

МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І
СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА
на тему:

РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ КАВ'ЯРНІ НА ANDROID
ЗАСОБАМИ МОВИ ПРОГРАМУВАННЯ KOTLIN

Здобувача вищої освіти першого (бакалаврського) рівня,
4 курсу, групи КБ-19-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології (за галузями)»


(підпис)

Савчука Андрія Васильовича
(прізвище, ім'я, по батькові)

Науковий керівник


(підпис)

Омельчук Антон Анатолійович,
к.т.н.
(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Керівник освітньо-
професійної програми


(підпис)

Федотова - Півень Ірина
Миколаївна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Допущено до захисту за рішенням кафедри
протокол від «26» травня 2023 № 13

Завідувач кафедри


(підпис)

Філоненко Михайло Миколайович,
к.ф.-м.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене
звання)

Реєстрація №6

05.06.2023

(дата)



(підпис відповідальної особи)

Результати перевірки

до захисту

(до захисту, на доопрацювання)

29.05.2023

(дата)



(підпис)

Результат захисту

5 (відмінно)

(оцінка)

14.06.2023

(дата)



(підпис)

Голова комісії:



(підпис)

В.М. Франчук

Члени комісії:



(підпис)

М.М. Філоненко



(підпис)

О.В. Редич



(підпис)

Є.В. Котух

БЛАНК ОЦІНЮВАННЯ ВИПУСКНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст випускної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	29		14.06.2023 Франчук В.М. <i>В.М. Франчук</i> Філоненко М.М. <i>М.М. Філоненко</i> Редич О.В. <i>О.В. Редич</i> Котух Є.В. <i>Є.В. Котух</i> Суховерхівка Л.П. <i>Л.П. Суховерхівка</i>
2. Дотримання вимог щодо оформлення	від 0 до 10	8		
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	3		
4. Рецензент	від 0 до 5	5		
РАЗОМ БАЛІВ	від 0 до 50	45		

Оцінювання кваліфікаційної роботи науковим керівником - 47 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
92	A	5 (відмінно)	14.06.2023	Франчук В.М. <i>В.М. Франчук</i> Філоненко М.М. <i>М.М. Філоненко</i> Редич О.В. <i>О.В. Редич</i> Котух Є.В. <i>Є.В. Котух</i> Суховерхівка Л.П. <i>Л.П. Суховерхівка</i>

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	
50-59	E	3 (задовільно)
35-49	FX	2 (незадовільно)

факультет фінансів та цифрових технологій
 кафедра комп'ютерних та інформаційних технологій і систем
 спеціальність 122 «Комп'ютерні науки»
 освітня програма «Інформаційні управляючі системи і технології (за галузями)»

Затверджую

Завідувач кафедри

 Філоненко М.М.
 «16» 11 2022

Завдання на кваліфікаційну роботу здобувачу вищої освіти

Савчуку Андрію Васильовичу

(прізвище, ім'я, по батькові)

Тема випускної кваліфікаційної (дипломної) роботи
розробка мобільного додатку для кав'ярні на Android засобами мови
програмування Kotlin

атверджена наказом ректора від «15» листопада 2022 №1205 (зі змінами наказ від
 04» квітня 2023 № 361)

Строк здачі закінченої роботи «26» травня 2023

Мета роботи: розробка клієнтоорієнтованого мобільного додатку для кав'ярні
на операційну систему Android.

Об'єкт дослідження: ринок мобільних додатків для кав'ярень в Україні.

Предмет дослідження: розробка мобільного додатку для кав'ярні, що охоплює
роботу визначення архітектури додатку, розробку серверної частини, API, розробку
інтерфейсу та тестування.

Об'єкт дослідження: кав'ярня Masaron, мобільні додатки для кав'ярень

Вихідні дані до роботи: методологія розробки та створений додаток для ОС
Android

Перелік графічного матеріалу: статистика користувачів мобільних пристроїв у
світі; головний екран та екран вибору кави додатка; чиста архітектура
мобільних додатків; залежність архітектури в ОС Android; схема бази даних
дodatка для кав'ярні; макет Product item.xml.

Зміст випускної кваліфікаційної (дипломної) роботи (перелік питань за кожним
 розділом):

Вступ, Розділ 1. Аналіз предметної області, 1.1 Історія та сучасний стан кав'ярень
Україні, 1.2. Актуальність створення додатка для кав'ярні, 1.3 Опис обраних
інструментів розробки додатка, Розділ 2. Технологія проектування додатку, 2.1
етапи створення мобільного додатка, 2.2. Дизайн мобільного додатку, 2.3
архітектура додатку, Розділ 3. Програмна реалізація додатку, 3.1 Моделювання
структури бази даних, 3.2. Реалізація додатка, 3.3. Використання додатка, Висновки,
дodatки

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	За фактом
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 15.11.2022	15.11.20
2	I розділ (провести аналіз предметної області та аналіз обраних інструментів)	01.02.2023	01.02.20
3	II розділ (методика реалізації дослідження)	03.04.2023	04.04.20
4	III розділ (проведені теоретичні і (або) експериментальні дослідження та проєктні рішення)	10.05.2023	09.05.20
5	Висновки	15.05.2022	15.05.20
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	26.05.2023	26.05.20
7	Отримання відгуку керівника	06.06.2023	06.06.20
8	Отримання зовнішньої рецензії	08.06.2023	08.06.20

8. Дата видачі завдання «16» листопада 2022

9. Керівник випускної кваліфікаційної роботи


 (підпис)

 Омельчук А.А.
 (прізвище, ініціали)

10. Керівник освітньо-професійної програми


 (підпис)

 Федотова-Пі
 (прізвище, ініціали)

11. Завдання до виконання отримав здобувач вищої освіти


 (підпис)

 Савчук А.В.
 (прізвище, ініціали)

АНОТАЦІЯ

Розробка мобільного додатку для кав'ярні на Android засобами мови програмування Kotlin

Мета роботи полягає в розробці мобільного додатку для кав'ярні враховуючи потреби споживачів.

