доступність видань або статус передплат.

Сайт Укрпошти як програмно-технічне забезпечення для оформлення передплати видань має кілька суттєвих недоліків, які можуть впливати на зручність і ефективність його використання клієнтами пошти. Важливо враховувати ці аспекти для подальшого покращення функціоналу та користувацького досвіду.

Врахувавши всі переваги та недоліки існуючих рішень, було створено програму з простим та зрозумілим для всіх інтерфейсом, яка виконує всі необхідні функції і призначена саме для роботи операторів поштового відділення передплати.

1.3 Визначення вимог до програмного забезпечення та розробка технічного завдання

Для того щоб виконати поставлену задачу, її необхідно розбити на окремі блоки, кожен з яких повинен бути логічно завершеним та виконувати певний набір функцій.

Задача розбита на такі блоки:

– визначення мети проєкту та завдання;
– аналіз предметної області;
– вибір програм для розробки програмного коду;
– проєктування структури вхідних і вихідних даних;
– методи організації вхідних і вихідних даних.

Під аналізом предметної області розуміється будування системи у вигляді трьох взаємозалежних моделей. Перша - об’єктна модель задачі, тобто визначення класів та об’єктів. Друга - динамічна модель задачі, тобто опис змін, що відбуваються під час роботи програми. Третя - це функціональна модель задачі, що являє собою визначення вхідних та вихідних даних опис функцій та опис обмежень.

Одним з необхідних пунктів для визначення вимог до прогрманого забезпечення є формалізований опис поставленої задачі. Це чітке, структуроване і однозначне представлення задачі, яке використовується для аналізу, розробки та впровадження рішень. Такий опис дозволяє уникнути неоднозначностей та забезпечує зрозуміле і точне розуміння вимог до задачі всіма учасниками процесу.

Формалізований опис поставленої задачі можна зробити у різних напрямках. Спершу потрібно розглянути задачі зі сторони взаємодії користувачів та програми.

Задача розглянута як взаємодія користувача та програми. Користувач може авторизуватися в уже наявний профіль або зареєструвати новий, зареєструвати передплатника або обрати наявного, оформити передплату або редагувати вже наявну, переглядати каталог видань, додавати нові видання в каталог, також в каталозі можна виконувати пошук. У користувача також є можливість змінити інформацію свого профілю.

Даний взаємозв’язок користувача і програми детально зображений на діаграмі варіантів використання на рисунку 1.3.1.

Діаграма варіантів використання (Use Case Diagram) є одним з основних засобів моделювання в UML (Unified Modeling Language). Вона використовується для опису функціональних вимог до системи та ілюструє взаємодію між користувачами (акторами) і системою для досягнення певної мети або виконання певної функції.

Основні елементи діаграми варіантів використання:

– актори (Actors) – зовнішні суб'єкти, які взаємодіють із системою;

– варіанти використання (Use Cases) – окремі функціональні одиниці або сценарії, які система виконує у відповідь на дії актора;

– зв'язки (Associations) – лінії, що з'єднують акторів з варіантами використання, показуючи, які актори взаємодіють з якими варіантами використання.

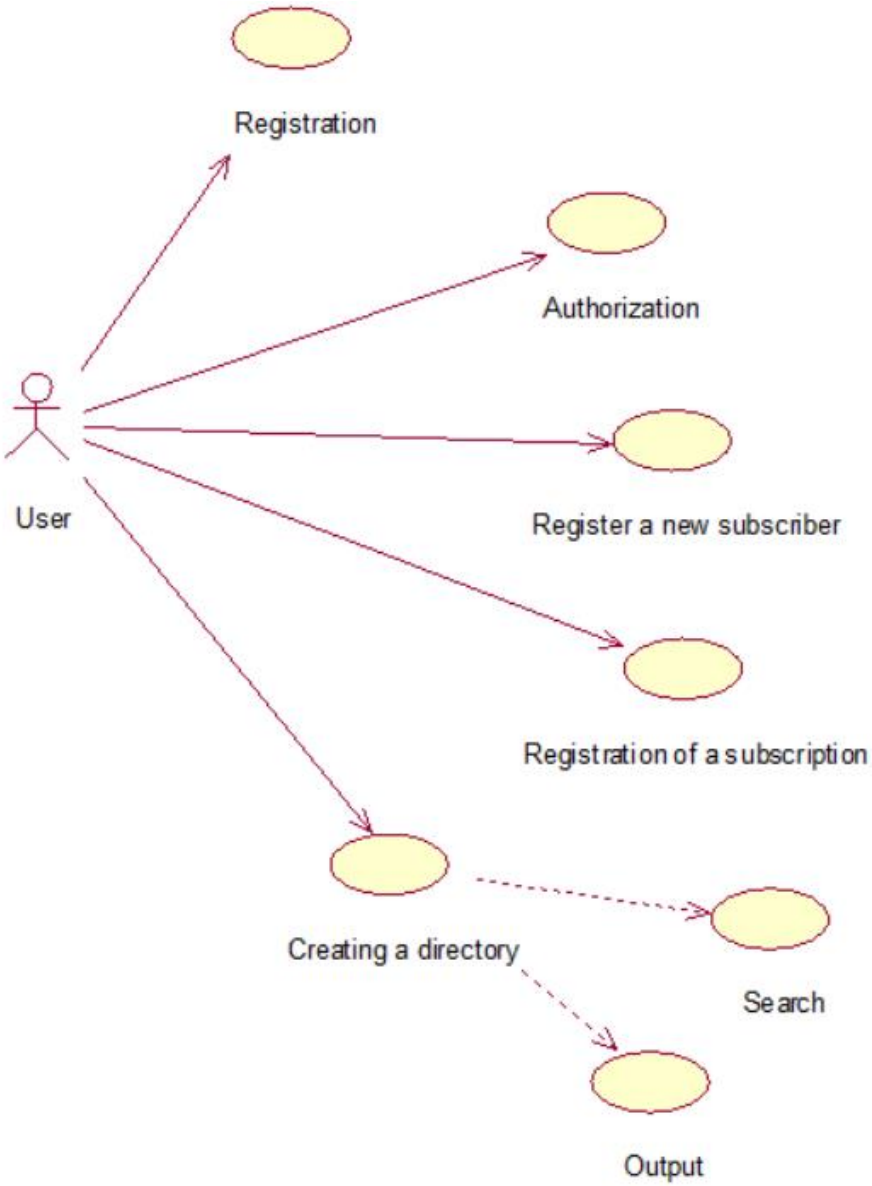


Рисунок 1.3.1 – Діаграма варіантів використання

Оскільки застосунок розробляється для операторів поштового відділення, неавторизовані користувачі не можуть виконувати жодних функцій в програмі. Тому на діаграмі варіантів використання (рисунок 1.3.1) зображено тільки одного актора, тобто авторизованого користувача.

Варіанти використання у застосунку :

– реєстрація – користувач реєструється в системі;

– авторизація – користувач входить до системи;

– створення каталогу – користувач створює видання, що додаються до каталогу;

– перегляд каталогу;

– пошук у каталозі;

– реєстрація передплатника – користувач реєструє нового передплатника в системі.

– оформлення передплати – користувач оформлює нову передплату.

Отже, формалізований опис завдання складається з різносторонніх описів взаємозв’язків користувача та програми

Відповідно до постановки завдання, у програмі необхідно реалізувати такий функціонал:

– авторизація або реєстрація працівника в програмі;

– створення та перегляд каталогу періодичних видань;

– фільтрація видань відносно певного параметра;

– реєстрація передплатника або вибір вже зареєстрованого;

– оформлення передплати, перегляд вже оформленої передплати;

– перегляд та пошук результатів передплати;

– можливість виконання пошуку в каталозі, в списку передплатників та списку оформленої передплати;

– передбачити збереження даних в базі даних.

Застосунок повинен мати:

– зручний та зрозумілий інтерфейс;

– функцію додавання нових даних;

– можливість редагування даних;

– можливість видалення даних;

– функцію збереження даних.

Отже, для коректної роботи програми необхідно забезпечити весь вище описаний функціонал.

1.4 Висновки до розділу 1

Отже, за підсумками першого розділу було проведено змістовий аналіз предметної області, її структурних та функціональних особливостей. Описано основні аспекти передплати періодичних видань, включаючи види підписки, способи доставки та актуальність теми для підприємств. Представлені переваги електронної підписки над традиційною паперовою формою. Надано статистичні дані та аналітику, що підтверджують актуальність теми.

Також було проведено аналіз наявного програмно-технічного забезпечення предметної області. Представлені переваги та недоліки розглянутих програмних засобів.

Останнім етапом було визначено функціональних та нефункціональних вимог до програмного забезпечення, описано основні функції застосунку: авторизація, реєстрація передплатників, оформлення та редагування передплат, перегляд каталогу видань, пошук у каталозі. Для більшого розуміння була представлена діаграма варіантів використання. Описано вимоги до функціональних і нефункціональних характеристик.

РОЗДІЛ 2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз та вибір архітектури мобільного застосунку

Архітектура програмного забезпечення - це невід'ємна частина загального дизайну системи та наріжний камінь усіх інновацій. Проектування архітектури програмного забезпечення є базою розробки ПЗ. Під проектуванням архітектури мається на увазі аналіз технічного завдання від замовника і подальше створення, згідно отриманого завдання, моделі бізнес-логіки додатка, специфікацій його компонентів, структури даних і їх відображення. Архітектура визначає логіку різних компонентів системи з точки зору рівня деталізації, необхідного для написання коду, а також визначає зв'язки між компонентами.

Головна мета архітектури програмного забезпечення полягає у відображенні та реалізації проектних рішень системи, яка будується. Також архітектура ПЗ допомагає візуалізувати, зрозуміти та проаналізувати якості системи. Архітектуру можна використовувати для оцінки проєктів перед їх реалізацією. Також не менш важливою функцією архітектури є прийняття ключових рішень для забезпечення якості роботи.

У своєму проєкті я використовувала багаторівневу архітектуру. Дана архітектура є чи не найпоширенішою, адже слугує як спосіб ефективного зберігання даних в таблицях. Цей підхід працює за принципом поділу відповідальності. ПЗ розділено на шари, що лежать один на одному, і кожен з них виконує певний обов'язок.

Багаторівнева архітектура ділить програму на три рівня (рисунок 2.1.1)

.

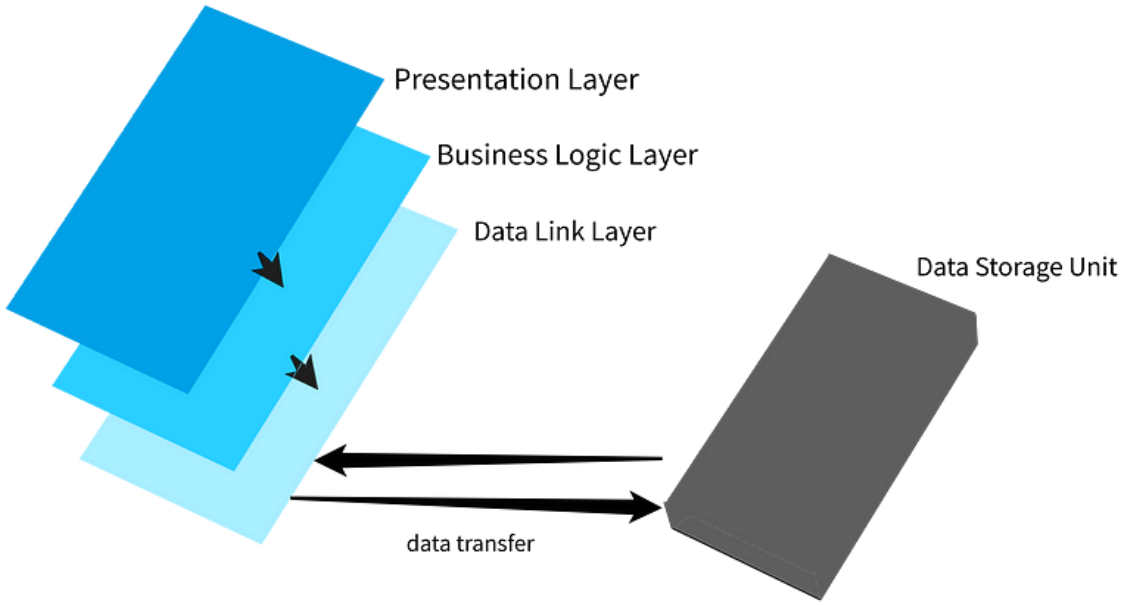


Рисунок 2.1.1 – багаторівнева архітектура

Рівень представлення (Presentation Layer) містить інтерфейс користувача і відповідає за забезпечення зрозумілого і ефективного користування програмою.

Рівень бізнес-логіки (Business Logic Layer), як випливає з назви, містить бізнес-логіку програми. Він відокремлює UI/UX від обчислень, пов'язаних із бізнесом. Це дозволяє з легкістю змінювати логіку в залежності від бізнес-вимог, що постійно змінюються, ніяк не впливаючи на інші шари.

Рівень передачі даних (Data Link Layer) відповідає за взаємодію з постійними сховищами, такими як бази даних, та інше опрацювання інформації.

Дані та елементи управління проходять через кожен рівень у дизайні та передаються від одного до іншого. Ця система також підвищує рівень абстракції та певною мірою навіть стабільність ПЗ. Також реалізація багаторівневої архітектури простіша порівняно з іншими підходами.

Одним із найважливіших інструментів архітектурного проектування є патерн. Патерни проектування є типовими рішеннями загальних проблем у

розробці програмного забезпечення. Кожен шаблон схожий на план, який можна налаштувати для вирішення певної проблеми у коді програми.

Три найбільш популярних шаблони проектування є MVC, MVP та MVVM. MVC означає модель, представлення та контролер, MVP - модель, представлення, презентер, а MVVM – це модель, представлення та модель представлення. Різницю між ними можна наочно побачили на рисунку

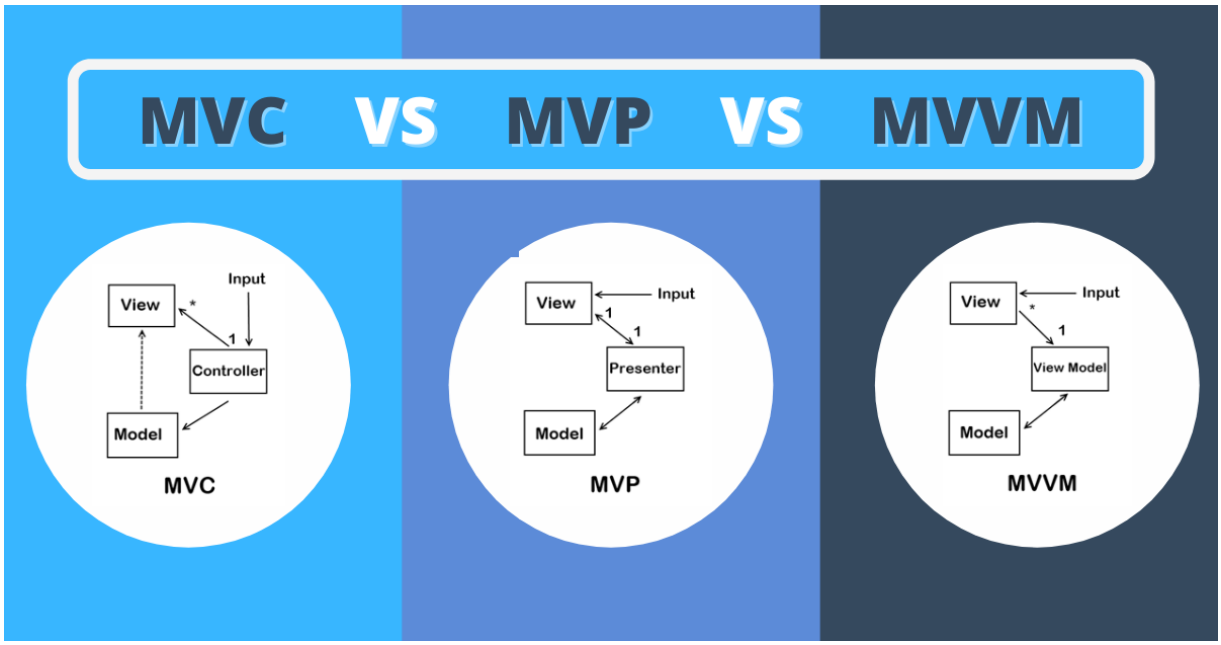


Рисунок 2.1.2 – Шаблони проектування

Для того, щоб детальніше розібратися у тому, який шаблон слід вибрати для реалізації проекту потрібно зробити їх порівняння.

MVP (Model-View-Presenter) складається з трьох основних компонентів: Model, яка містить бізнес-логіку та дані; View, яка відповідає за відображення інформації користувачу; та Presenter, який є посередником між Model і View. Presenter обробляє логіку взаємодії та оновлює View на основі змін у Model. Основною перевагою MVP є чітке розділення відповідальностей між компонентами, що дозволяє легко тестувати Presenter, оскільки він не залежить від View. Однак, у великих додатках Presenter може стати занадто складним.

MVC (Model-View-Controller) також має три компоненти: Model, яка містить дані та бізнес-логіку; View, яка відповідає за відображення даних; та Controller, який обробляє користувацький ввід та оновлює Model. Model автоматично оновлює View після змін. Головною перевагою MVC є простота реалізації та чітке відділення внутрішньої логіки від інтерфейсу. Проте, Controller може стати занадто великим і складним для підтримки, особливо у великих додатках.

MVVM (Model-View-ViewModel) складається з Model, яка містить дані та бізнес-логіку; View, яка відповідає за відображення та інтерфейс; та ViewModel, яка є абстракцією View і взаємодіє з Model, надаючи дані View. Головною перевагою MVVM є двонаправлений зв'язок між View і ViewModel, що полегшує синхронізацію даних та дозволяє легко тестувати ViewModel, оскільки він не залежить від View.

У таблиці 2.1.1 всі ці шаблони порівнюються за основними параметрами.

Таблиця 2.1.1 – порівняльна таблиця патернів

Параметр	MVP	MVC	MVVM
Компоненти	Model, View, Presenter	Model, View, Controller	Model, View, ViewModel
Розділення відповідальностей	Чітке розділення між компонетами	Відносне розділення	Чітке розділення між компонетами
Зв'язок між компонентами	View і Presenter взаємодіють безпосередньо	View оновлюється через Model	Двонаправлений зв'язок між View і ViewModel
Простота реалізації	Середня складність	Найпростіша	Середня складність
Переваги	Легкість тестування, чітке розділення відповідальностей	Простота реалізації	Двонаправлений зв'язок, легкість тестування
Недоліки	Можливе ускладнення Presenter	Controller може стати занадто великим і складним	Ускладнення при реалізації двонаправленого зв'язку

У моєму проєкті використовується патерн MVP. Model View Presenter — це патерн, заснований на патерні MVC. У MVP презентер такий самий, як контролер у патерні MVC. Проте є деяка відмінність: вхідною точкою в застосунок є представлення і дані не передаються безпосередньо з моделі до представлення, як у MVC. Презентер запитує модель, модель повертає дані презентеру, презентер обробляє отримані дані та передає їх у представлення. Презентер легко тестується, оскільки він не залежить від View, що дозволяє створювати більш якісні та надійні тести

Використовують MVP, коли програма має двонаправлений потік. Якщо при взаємодії з користувачем необхідно щось запитати з форми, і результат цього запиту негайно змінить інтерфейс користувача, слід використовувати патерн MVP. MVP забезпечує чітке розділення відповідальностей між компонентами, що сприяє підтримуваності та розширюваності кодуСаме тому для роботи над проєктом я обрала патерн MVP

2.2 Опис структури даних та моделі бази даних

База даних є ключовим компонентом мобільного застосунку, забезпечуючи його ефективність, надійність та зручність у використанні. База даних у мобільному застосунку необхідна для забезпечення багатьох важливих функцій і переваг, які значно покращують його роботу та користувацький досвід. Використання бази даних спрощує процес оновлення та редагування даних. Це дозволяє легко додавати нову інформацію, змінювати існуючу та видаляти застарілу.

Для того щоб коректно вносити дані слід продумати схему бази даних таблиці, та зв’язки між ними.

Проаналізувавши усі зібрані дані можна визначити сутності БД:

– Edition (міститиме інформацію про видання);
– User_info (міститиме інформацію про користувача);
– Subscriber (міститиме інформацію про передплатника);

– Subscription (міститиме інформацію про передплату);

Список таблиць, які є в базі даних зображений на рисунку 2.2.1.

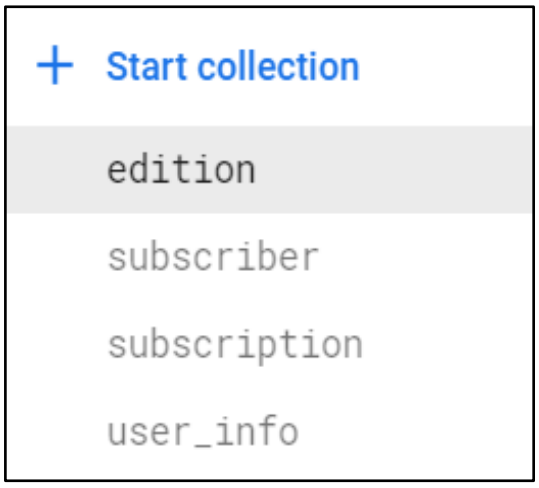


Рисунок 2.2.1 – Таблиці бази даних

Дані та їх типи, що містяться в таблицях представлені далі.

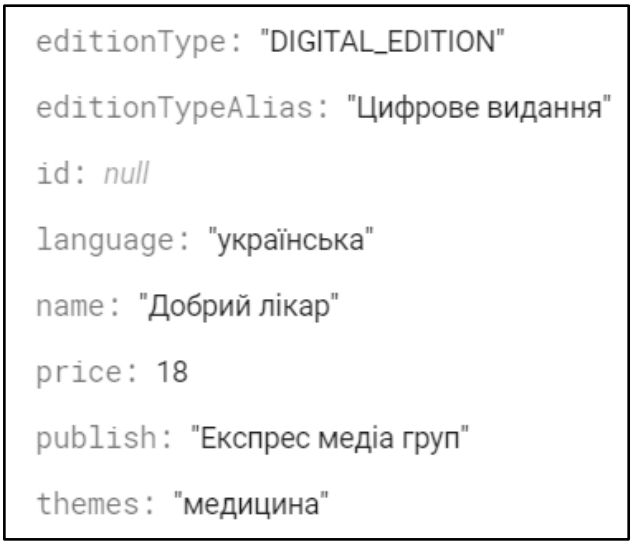


Рисунок 2.2.2 – Поля таблиці edition

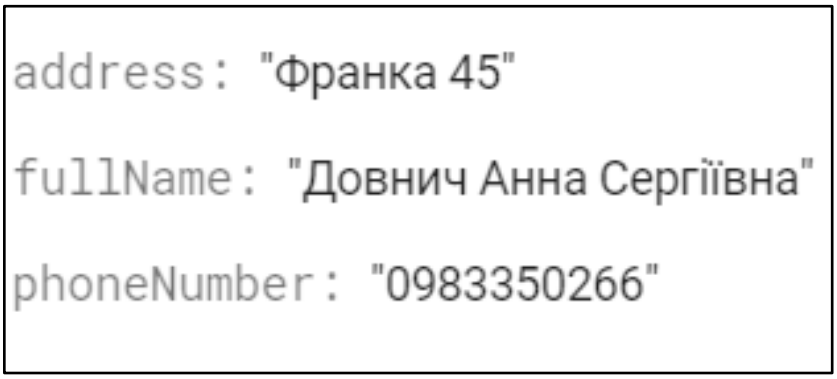


Рисунок 2.2.3– Поля таблиці subscriber

Рисунок 2.2.4 – Поля таблиці user_info

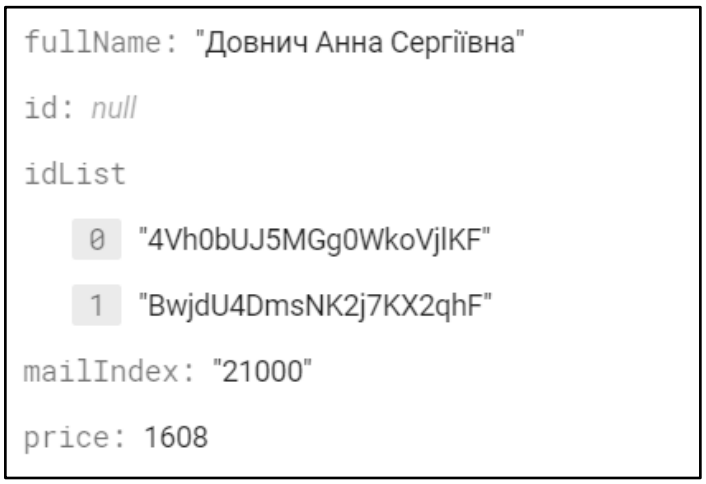


Рисунок 2.2.5 – Поля таблиці subscription

У висновку було створено базу даних зі згаданими вище таблицями і полями.

Після з’ясування зв’язків між таблицями та приведення бази даних до третьої нормальної форми було створено ER-діаграму бази даних (рисунок 2.2.6).

ER-діаграма (діаграма "сутність-зв'язок") є графічним представленням логічної структури бази даних. Вона показує сутності, які представляють таблиці, та зв'язки між ними.

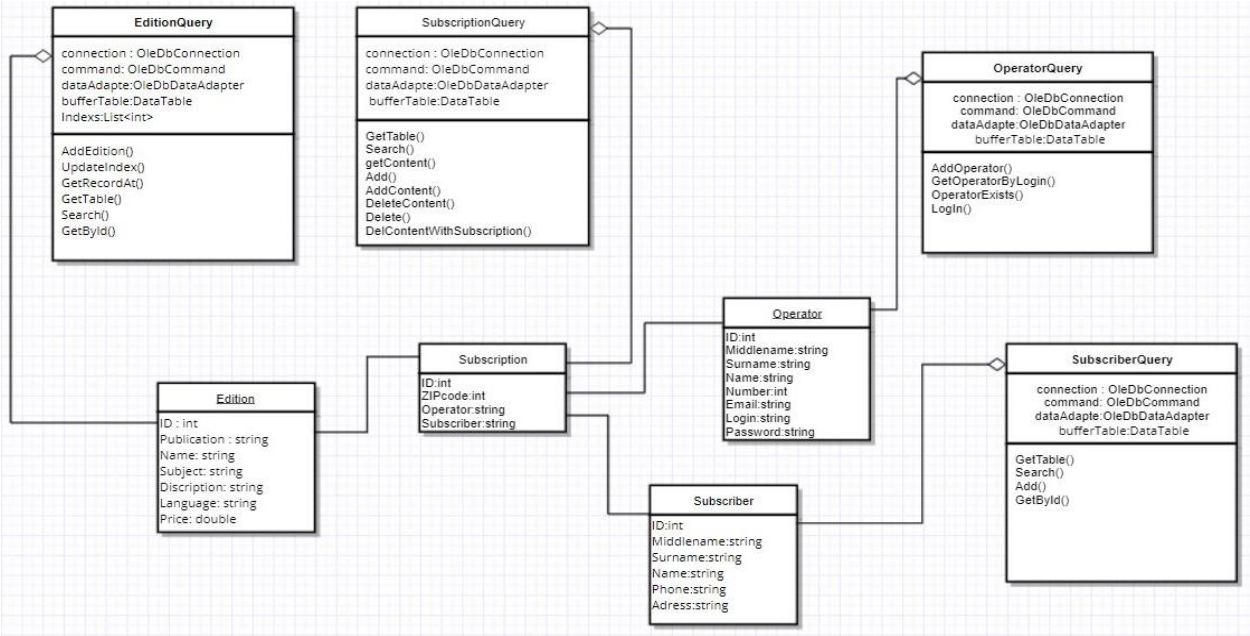


Рисунок 2.2.6 – ER-діаграма бази даних

2.3 Проєктування структур даних мобільного застосунку

Структура даних це всі ті змінні різних типів які використовуються в програмі. Існують локальні та глобальні змінні, якщо коротко то глобальні змінні – це ті, які можуть використовуватись у декількох функціях або методах, натомість локальні змінні створюються лише для методу в якому вони будуть використані.

Дані є найважливішою сутністю в інформатиці, а структури дозволяють зберігати їх в організованій формі. Яку б проблему не треба було вирішити, доводиться мати справу з даними – в незалежності чи це зарплата співробітника, ціни видань, список записів або навіть простий довідник.

Правильний підбір структур даних є надзвичайно важливим для ефективного функціонування відповідних алгоритмів їх обробки. Добре побудовані структури даних дозволяють оптимізувати використання машинного часу та пам'яті комп'ютера для виконання найкритичніших операцій.

У даному проєкті було реалізовано класи, а також їх поля, методи та об’єкти. Методи реалізовані у коді програми. Під час написання програми було використано такі принципи об’єктно-орієнтованого програмування:

– визначення атрибутів та методів класів;

– оголошення екземпляру класу;

– використання створених методів в основній програмі.

Коли взаємодії між проєктованою системою ПО і її оточенням визначені, ці дані можна використовувати як основу для розробки архітектури системи. При цьому необхідно застосовувати знання про загальні принципи проєктування системних архітектур і дані про конкретну предметну область.

Для створення нового класу потрібно вибрати пакет і натиснути ПКМ – Створити новий Java клас.

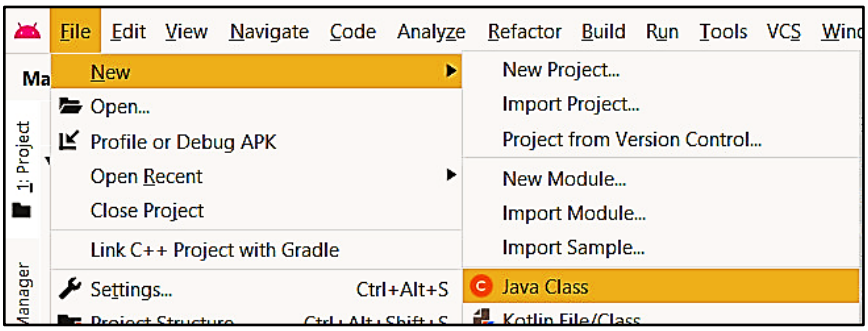


Рисунок 2.3.1– Створення нового класу

У програмі створено такі моделі: видавництво, передплатник, передплата та користувач.

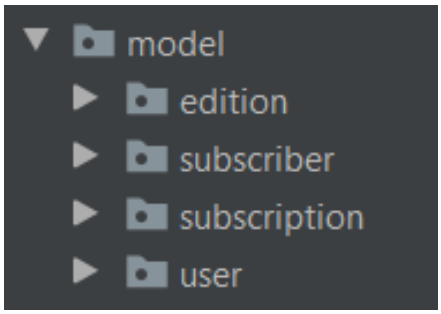


Рисунок 2.3.2 – Моделі проекту

Далі продемонстровано зразки класів, які містить програма та їхню структуру. Наприклад, даний клас відповідає за модель видавництва.

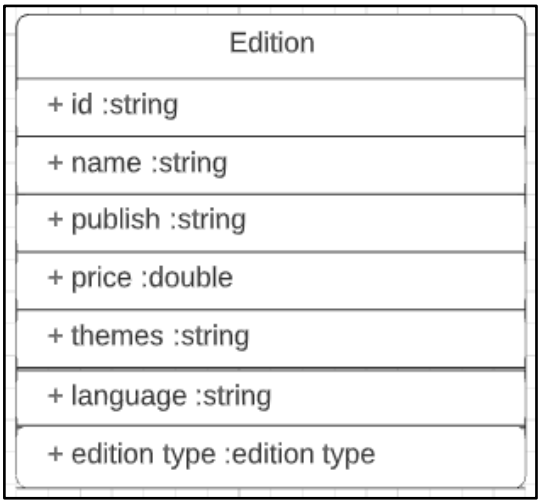


Рисунок 2.3.3 – Структура моделі Edition

Дана модель (рисунок 2.3.3) відповідає за організацію даних видання. При створенні нового видання користувач повинен ввести такі поля, як: назва, ціна, тема, мова, видавництво. Також потрібно вибрати тип видання з двох наявних. Усі видання після збереження формують список, що називається каталогом і знаходиться у відповідній вкладці програми.

Приклад реалізації моделі Edition.java в програмі:

```
public class Edition
 {
   private String id;
   private String name;
   private String publish;
   private Double price;
   private String themes;
   private String language;

   public String getId() {       return id;    }
   public void setId(String id) {       this.id = id;    }

   public String getName() {       return name;    }
   public void setName(String name) {       this.name = name;    }

   public String getPublish() {       return publish;    }
   public void setPublish(String publish) {       this.publish = publish;    }
```

```
  public Double getPrice() {       return price;    }
  public void setPrice(Double price) {       this.price = price;    }

  public String getThemes() {       return themes;    }
  public void setThemes(String themes) {       this.themes = themes;    }

  public String getLanguage() {       return language;    }
  public void setLanguage(String language) {this.language = language;    }

  public Edition(String name, String publish, Double price, String themes, String language)
{
    this.name = name;
    this.publish = publish;
    this.price = price;
    this.themes = themes;
    this.language = language;
  }}
```

Рисунок 2.3.4 – Структура моделі Subscription

Дана структура (рисунок 2.3.4) описує модель передплати, в ній знаходяться такі поля, як: ПІБ передплатника, поштовий індекс, початок та кінець терміну передплати, список видань, які передплатник хоче передплатити та ціна замовлення.

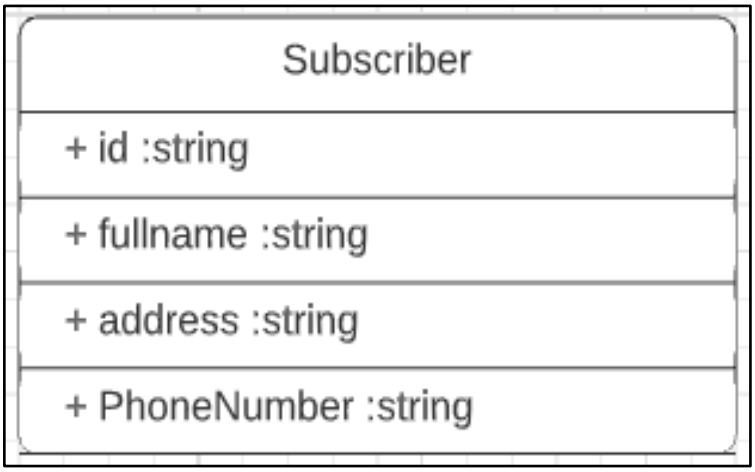


Рисунок 2.3.5 – Структура моделі Subscriber

Дана модель (рисунок 2.3.5) містить інформацію про передплатника, а саме: ПІБ, адресу і номер телефону. Дана структура призначена для оперування інформацією про передплатника, при оформленні передплати користувач обирає передплатника за іменем.

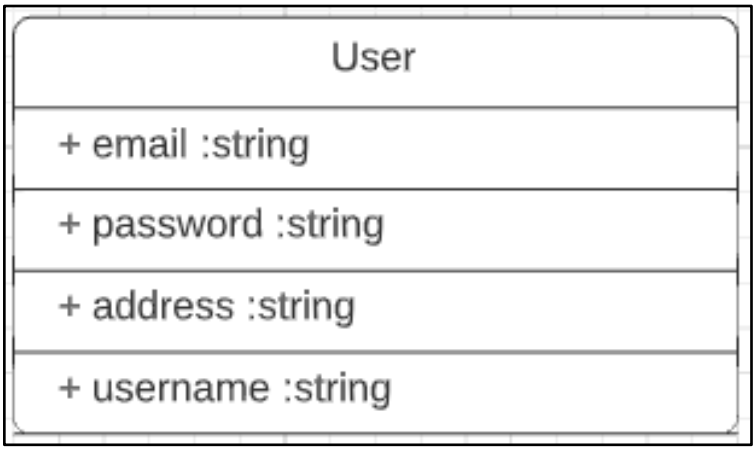


Рисунок 2.3.6 – Структура моделі User

Структура (рисунок 2.3.6) містить у собі інформацію про користувача, тобто про оператора поштового відділення. Має такі поля, як : електронна пошта, пароль до неї, номер відділення, ПІБ.

Для відображення функцій програми для користувача доцільно використати однойменну діаграму діяльності або по-іншому активності.

Діаграма активності дозволяє моделювати послідовності процесів або дій, реалізованих методами класів. Зазначені послідовності можуть являти

собою альтернативні галузей процесу обробки даних або галузям, які можуть виконуватися паралельно. Діаграми діяльності є аналогом блок-схеми будь-якого алгоритму.

Під час запуску програми виконується підключення до бази даних. Одразу після запуску програми з’являється вікно, де користувач може або зареєструватись, або авторизуватись. Наступною дією користувача є створення передплати та введення всіх необхідних даних. В залежності від коректності введених даних, користувач може або зберегти передплату, або буде необхідно знову ввести такі дані, щоб вони були коректні.

Все це детально описано у діаграмі активності, що зображена на рисунку 2.3.7

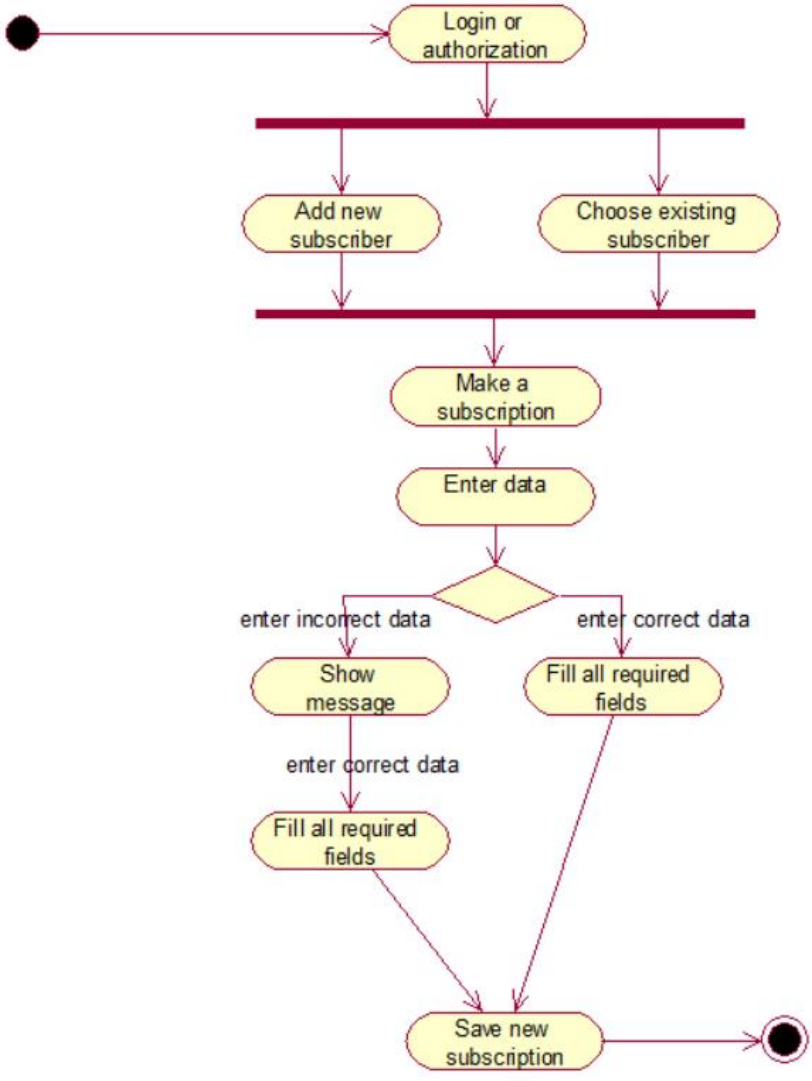


Рисунок 2.3.7 – Діаграма активності

Також для опису взаємодій доцільно створити діаграму кооперації. Головна особливість діаграми кооперації полягає в тому, що на ній графічно зображено не тільки послідовність взаємодії, але і всі структурні відношення між об'єктами, що беруть участь в цій взаємодії. За допомогою діаграми кооперації можна описати повний контекст взаємодій, як своєрідний часовий "зріз" сукупності об'єктів, взаємодіючих між собою для виконання певного завдання програмної системи. У вигляді прямокутників зображено об'єкти, що беруть участь у взаємодії, вказано асоціації між об'єктами у вигляді різних з'єднувальних ліній. Дані взаємозв'язки зображено на діаграмі кооперації.

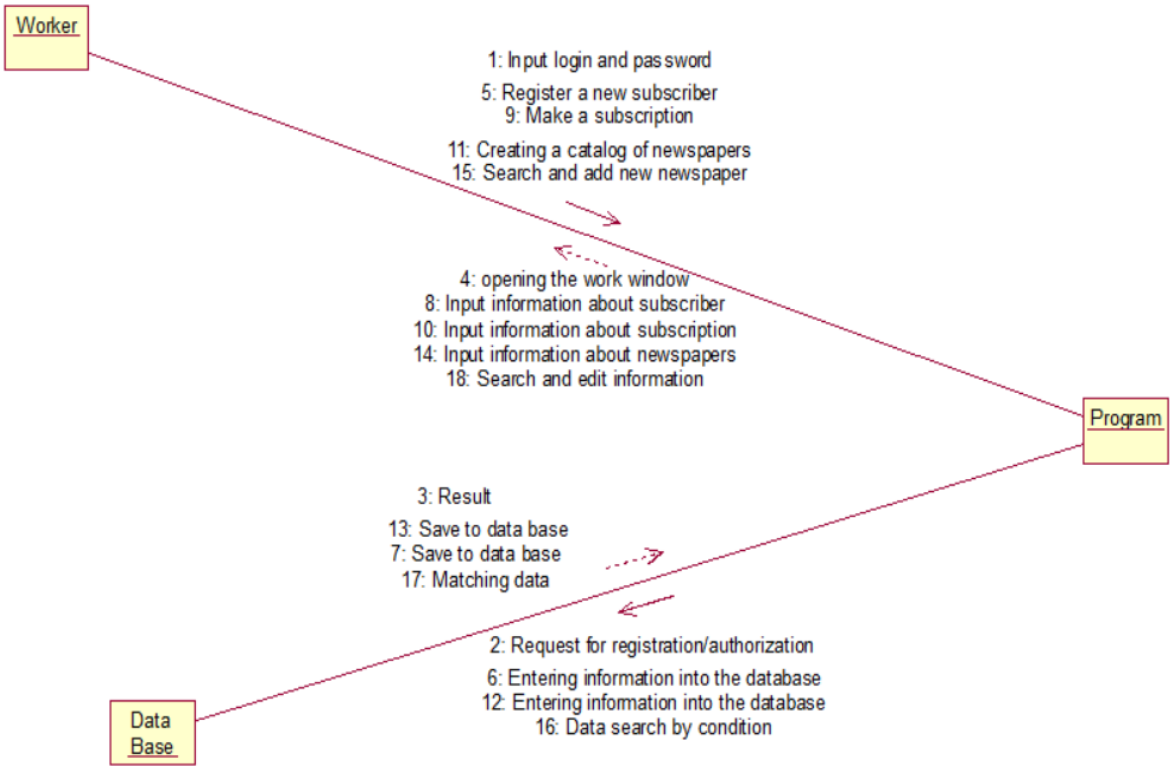


Рисунок 2.3.8 – Діаграма кооперації

Для того, щоб зрозуміти як функціонуватиме програма, необхідно розглянути діаграму послідовності. Діаграма послідовності відображає як елементи (користувачі, об'єкти, модулі) програми обмінюються між собою даними. Тобто як користувач вводить дані, як і де вони перевіряються, як дані опрацьовуються для досягнення певного результату.

У застосунку користувач взаємодіє з програмою, а програма з базою даних. Усі дані, що користувач вводить при авторизації, звіряються з даними, наявними в базі даних. Від результатів цієї перевірки залежать подальші дії. Усі дані, що користувач вводить в відведених полях для додавання нової інформації, зберігаються в базу даних. Дану взаємодію описано у діаграмі послідовності, що зображена на рисунку 2.3.9.

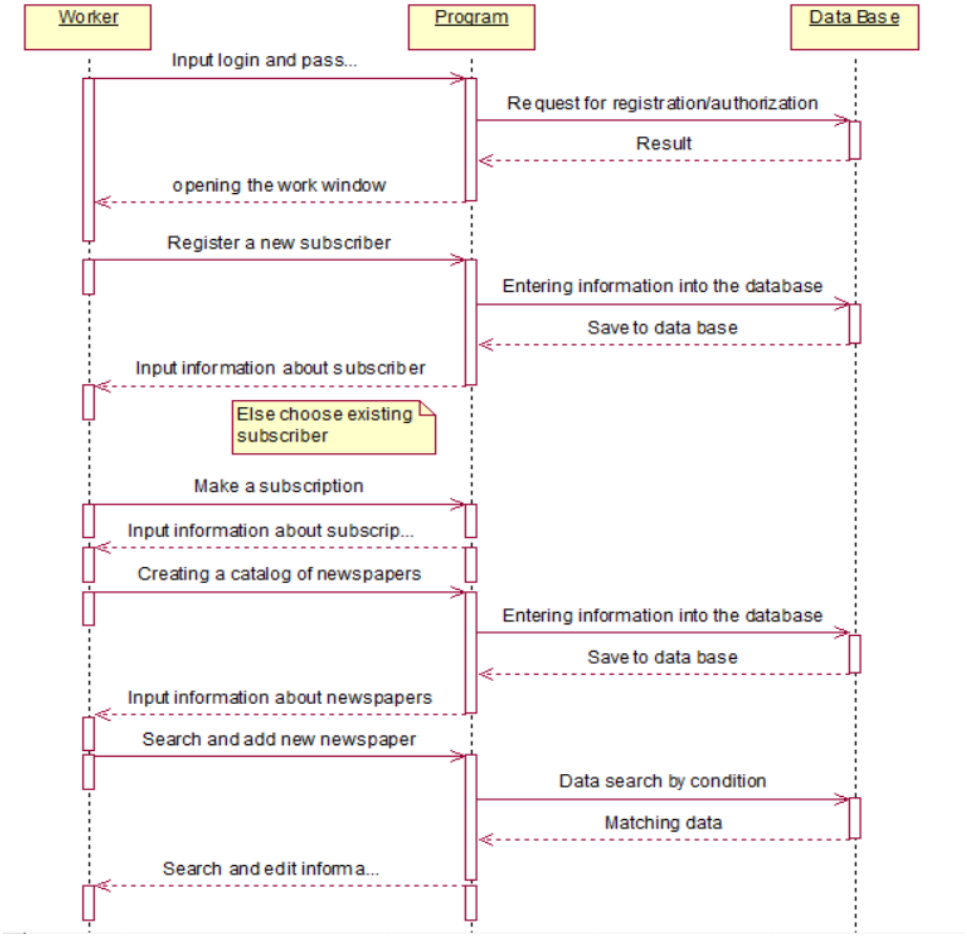


Рисунок 2.3.9 – Діаграма послідовності

Діаграма станів взаємно пов’язує події та стани. У разі приймання події наступний стан системи залежить як від поточного стану, так і від події. Зміна стану називається переходом.

Перше, що можна побачити – це вхідна та вихідна точки. Після запуску програма очікує на дії користувача. Під час отримання аргументів програма переходить в стан обробки запиту. Якщо користувач обирає операцію входу

чи реєстрацію – програма переходить в стан авторизації та очікує відповіді від бази даних, яка в свою чергу віддає результат запиту та повертає програму в стан обробки запиту.

Далі користувач реєструє нового передплатника і цієї дією програма знову повертається в стан реєстрації, проте цього разу вже передплатника, та очікує на обробку та збереження інформації в базі даних. Після реєстрації нового передплатника, ввівши всю необхідну інформацію про нього та після збереження даної інформації, користувач продовжує роботу і переходить в режим створення нової угоди про передплату. Далі оператор вводить усі необхідні дані для передплати і завершує роботу, переводячи програму в режим збереження та оновлення інформації в базі даних. Все це детально зображено на діаграмі станів, що представлена на рисунку

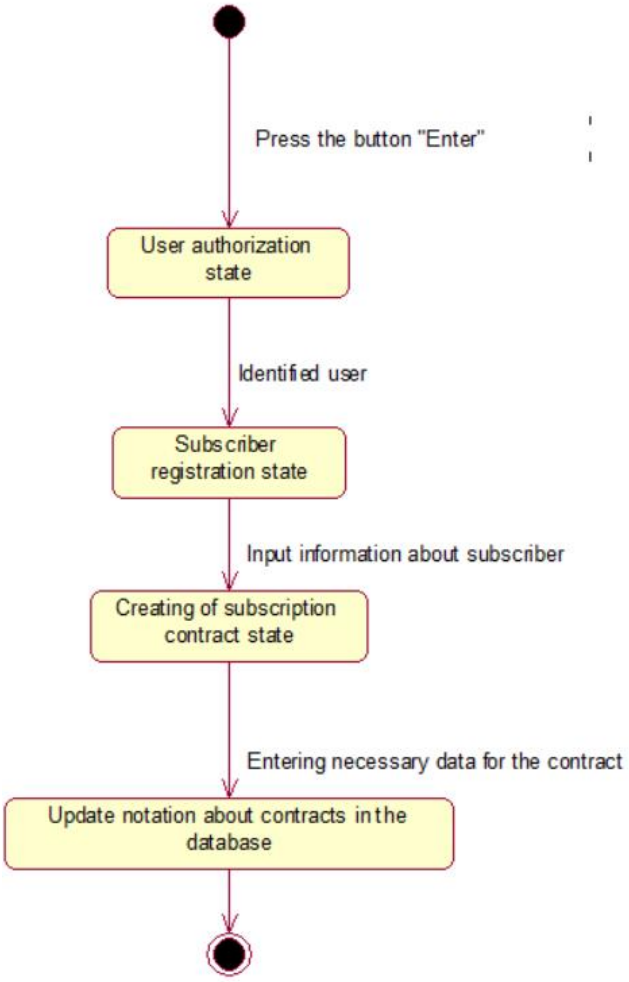


Рисунок 2.3.9 – Діаграма станів

Отже, було продемонстровано механізм-модельзастосунку, який наочно дає зрозуміти логіку його роботи , перетворення і обробку інформації, хід роботи користувача, який є передбачений застосунком.

2.4 Проєктування інтерфейсу мобільного застосунку

Інтерфейс користувача — засіб зручної взаємодії з інформаційною системою. Сукупність засобів для обробки та відбиття інформації, якнайбільше пристосованих для зручності користувача; у графічних системах інтерфейс користувача, втілюється багатовіконним режимом, змінами кольору, розміру, видимості вікон, їх розташуванням, гнучкими налаштуваннями як самих вікон, так і окремих їх елементів (файли, ярлики, шрифти тощо).

Для відображення форм використано метод, коли всі форми відкриваються в різних вікнах, такий спосіб вибрано тому що програма містить велику кількість полів введення та вибору.

Для проектування інтерфейсу форм потрібно спочатку їх уявити і зобразити графічно. Прикладом буде саме створення інтерфейсу деяких форм програми. Дана проекція інтерфейсу є однаковою для багатьох форм програми.

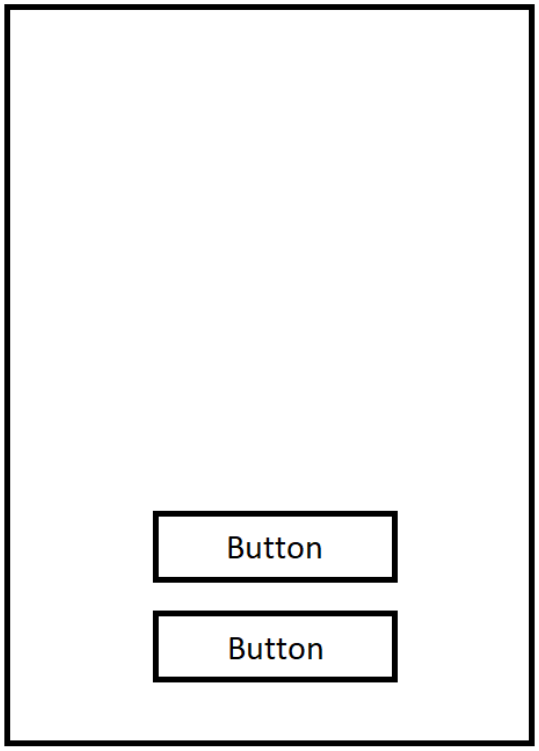


Рисунок 2.4.1 – Проекція інтерфейсу меню

Якщо натиснути на кнопку «Додати» в будь-якій з вкладок програми, з'явиться форма додавання з проекцією інтерфейсу, що зображена на рисунку 2.4.2. Інтерфейс дещо змінюється в залежності від вкладки, може бути більше полів введення або додаткові кнопки.

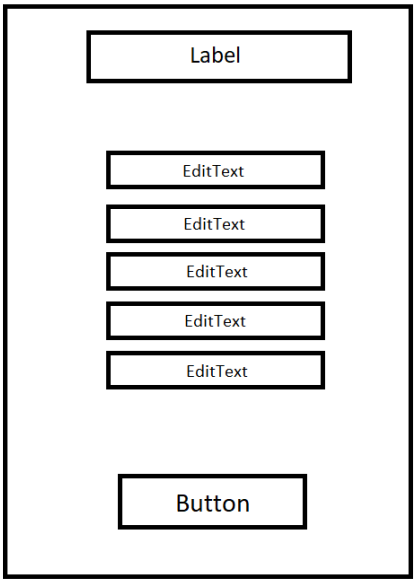


Рисунок 2.4.2– Проекція інтерфейсу форми додавання

Наприклад при додаванні нового видання з’являється додатковий елемент на формі – Toggle Buton (рисунок 2.4.3). Даний елемент використовується для того, щоб зазначити чи видання є цифровим чи друкованим.

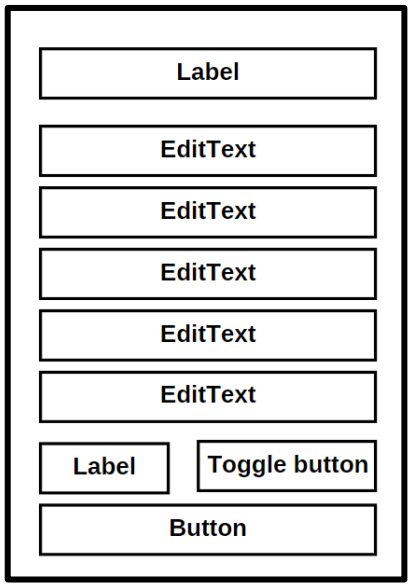


Рисунок 2.4.3– Проекція інтерфейсу форми додавання видання

При проектуванні інтерфейсу було взято до уваги особливості платформи під яку ведеться розробка, так як це мобільний застосунок потрібно зробити UI досить зрозумілим і щоб інтуїтивно можливо було розібратись у користуванні додатком. Також елементи UI зроблено досить великими, видимими і зручними, і зауважити те, що користувач на смартфоні може оперувати лише тапами і свайпами.

Отже, у цьому розділі продемонстровано структуру проекту, підбір типів даних, описано архітектуру системи, що розробляється, продемонстровано проекції інтерфейсу програми.

2.5 Аналіз та вибір технологій та методів реалізації мобільного застосунку

Оскільки варіантів реалізації програмного засобу є досить багато – це означає що і проєкт може бути написано абсолютно на різних мовах програмування. Це може бути Pascal, C++, C#, Java, Python та інші.

Для реалізації проєкту було вибрано об’єктно-орієнтовану мову програмування Java. Оскільки програмне забезпечення було призначено виключно на операційну систему Android то Java є найкращим варіантом мови програмування для реалізації цього проєкту. Java – це популярна мова програмування, яка з’явилася в 1995 році й наразі налічує дев’ять основних версій. Java стала технічним феноменом, що багато в чому пов’язано з її унікальною портативністю програми Java працюють на будь-якому пристрої або операційній системі. Середньостатистичному студенту, скоріш за все, викладають саме цю мову програмування: вона досить проста для розуміння базових принципів та водночас здатна вирішувати практично будь-які завдання розробки. Крім того, Java легко вчити завдяки синтаксису, який дозволяє ознайомитись з основами програмування за короткий час. Java-програмісти можуть розробляти застосунок на комп’ютері, а потім відкривати його на цільовій платформі – телефоні, сервері тощо. Саме з використанням цієї технології було створено програмний продукт.

Наразі найпопулярнішим середовищем програмування на Java у операційній системі Windows є Android Studio. Дане програмне середовище надає змогу використовувати нові багатофункціональні модулі для реалізації проєктів різної складності. Саме тому середовищем розробки програмного забезпечення було вибрано – Android Studio.