Завданнями роботи є: проведення аналізу предметної області; розробка архітектури та дизайну додатку; проєктування бази даних; реалізація додатку та тестування; розгортання додатку.

Предметом дослідження є розробка мобільного додатку для кав'ярні, що охоплює собою визначення архітектури додатку, розробку серверної частини, API, розробку інтерфейсу та тестування.

Об'єктом дослідження є ринок кав'ярень та їх додатків в Україні.

Базою дослідження виступає кав'ярня Macaron та додатки для кав'ярень.

За результатами дослідження сформульовані методика та етапи створення та впровадження мобільного додатка.

Одержані результати можуть бути використані для збільшення кількості клієнтів, порівняно з конкурентами, також для автоматизування певних процесів, таких як: збереження постійних клієнтів, впровадження швидкої програми лояльності.

Кваліфікаційна робота містить 63 сторінок, 5 таблиць, 14 рисунків, список використаних джерел 40 найменувань, 1 додаток.

Ключові слова: кава, кав'ярня, мобільний додаток, Android, Kotlin, Firebase, Firestore database, бізнес, диджиталізація, Android Studio, база даних, дизайн, онлайн-платежі, Clean architecture.

Рік виконання кваліфікаційної роботи 2023

Рік захисту роботи 2023

ABSTRACT

Development of a mobile application for a coffee shop on Android using the Kotlin programming language

The aim of this work is to develop a mobile application for a coffee shop, taking into account the needs of consumers.

The tasks of the work are as follows: conducting an analysis of the subject area; developing the architecture and design of the application; designing the database; implementing the application and conducting testing.

The subject of the research is the development of a mobile application for a coffee shop, which includes defining the application architecture, developing the server-side, API, interface design, and testing.

The object of the research is the market of coffee shops and their applications in Ukraine.

The research is based on the coffee shop "Macaron" and existing coffee shop applications.

The results of the research include the methodology and stages for creating and implementing the mobile application have been formulated.

The obtained results can be used to increase the number of customers compared to competitors and to automate certain processes, such as maintaining loyal customers and implementing a fast loyalty program.

The qualification work consists of 63 pages, 5 tables, 14 images, a list of 40 references, and 1 appendice.

Keywords: coffee, coffee shop, mobile application, Android, Kotlin, Firebase, Firestore database, business, digitization, Android Studio, database, design, online payments, Clean architecture.

Year of qualification work: 2023

Year of work defense: 2023

ЗМІСТ

ВСТУП	9
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	11
1.1 Історія та сучасний стан кав'ярень в Україні.	11
1.2. Актуальність створення додатка для кав'ярні.	15
1.3 Опис обраних інструментів розробки додатка	19
РОЗДІЛ 2 ТЕХНОЛОГІЯ ПРОЄКТУВАННЯ ДОДАТКУ	26
2.1 Етапи створення мобільного додатка	26
2.2. Дизайн мобільного додатку	30
2.3 Архітектура додатку	34
РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ДОДАТКУ	43
3.1 Моделювання моделі бази даних	43
3.2. Реалізація додатка	49
3.3. Використання додатка	53
ВИСНОВКИ	57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	59
ДОДАТКИ	64

ВСТУП

Ринок кав'ярень в Україні наразі перебуває на перехідному етапі зі зростання до зрілості. «Позитивно на ринок вплинуло укладення договорів про вільні поставки з провідними країнами-виробниками – В'єтнамом і Бразилією, а також вступ в дію угоди про асоціацію з ЄС щодо стандартизації товарів і послуг» [3]. У 2019 році Україна посіла третє місце на європейському ринку по кількості відкритих кав'ярень. Зростання продажів кави в кав'ярнях України за 2019 рік на 16% свідчить про те, що Україна має значний потенціал для розвитку кавової індустрії.

Актуальність теми зумовлена тим, що ринок кав'ярень як у Києві, так і в Україні стрімко зростає. Кава поза домом спочатку була трендом, а тепер стає необхідністю для кожної людини з доходом вище середнього. Зростання ринку кав'ярень пов'язане насамперед із зростанням ринку кави як такої. Український ринок кави розвивається відповідно до світової тенденції зростання виробництва, стабільного збільшення попиту та загалом зростання обсягу ринку кави у зв'язку із залежністю вітчизняного виробництва від імпортних поставок сировини. На теперішній час конкурентів на ринку кав'ярень формату «coffee to go» значно більше, ніж під час появи їх на ринку України. Внаслідок цього компанії зіткнулися з рядом проблем:

- обсяги продажів не так швидко зростають, як це було раніше;
- споживачі починають купляти продукцію конкурентів, а не продукцію компанії;
- розвивається культура споживання кави, внаслідок чого частині покупців не подобається технологія приготування;
- зростає кількість споживачів, яких не влаштовують стандарти обслуговування компанії;
- багато негативних відгуків на інтернет-ресурсах.

Виходячи з симптомів, які зазначені вище, маркетинговою управлінською проблемою є розроблення додатку для кав'ярні, що дозволить виділитися серед інших кав'ярень України.

Метою роботи є розробка мобільного додатку для кав'ярні враховуючи потреби споживачів.

У першому розділі проведений аналіз предметної області – історія появи та сучасний стан ринку кав'ярень, а також мобільних додатків для кав'ярень. Розглянуті існуючі додатки, їх функціональні можливості та особливості інтерфейсу. Також визначені потенційні переваги та недоліки створення додатку.

У другому розділі описані технології, які використовуються для проектування мобільного додатку для кав'ярень. Розглянуті основні принципи проектування мобільних додатків, визначені вимоги до архітектури та бази даних, описано методологію розробки додатку.

У третьому розділі описані етапи програмної реалізації мобільного додатку для кав'ярень. Описані основні модулі та функції додатку, а також інтерфейс користувача.

Об'єкт дослідження: ринок кав'ярень та їх додатків в Україні.

Предмет дослідження: розробка мобільного додатку для кав'ярні, що охоплює собою визначення архітектури додатку, розробку серверної частини, API, розробку інтерфейсу та тестування.

Методологія дослідження. Для розробки додатку були використані такі методи, як: аналіз, моделювання, проектування та розробка. При розробці мобільного додатку також використовувалось програмне забезпечення Android Studio, мова програмування Kotlin.

Практичне значення отриманих результатів. У майбутньому додаток можуть використовувати для збільшення кількості клієнтів, порівняно з конкурентами, також автоматизувати певні процеси, такі як: збереження

постійних клієнтів, впровадження швидкої програми лояльності та полегшення роботи персоналу.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Історія та сучасний стан кав'ярень в Україні.

Перші кав'ярні, як місце зустрічей громадськості, з'явилися у країнах Близького Сходу у XVI столітті, де чоловіки збиралися поспілкуватися, випити кави (як правило по-турецьки) або чаю, почерпнути ароматні витяжки кальяну, також грали в шахи, нарди, чи слухали музику та розповіді оповідача.

В Україні перші кав'ярні «європейського» типу були створені німцями у 1829 році, де Карл Гартман відкрив у Львові «Віденську кав'ярню», збудувавши для неї спеціальну будівлю, потім в 1840 році Готліб Фінке заснував під власним іменем першу кав'ярню в Києві [8].

Першого стрімкого розвитку кавова культура зазнала у другій половині 19 століття, коли містами, зокрема і Львовом, почала ширитися традиція відкриття кондитерських із запашною кавою. Одними з перших були літні майданчики-цукерні Яна Вольфа. На той час кав'ярні виступали місцями для збору різних політичних та ділових діячів, які за горнятком кави обговорювали важливі справи.

Згодом Друга світова війна та прихід радянської влади до Галичини спричинили руйнування традиції кав'ярень. Адже влада бажала знищити осередки, де представники інтелігенції могли зустрічатися та обмінюватися думками. Традиція кав'ярень у Львові почала відроджуватися лише на початку 60-х років.

Відновивши незалежність Україна почала швидко наздоганяти західні країни у багатьох питаннях, одним з яких була і кава. Саме з настанням буремних 90-х ми можемо говорити про початок розвитку кавової культури в Україні, так званих трьох хвиль.

Якщо у США та світі перша кавова хвиля розпочалася у середині 20 століття, то в Україні вона з'явилася лише з відновленням незалежності. Це пояснювалося дуже просто, адже комуністичний режим вважав каву буржуазним

продуктом, тому обмежував імпорт кавових зерен. У якості альтернативи на ринок випустили напої з цикорію та жолудів.

Перша хвиля характеризується споживанням розчинної кави. Розчинна кава – це продукт, для якого каву готують заздалегідь, а потім повністю забирають з неї вологу. У результаті виходить порошок, який містить кофеїн та здалека нагадує смак та аромат, який може дати філіжанка свіжозвареної кави.

Ще один популярний вид кави того часу — це кавові стіки 2в1 або 3в1, які містили у складі близько 10% кави, а решта — цукор, сухе молоко, стабілізатори та емульгатори. На відміну від України, у США перша хвиля принесла популярність ще каві з крапельних кавоварок.

Наступним важливим етапом стала поява таких сильних мереж кав'ярень, як Starbucks. Завдяки маркетинговій активності бренд став популярним та просунув у маси напої великого об'єму на основі еспreso з молоком, сиропами та різними додатками.

В Україні друга хвиля сталася у 2000 роки, коли з настанням більшої економічної стабільності у країні розвиваються кав'ярні з кавовими машинами та напоями на основі еспreso. Тоді також починають імпортувати свіжообсмажену каву з Європи, переважно італійських виробників, а на полицях магазинів переважають такі бренди, як Lavazza, Movenpick, Dallmayr [8].

Під час другої хвилі на перший план виходить отримання задоволення від споживання кави, тож важливою складовою було створення атмосфери кав'ярень та широкий вибір кавового меню. Щоправда, якраз під час другої хвилі люди вже починають звертати увагу на смак кави, розуміти різницю між робустою та арабікою та цікавитися, де були вирощені зерна.

Третя хвиля, це період, який почався на початку 2000-х років, а справжнього розголосу набув в останнє десятиліття. Третя хвиля виділяє не атмосферу навколо кави, а сам напій, його смак та аромат. В її основі лежить споживання кави високої якості.

Кава, про яку відомо усе – від країни походження зерен до методу обробки та ступеню обсмаження, є предметом пильної уваги науковців і експертів у галузі кавового бізнесу. За останні роки на ринку змінилися головні гравці – великі корпорації поступово замінюються «крафтовими» виробництвами, які мають за мету розвивати культуру кави, а не тільки заробляти гроші. Відтак, дослідження в галузі кави важливе не лише для розвитку кавового бізнесу, але й для розуміння тенденцій на ринку та формування споживчих пристрастей.