Це комплексне інтегроване середовище з широкими функціональними можливостями має удосконалений інтерфейс і містить нові інструменти для підтримки багатьох процесів. Розробники можуть створювати інноваційні і якісні додатки з привабливим інтерфейсом, які перевершать очікування

користувачів. Також це середовище дозволяє створювати програмне забезпечення не тільки на різних мовах програмування.

Для зберігання даних використано базу даних, створену за допомогою платформи Firebase. База даних – це інструмент, використовуючи який, можна збирати й упорядковувати інформацію. У базах даних можна зберігати відомості про людей, продукти, замовлення тощо.

Firebase – це платформа для розробки мобільних і веб-додатків, що дає змогу швидко створювати високоякісні додатки, залучати лояльних користувачів і збільшувати прибутки. До платформи включено комплекс інтегрованих функцій, які можна поєднувати й суміщати, зокрема мобільний сервер, засоби аналітики, а також інструменти вдосконалення й монетизації для максимальної успішності додатків.

Платформа Firebase має ряд переваг використання. Розглянемо основні з них. Першим і найвагомішим є безкоштовний початковий план, тобто користувачу не треба нічого платити, можливо ввійти в систему, використовуючи свій обліковий запис Google. Також важливою перевагою є швидкість розробки. Firebase надає величезну кількість готових до використання сервісів, які дозволяють розробникам уникнути створення шаблонного коду, «винаходу велосипеда» і написання бекенда з нуля.

Наскрізна платформа для розробки додатків є дуже зручним доповненням Firebase. Дана платформа повністю охоплює всі етапи розробки додатків, а також містить всі необхідні інструменти для створення, випуску та здійснення моніторингу додатків. Також Firebase надає інструменти для залучення і утримання проєктів у подальшому їх розвитку.

Досить зручним є те, що Firebase працює на платформі Google, внаслідок чого ідеально поєднується з іншими хмарними сервісами Google і інтегрується з багатьма сторонніми сервісами. Ще однією важливою перевагою є те, що дана платформа містить функцію моніторингу помилок. Firebase відслідковує як не критичні, так і фатальні помилки, всі звіти

генеруються на основі того, як помилки впливають на роботу користувачів. Також Firebase турбується про безпеку своїх користувачів та має функцію резервного копіювання, що унеможливлює втрату даних.

Інтеграція Firebase з Android Studio включає кілька основних кроків, які охоплюють налаштування проекту в Firebase, підключення Firebase SDK до Android проекту, а також налаштування та використання окремих служб Firebase.

Насамперед потрібно створити проєкт в Firebase:

– на сайті Firebase потрібно відкрити Firebase Console;

– натиснути кнопку "Створити проект";

– ввести назву проекту;

– прийняти умови використання та натиснути "Продовжити";

– вибрати налаштування Google Analytics, якщо необхідно, і завершити створення проекту.

Для того щоб підключити Android додаток до Firebase потрібно:

– у Firebase Console виберати необхідний проект:

– натиснути кнопку "Додати додаток" і виберати "Android";

– ввести назву пакету додатку (він повинен збігатися з назвою пакету в Android проекті);

– вказати "Псевдонім додатку" та "Сертифікат відбитка" (SHA-1) при необхідності;

– натиснути "Зареєструвати додаток".

Далі необхідно завантажити файл google-services.json:

– після реєстрації додатку завантажити конфігураційний файл google-services.json.

– помістити цей файл в каталог app Android проекту.

Звернувши увагу на всі переваги даної платформи, для зберігання даних було обрано Firebase.

2.6 Висновки до розділу 2

Отже, виконуючи роботу над розділом 2 було зроблено аналіз та вибір архітектури мобільного застосунку, детально описано багаторівневу архітектуру, виконано порівняння інших шаблонів проектування і обґрунтовано вибір шаблону MVP.

Було розглянуто схему БД, описано сутності бази даних та їх зв’язки, наведені структури таблиць та їх атрибути. Створено діаграму ERD (Entity-Relationship Diagram) для візуалізації зв’язків між таблицями. Описано використання об’єктно-орієнтованого підходу, зокрема класи, їх поля та методи. Наступним кроком було спроєктовано структуру моделей та інтерфейсу мобільного застосунку. Побудовано різні діаграми для демонстрації роботи мобільного застосунку. Заключним етапом стало проведення аналізу та вибір методів та технологій, за допомогою яких відбуватиметься реалізація мобільного застосунку. Описано переваги вибраних методів реалізації та процес їхньої інтеграції.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка бази даних мобільного застосунку

Як вже згадувалось раніше, для збереження даних використано платформу FireBase. Для того щоб створити базу даних в FireBase спочатку потрібно створити Firebase Console Project. Для цього треба перейти на головну сторінку консолі Firebase. Якщо використання Firebase Console вперше, буде запропоновано створити проект.

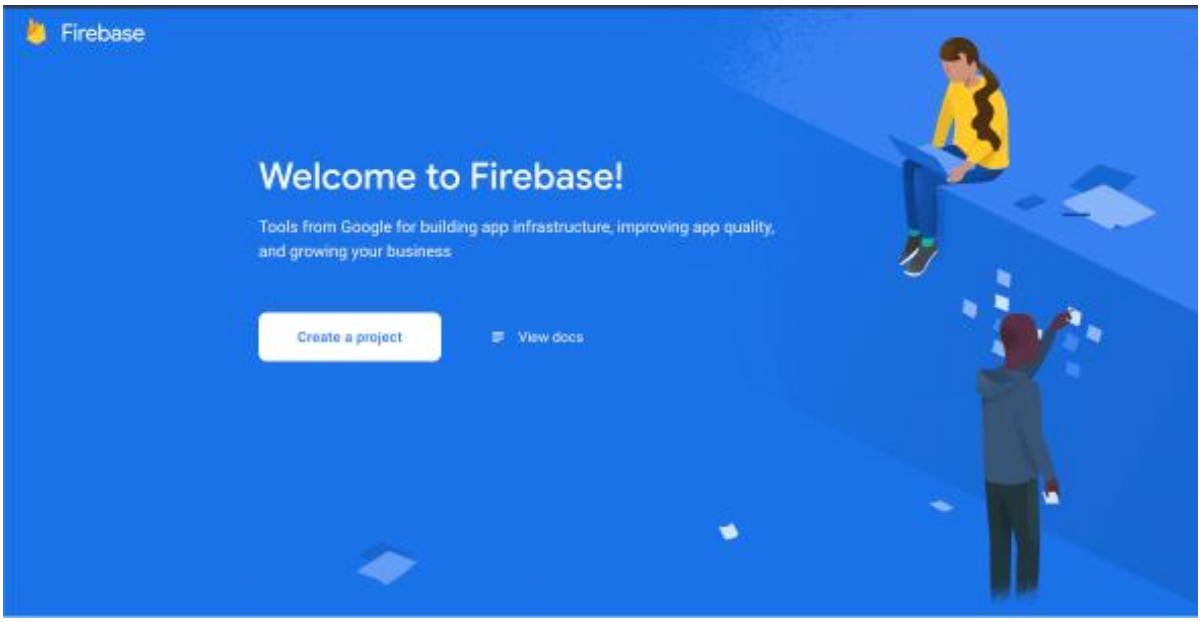


Рисунок 3.1.1 – Вітальна сторінка Firebase

Після натиснення кнопки «Create a project» з’явиться екран, де потрібно ввести назву проєкту. В даному випадку назва «MailApp».

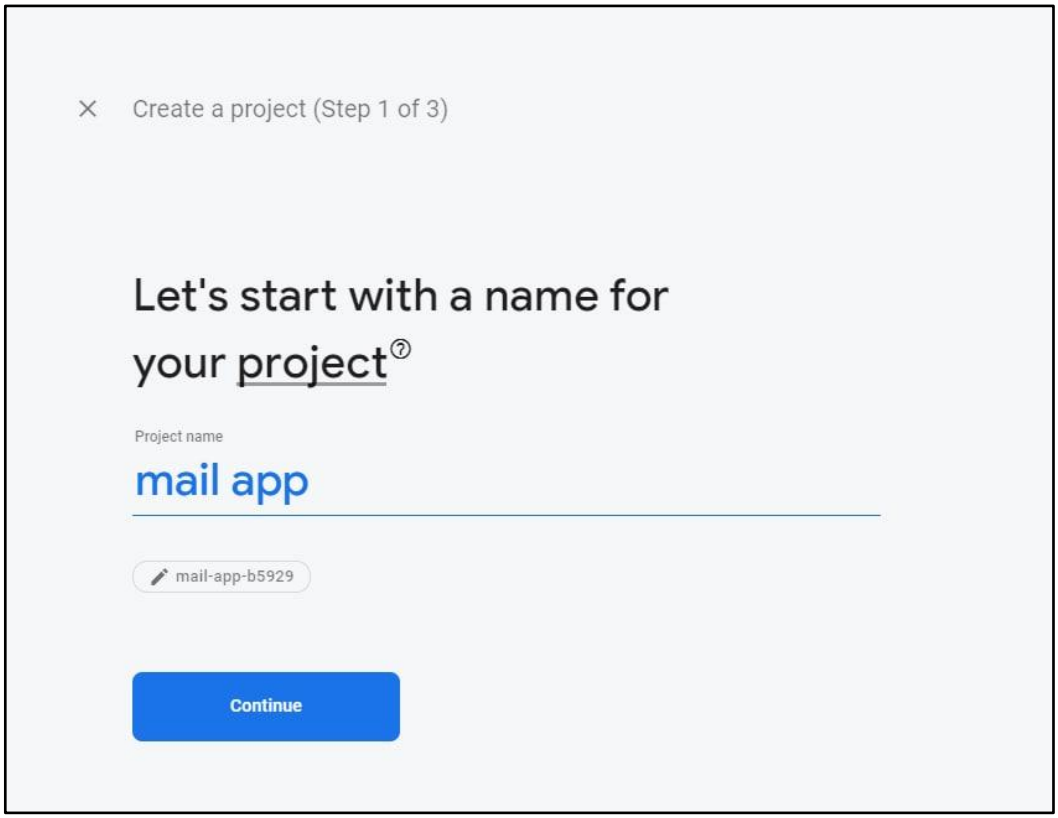


Рисунок 3.1.2 – Задання назви бази даних

Натиснувши кнопку «Continue» на наступному кроці буде запропоновано включити Google Analytics, але це необов'язково.

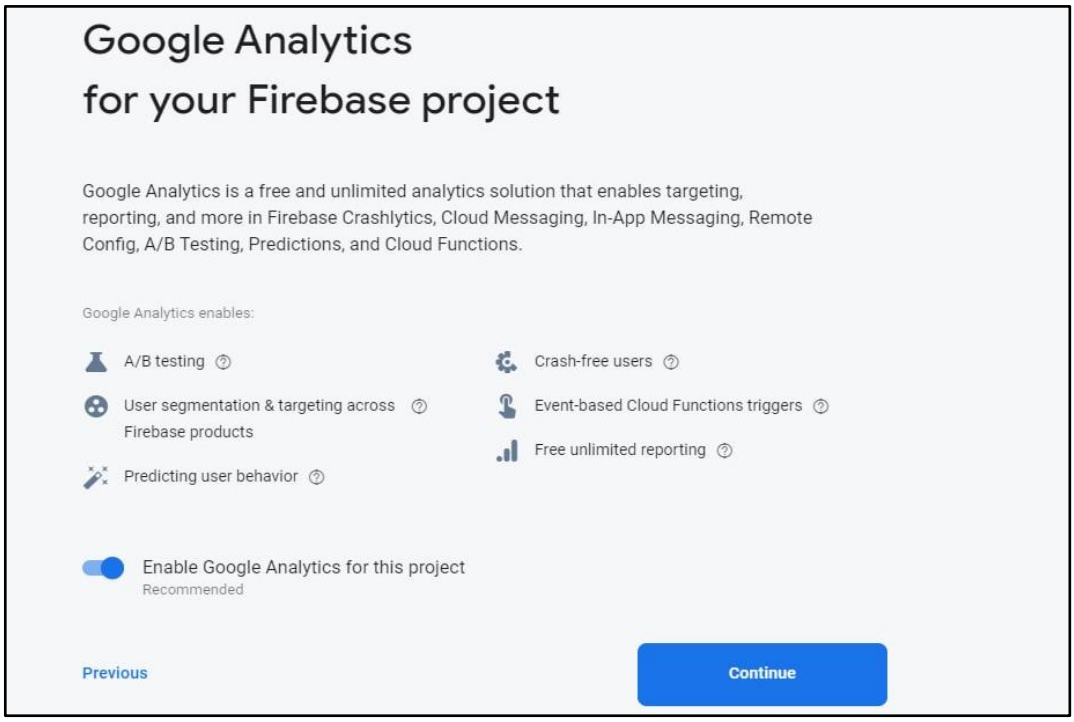


Рисунок 3.1.3– Пропозиція включити Google Analytics

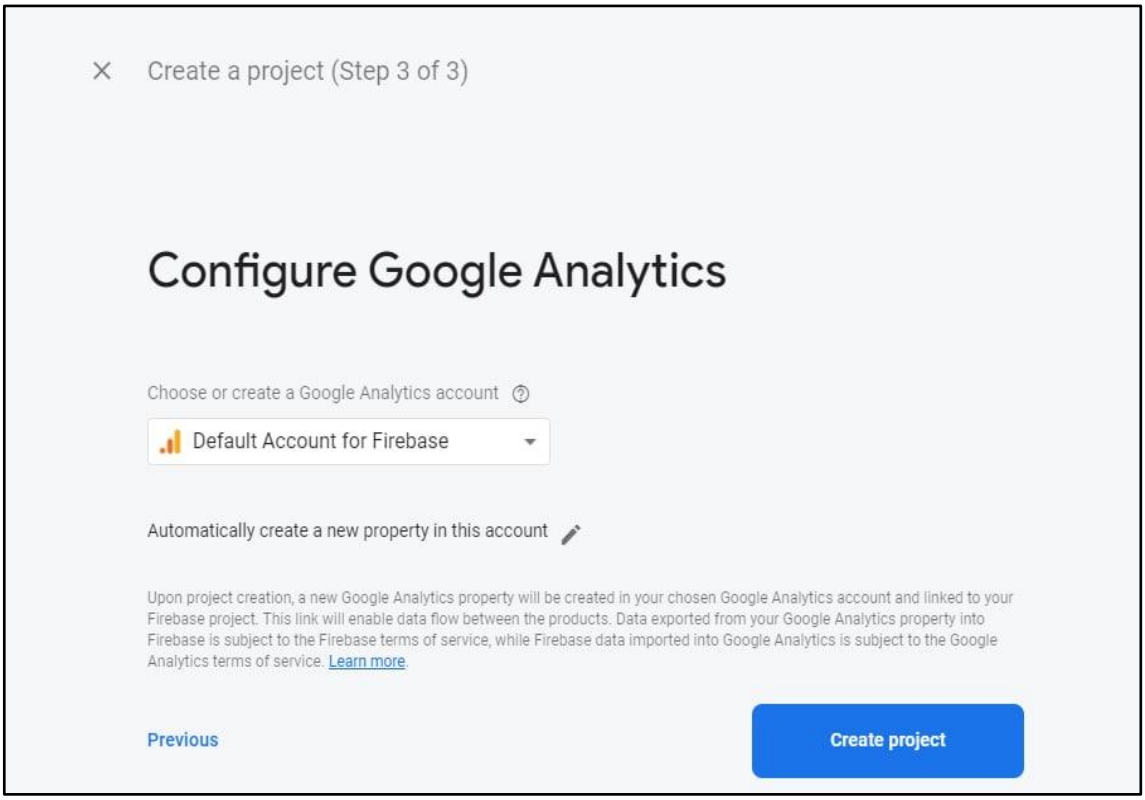


Рисунок 3.1.4 – Підключення Google Analytics

Натиснувши кнопку «Create project» з’явиться завантажувальний екран.

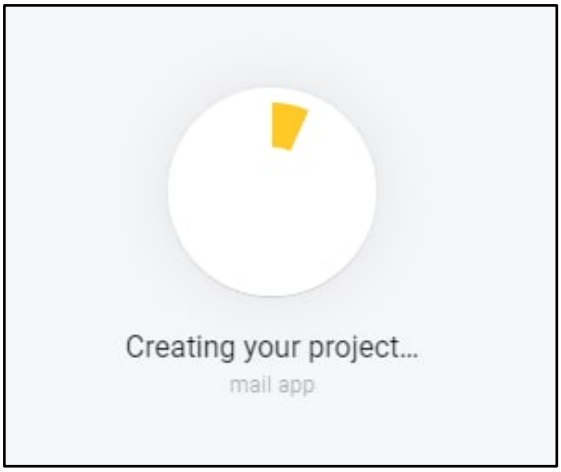


Рисунок 3.1.5 – Завантаження проєкту

Після того як завершиться завантаження з’явиться головний екран проєкту «MailApp». На головному екрані проекту потрібно натиснути на маленький білий значок Android, щоб запустити процес інтеграції.

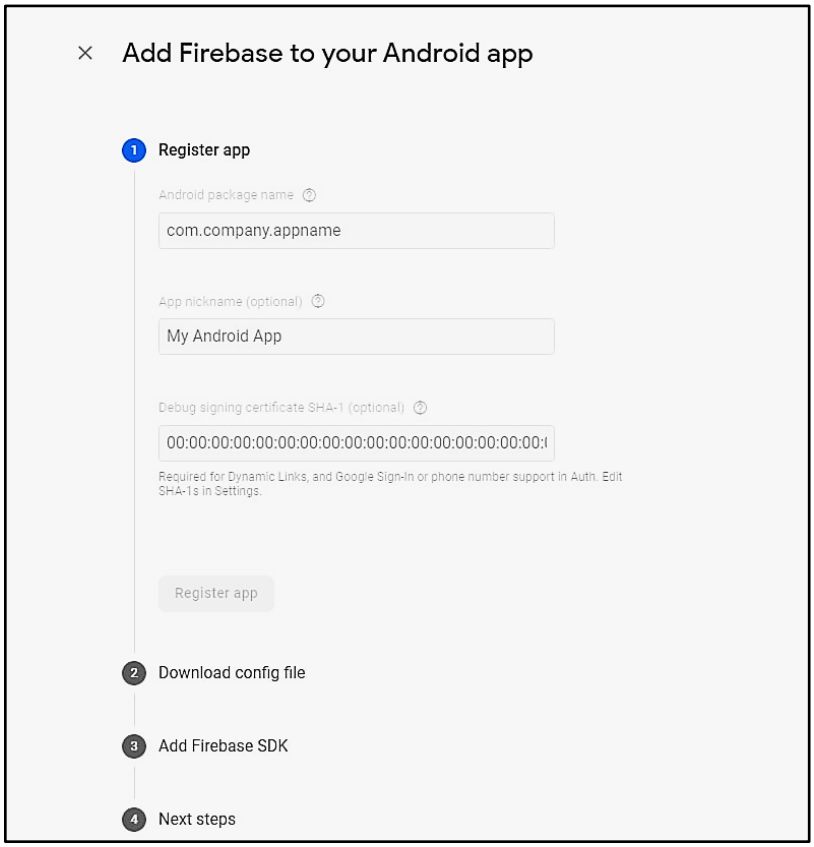


Рисунок 3.1.6 – Початок процесу інтеграції

Після введення всіх необхідних даних отримано JSON-файл google-services.json. На екрані є інструкція для його інтеграції в застосунок.

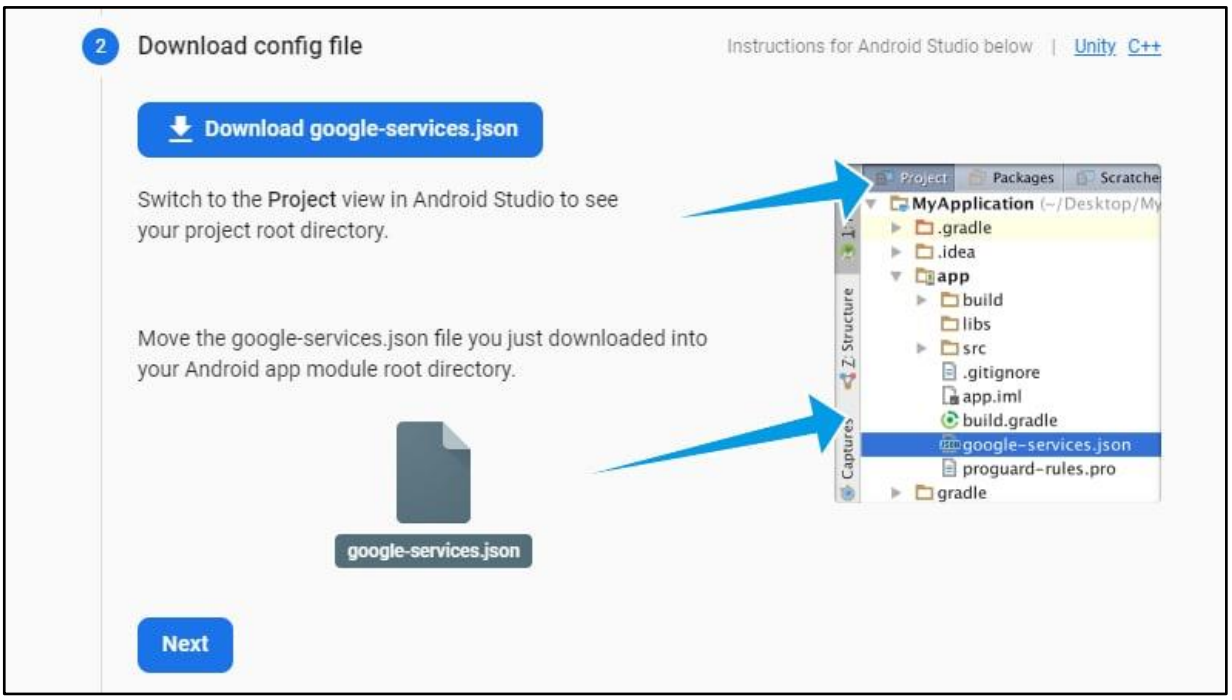


Рисунок 3.1.7 – Config file JSON для Firebase на Android

Далі в Android Studio потрібно відкрити Gradle Scripts > build.gradle (Project: MailApp) та додати цей код:

classpath "com.android.tools.build:gradle:4.1.2"

classpath 'com.google.gms:google-services:4.3.4'

У тій же папці потрібно відкрити Build.gradle (Module: MailApp.app) і додати наступний код:

```
apply plugin: 'com.android.application'
// Add this line
apply plugin: 'com.google.gms.google-services'

dependencies {
  // Import the Firebase BoM
  implementation platform('com.google.firebase:firebase-bom:26.7.0')

  // Add the dependency for the Firebase SDK for Google Analytics
  // When using the BoM, don't specify versions in Firebase dependencies
  implementation 'com.google.firebase:firebase-analytics'

  // Add the dependencies for any other desired Firebase products
  // https://firebase.google.com/docs/android/setup#available-libraries
}
```

Рисунок 3.1.8 – Код Build.gradle (Module: MailApp.app)

Після додавання проекту в Firebase Console і підключення до нього застосунку, потрібно налаштувати базу даних Firestore в консолі.

Натиснувши на кнопку «Create database» виведеться модальний екран. Спочатку він запитає про безпеку бази даних. Для розробки дипломного проєкту встановлено його в "production mode" для безпеки даних користувачів.