У 2018-2019 рр. споживання кави різко зросло, проте у 2020 році зменшилося на 4,6 тис. т в порівнянні з 2019 р. З березня 2020 року почалася епідемія COVID 19, оголосивши надзвичайний стан і кількомісячний локдаун.

Під впливом карантинних заходів, закрито було заклади кав'ярень, магазини непродовольчих товарів, перукарні, кінотеатри, а також призупинено пасажирський залізничний та повітряний транспорт. Протягом цього року було проведено кілька ревізій моделі карантину. Уряд послідовно послаблював обмеження, а потім знову вводив країну в новий період повного локдауну.

Найсерйознішою трагедією під час карантину стала поразка сектору малого бізнесу. Малі крамниці та кав'ярні на декілька місяців втратили дохід, оскільки вони не мали достатньої фінансової підтримки, щоб вижити під час кризи. Незважаючи на певні спроби держави допомогти підприємцям, багатьом з них так і не вдалося зберегти свій бізнес за карантинний рік.

Рятувалися від наслідків локдауну ресторани та кав'ярні переважно скороченнями працівників. Наприклад, заклади холдингу «!Fest» за підсумками першого року карантину не змогли повернути на роботу близько 20% працівників [15].

Якщо під час карантину люди не могли і не хотіли відвідувати заклади через ризик підхопити коронавірус, то з початком повномасштабної війни в Україні відвідування кав'ярень зовсім зникло з порядку денного сотень тисяч українців. Постійні обстріли, відтік населення, падіння зарплат, відсутність роботи – це все

те, що на сьогодні заважає українцям повернутися до колишнього життя та звичок, у тому числі ходити до закладів. З цих причин кав'ярням складно заробляти і відкривати нові заклади.

Ринок зменшився приблизно на 25% порівняно з лютим 2022 року. У деяких областях падіння ринку становило понад 50% (Харківська, Миколаївська, Запорізька, Луганська області), у Київській, Одеській, Дніпропетровській областях – падіння до 30% [4].

Разом з тим, у західних областях спостерігається позитивна динаміка. У Львові та області кількість ресторанів та кафе зросла приблизно на 30%, у Закарпатській, Чернівецькій, Івано-Франківській областях відмічено зростання приблизно на 20%.

З червня бізнес почав відновлюватись, почали відкриватися кав'ярні, ресторани у Києві, а також у західних областях. Загалом відкрилось понад 2 тисячі закладів.

Протягом періоду збройного конфлікту на сході України, внаслідок особливої ситуації, спостерігалось значне зростання кількості кав'ярень та ресторанів у місті Львові. Відповідно до статистичних даних, з моменту початку війни було зафіксовано понад 500 нових закладів у цьому регіоні. Серед усіх міст на заході України, Львів відрізнявся своєю високою космополітичністю та великою популярністю, що стало особливо помітним на тлі інших населених пунктів [12]. Згідно з висновками фахівців, під час війни більшість кав'ярень зуміла вижити та продовжує експлуатуватись. Ще з початку серпня кав'ярні у Києві досягли та перевищили попередні рівні продажів до війни.

Часті обстріли російськими військами енергетичної інфраструктури спричинили погіршення електропостачання. В багатьох кав'ярнях було вирішено придбати дизельні або бензинові генератори. На початку відсутність електроенергії та наявність генераторів стали конкурентною перевагою для кав'ярень, оскільки багато закладів ще не були готові до подібних випадків. У

кав'ярнях, де були генератори, спостерігався великий потік клієнтів, оскільки люди шукали місце, де можна було попрацювати, поїсти, випити та відпочити. Зараз використання генераторів є нормою для кав'ярень.

Кав'ярні наразі найпопулярніший вид бізнесу в Україні. Незважаючи на те, що вартість кави зростає, вона популярна серед різних соціальних груп, включаючи військових і мам з дітьми. Багато людей шукають в каві спосіб зняти стрес та відпочити, особливо супроводжуючи це солодкими та борошняними виробами. Таким чином, кав'ярні будуть символами задоволення на довгі роки, що сприятиме їх зростанню та поширенню, в тому числі і в цифровому просторі.

Цифровізація створює нові можливості для якісної діяльності кав'ярень. Вона забезпечує свободу взаємодії закладу та гостей, надає інтерактивні інструменти, якими зможе користуватись кожен.

Інтегровані цифрові системи даних, що надають ресторанним закладам переваги завдяки розумнішому управлінню, залученню нових клієнтів, на сьогодні залишаються ключовими пріоритетами для ресторанної галузі, а саме кав'ярень.

1.2. Актуальність створення додатка для кав'ярні.

На сьогодні цифрові інновації розвиваються в Україні з неймовірною швидкістю. Це призводить до стрімкого розвитку цифрових технологій у всіх сферах економіки. Завдяки відповідним цифровим інноваціям індустрія ресторанного бізнесу також суттєво змінилася в Україні за останні 20-25 років.

У наш час мати додаток для кав'ярні – це, гарна стратегія розвитку бізнесу. Клієнти можуть познайомитися з кав'ярнею перш ніж прийти туди.

Наявність додатка для кав'ярні надає можливість ще більше розширити аудиторію. Якісна перевага сучасного Інтернету полягає в тому, що кожен може легко знайти та встановити додаток. Наявність додатку дозволяє пошуковим системам дізнатися більше про бізнес та залучити більше клієнтів.

За допомогою додатка бізнес має можливість контролювати свій бренд. Незважаючи на те, що соціальні мережі є чудовим інструментом для бізнесу, наявність додатка надає глибшого контролю, дозволяючи зберегти імідж бренду у відповідності з вимогами компанії.

За даними статистики, більшість користувачів Інтернету шукає інформацію перед покупкою. Приблизно 83% користувачів смартфонів шукають необхідну інформацію в Інтернеті, перш ніж відправитися до магазину. Це означає, що постійні клієнти, які хочуть відвідати кав'ярню, також шукають інформацію в Інтернеті, де вони можуть знайти відгуки, фотографії та веб-сторінки. На сьогоднішній день більшість людей отримує інформацію через Інтернет. Використовуючи Інтернет як потужний інструмент, бізнес може надихнути потенційних клієнтів відвідати кав'ярню. Отже, наявність додатка дозволяє бізнесу залучити більше клієнтів та зберегти позиції на ринку.

Диджиталізація кав'ярень може бути розглянута як процес уможливлення, поліпшення та/або перетворення бізнес-операцій підприємств (закладів) ресторанного бізнесу та/або їхніх бізнес-функцій, та/або їхніх бізнес-моделей чи процесів, та/або ж усієї їхньої діяльності в цілому через застосування цифрових технологій та більш широке використання перетворених цифрових даних із метою одержання певних бізнес-переваг.

На практиці диджиталізація ресторанного бізнесу, а саме кав'ярень, досягається шляхом застосування закладами певного комплексу цифрових технологій та процесів, серед яких, здійснення дистанційної комунікації з клієнтами поза межами закладу, та через спеціалізовані вебсайти, сторінки у соціальних мережах, спеціалізовані мобільні застосунки, спеціалізовані чат-боти тощо, а також забезпечення можливості здійснення електронної оплати послуг закладу.

Дослідження та анкетування закордонних компаній свідчать про значне зростання популярності замовлення їжі онлайн або через мобільний додаток, а

також про можливість залучення значного обсягу нових клієнтів, особливо молоді, шляхом максимального запровадження можливої диджиталізації.

Своєю чергою закордонний досвід також свідчить про актуальність й ефективність запровадження у діяльності підприємств (закладів) готельного господарства різноманітних диджитал-інновацій, включаючи інновації у таких умовних напрямках як техніко-технологічні, сервісні, організаційно-технологічні й управлінські інновації [11].

Загалом процеси диджиталізації ресторанного бізнесу є корисними й ефективними. Однак на практиці вони мають й окремі проблеми.

Основними практичними проблемами запровадження мобільного додатку, є:

- 1) орієнтація переважно на молоде та середнього віку населення (як правило, споживач похилого віку не входить до цільової аудиторії запровадження додатка);
- 2) обов'язкова наявність у споживача смартфона;
- 3) зменшення кількості часу спілкування споживача з персоналом закладу, зокрема, баристою, що може одночасно зменшити у споживача відчуття високого рівня сервісу та необхідного обсягу приділеної йому уваги.

Проте, на відміну від створення власного додатку, програми управління закладом мають не потенційний, а реальний ефект вже з першого їхнього використання. Це сприяє залученню нових клієнтів та утримання існуючих. Останнім часом люди використовують телефони частіше, ніж комп'ютери. Так само й з доставкою їжі: простіше й швидше зробити замовлення зі смартфона.

Враховуючи чинний на сьогодні воєнний стан та одночасне продовження дії обмежувальних заходів пов'язаних з COVID-19 актуальними є перспективи подальшої диджиталізації кав'ярень в Україні, особливо, коли зростає попит споживачів на зручність, технологічні способи замовлення їжі, кави й швидкість обслуговування [13].

Перед розробленням прототипу додатку для кав'ярень з метою дотримання всіх поставлених вимог необхідно провести пошук та аналіз існуючих аналогів серед додатків для кав'ярень.

Aroma Kava - мобільний додаток для кав'ярень, який дозволяє клієнтам замовляти напої та страви, використовуючи свій мобільний телефон. Додаток має інтуїтивно зрозумілий інтерфейс, який дозволяє легко знайти потрібний товар та зробити замовлення в будь-який зручний час. Крім того, додаток дозволяє користувачам переглядати меню, акції та знижки, додавати товари до обраного та використовувати бонусну програму.

Характеристики та функції:

1. Замовлення напоїв та страв. Додаток дозволяє користувачам замовляти будь-які напої та страви, що доступні в кав'ярні, використовуючи свій мобільний телефон.
2. Меню та акції. Користувачі можуть переглядати меню кав'ярні, а також дізнаватися про акції та знижки, які діють в даний момент.
3. Обрані товари. Додаток дозволяє користувачам додавати товари до обраного, щоб потім легко знайти їх і замовити.
4. Бонусна програма. Користувачі можуть користуватися бонусною програмою кав'ярні, отримуючи бонусні бали за кожне замовлення та отримуючи знижки на наступні замовлення.

Серед основних недоліків додатку можна виділити:

1. Недостатня адаптивність. Дизайн додатку не адаптується до всіх типів екранів, що може призводити до проблем з відображенням деяких елементів на малих екранах.
2. Відсутність деяких важливих функцій. Додаток не має можливості здійснювати замовлення онлайн, а також не має інформації про розташування кав'ярень [25].

Додаток Vasara coffee призначений для замовлення кави та інших напоїв у кав'ярні Vasara Coffee. Додаток дозволяє користувачам замовляти каву та інші напої, вибираючи з меню, яке включає категорії напоїв, список складників та ціни. Користувачі можуть додавати товари до кошика, виконувати оплату з допомогою онлайн-платіжної системи та вказувати адресу доставки. Додаток також надає користувачам можливість зберігати свої улюблені напої у вкладці «Улюблені» та знаходити найближчу кав'ярню Vasara Coffee на карті.