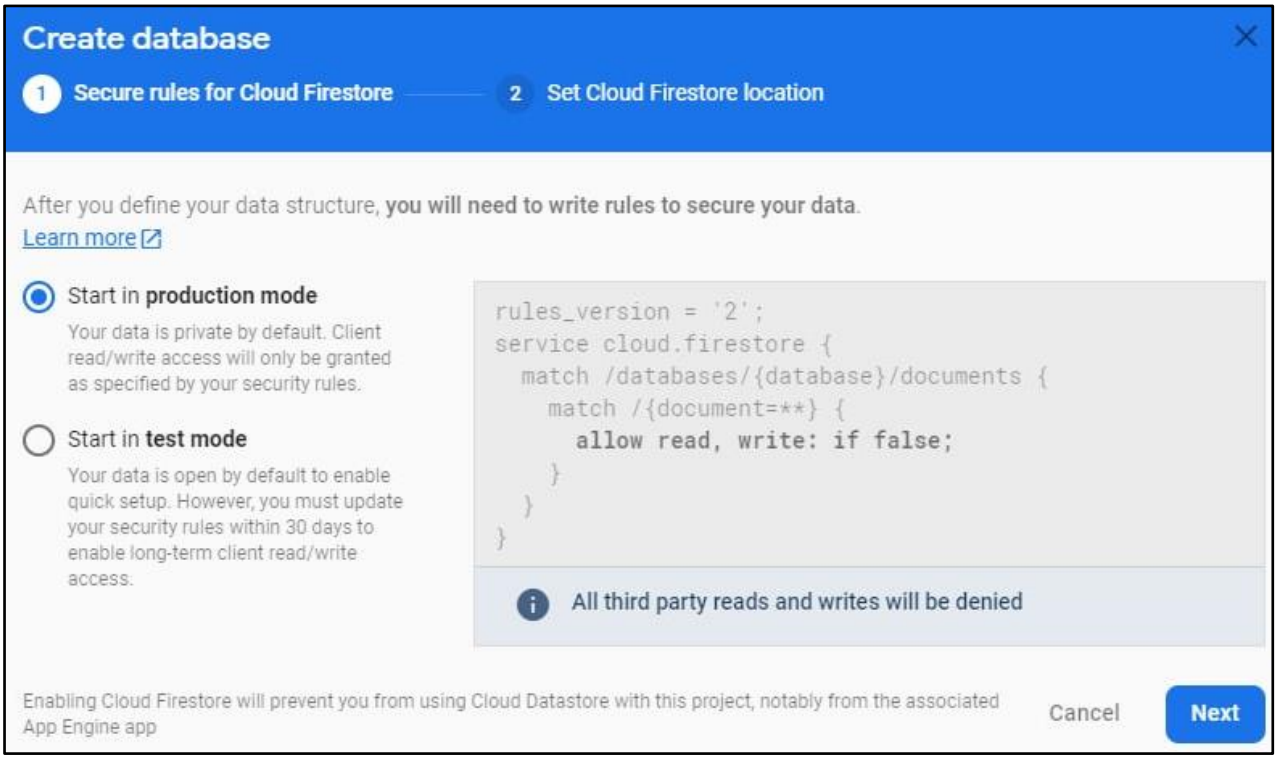


Рисунок 3.1.9 – Режими бази даних Firestore

Правила було змінено на такі:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write;
    }
  }
}
```

Рисунок 3.1.10 – Правила бази даних

Натиснувши кнопку Next, буде запропоновано вказати місце розташування сервера. Зазвичай рекомендується вибирати найбільш близьке до вас або ваших користувачів місце, так як це дозволить скоротити час завантаження. Консоль Firebase пропонує оптимальний варіант.

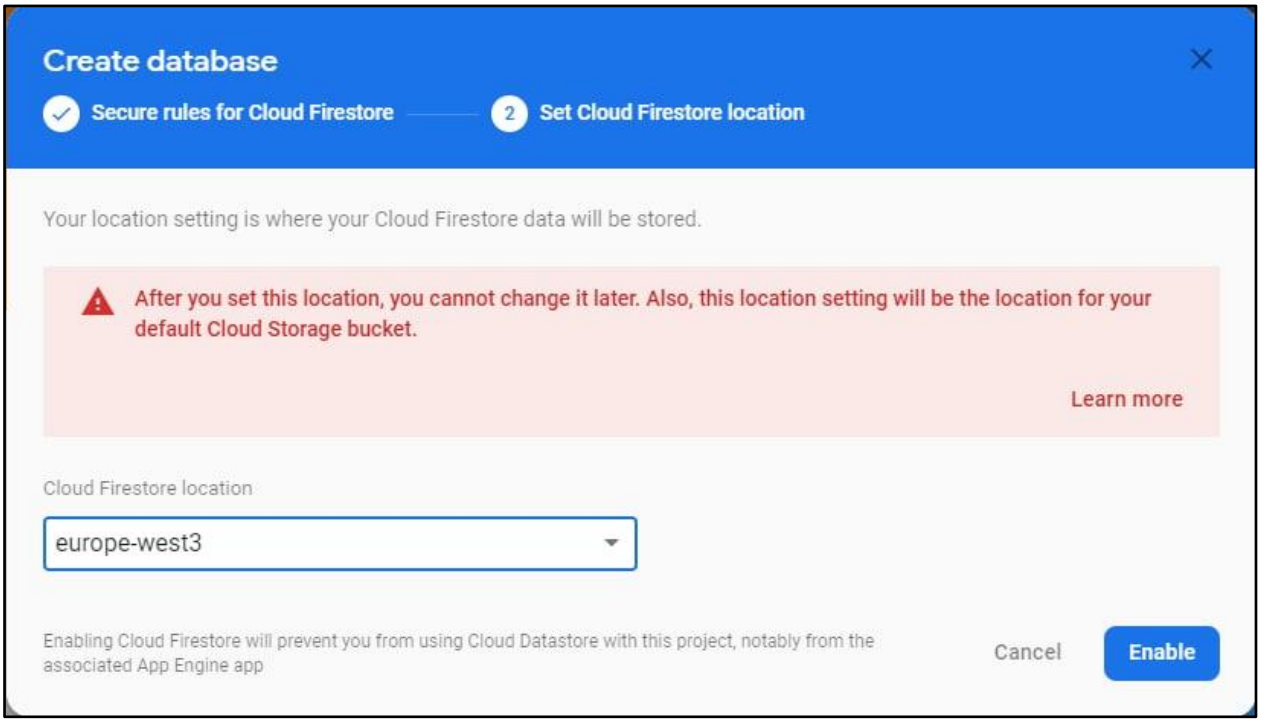


Рисунок 3.1.11 – Локація бази даних Firestore

Далі необхідно підключити Firestore до проєкту.

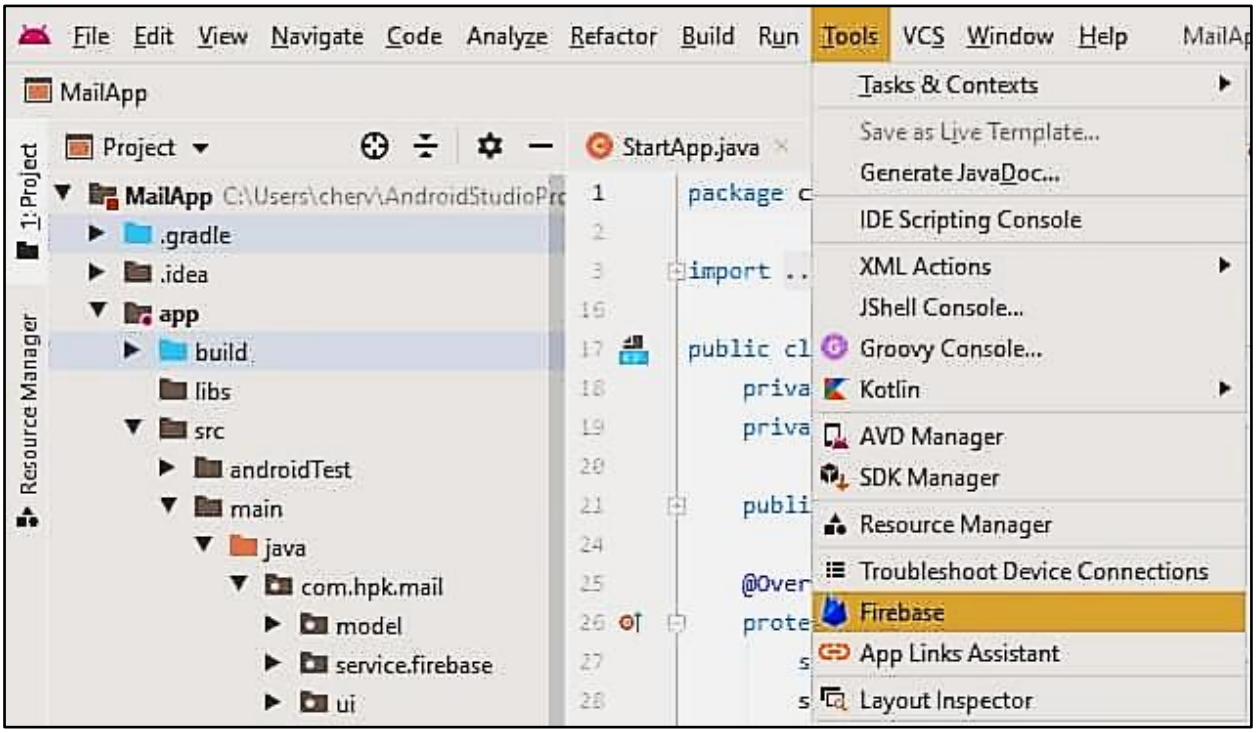


Рисунок 3.1.12 – Підключення бази даних до проєкту

Для демонстрації роботи з базою даних використано даний клас. Він відповідає за авторизацію користувача в програмі та зв’язаний з базою даних.

```
public class Auth {
private static final String COLLECTION_NAME = "user_info";

private final FirebaseAuth firebaseInstance = FirebaseAuth.getInstance();
private final FirebaseFirestore db = FirebaseFirestore.getInstance();

public void signUp(User userModel, Context context) {
firebaseInstance.createUserWithEmailAndPassword(userModel.getEmail(),
userModel.getPassword()).addOnSuccessListener(result -> {
    try {
        String userID = result.getUser().getUid();
UserInfo userInfo = new UserInfo(userModel.getUsername(),userModel.getAddress());

db.collection(COLLECTION_NAME).document(userID).set(userInfo).addOnSuccessListener(res -> {
CurrentUser.getNewInstance(userModel.getEmail(), userModel.getUsername(),
userModel.getAddress(), userID);

        StartApp.getCurrentInstance().toAppActivity();    });
    } catch (Exception exc) {
            Log.d("exception", exc.getMessage());  }    }
     ).addOnFailureListener(result -> {
            Log.d("error", result.getMessage());
            Toast.makeText(context, "Помилка при створені користувача",
            Toast.LENGTH_LONG).show();
            StartApp startApp = StartApp.getCurrentInstance();
            startApp.onFailSignUp();
        }     );
 }  }
```

Отже, створено консоль Firebase для застосунку на Android, який налаштований на використання Firestore з базою даних. База даних підключена до проєкту і використовується у класах для передавання даних з застосунку в базу.

3.2 Розробка програмних модулів мобільного застосунку

У цьому розділі описується загальна структура проєкту, розроблені програмні модулі, демонструється уривки коду програми, що відповідають кожному модулю програми.

На рисунку 3.2.1 наведена загальна структура проєкту.

Рисунок 3.2.1 – Структура проєкту

Було обрано технологію збірки проєкту Gradle. Gradle — система автоматичного збирання, розвинена на принципах, Apache Ant та Apache Maven. Для визначення порядку виконання завдань Gradle використовує орієнтований ациклічний граф ("DAG"). Gradle було розроблено для побудови мультипроєктів, які можуть розростатися, і підтримує інкрементальне збирання. Вона визначає, які частини було змінено, і виконує

тільки ті задачі, які залежать від цих частин. Основні плагіни призначені для розробки і розгортання Java, Groovy і Scala додатків, але готуються плагіни і для інших мов програмування.

Вміст файлу build.gradle зображено на рисунку 3.2.2.

Рисунок 3.2.2 – Build.gradle

В даному файлі прописані шляхи до всіх зовнішніх бібліотек та плагінів потрібних для роботи.

Форма реєстрації та авторизації є першими з якими взаємодіє користувач застосунку. На рисунку 3.2.3 представлено код реалізації реєстрації нового користувача в програмі.

```
private void onSignUpBtnClick(View v) {
    v.setVisibility(View.INVISIBLE);
    findViewById(R.id.progressBar_sign_up).setVisibility(View.VISIBLE);
    EditText passwordField = findViewById(R.id.password_sign_up);
    EditText confirmPasswordField = findViewById(R.id.confirm_password_sign_up);
    EditText emailField = findViewById(R.id.email_sign_up);
    EditText addressField = findViewById(R.id.address_sign_up);
    EditText usernameField = findViewById(R.id.username_sign_up);

    String email = emailField.getText().toString();
    String password = passwordField.getText().toString();
    String confirmPassword = confirmPasswordField.getText().toString();
    String address = addressField.getText().toString();
    String username = usernameField.getText().toString();

    if (!(isValidEmail(email) || password.isEmpty() || password.equals(confirmPassword)) || address.isEmpty() || username.isEmpty()) {
        new AlertDialog.Builder(this).setTitle("Помилка").setMessage("Введіть корректно дані").setPositiveButton("OK", null).show();
        onFailSignUp();
        return;
    }

    User user = new User(email, password, username, address);

    Auth auth = new Auth();

    auth.signUp(user, this);
}
```

Рисунок 3.2.3 – Уривок коду модулю реєстрації

Якщо користувач вже зареєстрований в програмі, йому необхідно авторизуватись. На рисунку 3.2.4 представлено код реалізації авторизації користувача в програмі.

```
private void onSignInBtnClick(View v) {
    v.setVisibility(View.INVISIBLE);
    findViewById(R.id.progressBar_sign_in).setVisibility(View.VISIBLE);

    EditText passwordField = findViewById(R.id.password_sign_in);
    EditText emailField = findViewById(R.id.email_sign_in);

    String email = emailField.getText().toString();
    String password = passwordField.getText().toString();

    if (!(isValidEmail(email) || !password.isEmpty())) {
        new AlertDialog.Builder(this).setTitle("Помилка").setMessage("Введіть корректно дані").setPositiveButton("OK", null).show();
        onFailSignIn();
        return;
    }

    User user = new User(email, password);
    Auth auth = new Auth();

    auth.signIn(user, this);
}
```

Рисунок 3.2.4 – Уривок коду модулю авторизації

Для застосунку, який використовується для обліку передплат у поштовому відділенні, наявність списку видань для передплати є основою програми. Тому створення нового запису про видання є одним із найважливіших модулів програми. На рисунку 3.2.5 зображено уривок коду, що відповідає за створення нового видання.

```
public EditionType getEditionType() {
    return editionType;
}

public String getEditionTypeAlias() {
    return editionType == EditionType.DIGITAL_EDITION ? "Цифрове видання" : "Друковане видання";
}

public void setEditionType(EditionType editionType) {
    this.editionType = editionType;
}

public String getId() {
    return id;
}

public void setId(String id) {
    this.id = id;
}

public String getName() {
    return name;
}
```

Рисунок 3.2.5 – Уривок коду створення нового видання

Під час користування застосунком, можлива ситуація, коли потрібно змінити опис, ціну чи тип видання. На рисунку 3.2.6 продемонстровано реалізацію коду, що відповідає за редагування запису про видання.

```
try {
    name = nameEditText.getText().toString();
    language = languageEditText.getText().toString();
    price = Double.parseDouble(priceEditText.getText().toString());
    themes = themesEditText.getText().toString();
    publish = publishEditText.getText().toString();
    isDigital = isDigitalSwitch.isChecked();
} catch (Exception e) {
    Toast.makeText(this, "Введіть всі дані", Toast.LENGTH_LONG).show();
    findViewById(R.id.add_edition_btn).setVisibility(View.VISIBLE);
    return;
}

if (name.isEmpty() || price == 0 || themes.isEmpty() || language.isEmpty() || publish.isEmpty()) {
    Toast.makeText(this, "Введіть всі дані", Toast.LENGTH_LONG).show();
    findViewById(R.id.add_edition_btn).setVisibility(View.VISIBLE);
    return;
}

Edition edition = new Edition(name, publish, price, themes, language, isDigital ? EditionType.DIGITAL_EDITION : EditionType.PRINT_EDITION);
storage.editData("edition", EditEdition.edition.getId(), edition);
finish();
}
```

Рисунок 3.2.6 – Уривок коду редагування інформації про видання

Основним модулем застосунку звичайно є створення передплати. На рисунку 3.2.7 зображено франгмент коду, що відповідає за створення запису про передлату. Завдяки цьому модулю користувач може створити передплату, попередньо обравши передплатника, видання та його кількість, та період передплати.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.fragment_add_subscription);
    instance = this;
    findViewById(R.id.add_subscription_picker).setVisibility(View.INVISIBLE);
    findViewById(R.id.subscription_add_edition).setOnClickListener(this::toAddEdition);
    findViewById(R.id.subscription_add_data).setOnClickListener(this::addData);
    findViewById(R.id.subscription_edit_edition_list).setOnClickListener(this::toEditEdition);
    Storage storage = new Storage();
    storage.getData("subscriber", new AddSubscriptionActions());
    editions = new ArrayList<>();
    ((EditText) findViewById(R.id.start)).setOnEditorActionListener(
            new TextView.OnEditorActionListener() {
                @Override
                public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
                    if (actionId == EditorInfo.IME_ACTION_SEARCH ||
                            actionId == EditorInfo.IME_ACTION_DONE ||
                            event != null &&
                                    event.getAction() == KeyEvent.ACTION_DOWN &&
                                    event.getKeyCode() == KeyEvent.KEYCODE_ENTER) {
                        if (event == null || !event.isShiftPressed()) {
                            // the user is done typing.
                            return true; // consume.
                        }
                    }
                    calculatePrice();
                    return false; // pass on to other listeners.
                }
            }
    );
```

Рисунок 3.2.7 – Уривок коду створення запису передплати

3.3 Керівництво користувача

У цьому розділі здійснюється покроковий опис користування мобільним застосунком для користувачів. Після запуску програми з’являється початкове вікно програми (рисунок 3.3.1), де знаходяться дві кнопки: кнопка авторизації (Sign in) та кнопка реєстрації (Sign up).

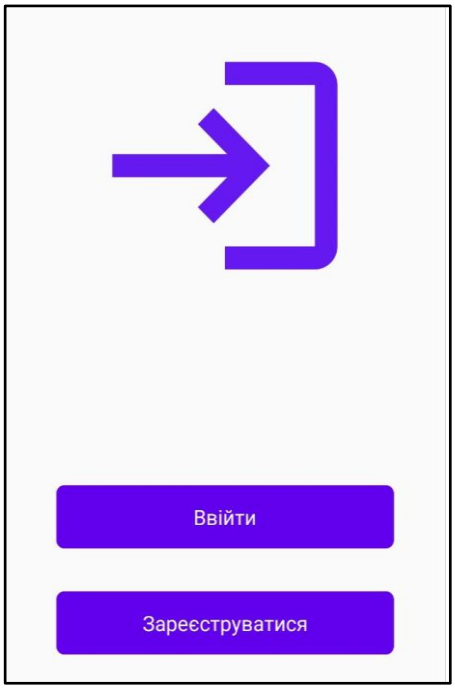


Рисунок 3.3.1 – Вигляд початкової форми програми

Якщо натиснути на кнопку SIGN UP, перед користувачем з’явиться форма реєстрації, де потрібно ввести необхідні поля.

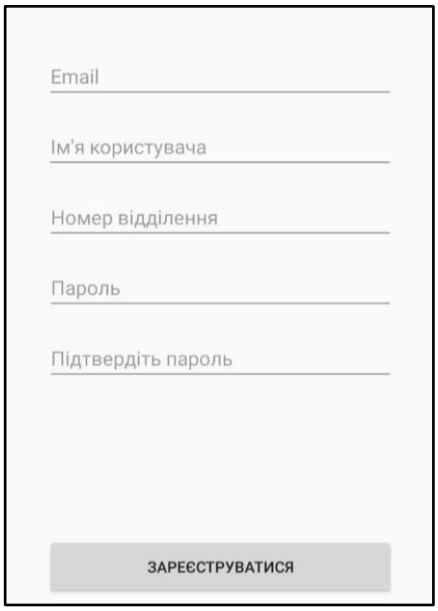


Рисунок 3.3.2 – Форма реєстрації

Оскільки програма зв’язана з базою даних, поле «Пароль» перевіряється на достовірність і в залежності від результату перевірки,

користувач або заходить в програму або виводиться повідомлення про помилку (рисунок 3.3.3)

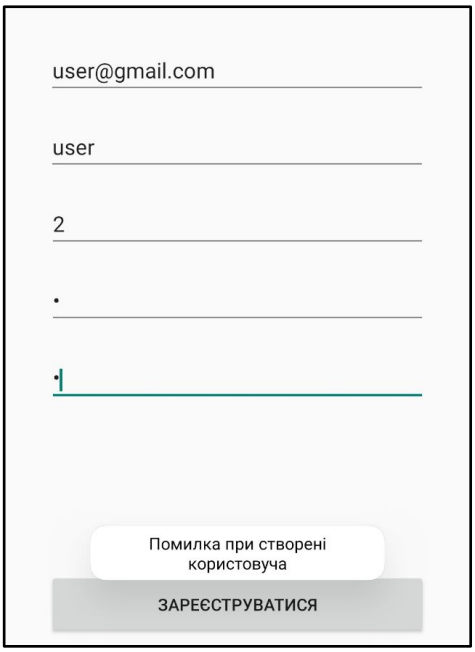


Рисунок 3.3.3 – Помилка при реєстрації

Після входу на обліковий запис користувача з’являється бокове меню (рисунок 3.3.4), де є перелік функцій програми та зверху вказана інформація про користувача під іменем якого зараз працює програма.

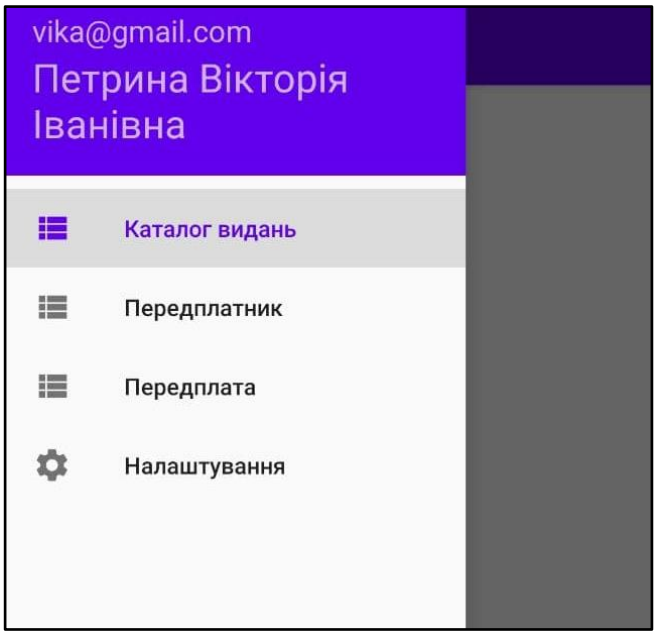


Рисунок 3.3.4 – Меню програми

Перша вкладка головного меню – це вкладка каталогу видань. На ній є дві кнопки «Список видань» та «Додати видання».

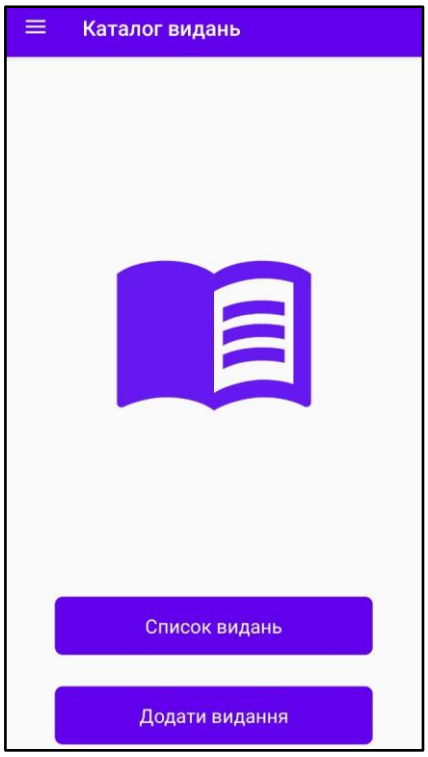


Рисунок 3.3.5 – Вкладка каталогу видань

Натиснувши на кнопку «Додати видання» відкривається нова форма для додавання (рисунок 3.3.6). На поле вартість встановлено обмеження введення. Можливо вводити тільки цифри.

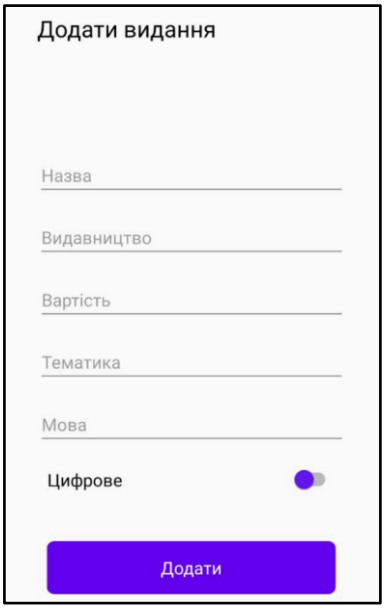


Рисунок 3.3.6 – Форма додавання видання

Натиснувши на кнопку «Список видань» з’являється форма, на якій представлена інформація про всі наявні видання. На формі є кнопка сортування, натиснувши на яку видання сортуються по назві в алфавітному або зворотньому порядку. На рисунку 3.3.7 зображено вже відсортований список видань.

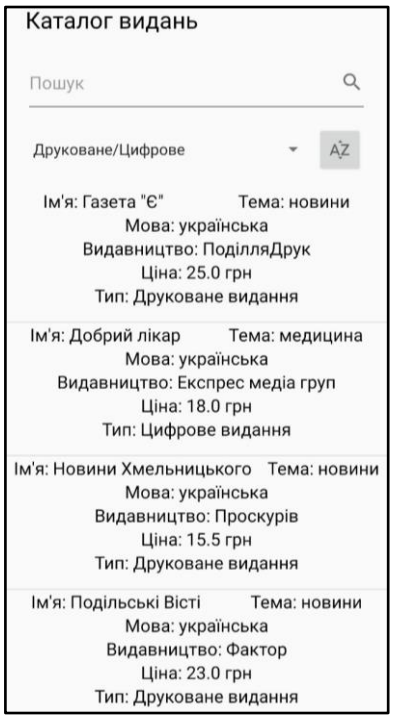


Рисунок 3.3.7 – Каталог видань

Для редагування видання достатньо натиснути на нього, після чого з’явиться повідомлення про можливі дії з цим виданням.

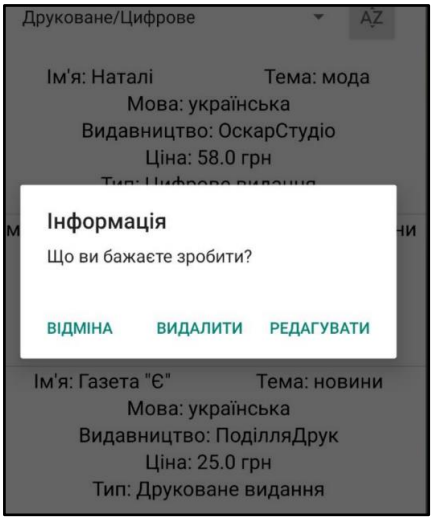


Рисунок 3.3.8 – Повідомлення

Натиснувши на кнопку «Редагувати» з’являється форма (рисунок 3.3.9) з заповненими даними про видання, які можна змінити. Для видалення видання потрібно натиснути на кнопку «Видалити».

Рисунок 3.3.9– Редагування видання

Для того щоб відфільтрувати видання по типу потрібно вибрати з випадаючого списку потрібний тип.

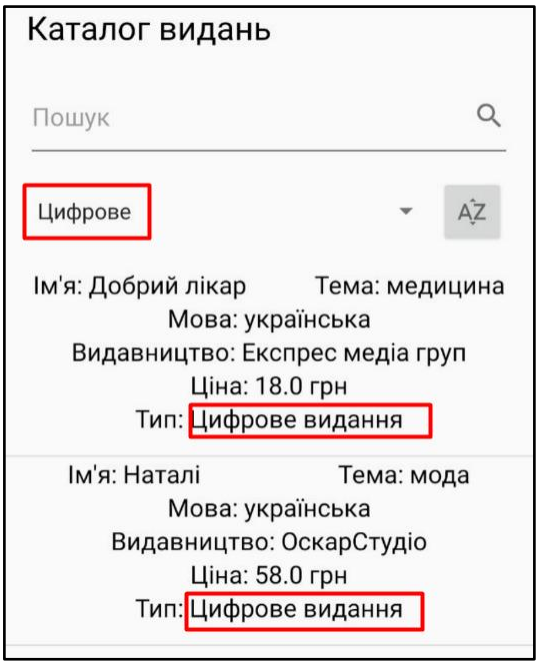


Рисунок 3.3.10 – Відфільтрований каталог видань

Для того щоб виконати пошук по каталогу потрібно ввести назву видання в відповідне поле.

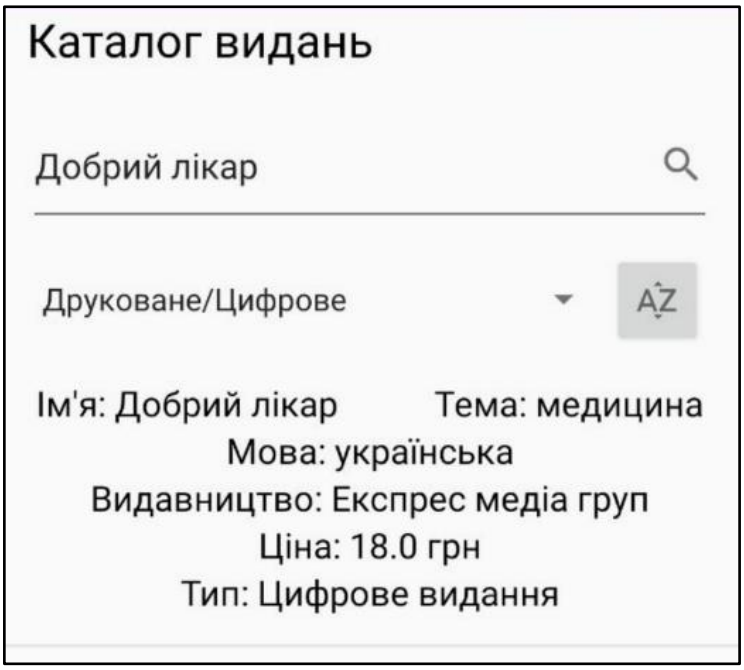


Рисунок 3.3.11– Пошук в каталозі

Наступною є вкладка передплатників (рисунок 3.3.12), на формі є 2 кнопки «Переглянути передплатників» та «Додати передплатника»

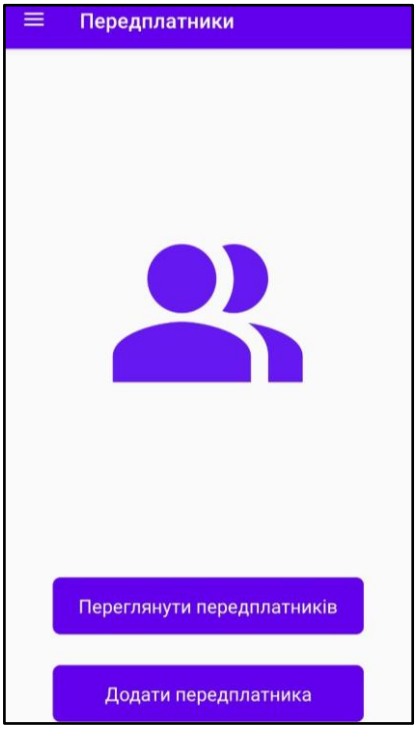


Рисунок 3.3.12– Вкладка передплатників

Натиснувши на кнопку «Переглянути передплатників», перед користувачем з’являється форма з інформацією про всіх зареєстрованих передплатників. На даній формі за іменем можна здійснювати пошук і сортувати передплатників.

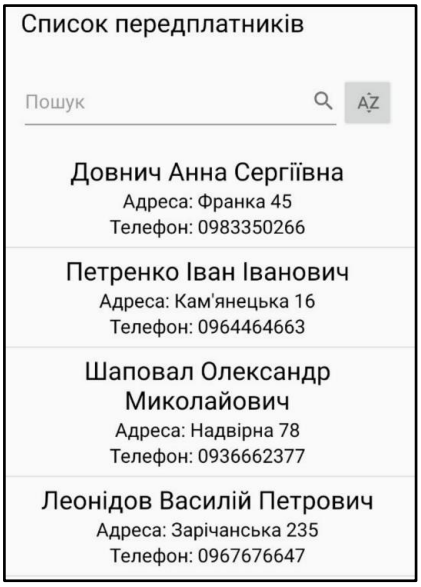


Рисунок 3.3.13– Список наявних передплатників

Для редагування передплатника потрібно натиснути на нього, після чого з’явиться повідомлення про можливі дії з цим передплатником (редагувати, видалити).

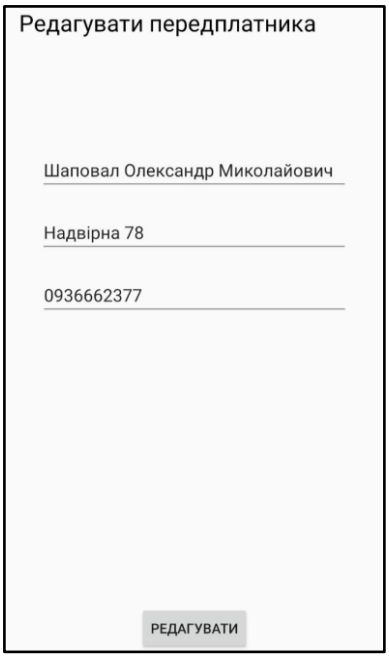


Рисунок 3.3.14 – Редагування інформації передплатника

При натисненні на кнопку «Додати передплатника», з’являється форма для додання нового передплатника в базу, з необхідними полями.

Рисунок 3.3.15 – Форма для створення нових передплатників

Для роботи з базою даних використано Firebase, усі дані, що вносились в програмі, були додані в базу даних.

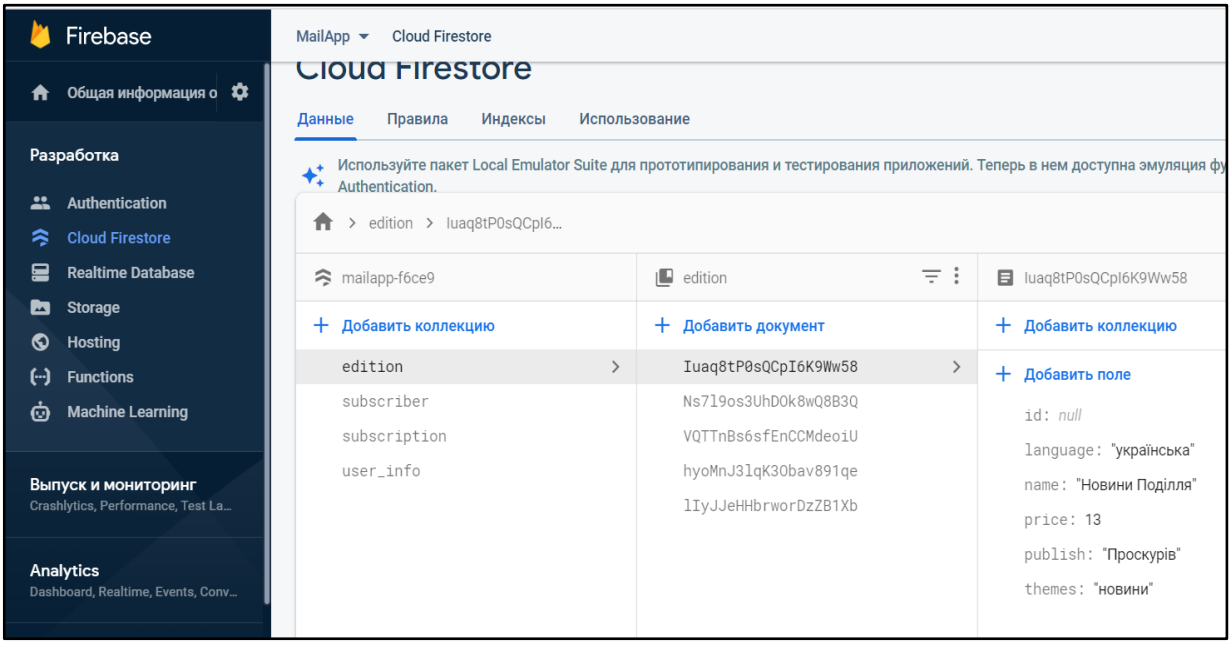


Рисунок 3.3.16 – База даних

У вкладці «Налаштування» є кнопка «Вихід» за допомогою якої можна вийти з цього облікового запису та змінити користувача на іншого.

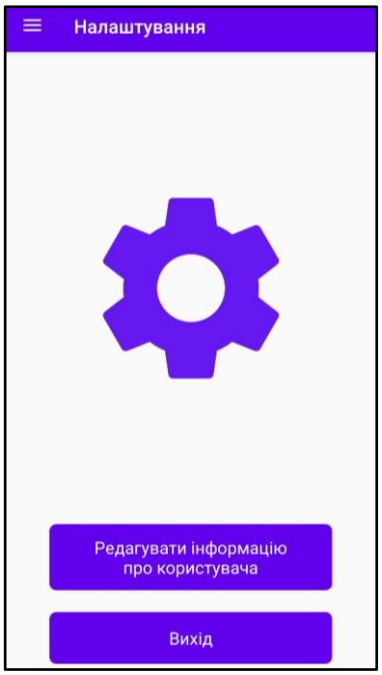


Рисунок 3.3.17 – Вкладка налаштувань

Також у вкладці налаштувань є кнопка «Редагувати інформацію про користувача», натиснувши на неї, відкриється форма, де можна це зробити.

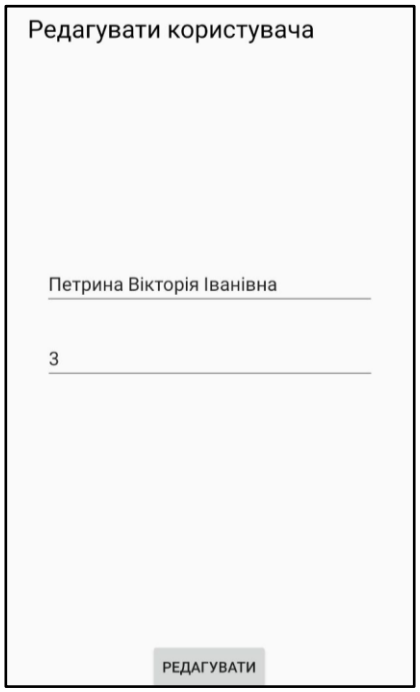


Рисунок 3.3.18 – Редагування користувача

Наступна вкладка – це вкладка, де можна оформити передплату (рисунок 3.3.19). На формі знаходяться дві кнопки «Переглянути передплату» та «Додати передплату».

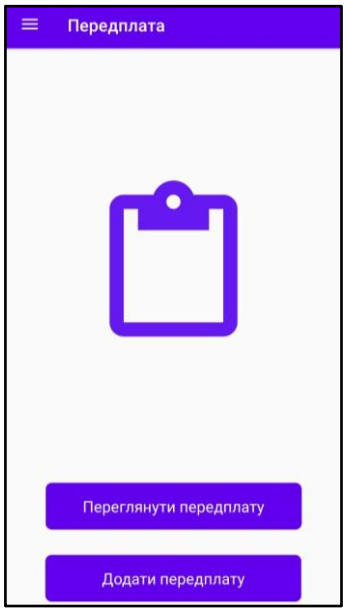


Рисунок 3.3.19 – Вкладка передплати

Якщо натиснути на кнопку «Переглянути передплату», з’явиться список вже оформлених передплат. На цій формі можна виконувати пошук передплати по ПІБ-у передплатника, а також сортувати в алфавітному і зворотньому порядку за іменем передплатника.

Рисунок 3.3.20 – Список оформлених передплат

Видалити передплату можна натиснувши на неї, з’явиться таке попередження про видалення

Рисунок 3.3.21 – Попередження про видалення

Після натиснення кнопки «Додати передплату» з’являється форма для оформлення нової передплати, де необхідно вибрати передплатника зі списку та вписати поштовий індекс, також необхідно вписати початок і кінець терміну передплати і обрати видання. Також можна переглянути вже обрані видання і при потребі видалити зі списку.

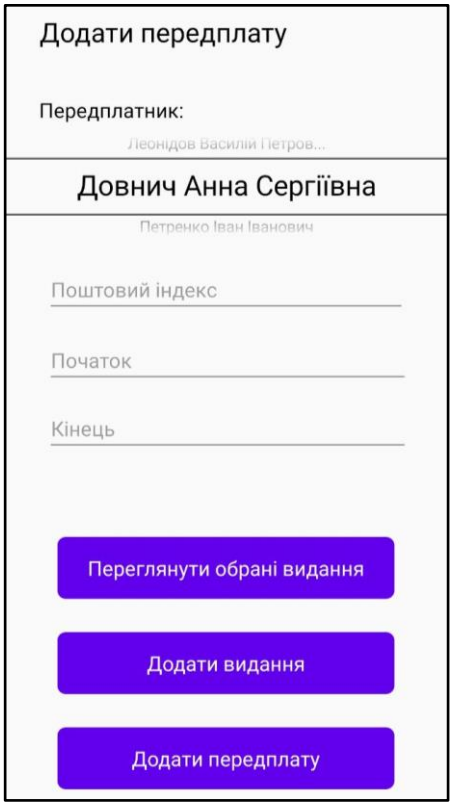


Рисунок 3.3.22 – Додання передплати

Натиснувши кнопку «Додати передплату», користувач завершить операцію та передплата буде додана у базу даних. Про це також з’явиться повідомлення.

Отже, в даному розділі описано покрокову інструкцію як користуватись програмою, які вона має функції та можливості.

3.4 Технічні характеристики мобільного застосунку

Щоб програма повноцінно працювала, потрібен телефон з наступними системними вимогами:

- оперативної пам’яті, мінімум 1 ГБ;
- операційна система Android 6 і вище;
- 50 Мб вільного місця на жорсткому диску;
- процесор з тактовою частотою 1.5 ГГц і вище.

Усі описані вимоги для нормального функціонування мобільного застосунку є мінімальними.

3.5 Тестування проєкту

Важливою складовою є тестування. Тестування програмного забезпечення — це процес технічного дослідження, призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись. Техніка тестування також включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою оцінки. Тестування пронизує весь життєвий цикл ПЗ. Це ітераційний процес. До цього процесу входить виконання програми з метою пошуку помилок. Це реалізовується за допомогою введення наперед продуманих даних, які можуть викликати збої і спостереження за тим, як поводить себе програма при такому ході подій.

Для перевірки програми на виявлення помилок було обрано метод мануального тестування. Ручне тестування здійснюється за допомогою

натискання кнопок і перевірки результатів їх роботи в самому інтерфейсі мобільного додатку. Для початку тестування потрібно створити користувача та авторизуватись, а далі просто натискати на клавіші і перевіряти відповідність їм наданої інформації.

В процесі користування програмою виявлено та виправлено наступні помилки:

– некоректна робота форма реєстрації(пропуск занадто малих паролів);

– дозвіл залишати пустими поля, які мусять бути заповненими;

– некоректне відображення інформації на формі;

– помилки з базою даних.

Внаслідок виправлення виявлених помилок програма працює коректно і видає помилки при неправильному введенні даних.

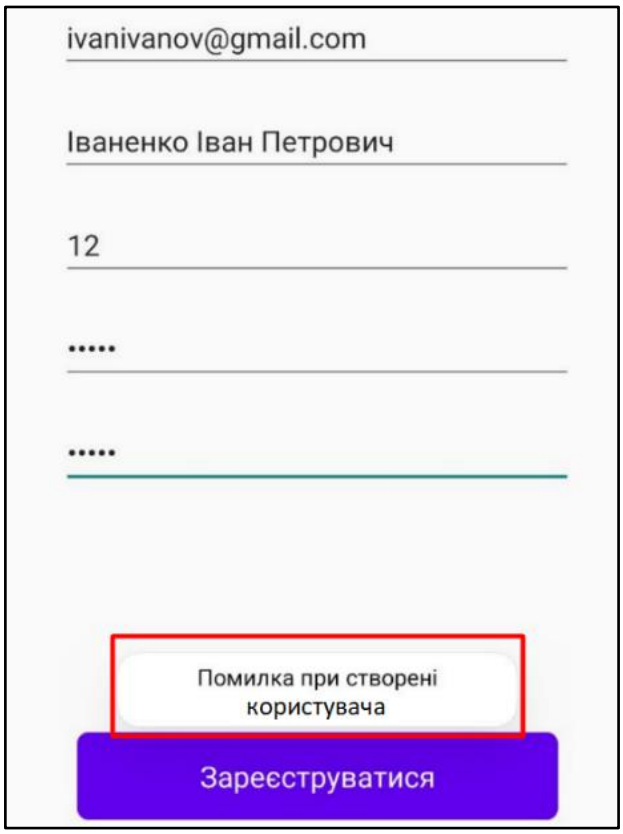


Рисунок 3.5.1 – Перевірка довжини паролю

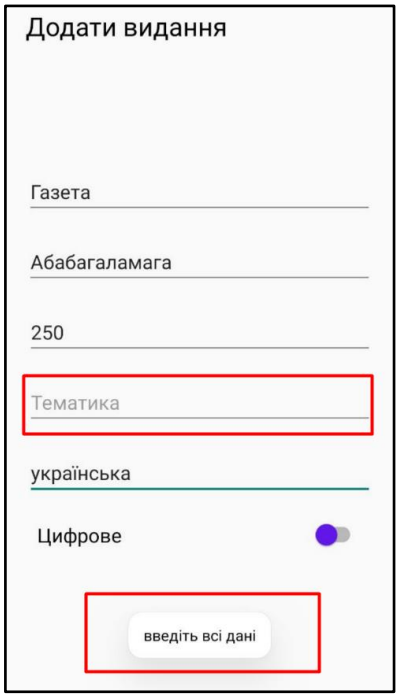


Рисунок 3.5.2 – Помилка при створенні нового видання

Отже, процес тестування відіграє значну роль у реалізації готового проєкту, щоб запобігти помилок в процесі експлуатації.

3.6 Висновки до розділу 3

У розділі 3 було описано декілька етапів розробки застосунку. Перше, що було продемонстровано - розробка бази даних мобільного застосунку. Розділ детально описує створення бази даних у Firebase, починаючи з створення проекту в консолі Firebase до налаштування Firestore. Також описано процес інтеграції бази даних у застосунок, наведено приклади коду для налаштування Firebase у проекті. Наведено фрагменти коду для створення нового користувача та збереження його інформації у Firestore. Далі було продемонстровано загальну структуру проєкту та описано використання Gradle для збірки проєкту. Наведено фрагменти коду для різних функцій застосунку. Детально описано процес користування застосунком зі скріншотами усіх форм застосунку. Також було подано технічні характеристики мобільного застосунку.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було реалізовано мобільний застосунок для обліку передплат у поштовому відділенні під операційну систему Android. Під час розробки було проведено аналіз вимог предметної області, аналіз існуючих рішень, визначено їх основні переваги та недоліки, для врахування під час реалізації додатку. Проведено аналіз засобів розробки та здійснено їх обґрунтований вибір. Було обрано середовище розробки «Android Studio», мову програмування Java, для створення і реалізації бази даних була використана платформа Firebase.

Після попередніх етапів розробки проєкту, було здійснено проєктування, написання коду до відповідних елементів програми, було створено логіку програми, логіку самого користувача. Для коректного відображання процесів та легкого переведення діаграм в програмний код були створені діаграми UML.

В результаті було створено програму для операторів поштового відділення.

Переваги програми:

- зручний та зрозумілий інтерфейс;
- програма працює швидко;
- не займає багато місця на телефоні;
- збереження даних для входу в систему.

Недоліки у програмі:

- не зручний перехід між вікнами;
- вимагає підключення до Інтернету.

У результаті виконаної роботи було реалізовано мобільний застосунок для обліку передплат у поштовому відділенні, який має простий та інтуїтивно-зрозумілий інтерфейс.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інтернет-ресурс для вивчення програмування [Електронний ресурс] – Режим доступу: https://metanit.com/;
2. Інтернет-ресурс для вивчення програмування [Електронний ресурс] – Режим доступу: https://metanit.com/;
3. Інтернет-форум для пошуку вирішення запитань прикладного програмування [Електронний ресурс] – Режим доступу: https://stackoverflow.com/;
4. Інтернет-ресурс для вивчення Java core docs[Електронний ресурс] – Режим доступу: https://docs.oracle.com/en/java/;
5. Інтернет-ресурс для вивчення Firebase [Електронний ресурс] – Режим доступу: https://firebase.google.com/.
6. Інтернет-ресурс для вивчення Firebase [Електронний ресурс] – Режим доступу: https://blog.back4app.com/ru/%D1%87%D1%82%D0%BE-%D1%82%D0%B0%D0%BA%D0%BE%D0%B5-firebase/.
7. Форум програмістів та сисадмінів [Електронний ресурс] – Режим доступу: http://www.cyberforum.ru/.
8. Запорожець, О.В. "Системи управління базами даних та знань в інтернет-технологіях" / О.В. Запорожець. - Київ: НУХТ, 2013. - 192 с.
9. Гарбуз, С.В. "Основи інтернет-технологій" / С.В. Гарбуз. - Київ: Центр учбової літератури, 2014. - 320 с.
10. Котляр, І.В. "Основи програмування в середовищі Visual Studio 2019" / І.В. Котляр. - Київ: Видавничий дім "Ін Юре", 2019. - 400 с.
11. Інтернет-ресурс для вивчення Java [Електронний ресурс] – Режим доступу: https://javarush.com/ua/groups/posts/;
12. Firebase Documentation. Google. [Електронний ресурс] – Режим доступу:https://firebase.google.com/docs
13. Tushar, R., & Kurose, M. / Learning Firebase for Android Development. Packt Publishing., 2018

14. Buining, T., & Klein, A. / Firebase Essentials – Android Edition. Leanpub., 2017

15. Chung, L. / Firebase Cookbook: Over 70 recipes to help you create real-time web and mobile applications with Firebase. Packt Publishing., 2017

16. Кухаренко, В. М., та Завгородній, Ю. М. / Об’єктно-орієнтоване програмування на Java. - Харків: ХНУРЕ. - 268 с.

17. Яковлева, І. В. / Програмування мовою Java. - Київ: Видавничий дім "Кондор", 2020 - 350 с.

18. Задорожний, О. В. / Розробка мобільних додатків на Java для Android. - Київ: Ліра-К., 2020 – 258 с.

19. Гордієнко, О.В., Гордієнко, О.О. "ASP.NET: Використання фреймворка" / О.В. Гордієнко, О.О. Гордієнко. - Київ: Видавничий дім "Ін Юре", 2019. - 352 с

20. Сергеєв, О. П. / Java: Основи програмування. Київ: Академвидав., 2017 – 296 с.

21. Шумило, М. В. / Java. Структури даних та алгоритми. Київ: Вид. Група BHV. , 2018 – 304 с.

22. Іванченко, В. І. / Сучасні технології розробки програмних систем. Харків: ХНУРЕ. , 2019 – 297 с.

23. Дерев'янко, І. В. / Розробка мобільних застосунків для iOS та Android. Київ: Видавничий дім "Кондор"., 2020 – 182 с.

24. Пахомов, А. О., та Кравець, О. В. / Хмарні обчислення в сучасних інформаційних системах. Київ: Вид-во НТУУ "КПІ". , 2018 – 198 с

25. Гладкий, А. О. /. Основи програмування для Android. Київ: НТУУ "КПІ"., 2018 – 78 с

26. Сергеєв, О. П. / Архітектура програмних додатків. Київ: Академвидав. , 2018 – 455 с

27. Костюк, В. В. / Архітектура програмних систем. Київ: Видавничий дім "Кондор"., 2019 – 452 с

ДОДАТКОК А

Код проєкту

Edition.java

```
package com.hpk.mail.model.edition;

public class Edition {

    private String id;
    private String name;
    private String publish;
    private Double price;
    private String themes;
    private String language;
    private EditionType editionType;

    public EditionType getEditionType() {
        return editionType;
    }
    public String getEditionTypeAlias() {
        return editionType == EditionType.DIGITAL_EDITION ?
"Цифрове видання" : "Друковане видання";
    }
    public void setEditionType(EditionType editionType) {
        this.editionType = editionType;
    }
    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getPublish() {
        return publish;
    }
    public void setPublish(String publish) {
        this.publish = publish;
    }
    public Double getPrice() {
        return price;
    }
    public void setPrice(Double price) {
        this.price = price;
    }
    public String getThemes() {
        return themes;
    }
    public void setThemes(String themes) {
        this.themes = themes;
    }
    public String getLanguage() {
        return language;
    }
    public void setLanguage(String language) {
        this.language = language;
    }

    public Edition(String name, String publish, Double price, String
themes, String language, EditionType editionType) {
        this.name = name;
        this.publish = publish;
        this.price = price;
        this.themes = themes;
        this.language = language;
        this.editionType = editionType;
    }
}
```

Editiontype.java

```
public enum EditionType {
    DIGITAL_EDITION,
    PRINT_EDITION
}
```

Subscriber.java

```
package com.hpk.mail.model.subscriber;

public class Subscriber {
    private String id;
    private String fullName;
    private String address;
    private String phoneNumber;

    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getFullName() {
        return fullName;
    }
    public void setFullName(String fullName) {
        this.fullName = fullName;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }
    public String getPhoneNumber() {
        return phoneNumber;
    }
    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }
    public Subscriber(String fullName, String address, String
phoneNumber) {
        this.fullName = fullName;
        this.address = address;
        this.phoneNumber = phoneNumber;
    }
    public Subscriber() {
    }
}
```

Subscription.java

```
package com.hpk.mail.model.subscription;

import java.util.ArrayList;
import java.util.Date;

public class Subscription {
    private String id;
    private String fullName;
    private String mailIndex;
    private Date start;
    private Date end;
    private ArrayList<String> idList;
    private double price;

    public String getFullName() {
        return fullName;
    }
```

```
    public void setFullName(String fullName) {
        this.fullName = fullName;
    }
    public String getMailIndex() {
        return mailIndex;
    }
    public void setMailIndex(String mailIndex) {
        this.mailIndex = mailIndex;
    }
    public ArrayList<String> getIdList() {
        return idList;
    }
    public void setIdList(ArrayList<String> idList) {
        this.idList = idList;
    }
    public double getPrice() {
        return price;
    }
    public void setPrice(double price) {
        this.price = price;
    }
    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public Date getStart() {
        return start;
    }
    public void setStart(Date start) {
        this.start = start;
    }
    public Date getEnd() {
        return end;
    }
    public void setEnd(Date end) {
        this.end = end;
    }
    public Subscription(String fullName, String mailIndex,
ArrayList<String> idList, Date start, Date end, double price) {
        this.fullName = fullName;
        this.mailIndex = mailIndex;
        this.idList = idList;
        this.price = price;
        this.start = start;
        this.end = end;
    }
}
```

CurrentUser.java

```
package com.hpk.mail.model.user;

public class CurrentUser extends User {

    private String email;
    private String address;
    private String username;
    private String id;

    private static CurrentUser instance;

    public static CurrentUser getNewInstance(String email, String
username, String address, String id) {
        instance = new CurrentUser(email, username, address, id);
        return instance;
    }
    public static CurrentUser getCurrentInstance() {
        return instance;
    }

    public static void clearUser() {
        instance = null;
    }
    private CurrentUser(String email, String username, String
address, String id) {
        this.email = email;
        this.address = address;
        this.username = username;
        this.id = id;
    }

    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getUsername() {
        return username;
    }
    public void setUserName(String username) {
        this.username = username;
    }
    pblic String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }
}
```

User.java

```
package com.hpk.mail.model.user;

public class User {
    private String email;
    private String password;
    private String address;
    private String username;

    public String getUsername() {
        return username;
    }
    public void setUserName(String username) {
        this.username = username;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public String getPassword() {
        return password;
    }
    public User() {}
    public User(String email, String password){
        this.email = email;
        this.password = password;
    }
    public User(String email, String password,String
username,String address){
        this.email = email;
        this.password = password;
```

```
        this.username = username;
        this.address = address;    }}
```

UserInfo.java

```
package com.hpk.mail.model.user;

public class UserInfo {
    private String address;
    private String username;

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public void setAddress(String address){
        this.address = address;
    }

    public String getAddress(){
        return address;
    }

    public UserInfo(){}

    public UserInfo(String username,String address){
        this.username = username;
        this.address = address;
    }
}
```

Auth.java

```
package com.hpk.mail.service.firebase;

import android.content.Context;
import android.util.Log;
import android.widget.Toast;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.FirebaseFirestore;
import com.hpk.mail.StartApp;
import com.hpk.mail.model.user.CurrentUser;
import com.hpk.mail.model.user.User;
import com.hpk.mail.model.user.UserInfo;

public class Auth {
    private static final String COLLECTION_NAME = "user_info";

    private final FirebaseAuth firebaseInstance =
FirebaseAuth.getInstance();
    private final FirebaseFirestore db =
FirebaseFirestore.getInstance();

    public Auth() {
    }

    public boolean checkCurrentUser() {
        final FirebaseUser firebaseUser =
firebaseInstance.getCurrentUser();
        return firebaseUser != null;
    }

    public String getEmail() {
        final FirebaseUser firebaseUser =
firebaseInstance.getCurrentUser();
        return firebaseUser != null ? firebaseUser.getEmail() : null;
    }

    public void signUp(User userModel, Context context) {

firebaseInstance.createUserWithEmailAndPassword(userModel.ge
tEmail(), userModel.getPassword()).addOnSuccessListener(result -
> {
                try {
                    String userID = result.getUser().getUid();
                    UserInfo userInfo = new
UserInfo(userModel.getUsername(), userModel.getAddress());

db.collection(COLLECTION_NAME).document(userID).set(userI
nfo).addOnSuccessListener(res -> {

CurrentUser.getNewInstance(userModel.getEmail(),
userModel.getUsername(), userModel.getAddress(), userID);
                        StartApp.getCurrentInstance().toAppActivity();
                    });
                } catch (Exception exc) {
                    Log.d("exception", exc.getMessage());
                }
            }
        ).addOnFailureListener(result -> {
                Log.d("error", result.getMessage());
                Toast.makeText(context, "Помилка при створені
користувча", Toast.LENGTH_LONG).show();
                StartApp startApp = StartApp.getCurrentInstance();
                startApp.onFailSignUp();
            }
        );
    }

    public void logout() {
        firebaseInstance.signOut();
        CurrentUser.clearUser();
    }

    public void getInfo(Context context) {
        FirebaseUser user = firebaseInstance.getCurrentUser();
        String id = user.getUid();


db.collection(COLLECTION_NAME).document(id).get().addOnS
uccessListener(documentSnapshot -> {
            String email =
firebaseInstance.getCurrentUser().getEmail();
            String username =
documentSnapshot.get("username").toString();
            String address =
documentSnapshot.get("address").toString();

            CurrentUser.getNewInstance(email, username, address, id);
            StartApp.getCurrentInstance().toAppActivity();
        }).addOnFailureListener(fail -> {
            Log.d("fail", fail.toString());
            Toast.makeText(context, "Помилка при вході",
Toast.LENGTH_LONG).show();
            logout();
            StartApp startApp = StartApp.getCurrentInstance();
            startApp.onFailSignIn();
        });
    }

    public void signIn(User userModel, Context context) {

firebaseInstance.signInWithEmailAndPassword(userModel.getEm
ail(), userModel.getPassword()).addOnFailureListener(result -> {
                Toast.makeText(context, "Помилка при вході",
Toast.LENGTH_LONG).show();
                StartApp startApp = StartApp.getCurrentInstance();
                startApp.onFailSignIn();
            }
        ).addOnSuccessListener(result -> {
            getInfo(context);
        });
    }
}
```

Actions.java

```
package com.hpk.mail.service.firebase;

import com.google.firebase.firestore.QuerySnapshot;

public interface IActions {
```

```
    void onSuccessListener();
    void onSuccessListener(QuerySnapshot documentSnapshot);
    void onFailListener();
}
```

Storage.java

```
public class Storage {
    FirebaseFirestore firebaseFirestore =
FirebaseFirestore.getInstance();

    public void setData(String collectionName, Object value,
IActions storageValue) {

firebaseFirestore.collection(collectionName).add(value).addOnFail
ureListener(result -> {
            storageValue.onFailListener();
        }).addOnSuccessListener(result -> {
            storageValue.onSuccessListener();
        });
    }

    public void getData(String collectionName, IActions
storageValue) {

firebaseFirestore.collection(collectionName).get().addOnSuccessLi
stener(doc -> {
            storageValue.onSuccessListener(doc);
        }).addOnFailureListener(result -> {
            Log.d("error", result.getMessage());
            storageValue.onFailListener();
        });
    }

    public void editData(String collectionName, String id, Object
value) {

firebaseFirestore.collection(collectionName).document(id).set(valu
e).addOnFailureListener(result -> {
            Log.d("error", result.getMessage());
        });
    }

    public void removeData(String collectionName, String id) {

firebaseFirestore.collection(collectionName).document(id).delete()
.addOnFailureListener(result -> {
            Log.d("error", result.getMessage());
        });
    }

    public void removeData(String collectionName, String id,
IActions storageValue) {

firebaseFirestore.collection(collectionName).document(id).delete()
.addOnSuccessListener(result -> getData(collectionName,
storageValue)).addOnFailureListener(result -> {
            Log.d("error", result.getMessage());
        });
    }
}
```