Серед недоліків можна виділити:

1. Обмежена функціональність: Додаток не має деяких корисних функцій, таких як замовлення їжі та закусок.
2. Відсутність функції відстеження замовлення: Додаток не надає можливості відстеження статусу замовлення, що може створювати незручності для користувачів.
3. Відсутність можливості змінити замовлення: Користувачі не можуть змінити своє замовлення після оформлення, що може створювати проблеми в разі помилкового замовлення [5].

Мобільний додаток «Prostir Coffee» розроблено для замовлення та безпечної оплати кавових напоїв в кав'ярні «Простір». Застосунок надає користувачам можливість замовляти каву та інші напої з доступного меню кав'ярні, використовуючи категорії напоїв та додавання необхідних інгредієнтів. Після вибору товарів користувачі можуть додати їх до кошика, обрати спосіб оплати та вказати адресу доставки. Додаток також містить розділ «Favorites», де користувачі можуть зареєструватися та отримувати бонуси та знижки. Варто відзначити, що на даний момент відсутня можливість зберігання улюблених напоїв, що може становити недолік застосунку [34].

1.3 Опис обраних інструментів розробки додатка

Розробка мобільного додатку, з технічної точки зору – це процес або процедура створення програмного забезпечення, що працює на смартфонах. Додаток завантажується та встановлюється користувачем з магазинів додатків (Google Play, App Store).

Для створення мобільного додатка необхідно враховувати безліч факторів, таких як розмір екрана, функції, платформа розробки додатків, дизайн тощо.

Насамперед треба вибрати платформу розробки мобільного додатку, це досягається шляхом визначення цільової аудиторії. Взявши до уваги статистику користувачів мобільних пристроїв у всьому світі за 2022 р. (Рис 1.1).

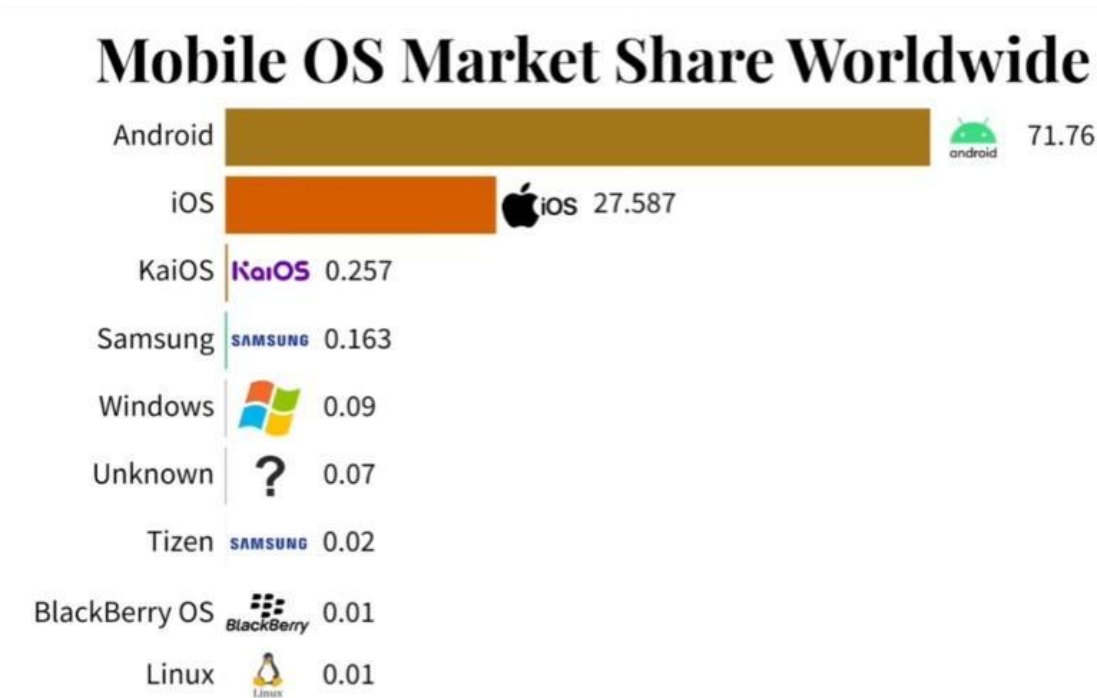


Рис. 1.1. Огляд користувачів мобільних пристроїв у всьому світі [20].

Зважаючи на сферу діяльності, необхідно провести аналіз основних функцій, які будуть надані користувачам. Запровадження технологічного стеку є ключовим завданням, яке потрібно виконати за допомогою фахівців у галузі розробки мобільних додатків. Термін "технологічний стек", також відомий як "стек рішень", "технологічна інфраструктура" або "екосистема даних", визначає перелік всіх

технологічних сервісів, які використовуються для створення та розгортання програмного продукту. Впровадження популярного та надійного технологічного стеку виступає ключовим елементом для забезпечення безперебійної роботи додатка.

Для розробки додатка буде використовуватись стек технологій:

- Операційна система Android,
- Мова програмування Kotlin,
- Платформа Firebase
- Realtime Database (База даних)
- Середовище розробки Android Studio
- Та інші допоміжні бібліотеки, для написання ефективного коду.