EditionFragment.java

```
 public class EditionFragment extends Fragment {


    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle
savedInstanceState) {
        View root = inflater.inflate(R.layout.fragment_edition,
container, false);

root.findViewById(R.id.create_edition_btn).setOnClickListener(v-
>{
            Intent intent = new Intent(this.getContext(),
AddEdition.class);
            startActivity(intent);
        });

root.findViewById(R.id.list_edition_btn).setOnClickListener(v->{
            Intent intent = new Intent(this.getContext(),
GetEdition.class);
            startActivity(intent);
        });
        return root;
    }
}
```

SettingFragment.java

```
package com.hpk.mail.ui.setting;

public class SettingFragment extends Fragment {
    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle
savedInstanceState) {
        View root = inflater.inflate(R.layout.fragment_setting,
container, false);

root.findViewById(R.id.edit_user_info_btn).setOnClickListener(th
is::editUserInfo);

root.findViewById(R.id.setting_logout_btn).setOnClickListener(th
is::logout);
        return root;
    }

    private void logout(View v){
        Auth auth = new Auth();
        auth.logout();
        CurrentUser.clearUser();
        Intent intent = new Intent(getContext(), StartApp.class);
        startActivity(intent);
        App.getInstance().finish();
    }

    private void editUserInfo(View v){
        Intent intent = new Intent(getContext(), EditUser.class);
        startActivity(intent);
    }
}
```

SubscriberFragment.java

```
public class SubscriberFragment extends Fragment {

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle
savedInstanceState)
 {
        View root = inflater.inflate(R.layout.fragment_subscriber,
container, false);

root.findViewById(R.id.show_subscriber_btn).setOnClickListener
(this::toShowSubscriber);

root.findViewById(R.id.add_subscriber_btn).setOnClickListener(t
his::toAddSubscriber);
        return root;
    }

    private void toAddSubscriber(View v) {
        Intent intent = new Intent(this.getContext(),
AddSubscriber.class);
        startActivity(intent);
    }

    private void toShowSubscriber(View v) {
        Intent intent = new Intent(this.getContext(),
GetSubscriber.class);
        startActivity(intent);
    }
}
```

SubscriptionFragment.java

```
public class SubscriptionFragment extends Fragment {

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle
savedInstanceState) {
        View root = inflater.inflate(R.layout.fragment_subscription,
container, false);

root.findViewById(R.id.show_subscription_btn).setOnClickListen
er(this::toShowSubscription);

root.findViewById(R.id.add_subscription_btn).setOnClickListener
(this::toAddSubscription);
        return root;
    }

    private void toAddSubscription(View v) {
        Intent intent = new Intent(this.getContext(),
AddSubscription.class);
        startActivity(intent);
    }

    private void toShowSubscription(View v) {
        Intent intent = new Intent(this.getContext(),
GetSubscription.class);
        startActivity(intent);
    }
}
```

App.java

```
public class App extends AppCompatActivity{

    private AppBarConfiguration mAppBarConfiguration;
    private static App instance;
    public static App getInstance() {
        return instance;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        instance = this;
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main_screen);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        NavigationView navigationView =
findViewById(R.id.nav_view);
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_edition, R.id.nav_subscriber,
R.id.nav_subscription, R.id.nav_setting)
                .setDrawerLayout(drawer)
                .build();
        NavController navController =
Navigation.findNavController(this, R.id.nav_host_fragment);
        NavigationUI.setupActionBarWithNavController(this,
navController, mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView,
navController);
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        TextView email = findViewById(R.id.user_email);
        TextView username = findViewById(R.id.user_username);
        Auth auth = new Auth();
        CurrentUser currentUser = CurrentUser.getCurrentInstance();
        email.setText(auth.getEmail());
        username.setText(currentUser.getUsername());
        return true;
    }

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController =
Navigation.findNavController(this, R.id.nav_host_fragment);
        return NavigationUI.navigateUp(navController,
mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
}
```

StartApp.java

```
public class StartApp extends AppCompatActivity {

    private boolean canBack = false;
    private static StartApp startApp;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        startApp = this;
        Auth auth = new Auth();
        if (auth.checkCurrentUser()) {
            auth.getInfo(this);
        }else{
            onStartLayout();
        }
    }

    @Override
    public void onBackPressed() {
        if (canBack) {
            onStartLayout();
            canBack = false;
        } else {
            System.exit(0);
        }
    }

    public static StartApp getCurrentInstance() {
        return startApp;
    }

    public void toAppActivity() {
        Intent intent = new Intent(this, App.class);
        startActivity(intent);
        finish();
    }

    private boolean isValidEmail(CharSequence target) {
        return (!TextUtils.isEmpty(target) &&
Patterns.EMAIL_ADDRESS.matcher(target).matches());
    }

    private void onSignInBtnClick(View v) {
        v.setVisibility(View.INVISIBLE);

findViewById(R.id.progressBar_sign_in).setVisibility(View.VISI
BLE);

        EditText passwordField =
findViewById(R.id.password_sign_in);
        EditText emailField = findViewById(R.id.email_sign_in);

        String email = emailField.getText().toString();
        String password = passwordField.getText().toString();

        if (!(isValidEmail(email) || !password.isEmpty())) {
            new
AlertDialog.Builder(this).setTitle("Помилка").setMessage("Введі
ть корректно дані").setPositiveButton("OK", null).show();
            onFailSignIn();
            return;
        }
```

```
        User user = new User(email, password);
        Auth auth = new Auth();

        auth.signIn(user, this);
    }

    public void onFailSignIn(){

findViewById(R.id.sign_in_btn).setVisibility(View.VISIBLE);

findViewById(R.id.progressBar_sign_in).setVisibility(View.INVI
SIBLE);
    }

    public void onFailSignUp(){

findViewById(R.id.sign_up_btn).setVisibility(View.VISIBLE);

findViewById(R.id.progressBar_sign_up).setVisibility(View.INVI
SIBLE);
    }

    private void onSignUpBtnClick(View v) {
        v.setVisibility(View.INVISIBLE);

findViewById(R.id.progressBar_sign_up).setVisibility(View.VISI
BLE);
        EditText passwordField =
findViewById(R.id.password_sign_up);
        EditText confirmPasswordField =
findViewById(R.id.confirm_password_sign_up);
        EditText emailField = findViewById(R.id.email_sign_up);
        EditText addressField =
findViewById(R.id.address_sign_up);
        EditText usernameField =
findViewById(R.id.username_sign_up);

        String email = emailField.getText().toString();
        String password = passwordField.getText().toString();
        String confirmPassword =
confirmPasswordField.getText().toString();
        String address = addressField.getText().toString();
        String username = usernameField.getText().toString();

        if (!(isValidEmail(email) || password.isEmpty() ||
password.equals(confirmPassword)) || address.isEmpty() ||
username.isEmpty()) {
            new
AlertDialog.Builder(this).setTitle("Помилка").setMessage("Введіт
ь корректно дані").setPositiveButton("OK", null).show();
            onFailSignUp();
            return;
        }

        User user = new User(email, password, username, address);

        Auth auth = new Auth();

        auth.signUp(user, this);
    }

    private void onStartLayout() {
        setContentView(R.layout.login_layout);
        findViewById(R.id.to_sign_in_btn).setOnClickListener(v ->
onSignInLayout());
        findViewById(R.id.to_sign_up_btn).setOnClickListener(v ->
onSignUpLayout());
    }

    private void onSignUpLayout() {
        canBack = true;
        setContentView(R.layout.sign_up_layout);

findViewById(R.id.sign_up_btn).setOnClickListener(this::onSign
UpBtnClick);
    }

    private void onSignInLayout() {
        canBack = true;
        setContentView(R.layout.sign_in_layout);

findViewById(R.id.sign_in_btn).setOnClickListener(this::onSignI
nBtnClick);
    }
}
```

EditionActions.java

```
public class EditionActions implements IActions {

    @Override
    public void onSuccessListener() {
    }

    @Override
    public void onSuccessListener(QuerySnapshot
documentSnapshot) {
        EditionList getEdition = EditionList.getInstance();
        List<DocumentSnapshot> documents =
documentSnapshot.getDocuments();
        ArrayList<Edition> editions = new ArrayList<>();
        for (DocumentSnapshot element : documents) {
            Edition edition = new Edition(element.getString("name"),
element.getString("publish"), (Double) element.get("price"),
element.getString("themes"), element.getString("language"),
element.getString("editionType").equals(EditionType.DIGITAL_E
DITION) ? EditionType.DIGITAL_EDITION :
EditionType.PRINT_EDITION);
            edition.setId(element.getId());
            editions.add(edition);
        }
        getEdition.onSuccessListener(editions);
    }

    @Override
    public void onFailListener() {
        EditionList addEdition = EditionList.getInstance();
        Toast.makeText(addEdition, "Fail",
Toast.LENGTH_LONG).show();
    }
}
```

EditionAdapter.java

```
public class EditionAdapter extends BaseAdapter {

    private Activity activity;
    private ArrayList<Edition> editions;
    private LayoutInflater layoutInflater;

    public EditionAdapter(Activity activity, ArrayList<Edition>
editions) {
        this.activity = activity;
        this.editions = editions;
        layoutInflater = (LayoutInflater)
activity.getSystemService(Context.LAYOUT_INFLATER_SERVI
CE);
    }

    @Override
    public int getCount() {
        return editions.size();
    }

    @Override
    public Object getItem(int position) {
        return editions.get(position);
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
```

```
    public View getView(int position, View view, ViewGroup
viewGroup) {
        View finalView = view;
        finalView =
layoutInflater.inflate(R.layout.edition_list_element, null);
        TextView publishTextView =
finalView.findViewById(R.id.publish);
        TextView languageTextView =
finalView.findViewById(R.id.language);
        TextView priceTextView =
finalView.findViewById(R.id.price);
        TextView themesTextView =
finalView.findViewById(R.id.themes);
        TextView nameTextView =
finalView.findViewById(R.id.name);
        TextView typeTextView =
finalView.findViewById(R.id.type);

        publishTextView.setText("Видавництво: " +
editions.get(position).getPublish());
        languageTextView.setText("Мова: " +
editions.get(position).getLanguage());
        priceTextView.setText("Ціна: " +
editions.get(position).getPrice() + " грн");
        themesTextView.setText("Тема: " +
editions.get(position).getThemes());
        nameTextView.setText("Ім'я: " +
editions.get(position).getName());
        typeTextView.setText("Тип: " +
editions.get(position).getEditionTypeAlias());

        AddSubscription addSubscription =
AddSubscription.getInstance();

        finalView.setOnClickListener(v -> {
            activity.finish();
            addSubscription.addEdition(editions.get(position));
        });
        return finalView;
    }
}
```

EditionList.java

```
public class EditionList extends AppCompatActivity {
    private static EditionList instance;

    public static EditionList getInstance() {
        return instance;
    }
    private ArrayList<Edition> editions;
    private Boolean descSort = false;

    @RequiresApi(api = Build.VERSION_CODES.N)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.fragment_get_edition);
        instance = this;
        Storage storage = new Storage();
        storage.getData("edition", new EditionActions());
        final EditText edittext = (EditText)
findViewById(R.id.search_edition_input);
        edittext.setOnKeyListener(new View.OnKeyListener() {
            public boolean onKey(View v, int keyCode, KeyEvent
event) {
                if ((event.getAction() == KeyEvent.ACTION_DOWN)
&&
                        (keyCode == KeyEvent.KEYCODE_ENTER)) {
                    InputMethodManager imm = (InputMethodManager)
instance.getSystemService(Activity.INPUT_METHOD_SERVICE
);
                    View view = instance.getCurrentFocus();
                    if (view == null) {
                        view = new View(instance);
                    }

imm.hideSoftInputFromWindow(view.getWindowToken(), 0);
                    ArrayList<Edition> temp = new ArrayList<>();
                    String searchText = edittext.getText().toString();
                    for (Edition edition : editions) {
                        if (edition.getName().contains(searchText) ||
                                edition.getLanguage().contains(searchText) ||

edition.getPrice().toString().contains(searchText) ||
                                edition.getThemes().contains(searchText) ||

edition.getEditionTypeAlias().contains(searchText) ||
                                edition.getPublish().contains(searchText)) {
                            temp.add(edition);
                        }
                    }
                    onSuccessListener(temp);
                    return true;
                }
                return false;
            }
        });
        final Spinner spinner = findViewById(R.id.spinner);
        ArrayAdapter<?> adapter =
                ArrayAdapter.createFromResource(this,
R.array.edition_type, android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spin
ner_dropdown_item);
        spinner.setAdapter(adapter);
        spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            public void onItemSelected(AdapterView<?> parent,
                                       View itemSelected, int
selectedItemPosition, long selectedId) {
                if (editions != null) {
                    ArrayList<Edition> temp = new ArrayList<>();
                    for (Edition edition : editions) {
                        if (edition.getEditionType() ==
EditionType.DIGITAL_EDITION && 2 == selectedItemPosition)
{
                            temp.add(edition);
                        }
                        if (edition.getEditionType() ==
EditionType.PRINT_EDITION && 1 == selectedItemPosition) {
                            temp.add(edition);
                        }
                        if (0 == selectedItemPosition) {
                            temp.add(edition);
                        }
                    }
                    onSuccessListener(temp);
                }
            }

            public void onNothingSelected(AdapterView<?> parent) {
            }
        });
        findViewById(R.id.sort_edition_btn).setOnClickListener(v ->
{
            if(!descSort) {
                editions.sort((o1, o2) ->
o1.getName().compareTo(o2.getName()));
                descSort = true;
            }else{
                editions.sort((o1, o2) ->
o2.getName().compareTo(o1.getName()));
                descSort = false;
            }
            onSuccessListener(editions);
        });
    }

    public void onSuccessListener(ArrayList<Edition> editions) {
        if (this.editions == null) {
            this.editions = editions;
        }
        EditionAdapter editionAdapter = new EditionAdapter(this,
editions);
```

```
        ListView listView = findViewById(R.id.edition_list);
        listView.setAdapter(editionAdapter);
    }

}
```

AddEdition.java

```
public class AddEdition extends AppCompatActivity {
    private static AddEdition instance;

    public static AddEdition getInstance() {
        return instance;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.fragment_create_edition);

findViewById(R.id.add_edition_btn).setOnClickListener(this::add
Edition);
        instance = this;
    }

    public void addEdition(View v) {
        Storage storage = new Storage();
        EditionActions editionActions = new EditionActions();

findViewById(R.id.add_edition_btn).setVisibility(View.INVISIB
LE);

        EditText nameEditText = findViewById(R.id.name_edition);
        EditText languageEditText =
findViewById(R.id.language_edition);
        EditText priceEditText = findViewById(R.id.price_edition);
        EditText themesEditText =
findViewById(R.id.themes_edition);
        EditText publishEditText =
findViewById(R.id.publish_edition);
        Switch isDigitalSwitch =
findViewById(R.id.is_digital_switch);

        String name = nameEditText.getText().toString();
        String language = languageEditText.getText().toString();
        Double price =
Double.parseDouble(priceEditText.getText().toString());
        String themes = themesEditText.getText().toString();
        String publish = publishEditText.getText().toString();
        Boolean isDigital = isDigitalSwitch.isChecked();

        if (name.isEmpty() || price == 0 || themes.isEmpty() ||
language.isEmpty() || publish.isEmpty()) {
            Toast.makeText(this, "введіть всі дані",
Toast.LENGTH_LONG).show();
            return;
        }

        Edition edition = new Edition(name, publish, price, themes,
language, isDigital ? EditionType.DIGITAL_EDITION :
EditionType.PRINT_EDITION);
        storage.setData("edition", edition, editionActions);
    }
}
```

EditEdition.java

```
public class EditEdition extends AppCompatActivity {
    private static Edition edition;
    private Storage storage = new Storage();

    public static void setEdition(Edition edition) {
        EditEdition.edition = edition;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.fragment_create_edition);

        EditText name = findViewById(R.id.name_edition);
        EditText publish = findViewById(R.id.publish_edition);
        EditText price = findViewById(R.id.price_edition);
        EditText theme = findViewById(R.id.themes_edition);
        EditText language = findViewById(R.id.language_edition);

        name.setText(edition.getName());
        price.setText(edition.getPrice().toString());
        publish.setText(edition.getPublish());
        theme.setText(edition.getThemes());
        language.setText(edition.getLanguage());

        Button btn = findViewById(R.id.add_edition_btn);
        btn.setText("Редагувати");
        TextView textView = findViewById(R.id.textView2);
        textView.setText("Редагувати видання");
        btn.setOnClickListener(this::editEdition);
    }

    private void editEdition(View v) {

findViewById(R.id.add_edition_btn).setVisibility(View.INVISIB
LE);

        EditText nameEditText = findViewById(R.id.name_edition);
        EditText languageEditText =
findViewById(R.id.language_edition);
        EditText priceEditText = findViewById(R.id.price_edition);
        EditText themesEditText =
findViewById(R.id.themes_edition);
        EditText publishEditText =
findViewById(R.id.publish_edition);
        Switch isDigitalSwitch =
findViewById(R.id.is_digital_switch);

        String name = nameEditText.getText().toString();
        String language = languageEditText.getText().toString();
        Double price =
Double.parseDouble(priceEditText.getText().toString());
        String themes = themesEditText.getText().toString();
        String publish = publishEditText.getText().toString();
        Boolean isDigital = isDigitalSwitch.isChecked();

        if (name.isEmpty() || price == 0 || themes.isEmpty() ||
language.isEmpty() || publish.isEmpty()) {
            Toast.makeText(this, "введіть всі дані",
Toast.LENGTH_LONG).show();
            return;
        }

        Edition edition = new Edition(name, publish, price, themes,
language, isDigital ? EditionType.DIGITAL_EDITION :
EditionType.PRINT_EDITION);
        storage.editData("edition", EditEdition.edition.getId(),
edition);
        finish();
    }
}
```

GetEdition.java

```
public class GetEdition extends AppCompatActivity {
    private static GetEdition instance;

    public static GetEdition getInstance() {
        return instance;
    }

    private ArrayList<Edition> editions;
    private Boolean descSort = false;

    @RequiresApi(api = Build.VERSION_CODES.N)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.fragment_get_edition);
        instance = this;
        Storage storage = new Storage();
        storage.getData("edition", new EditionActions());
        final EditText edittext = (EditText)
findViewById(R.id.search_edition_input);
        edittext.setOnKeyListener(new View.OnKeyListener() {
            public boolean onKey(View v, int keyCode, KeyEvent
event) {
                if ((event.getAction() == KeyEvent.ACTION_DOWN)
&&
                        (keyCode == KeyEvent.KEYCODE_ENTER)) {
                    InputMethodManager imm = (InputMethodManager)
instance.getSystemService(Activity.INPUT_METHOD_SERVICE
);
                    View view = instance.getCurrentFocus();
                    if (view == null) {
                        view = new View(instance);
                    }

imm.hideSoftInputFromWindow(view.getWindowToken(), 0);
                    ArrayList<Edition> temp = new ArrayList<>();
                    String searchText = edittext.getText().toString();
                    for (Edition edition : editions) {
                        if (edition.getName().contains(searchText) ||
                                edition.getLanguage().contains(searchText) ||

edition.getPrice().toString().contains(searchText) ||
                                edition.getThemes().contains(searchText) ||

edition.getEditionTypeAlias().contains(searchText) ||
                                edition.getPublish().contains(searchText)) {
                            temp.add(edition);
                        }
                    }
                    onSuccessListener(temp);
                    return true;
                }
                return false;
            }
        });
        final Spinner spinner = findViewById(R.id.spinner);
        ArrayAdapter<?> adapter =
                ArrayAdapter.createFromResource(this,
R.array.edition_type, android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spin
ner_dropdown_item);
        spinner.setAdapter(adapter);
        spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            public void onItemSelected(AdapterView<?> parent,
                                       View itemSelected, int
selectedItemPosition, long selectedId) {
                if (editions != null) {
                    ArrayList<Edition> temp = new ArrayList<>();
                    for (Edition edition : editions) {
                        if (edition.getEditionType() ==
EditionType.DIGITAL_EDITION && 2 == selectedItemPosition)
{
                            temp.add(edition);
                        }
                        if (edition.getEditionType() ==
EditionType.PRINT_EDITION && 1 == selectedItemPosition) {
                            temp.add(edition);
                        }
                        if (0 == selectedItemPosition) {
                            temp.add(edition);
                        }
                    }
                    onSuccessListener(temp);
                }
            }

            public void onNothingSelected(AdapterView<?> parent) {
            }
        });
        findViewById(R.id.sort_edition_btn).setOnClickListener(v ->
{
            if(!descSort) {
                editions.sort((o1, o2) ->
o1.getName().compareTo(o2.getName()));
                descSort = true;
            }else{
                editions.sort((o1, o2) ->
o2.getName().compareTo(o1.getName()));
                descSort = false;
            }
            onSuccessListener(editions);
        });
    }

    public void onSuccessListener(ArrayList<Edition> editions) {
        if (this.editions == null) {
            this.editions = editions;
        }
        EditionAdapter editionAdapter = new EditionAdapter(this,
editions);
        ListView listView = findViewById(R.id.edition_list);
        listView.setAdapter(editionAdapter);
    }
```

EditEditionList.java

```
public class EditEditionList extends AppCompatActivity {
    private ArrayList<Edition> editions;
    private Boolean descSort = false;

    @RequiresApi(api = Build.VERSION_CODES.N)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.fragment_get_edition);
        EditEditionList instance = this;

        final EditText edittext = (EditText)
findViewById(R.id.search_edition_input);
        edittext.setOnKeyListener(new View.OnKeyListener() {
            public boolean onKey(View v, int keyCode, KeyEvent
event) {
                if ((event.getAction() == KeyEvent.ACTION_DOWN)
&&
                        (keyCode == KeyEvent.KEYCODE_ENTER)) {
                    InputMethodManager imm = (InputMethodManager)
instance.getSystemService(Activity.INPUT_METHOD_SERVICE
);
                    View view = instance.getCurrentFocus();
                    if (view == null) {
                        view = new View(instance);
                    }

imm.hideSoftInputFromWindow(view.getWindowToken(), 0);
                    ArrayList<Edition> temp = new ArrayList<>();
                    String searchText = edittext.getText().toString();
                    for (Edition edition : editions) {
                        if (edition.getName().contains(searchText) ||
                                edition.getLanguage().contains(searchText) ||

edition.getPrice().toString().contains(searchText) ||
                                edition.getThemes().contains(searchText) ||

edition.getEditionTypeAlias().contains(searchText) ||
                                edition.getPublish().contains(searchText)) {
                            temp.add(edition);
                        }
                    }
                    onSuccessListener(temp);
                    return true;
                }
                return false;
            }
        });
        final Spinner spinner = findViewById(R.id.spinner);
        ArrayAdapter<?> adapter =
```

```
            ArrayAdapter.createFromResource(this,
R.array.edition_type, android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spin
ner_dropdown_item);
        spinner.setAdapter(adapter);
        spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            public void onItemSelected(AdapterView<?> parent,
                                       View itemSelected, int
selectedItemPosition, long selectedId) {
                if (editions != null) {
                    ArrayList<Edition> temp = new ArrayList<>();
                    for (Edition edition : editions) {
                        if (edition.getEditionType() ==
EditionType.DIGITAL_EDITION && 2 == selectedItemPosition)
{
                            temp.add(edition);
                        }
                        if (edition.getEditionType() ==
EditionType.PRINT_EDITION && 1 == selectedItemPosition) {
                            temp.add(edition);
                        }
                        if (0 == selectedItemPosition) {
                            temp.add(edition);
                        }
                    }
                    onSuccessListener(temp);
                }
            }

            public void onNothingSelected(AdapterView<?> parent) {
            }
        });
        findViewById(R.id.sort_edition_btn).setOnClickListener(v ->
{
            if(!descSort) {
                editions.sort((o1, o2) ->
o1.getName().compareTo(o2.getName()));
                descSort = true;
            }else{
                editions.sort((o1, o2) ->
o2.getName().compareTo(o1.getName()));
                descSort = false;
            }
            onSuccessListener(editions);
        });

onSuccessListener(AddSubscription.getInstance().getEditions());
    }

    public void onSuccessListener(ArrayList<Edition> editions) {
        if (this.editions == null) {
            this.editions = editions;
        }
        EditionAdapter editionAdapter = new EditionAdapter(this,
editions);
        ListView listView = findViewById(R.id.edition_list);
        listView.setAdapter(editionAdapter);
    }

}
```

EditEditionAdapter.java

```
public class EditionAdapter extends BaseAdapter {

    private Activity activity;
    private ArrayList<Edition> editions;
    private LayoutInflater layoutInflater;

    public EditionAdapter(Activity activity, ArrayList<Edition>
editions) {
        this.activity = activity;
        this.editions = editions;
        layoutInflater = (LayoutInflater)
activity.getSystemService(Context.LAYOUT_INFLATER_SERVI
CE);
    }

    @Override
    public int getCount() {
        return editions.size();
    }

    @Override
    public Object getItem(int position) {
        return editions.get(position);
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View view, ViewGroup
viewGroup) {
        view = layoutInflater.inflate(R.layout.edition_list_element,
null);
        TextView publishTextView =
view.findViewById(R.id.publish);
        TextView languageTextView =
view.findViewById(R.id.language);
        TextView priceTextView = view.findViewById(R.id.price);
        TextView themesTextView =
view.findViewById(R.id.themes);
        TextView nameTextView = view.findViewById(R.id.name);
        TextView typeTextView = view.findViewById(R.id.type);

        publishTextView.setText("Видавництво: " +
editions.get(position).getPublish());
        languageTextView.setText("Мова: " +
editions.get(position).getLanguage());
        priceTextView.setText("Ціна: " +
editions.get(position).getPrice() + " грн");
        themesTextView.setText("Тема: " +
editions.get(position).getThemes());
        nameTextView.setText("Ім'я: " +
editions.get(position).getName());
        typeTextView.setText("Тип: " +
editions.get(position).getEditionTypeAlias());

        view.setOnClickListener(v -> {
            AlertDialog.Builder builder = new
AlertDialog.Builder(activity);
            builder.setTitle("Інформація").setMessage("Що ви
бажаєте зробити?").setPositiveButton("Редагувати", new
DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {
                    EditEdition.setEdition(editions.get(position));
                    Intent intent = new Intent(activity, EditEdition.class);
                    activity.startActivity(intent);
                    activity.finish();
                }
            }).setNeutralButton("Відміна", new
DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {
                }
            }).setNegativeButton("Видалити", new
DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {
                    new Storage().removeData("edition",
editions.get(position).getId(), new EditionActions());
                }
            }).create().show();

        });
        return view;
    }
}
```

AddSubscriber.java

```
public class AddSubscriber extends AppCompatActivity {
    private static AddSubscriber instance;

    public static AddSubscriber getInstance() {
        return instance;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.fragment_add_subscriber);
        instance = this;

findViewById(R.id.add_subscriber).setOnClickListener(this::addS
ubscriber);
    }

    public void addSubscriber(View v) {
        Storage storage = new Storage();

findViewById(R.id.add_subscriber).setVisibility(View.INVISIBL
E);
        SubscriberActions subscriberActions = new
SubscriberActions();

        EditText fullNameEditText =
findViewById(R.id.subscriber_full_name);
        EditText addressEditText =
findViewById(R.id.subscriber_address);
        EditText phoneNumberEditText =
findViewById(R.id.subscriber_phone_number);

        String fullName = fullNameEditText.getText().toString();
        String address = addressEditText.getText().toString();
        String phoneNumber =
phoneNumberEditText.getText().toString();

        if (fullName.isEmpty() || address.isEmpty() ||
phoneNumber.isEmpty()) {
            Toast.makeText(this, "введіть всі дані",
Toast.LENGTH_LONG).show();
            return;
        }

        Subscriber subscriber = new Subscriber(fullName, address,
phoneNumber);
        storage.setData("subscriber", subscriber, subscriberActions);
    }
}
```