Kotlin – це статично типізована мова програмування загального призначення, розроблена компанією JetBrains. Вона має як риси об'єктно-орієнтованого програмування, так і функціонального програмування. Kotlin сумісна з Java, що означає, що обидві мови можуть обмінюватися інформацією та використовувати її. Як і компілятор Java, компілятор Kotlin також генерує байт-код, який може працювати на JVM [31].

Kotlin широко використовується для розробки програм на стороні сервера, додатків для Android, мульти-платформенної мобільної розробки та багато іншого. Ця мова програмування може бути розглянута як вдосконалена версія мови Java з додатковими функціями. Крім того, в проєктах Kotlin можна використовувати бібліотеки Java. Незважаючи на те, що Java є високопродуктивною мовою програмування, розробники Kotlin більше зосереджені на тому, щоб спростити код та підвищити читабельність, що зумовило активну підтримку цієї мови програмування з боку Google.

Є багато особливостей мови Kotlin, а саме:

- Мова Kotlin може використовуватися на кількох платформах, що скорочує час, витрачений на написання та підтримку коду на різних платформах. Це не залежить від віртуальної машини на цільовій платформі.

- Kotlin має деякі сучасні функції, такі як нульова безпека (Null Safety), яка є функцією для зменшення проблем, пов'язаних із нульовими посиланнями. Kotlin підтримує такі функції, як функція Lambda (анонімна функція) і Smart casts (використовується для додавання умов усередині виразу if).

- У мові Kotlin є багато розширень, які дозволяють додавати функціональні можливості без успадкування від класу, використовуючи декоративний шаблон. Ця функція робить код легким для підтримки та читання.

- Kotlin сумісний із Java, ви можете конвертувати файл Java у файл Kotlin лише за допомогою сценарію. Крім того, він працює на JVM, а бібліотеки Java добре працюють на Kotlin.

- У функції мови Kotlin можна передати і параметри функції, що означає, що функція може діяти як змінна, і так само, звичайну змінну. Такі функції відомі, як функції вищого порядку. Їх функціональність виконує дві речі: по-перше, приймає функції як параметри, а по-друге, повертає функцію.

- Kotlin має клас даних (Data class). Клас даних – це клас, який не виконує жодних операцій і містить лише стан. Перевагою цього класу є те, що він містить автоматично згенерований код та уникає реалізації методів у класі [35].

Kotlin досить популярна мова програмування, в наш час його активно використовують компанії гіганти, а саме:

- Airbnb, американська компанія, що займається керуванням онлайн-платформою для оренди житла та проживання у сім'ях для відпочинку, також використовує мову Kotlin для свого Android додатку для безвідмовної роботи.

- Google широко використовує мову Kotlin у своєму виробничому коді. Google також застосував Kotlin у більш ніж 60 програмах, таких як Google Homes, Google Maps, Google Drive, Google Pay, Google Sheets тощо.

- Zomato, найбільша компанія з доставки їжі в Індії, також використовує Kotlin для додатку на Android, це допомогло зменшити рядки коду та зробити код набагато лаконічнішим.

- Провідний у світі потоковий сервіс Netflix використав мову Kotlin, щоб перебудувати свій програвач інтерфейсу користувача для додатку Netflix на Android.

- Uber, американська компанія доставки, яка надає такі послуги, як доставка їжі, замовлення поїздок, доставка посилок, кур'єрська служба тощо, використовує мову Kotlin через її популярність у розробці Android і її взаємодію з Java [23].

«Android – мобільна операційна система з відкритим кодом, яку створила та розробила компанія Google. Основою для її створення було ядро Linux. Android – це найбільш популярна ОС у світі, яка використовується на смартфонах (починаючи з 2011 року). З цієї мобільної ОС частіше всього виходять в мережу Інтернет» [20].

Для розповсюдження додатків для користувачів компанія Google використовує власний офіційний інтернет-магазин додатків Google Play. Також Google Play надає користувачам похідні сервіси для пошуку та завантаження книг, музики та фільмів. Загалом в Google Play налічується близько 3,3 мільйона додатків [24].

Для розробки додатків для ОС Android можна використовувати різні середовища розробки. Найбільш популярним є Android Studio від компанії Google.

Android Studio має ряд переваг:

- вбудовані SDK, які автоматично завантажують необхідні пакети та інструменти для розробки;
- підтримка декількох мов програмування (Java, C, C++, Dart);
- підтримка компанії Google.

Тобто через Android Studio компанія Google реалізовує своє бачення розвитку Android-додатків. Але значним мінусом цієї IDE є повільна робота вбудованого емулятора Android. Для його коректної роботи розробник системи повинен мати якісне та потужне апаратне забезпечення, яке має високу вартість [21].

Перевагою мобільної ОС Android є те, що користувачу доступно велика кількість пристроїв від різних виробників, з варіацією параметрів та ціновим діапазоном. Але для розробників це значний недолік. Через встановлення мобільної ОС Android на пристроях різних компаній, ОС втрачає свою якість, бо неможливо створити настільки універсальну ОС, та приходиться адаптуватися до всіх пристроїв.

«Firebase – платформа для розробки програмних додатків, яку розробила компанія Firebase. У 2014 році її викупила компанія Google. На цей час Firebase має 18 програмних продуктів, які використовують 1,5 мільйона програмних додатків» [27]. Найвідоміші програмні додатки – аналітика Firebase, Firebase Cloud Messaging, Firebase Auth, Firebase Realtime, хостинг Firebase.

Firebase створила інфраструктуру для тестування програмних додатків, для мобільних ОС Android та IOS. Також є можливість автоматичного тестування при умові, що розробник ПЗ не передбачив тестовий код для розробленої системи [22].