EditSubscriber.java

```
public class EditSubscriber extends AppCompatActivity {
    private static Subscriber subscriber;
    private Storage storage = new Storage();

    public static void setSubscriber(Subscriber subscriber) {
        EditSubscriber.subscriber = subscriber;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.fragment_add_subscriber);

        EditText fullName =
findViewById(R.id.subscriber_full_name);
        EditText address = findViewById(R.id.subscriber_address);
        EditText phoneNumber =
findViewById(R.id.subscriber_phone_number);

        fullName.setText(subscriber.getFullName());
        address.setText(subscriber.getAddress());
        phoneNumber.setText(subscriber.getPhoneNumber());

        Button btn = findViewById(R.id.add_subscriber);
        btn.setText("Редагувати");
        btn.setOnClickListener(this::editSubscriber);

        TextView textView = findViewById(R.id.textView);
        textView.setText("Редагувати передплатника");
    }

    private void editSubscriber(View v) {

findViewById(R.id.add_subscriber).setVisibility(View.INVISIBL
E);

        EditText fullNameField =
findViewById(R.id.subscriber_full_name);
        EditText addressField =
findViewById(R.id.subscriber_address);
        EditText phoneNumberField =
findViewById(R.id.subscriber_phone_number);

        String fullName = fullNameField.getText().toString();
        String address = addressField.getText().toString();
        String phoneNumber =
phoneNumberField.getText().toString();

        if (fullName.isEmpty() || address.isEmpty() ||
phoneNumber.isEmpty()) {
            Toast.makeText(this, "введіть всі дані",
Toast.LENGTH_LONG).show();
            return;
        }

        Subscriber subscriber = new Subscriber(fullName, address,
phoneNumber);
        storage.editData("subscriber",
EditSubscriber.subscriber.getId(), subscriber);
        finish();
    }

    private void removeSubscriber(View v) {
        storage.removeData("subscriber",
EditSubscriber.subscriber.getId());
        finish();
    }
}
```

GetSubscriber.java

```
public class GetSubscriber extends AppCompatActivity {
    private static GetSubscriber instance;

    public static GetSubscriber getInstance(){
        return instance;
    }
    private ArrayList<Subscriber> subscribers = new ArrayList<>();
    private Boolean descSort = false;


    @RequiresApi(api = Build.VERSION_CODES.N)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.fragment_show_subscriber);
        instance = this;
        Storage storage = new Storage();
        storage.getData("subscriber", new SubscriberActions());
        final EditText edittext = (EditText)
findViewById(R.id.search_subscriber_input);
        edittext.setOnKeyListener(new View.OnKeyListener() {
            public boolean onKey(View v, int keyCode, KeyEvent
event) {
                if ((event.getAction() == KeyEvent.ACTION_DOWN)
&&
                        (keyCode == KeyEvent.KEYCODE_ENTER)) {
                    InputMethodManager imm = (InputMethodManager)
instance.getSystemService(Activity.INPUT_METHOD_SERVICE
);
                    View view = instance.getCurrentFocus();
```

```
                if (view == null) {
                    view = new View(instance);
                }

imm.hideSoftInputFromWindow(view.getWindowToken(), 0);
                ArrayList<Subscriber> temp = new ArrayList<>();
                String searchText = edittext.getText().toString();
                for (Subscriber subscriber : subscribers) {
                    if (subscriber.getFullName().contains(searchText) ||
subscriber.getPhoneNumber().contains(searchText) ||
subscriber.getAddress().contains(searchText)) {
                        temp.add(subscriber);
                    }
                }
                onSuccessListener(temp);
                return true;
            }
            return false;
        }
    });

findViewById(R.id.sort_subscriber_btn).setOnClickListener(v -> {
        if(!descSort) {
            subscribers.sort((o1, o2) ->
o1.getFullName().compareTo(o2.getFullName()));
            descSort = true;
        }else{
            subscribers.sort((o1, o2) ->
o2.getFullName().compareTo(o1.getFullName()));
            descSort = false;
        }
        onSuccessListener(subscribers);
    });
  }

  public void onSuccessListener(ArrayList<Subscriber>
subscribers) {
    if(this.subscribers!=null){
      this.subscribers = subscribers;
    }
    SubscriberAdapter subscriberAdapter = new
SubscriberAdapter(this, subscribers);
    ListView listView = findViewById(R.id.subscriber_list);
    listView.setAdapter(subscriberAdapter);
  }
}
```

SubscriberActions.java

```
public class SubscriberActions implements IActions {
  @Override
  public void onSuccessListener() {
    AddSubscriber addSubscriber = AddSubscriber.getInstance();
    Toast.makeText(addSubscriber, "Success",
Toast.LENGTH_LONG).show();
    addSubscriber.finish();
  }

  @Override
  public void onSuccessListener(QuerySnapshot
documentSnapshot) {
    GetSubscriber getSubscriber = GetSubscriber.getInstance();
    List<DocumentSnapshot> documents =
documentSnapshot.getDocuments();
    ArrayList<Subscriber> subscribers = new ArrayList<>();
    for (DocumentSnapshot element : documents) {
      Subscriber subscriber = new
Subscriber(element.getString("fullName"),
element.getString("address"), element.getString("phoneNumber"));
      subscriber.setId(element.getId());
      subscribers.add(subscriber);
    }
    getSubscriber.onSuccessListener(subscribers);
  }

  @Override
  public void onFailListener() {
    AddSubscriber addSubscriber = AddSubscriber.getInstance();
    Toast.makeText(addSubscriber, "Fail",
Toast.LENGTH_LONG).show();

addSubscriber.findViewById(R.id.add_subscriber).setVisibility(Vi
ew.VISIBLE);
  }
}
```

SubscriberAdapter.java

```
public class SubscriberAdapter extends BaseAdapter {

  private Activity activity;
  private ArrayList<Subscriber> subscribers;
  private LayoutInflater layoutInflater;

  public SubscriberAdapter(Activity activity,
ArrayList<Subscriber> subscribers) {
    this.activity = activity;
    this.subscribers = subscribers;
    layoutInflater = (LayoutInflater)
activity.getSystemService(Context.LAYOUT_INFLATER_SERVI
CE);
  }

  @Override
  public int getCount() {
    return subscribers.size();
  }

  @Override
  public Object getItem(int position) {
    return subscribers.get(position);
  }

  @Override
  public long getItemId(int position) {
    return position;
  }

  @Override
  public View getView(int position, View view, ViewGroup
viewGroup) {
    view = layoutInflater.inflate(R.layout.subscriber_list_element,
null);
    TextView fullNameTextView =
view.findViewById(R.id.subscriber_list_full_name);
    TextView addressTextView =
view.findViewById(R.id.subscriber_list_address);
    TextView phoneNumberTextView =
view.findViewById(R.id.subscriber_list_phone_number);


fullNameTextView.setText(subscribers.get(position).getFullName(
));
    addressTextView.setText("Адреса: " +
subscribers.get(position).getAddress());
    phoneNumberTextView.setText("Телефон: " +
subscribers.get(position).getPhoneNumber());

    view.setOnClickListener(v -> {
      AlertDialog.Builder builder = new
AlertDialog.Builder(activity);
      builder.setTitle("Інформація").setMessage("Що ви
бажаєте зробити?").setPositiveButton("Редагувати", new
DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {

EditSubscriber.setSubscriber(subscribers.get(position));
          Intent intent = new Intent(activity,
EditSubscriber.class);
          activity.startActivity(intent);
          activity.finish();
        }
      }).setNeutralButton("Відміна", new
DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
        }
```

```
        }).setNegativeButton("Видалити", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                new Storage().removeData("subscriber",
subscribers.get(position).getId(), new SubscriberActions());
            }
        }).create().show();
    });
    return view;
  }
}
```

AddSubscription.java

```
public class AddSubscription extends AppCompatActivity {
  private static AddSubscription instance;

  public static AddSubscription getInstance() {
    return instance;
  }

  private ArrayList<Edition> editions;

  public ArrayList<Edition> getEditions() {
    return editions;
  }

  private String[] labels;
  private double price;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.fragment_add_subscription);
    instance = this;

findViewById(R.id.add_subscription_picker).setVisibility(View.I
NVISIBLE);

findViewById(R.id.subscription_add_edition).setOnClickListener(
this::toAddEdition);

findViewById(R.id.subscription_add_data).setOnClickListener(thi
s::addData);

findViewById(R.id.subscription_edit_edition_list).setOnClickList
ener(this::toEditEdition);
    Storage storage = new Storage();
    storage.getData("subscriber", new
AddSubscriptionActions());
    editions = new ArrayList<>();
    ((EditText)
findViewById(R.id.start)).setOnEditorActionListener(
        new TextView.OnEditorActionListener() {
          @Override
          public boolean onEditorAction(TextView v, int
actionId, KeyEvent event) {
            if (actionId ==
EditorInfo.IME_ACTION_SEARCH ||
                actionId == EditorInfo.IME_ACTION_DONE
||
                event != null &&
                    event.getAction() ==
KeyEvent.ACTION_DOWN &&
                    event.getKeyCode() ==
KeyEvent.KEYCODE_ENTER) {
              if (event == null || !event.isShiftPressed()) {
                // the user is done typing.
                return true; // consume.
              }
            }
            calculatePrice();
            return false; // pass on to other listeners.
          }
        }
    );
    ((EditText)
findViewById(R.id.end)).setOnEditorActionListener(
        new TextView.OnEditorActionListener() {
          @Override
          public boolean onEditorAction(TextView v, int
actionId, KeyEvent event) {
            if (actionId ==
EditorInfo.IME_ACTION_SEARCH ||
                actionId == EditorInfo.IME_ACTION_DONE
||
                event != null &&
                    event.getAction() ==
KeyEvent.ACTION_DOWN &&
                    event.getKeyCode() ==
KeyEvent.KEYCODE_ENTER) {
              if (event == null || !event.isShiftPressed()) {
                // the user is done typing.
                return true; // consume.
              }
            }
            calculatePrice();
            return false; // pass on to other listeners.
          }
        }
    );
  }

  public void onSuccessListener(ArrayList<Subscriber>
subscribers) {
    NumberPicker numberPicker =
findViewById(R.id.add_subscription_picker);
    String[] labels = new String[subscribers.size()];
    this.labels = new String[subscribers.size()];
    for (int i = 0; i < subscribers.size(); i++) {
      String fullName = subscribers.get(i).getFullName();
      this.labels[i] = fullName;
      if (fullName.length() < 23) {
        labels[i] = (fullName);
      } else {
        labels[i] = fullName.substring(0, 23) + "...";
      }
    }
    numberPicker.setMinValue(1);
    numberPicker.setMaxValue(labels.length);
    numberPicker.setDisplayedValues(labels);
    numberPicker.setValue(1);
    numberPicker.setVisibility(View.VISIBLE);

findViewById(R.id.subscription_progressBar).setVisibility(View.I
NVISIBLE);
  }

  private void toAddEdition(View v) {
    Intent intent = new Intent(getApplicationContext(),
EditionList.class);
    startActivity(intent);
  }

  private void toEditEdition(View v) {
    Intent intent = new Intent(getApplicationContext(),
EditEditionList.class);
    startActivity(intent);
  }

  private void calculatePrice() {
    SimpleDateFormat format = new
SimpleDateFormat("dd/MM/yyyy");
    if (editions.size() == 0) {
      ((TextView)
findViewById(R.id.subscription_price)).setText("");
    }
    String startDateString = ((TextView)
findViewById(R.id.start)).getText().toString();
    String endDateString = ((TextView)
findViewById(R.id.end)).getText().toString();
    try {
      Date startDate = format.parse(startDateString);
      Date endDate = format.parse(endDateString);
```

```
        if (!editions.isEmpty() && startDate != null && endDate
!= null) {
            double price = 0;
            for (Edition edition : editions) {
               Log.i("edition", String.valueOf(edition.getPrice()));
               price += edition.getPrice();
            }
            long month = (endDate.getTime() - startDate.getTime())
/ Long.valueOf("2592000000");
            price *= month;
            ((TextView)
findViewById(R.id.subscription_price)).setText(String.valueOf(pri
ce));
            this.price = price;
         }
      } catch (ParseException e) {
         e.printStackTrace();
         ((TextView)
findViewById(R.id.subscription_price)).setText("");
         new
AlertDialog.Builder(this).setTitle("Помилка").setMessage("Введі
ть корректно дату").setPositiveButton("OK", null).show();
      }
   }

   public void addEdition(Edition edition) {
      editions.add(edition);
      calculatePrice();
   }

   public void removeEdition(int position) {
      editions.remove(position);
      calculatePrice();
   }

   private void addData(View v) {

findViewById(R.id.subscription_add_data).setVisibility(View.IN
VISIBLE);
      calculatePrice();
      TextView mailIndexTextView =
findViewById(R.id.subscription_mail_index);
      NumberPicker numberPicker =
findViewById(R.id.add_subscription_picker);

      int index = numberPicker.getValue() - 1;
      String mailIndex = mailIndexTextView.getText().toString();

      ArrayList<String> list = new ArrayList<>();

      for (Edition edition : editions) {
         list.add(edition.getId());
      }

      if (mailIndex.isEmpty() || list.isEmpty()) {
         Toast.makeText(this, "Будь ласка, введіть дані!",
Toast.LENGTH_LONG).show();
         return;
      }
      SimpleDateFormat format = new
SimpleDateFormat("dd/MM/yyyy");
      String startDateString = ((TextView)
findViewById(R.id.start)).getText().toString();
      String endDateString = ((TextView)
findViewById(R.id.end)).getText().toString();
      try {
         Date startDate = format.parse(startDateString);
         Date endDate = format.parse(endDateString);
         Subscription subscription = new
Subscription(labels[index], mailIndex, list, startDate, endDate,
price);
         Storage storage = new Storage();
         storage.setData("subscription", subscription, new
AddSubscriptionActions());
      } catch (Exception e) {
         new
AlertDialog.Builder(this).setTitle("Помилка").setMessage("Введі
ть корректно дату").setPositiveButton("OK", null).show();
      }
   }
}
```

GetSubscription.java

```
public class GetSubscription extends AppCompatActivity {
   private static GetSubscription instance;

   public static GetSubscription getInstance() {
      return instance;
   }

   private ArrayList<Subscription> subscriptions;
   private Boolean descSort = false;


   @RequiresApi(api = Build.VERSION_CODES.N)
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.fragment_show_subscription);
      instance = this;
      Storage storage = new Storage();
      storage.getData("subscription", new SubscriptionActions());
      final EditText edittext = (EditText)
findViewById(R.id.search_subscription_input);
      edittext.setOnKeyListener(new View.OnKeyListener() {
         public boolean onKey(View v, int keyCode, KeyEvent
event) {
            if ((event.getAction() == KeyEvent.ACTION_DOWN)
&&
                  (keyCode == KeyEvent.KEYCODE_ENTER)) {
               InputMethodManager imm = (InputMethodManager)
instance.getSystemService(Activity.INPUT_METHOD_SERVICE
);
               View view = instance.getCurrentFocus();
               if (view == null) {
                  view = new View(instance);
               }

imm.hideSoftInputFromWindow(view.getWindowToken(), 0);
               ArrayList<Subscription> temp = new ArrayList<>();
               String searchText = edittext.getText().toString();
               for (Subscription subscription : subscriptions) {
                  if (subscription.getFullName().contains(searchText)
|| String.valueOf(subscription.getPrice()).contains(searchText) ||
subscription.getMailIndex().contains(searchText)) {
                     temp.add(subscription);
                  }
               }
               onSuccessListener(temp);
               return true;
            }
            return false;
         }
      });

findViewById(R.id.sort_subscription_btn).setOnClickListener(v -
> {
         if(!descSort) {
            subscriptions.sort((o1, o2) ->
o1.getFullName().compareTo(o2.getFullName()));
            descSort = true;
         }else{
            subscriptions.sort((o1, o2) ->
o2.getFullName().compareTo(o1.getFullName()));
            descSort = false;
         }
         onSuccessListener(subscriptions);
      });
   }
```

SubscriptionAction.java

```
    public void onSuccessListener(ArrayList<Subscription>
subscriptions) {
        if (this.subscriptions == null) {
            this.subscriptions = subscriptions;
        }
        SubscriptionAdapter subscriberAdapter = new
SubscriptionAdapter(this, subscriptions);
        ListView listView = findViewById(R.id.subscription_list);
        listView.setAdapter(subscriberAdapter);
    }
}

public class SubscriptionActions implements IActions {
    @Override
    public void onSuccessListener() {

    }

    @Override
    public void onSuccessListener(QuerySnapshot
documentSnapshot) {
        GetSubscription getSubscription =
GetSubscription.getInstance();
        List<DocumentSnapshot> documents =
documentSnapshot.getDocuments();
        ArrayList<Subscription> subscriptions = new ArrayList<>();
        for (DocumentSnapshot element : documents) {
            Subscription subscription = new
Subscription(element.getString("fullName"),
element.getString("mailIndex"), (ArrayList) element.get("idList"),
element.getDate("start"),element.getDate("end"),
element.getDouble("price"));
            subscription.setId(element.getId());
            subscriptions.add(subscription);
        }
        getSubscription.onSuccessListener(subscriptions);
    }

    @Override
    public void onFailListener() {
        GetSubscription getSubscription =
GetSubscription.getInstance();
        Toast.makeText(getSubscription, "Fail",
Toast.LENGTH_LONG).show();
    }
}
```

AddSubscriptionAction.java

```
public class AddSubscriptionActions implements IActions {
    @Override
    public void onSuccessListener() {
        AddSubscription addSubscription =
AddSubscription.getInstance();
        Toast.makeText(addSubscription.getApplicationContext(),
"Success", Toast.LENGTH_LONG).show();
        addSubscription.finish();
    }

    @Override
    public void onSuccessListener(QuerySnapshot
documentSnapshot) {
        AddSubscription addSubscription =
AddSubscription.getInstance();
        List<DocumentSnapshot> documents =
documentSnapshot.getDocuments();
        ArrayList<Subscriber> subscribers = new ArrayList<>();
        for (DocumentSnapshot element : documents) {
            Subscriber subscriber = new
Subscriber(element.getString("fullName"),
element.getString("address"), element.getString("phoneNumber"));
            subscribers.add(subscriber);
        }
        addSubscription.onSuccessListener(subscribers);
    }

    @Override
    public void onFailListener() {
        AddSubscription addSubscription =
AddSubscription.getInstance();
        Toast.makeText(addSubscription.getApplicationContext(),
"Fail", Toast.LENGTH_LONG).show();

addSubscription.findViewById(R.id.subscription_add_data).setVis
ibility(View.VISIBLE);
    }
}
```

BuildConfig.java

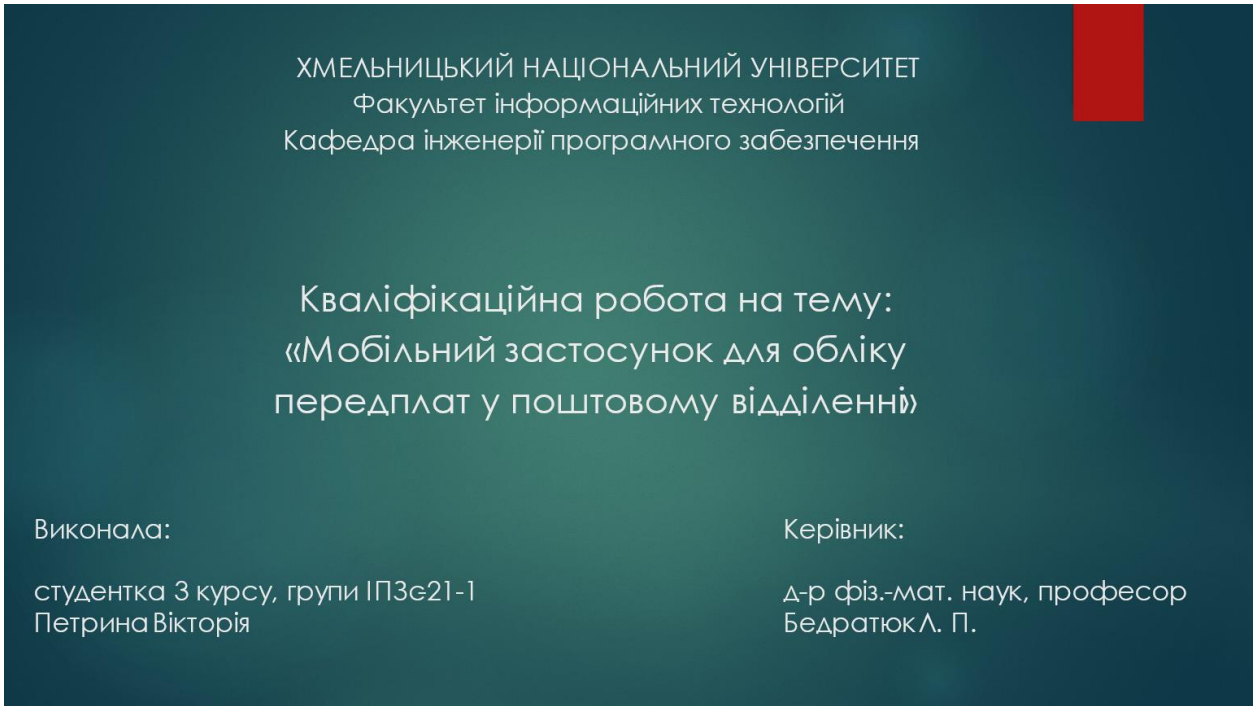
```
package com.hpk.mail;

public final class BuildConfig {
  public static final boolean DEBUG =
Boolean.parseBoolean("true");
  public static final String APPLICATION_ID = "com.hpk.mail";
  public static final String BUILD_TYPE = "debug";
  public static final int VERSION_CODE = 1;
  public static final String VERSION_NAME = "1.0";
}
```

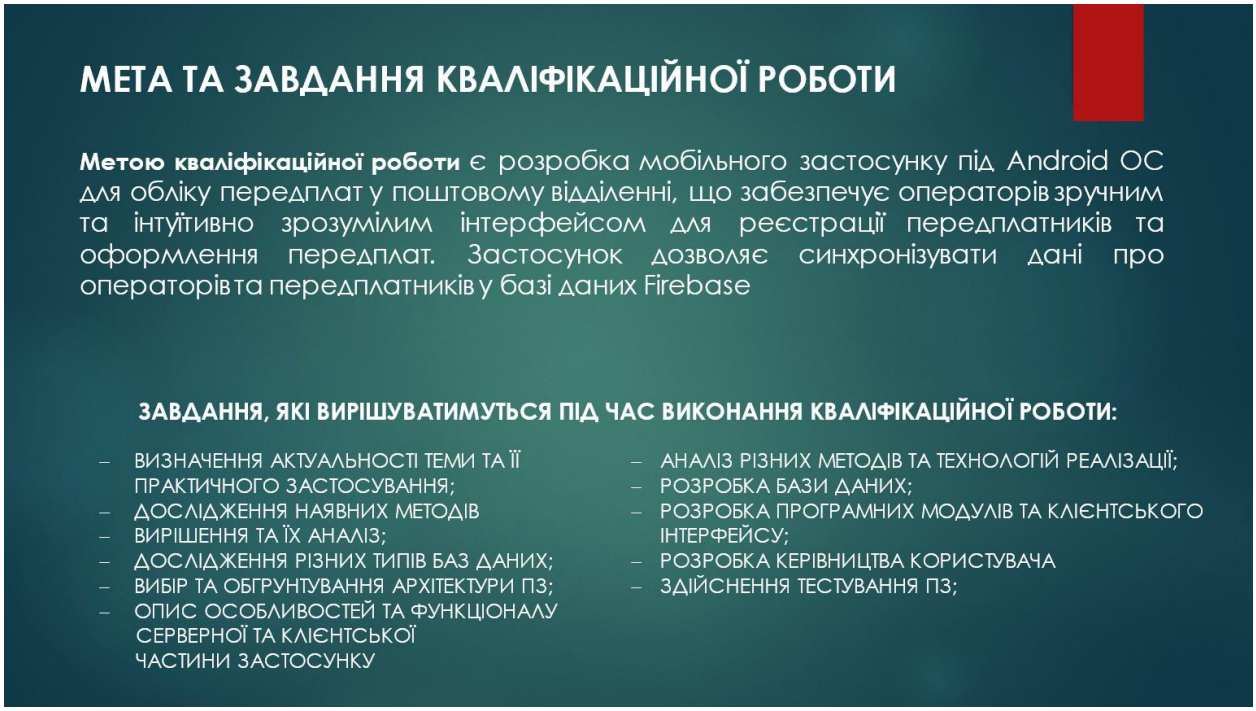
EditUser.java

```
public class EditUser extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.fragment_edit_user);
        CurrentUser currentUser = CurrentUser.getCurrentInstance();
        ((EditText)
findViewById(R.id.edit_full_name)).setText(currentUser.getUsern
ame());
        ((EditText)
findViewById(R.id.edit_address)).setText(currentUser.getAddress
());

findViewById(R.id.edit_user_btn).setOnClickListener(this::edit);
    }

    private void edit(View v) {
        FirebaseAuth firebaseInstance = FirebaseAuth.getInstance();
        String id = CurrentUser.getCurrentInstance().getId();
        String fullName = ((EditText)
findViewById(R.id.edit_full_name)).getText().toString();
        String address = ((EditText)
findViewById(R.id.edit_address)).getText().toString();
        String email = firebaseInstance.getCurrentUser().getEmail();

        CurrentUser currentUser =
CurrentUser.getNewInstance(email, fullName, address, id);


FirebaseFirestore.getInstance().collection("user_info").document(i
d).update("address", currentUser.getAddress(), "username",
currentUser.getUsername()).addOnSuccessListener((doc) ->
this.finish()).addOnFailureListener((doc) -> Toast.makeText(this,
"Помилка оновлення", Toast.LENGTH_LONG).show());
    }

}
```

ДОДАТОК Б

(графічна частина)





АКТУАЛЬНІСТЬ ТЕМИ

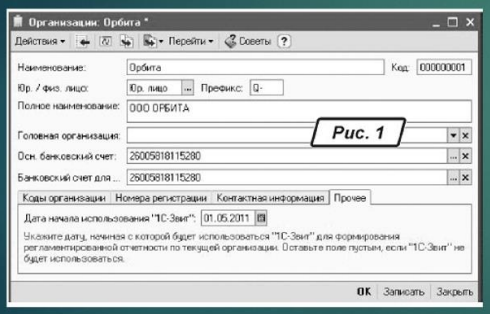
Багато підприємств активно користуються спеціалізованими періодичними виданнями у своїй діяльності. Ці видання можуть містити важливу для бізнесу інформацію, новини, аналізи та інші матеріали, які сприяють розвитку компанії та підвищенню професійної компетентності співробітників.

Передплата періодичних видань в Україні зазвичай оформлюється
- у відділеннях пошти;
- у листонош на доставних дільницях;
- передплатних агенціях;

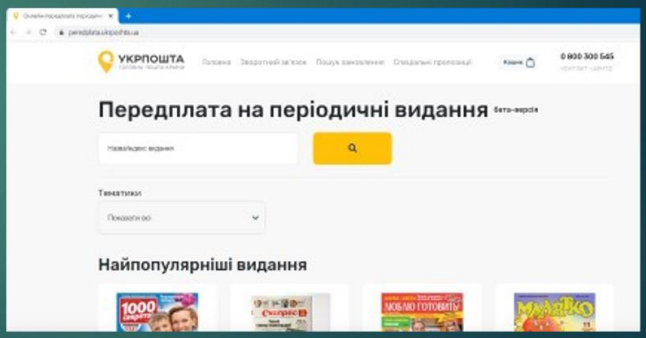
Сьогодні широкої популярності набуває підписка на видання в електронній формі, які можуть доставлятися передплатнику за допомогою Інтернету. Цей вид підписки відрізняється від традиційної паперової форми тим, що доставляється передплатнику практично миттєво. Із тенденцією переходу на використання такої послуги, як онлайн передплата, зростає попит на спрощення реалізації її оформлення. Саме тому обрана тема дипломного проекту є актуальною і затребуваною.

НАЯВНІ МЕТОДИ ВИРІШЕННЯ

«АРМ Вз»



Сайт Укрпошти

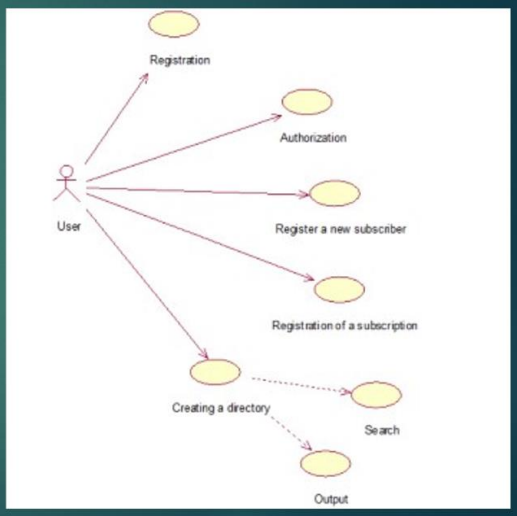


ОСНОВНІ АКТОРИ

Визначення основних акторів та побудова діаграми варіантів використання - етапи, що значно спрощують визначення вимог до програмного забезпечення

На Use Case діаграмі зображено, що в моїй програмі користувач може

- авторизуватися в уже наявний профіль або зареєструвати новий,
- зареєструвати передплатника або обрати наявного,
- оформити передплату або редагувати вже наявну,
- переглядати каталог видань, додавати нові видання в каталог,
- виконувати пошук в каталозі.

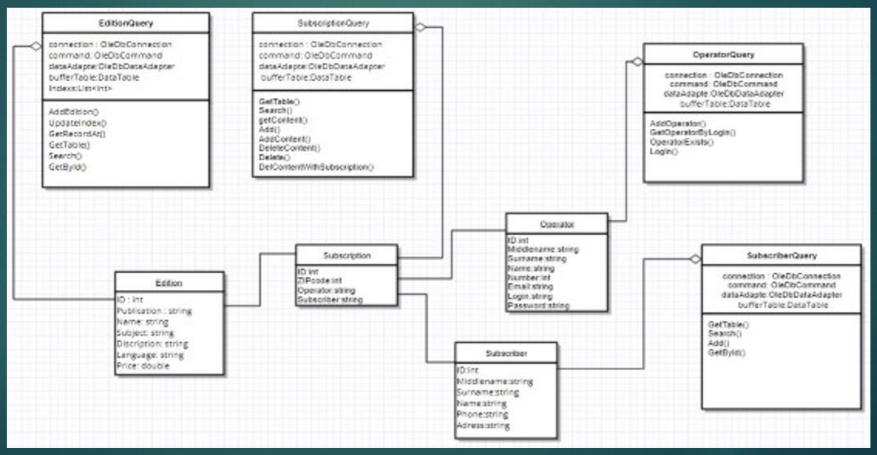


ВИЗНАЧЕННЯ ВИМОГ

- Авторизація або реєстрація працівника в програмі;
- Створення, редагування та видалення видання
- Фільтрація, сортування та пошук в каталозі товарів
- Створення, редагування та видалення передплатників
- Пошук серед передплатників
- Створення, видалення передплати
- Пошук серед наявної передплати

Логічна модель бази даних

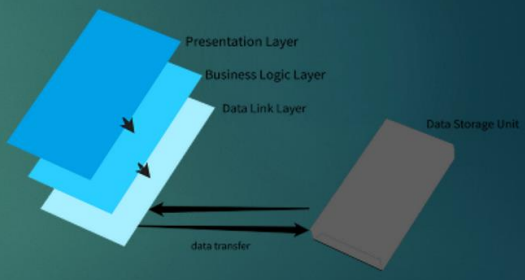
На основі усіх зібраних даних, після з'ясування усіх сутностей, що будуть у базі даних, з'ясування атрибутів, що будуть у кожній сутності, визначення обмежень, які будуть застосовуватися для сутностей і проведення нормалізації бази даних було побудовано логічну модель бази даних.

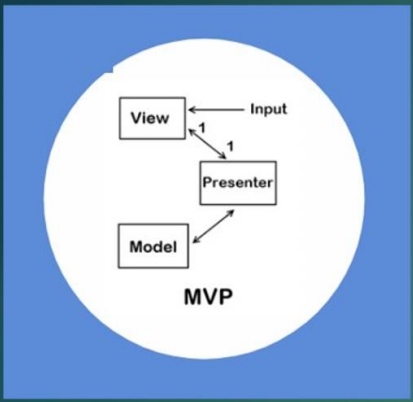


Архітектура мобільного додатку

Багаторівнева архітектура є чи не найпоширенішою, адже слугує як спосіб ефективного зберігання даних в таблицях. Цей підхід працює за принципом поділу відповідальності. ПЗ розділено на шари, що лежать один на одному, і кожен з них виконує певний обов'язок.

- Рівень представлення (Presentation Layer) містить інтерфейс користувача і відповідає за забезпечення зрозумілого і ефективного користування програмою.
- Рівень бізнес-логіки (Business Logic Layer), як випливає з назви, містить бізнес-логіку програми.
- Рівень передачі даних (Data Link Layer) відповідає за взаємодію з постійними сховищами, такими як бази даних.





У моєму проєкті використовується патерн MVP. Model View Presenter — це патерн, заснований на патерні MVC. У MVP презентер такий самий, як контролер у патерні MVC. Проте є деяка відмінність: вхідною точкою в додаток є представлення і дані не передаються безпосередньо з моделі до представлення, як у MVC. Презентер запитує модель, модель повертає дані презентеру, презентер обробляє отримані дані та передає їх у представлення.

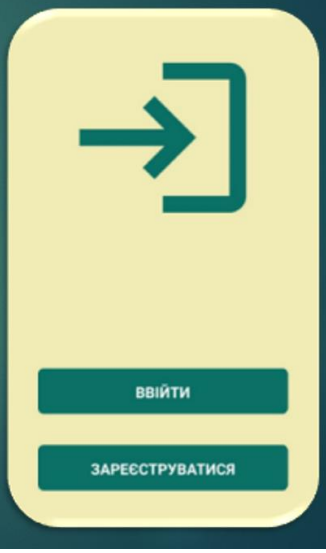
Використовують MVP, коли програма має двонаправлений потік. Якщо при взаємодії з користувачем необхідно щось запитати з форми, і результат цього запиту негайно змінить інтерфейс користувача, слід використовувати патерн MVP.

Засоби розробки ПЗ

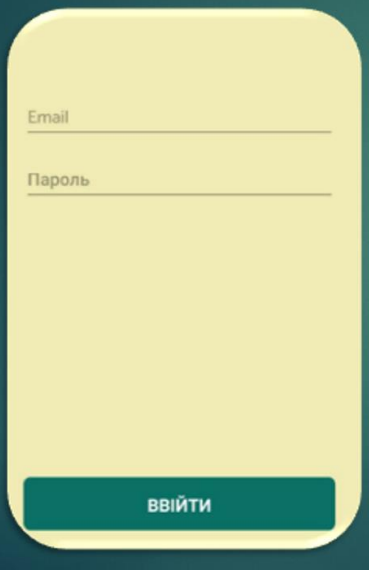
Для реалізації проєкту було вибрано об'єктно-орієнтовану мову програмування Java. Оскільки програмне забезпечення було призначено виключно на операційну систему Android то Java є найкращим варіантом мови програмування для реалізації цього проєкту.
Для зберігання даних використано базу даних, створену за допомогою платформи Firebase. Firebase – це платформа для розробки мобільних і веб-додатків, що дає змогу швидко створювати високоякісні додатки, залучати лояльних користувачів і збільшувати прибутки. До платформи включено комплекс інтегрованих функцій, які можна поєднувати й суміщати, зокрема мобільний сервер, засоби аналітики, а також інструменти вдосконалення й монетизації для максимальної успішності додатків.

Реалізація додатку

Форма входу або
реєстрації користувача

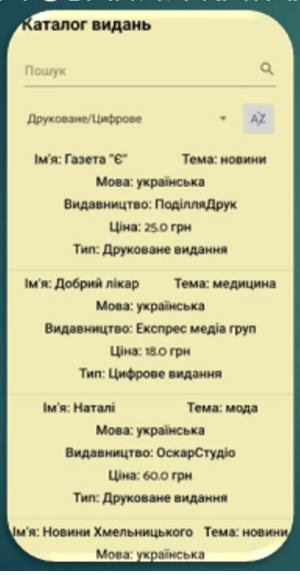


Форма аутентифікації користувача

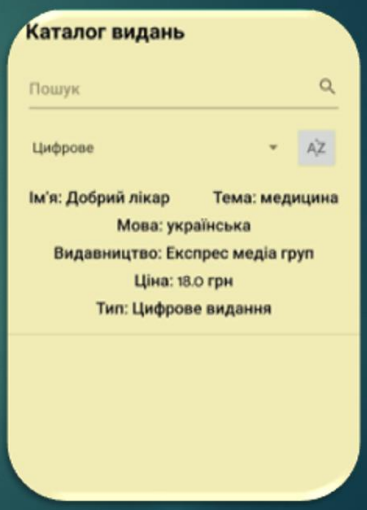


Каталог видань

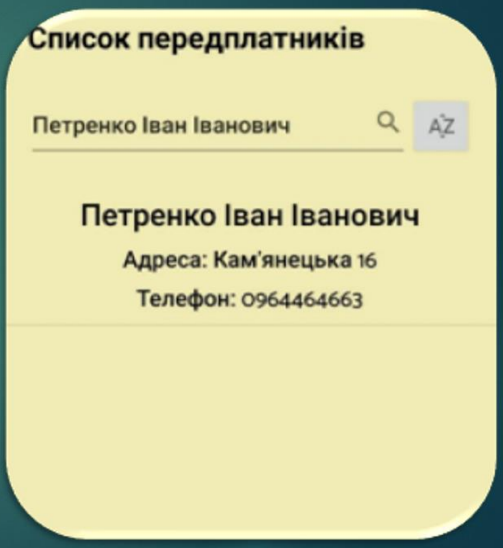
ВІДСОРТОВАНИЙ КАТАЛОГ



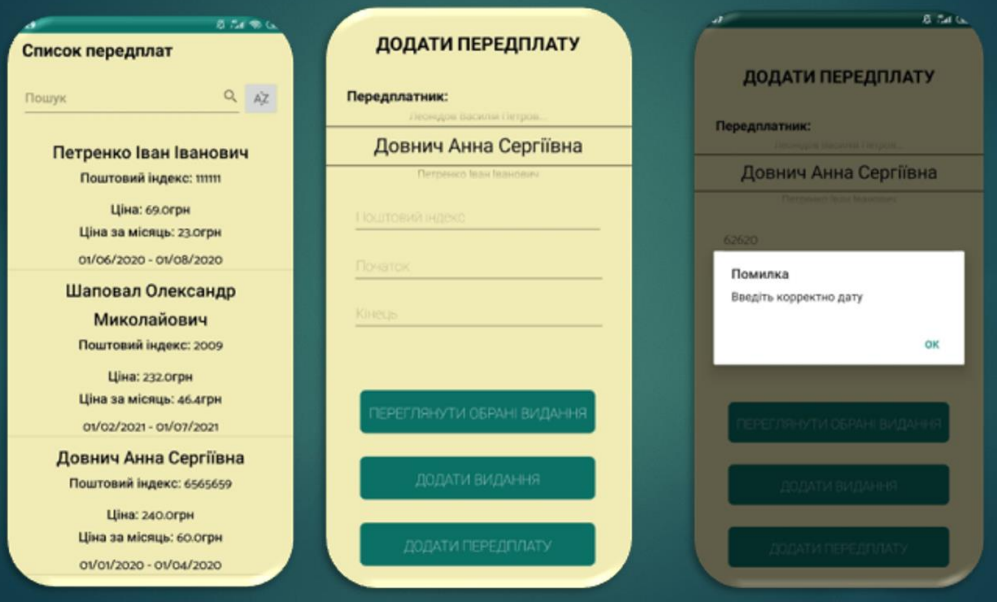
ВІДФІЛЬТРОВАНИЙ КАТАЛОГ



Список передплатників



Оформлення нової передплати



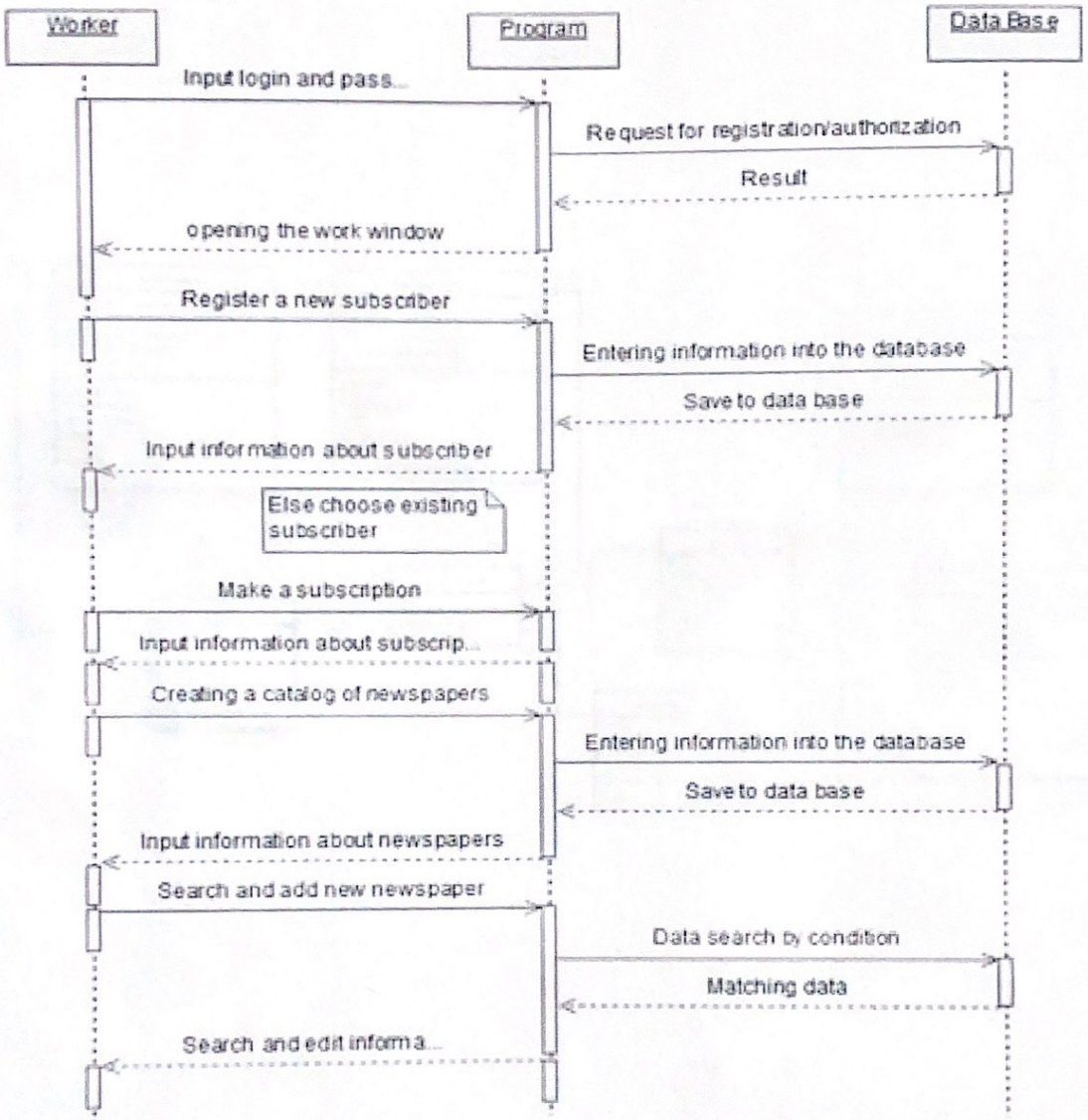
Висновки

У процесі виконання кваліфікаційної роботи було реалізовано мобільний застосунок для обліку передплат у поштовому відділенні під операційну систему Android. Під час розробки було проведено аналіз вимог предметної області, аналіз існуючих рішень, визначено їх основні переваги та недоліки, для врахування під час реалізації додатку. Проведено аналіз засобів розробки та здійснено їх обґрунтований вибір.

Було здійснено проєктування, написання коду до відповідних елементів програми, було створено логіку програми, логіку самого користувача. Для коректного відображання процесів та легкого переведення діаграм в програмний код були створені діаграми UML

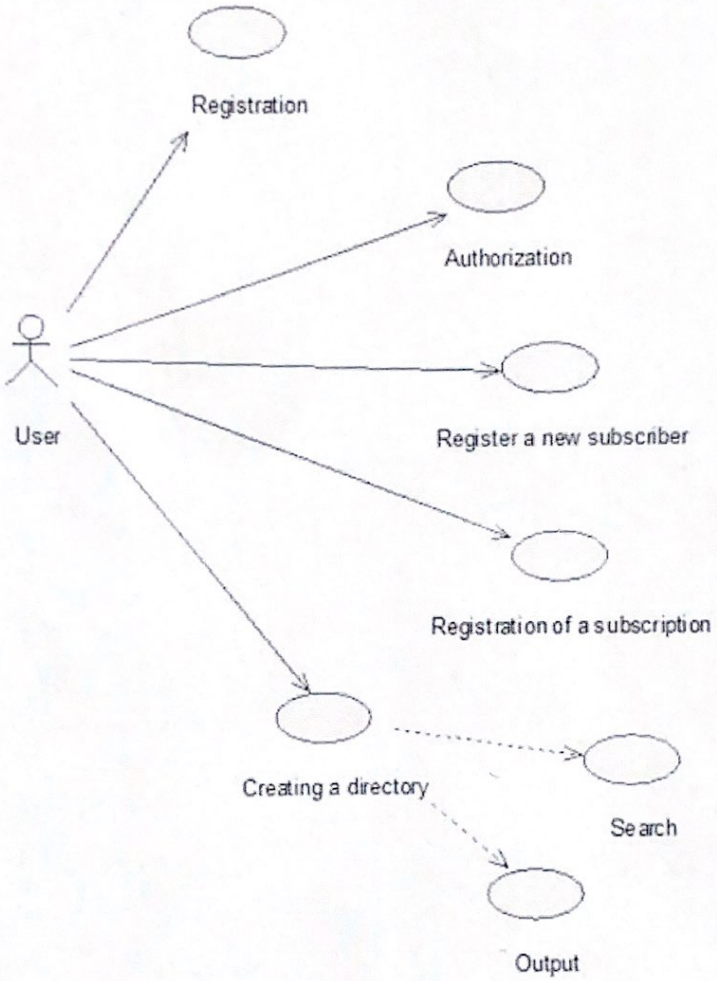
У результаті виконаної роботи було реалізовано мобільний застосунок для обліку передплат у поштовому відділенні, який має простий та інтуїтивно-зрозумілий інтерфейс.

Графічна частина



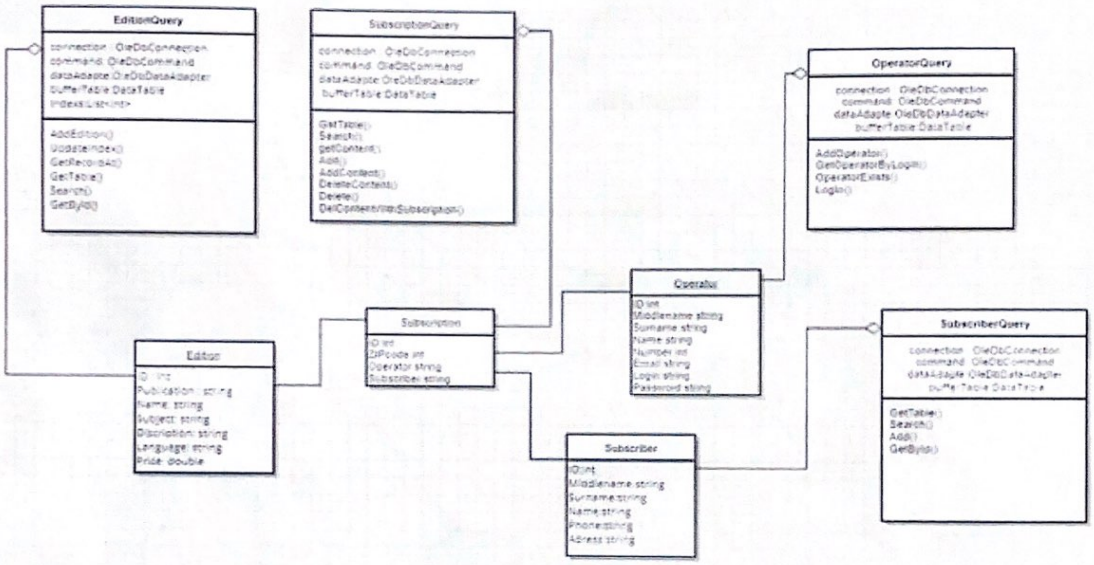
Зм.	Арк.	№ докум.	Підпис	Дата
Розробив		Петрина В.І.		25.05
Керівник		Бедратюк Л.Г.		28.05
Консульт.				
Н. Контр.		Бедратюк Г.І.		03.06
Зав. Каф.		Бедратюк Л.Г.		03.06

Літера	Маса	Масштаб
Аркуш	Аркушів	



Зм.	Арк.	№ докум.	Підпис	Дата
Розробив		Петрина В.І.		05.0
Керівник		Бедратюк Л.Г.		05.0
Консульт.				
Н. Контр.		Бедратюк Г.І.		05.0
Зав. Каф.		Бедратюк Л.Г.		05.0

Літера	Маса	Масштаб
Аркуш	Аркушів	



Зм.	Арк.	№ докум.	Підпис	Дата
Розробив		Петрина В.І.		05.06
Керівник		Бедратюк Л.Г.		05.06
Консульт.				
Н. Контр.		Бедратюк Г.І.		05.06
Зав. Каф.		Бедратюк Л.Г.		05.06

Літера	Маса	Масштаб
Аркуш	Аркушів	

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Петрина Вікторія Іванівна

Тема Мобільний застосунок для обліку передплат у поштовому відділенні

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3; кількість сторінок записки 72

1.Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі Петрини Вікторії Іванівни розглянуто створення мобільного застосунку для обліку передплат у поштовому відділенні. Проект включає аналіз предметної області, проектування архітектури системи, програмну реалізацію та тестування застосунку. Використано мову програмування Java, середовище розробки Android Studio та базу даних
Firebase.

2. Висновок про відповідність роботи поставленому завданню Робота повністю відповідає поставленому завданню. Виконані всі необхідні етапи розробки: дослідження предметної області, проектування, реалізація та тестування мобільного застосунку.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи Дослідження предметної області: Виконано детальний аналіз предметної області, визначено вимоги до програмного забезпечення. Проектування програмного забезпечення: Розроблено архітектуру системи, моделі бази даних, інтерфейс користувача. Використано UML діаграми для візуалізації структури системи. Програмна реалізація: Реалізовано функціонал мобільного застосунку з використанням Java, Android Studio та Firebase. Включено можливість авторизації користувачів, управління передплатниками, створення передплат, перегляд каталогу видань. Тестування системи: Проведено детальне тестування функціоналу застосунку, описано результати тестування.

4. Позитивні сторони роботи Використання сучасних технологій і передових методів розробки. Високий рівень деталізації проектування та реалізації. Інтеграція з Firebase забезпечує надійне зберігання даних. Комплексне тестування, яке забезпечує високу якість кінцевого продукту.

5. Негативні сторони роботи Деякі аспекти роботи з оптимізацією продуктивності могли бути розглянуті більш детально, зокрема питання оптимізації швидкості завантаження даних з серверу та зменшення часу відгуку системи.

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення та пояснювальна записка виконані на високому рівні, містять усі необхідні схеми, діаграми та ілюстрації, що робить їх зрозумілими і зручними для користувачів.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота Петрини Вікторії Іванівни є завершеним проектом, який демонструє високий рівень знань та навичок у галузі інженерії програмного забезпечення. Робота відповідає поставленим завданням та вимогам, містить теоретичні та практичні результати, які можуть бути використані в подальшій розробці мобільних застосунків.

8. Інші зауваження Рекомендується додати більше тестових сценаріїв для різних ситуацій використання, що дозволить більш повно оцінити стабільність та функціональністьзастосунку.

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана у повному обсязі, відповідає поставленій задачі та заслуговує на оцінку «добре».

РЕЦЕНЗЕНТ Говорущенко Тетяна Олександрівна, доктор технічних наук, професор, комп'ютерної інженерії та інформаційних систем (КІІС) ХНУ

"5" 06 2024 р. _____
(підпис)

Завідувачу кафедри
інженерії програмного забезпечення
проф. Бедратюку Л. П.
студента групи ПЗс-21-1
Петрина В. Т.
Прізвище, ініціали

ЗАЯВА

Прошу закріпити за мною тему кваліфікаційної роботи освітнього ступеня «бакалавр» за спеціальністю 121 «Інженерія програмного забезпечення»: Мобільний застосунок для обліку передплат у поштовому відділенні

(керівник роботи – Бедратюк Леонід Петрович)
Прізвище, ім'я, по батькові

02.01.2024
Дата

Підпис студента

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатами звіту/звітів перевірки роботи, продукованими програмно-технічним засобом (ами), на наявність текстових збігів:

Назва кваліфікаційної роботи: «Мобільний застосунок для обліку передплат у поштовому відділенні »
Автор: Петрина Вікторія Іванівна
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Спеціальність: 121 – Інженерія програмного забезпечення
Науковий керівник: Бедратюк Леонід Петрович, д-р фіз.-мат. наук, проф.
Після аналізу звіту/звітів зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої й електронної версії роботи	
3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнуті. Робота може бути допущена до захисту після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені у роботі, є законними і не є плагіатом, оскільки:

1) у тексті кваліфікаційної роботи системою перевірки на плагіат Unicheck виявлено схожість з деякими документами у частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, відомість документів), у структурі змісту, назвах розділів/підрозділів, у рамках основних написів, у назвах публікацій переліку джерел посилання;

2) в якості запозичень системою Unicheck було зафіксовано деякі послідовності вихідного коду і посилання на бібліотеки, які є стандартними мовними конструкціями програмування та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) запозичення, виявлені в тексті роботи, є фрагментарними.

Максимальний обсяг запозичень, визначений системою Anti-Plagiarism, складає 11.0%. Обсяг запозичень, визначений системою Unicheck виявлення збігів ідентичності/схожості, складає 16.2% і адресується до 769 джерел з Інтернету і 153 джерела з бібліотеки, що, з урахуванням наведених обгрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Дата 05.06.2024

Завідувач кафедри — Леонід БЕДРАТЮК

Гарант освітньої програми — Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи — Леонід БЕДРАТЮК

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти

Петрини В.І.

Прізвище, ініціали

факультет ІТ, 3 курс, група ІПЗс-21-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності в Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та/або Anti-Plagiarism) і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

02.06.2024
дата

підпис

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 11.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 10%**

	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
ID: 128471 Назва: БКР Мобільний застосунок для обліку передплат у поштовому відділенні Додано в БД: 2024-06-05 Автора: Петрина В.І. Керівники: Бедратюк Л. П., д-р фіз.-мат. наук, проф. Консультанти: Опоненти:	60491	571	11065 (18%)	112 (20%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми
118414	Назва: КП з дисципліни «Архітектура та проектування програмного забезпечення» на тему: «Архітектура та компоненти додатку для обліку передплатників поштового відділення в розрізі операторів» Додано в БД: 2023-06-29 Автора: Петрина В.І. Керівники: Форкун Ю.В. к.т.н. доц. Консультанти: Опоненти:	6467 (11.0%)	63 (11.0%)

UNICHECK
by Turnitin

Ім'я користувача:
ІПЗ

ID перевірки:
1016316941

Дата перевірки:
03.06.2024 23:51:44 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
04.06.2024 09:09:04 EEST

ID користувача:
100012953

Назва документа: БКР_Мобільний_застосунок_для_обліку_передплат_у_поштовому_відділенні_Петрина_Бедр...

Кількість сторінок: 72 **Кількість слів:** 8862 **Кількість символів:** 72188 **Розмір файлу:** 6.49 MB **ID файлу:** 1016114642

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

16.2%
Схожість

Найбільша схожість: 2.82% з джерелом з Бібліотеки (ID файлу: 1015045648)

15.3% Джерела з Інтернету | 769 ..... Сторінка 74

5.41% Джерела з Бібліотеки | 153 ..... Сторінка 79

0% Цитат

Не знайдено жодних цитат

Не знайдено жодних посилань

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи | 4

Підозріле форматування | 30 сторінок