Firebase Realtime – це сервіс, який надає БД, в основі якої лежить хмарна технологія. Дані зберігаються у форматі JSON і синхронізуються в реальному часі. Використовуючи SDK Firebase для розробки під мобільну ОС, то розробники отримують автоматичне оновлення з новими даними. Дані залишаються доступними при переході в автономний режим користування [27].

Основні функції Firebase:

- синхронізація даних в режимі реального часу;
- швидкий доступ до документів і колекцій, що зберігаються в БД;
- SDK для iOS, Android і Web з офлайн-доступом до даних;

- серверні SDK для Node, Python, Go і Java;
 - простота у використанні, що є пріоритетом для всіх продуктів Firebase;
 - синхронізація даних між пристроями в режимі реального часу, що дозволяє створювати додатки з автоматичною синхронізацією даних;
 - використання колекцій і документів для структурування та обробки даних.
- При цьому підході продуктивність запитів не залежить від обсягу даних, і результат роботи з базою з 100 і 100 мільйонів документів буде однаково ефективний [27].

Firebase надає в режимі реального часу базу даних та бекенд як службу. Ця служба надає розробникам застосунків API, який дозволяє синхронізувати дані застосунків між клієнтами та зберігати їх у хмарі Firebase.

Компанія також надає клієнтські бібліотеки, які дозволяють інтеграцію із застосунками Android, iOS, Java, Objective-C, Swift. REST API використовує протокол подій із сервером, який є інтерфейсом для створення HTTP-з'єднань для отримання push-повідомлень від сервера. Розробники, які використовують Realtime Database, можуть захищати свої дані за допомогою правил безпеки, що застосовуються на сервері [29].

Для розробки мобільного застосунку було вирішено обрати ОС Android, тому що:

- більш зручне середовище розробки та інструмент для створення графічного інтерфейсу в порівнянні з iOS;
- більша аудиторія користувачів;
- більш якісні програмні додатки через політику компанії Google.

В якості системи керування базами даних вирішено обрати Firebase через її швидкодію. Доступ до цих файлів дуже простий як для користувача, так і для розробника. Не потрібно створювати велику кількість допоміжних файлів, натомість треба лише зареєструвати додаток на офіційному сайті Firebase.

РОЗДІЛ 2 ТЕХНОЛОГІЯ ПРОЄКТУВАННЯ ДОДАТКУ

2.1 Етапи створення мобільного додатка

Створення мобільного додатку є складним і багатоетапним процесом, який включає в себе декілька ключових етапів. Незалежно від того, чи ви розробляєте додаток для Android, чи iOS, процес створення вимагає від розробників високої кваліфікації та глибоких знань.

Можливі наступні етапи розробки мобільних додатків, які можуть бути виділені з наукової точки зору:

Перший етап полягає у пошуку основної ідеї програми та її концепту, який може передбачати розробку чогось унікального або удосконалення вже існуючого. На цьому етапі також проводиться аналіз ринку та вже існуючих продуктів, проводяться інтерв'ю зі інвесторами. Задля успішної подальшої розробки продукту необхідно детально описати бізнес-вимоги, визначити цільову аудиторію, конкурентоспроможність, вимоги бренду та показники успіху. До цього також належить визначення персони головних клієнтів та їх потреб.

Другий етап передбачає визначення базового функціонала – мінімально життєздатного продукту (MVP). Найчастіше мобільні додатки поступово народжуються, спочатку виходить демоверсія зі спрощеним функціоналом, а потім в наступних релізах виходять нові особливості. Розробник збирає зворотній зв'язок від користувачів і розуміє на що робити ставку в подальшій розробці. Цей підхід найчастіше є фінансово виправданим та може призвести до мінімального прибутку. На цьому етапі визначається user experience map, яка містить в собі набір базових функцій та можливостей, доступних користувачеві, та їхня взаємодія. З наукового погляду важливо відзначити базову модель поширення з точки зору маркетингу, оскільки це визначає основні сторінки додатка. При цьому необхідно детально визначити послідовність дій в просуванні, включаючи джерела / канали поширення та цільові дії користувача [24].

На третьому етапі розробки UI / UX design дизайнер розпочинає роботу з конкретним функціоналом та чітким розумінням кроків клієнта. Тут створюється інформаційна архітектура, включаючи основну структуру навігаційної системи програми, сценарій використання, системні аналітики, прототипи та дизайн взаємодій. Варто зауважити, що на цьому етапі не йдеться про безпосередньо дизайн додатків, а лише про прототипи або прості макети.

Один з етапів розробки дизайну – валідація прототипів в інструментальних тестах – включає дуже просте базове тестування макетів та прототипів. На цьому етапі дизайнер перевіряє, чи можуть користувачі розуміти цінність додатка, що було спочатку визначено, та чи можуть вони зрозуміти, як його використовувати. Часто замовником буває очевидним функціонал сторінки додатка, але абсолютно не зрозумілим користувачеві. Дизайнер вносить зміни в дизайн на основі результатів тестів [30].

Наступний етап – це створення розгорнутого технічного завдання (ТЗ) за допомогою дизайнера, проджект-менеджера та інших фахівців, а також безпосередньо саму розробку додатка та його тестування.

Процес розробки мобільного додатка є поетапним і має включати плани релізів, щоб забезпечити здійснення замовлення в строк та відповідно до плану просування додатка. На кожному етапі розробки можуть з'являтися нові ідеї щодо функціонала та дизайну, які можуть вплинути на швидкість процесу розробки та план його просування.

Однак, план просування додатка є надзвичайно важливим етапом, без якого проєкт не зможе успішно розпочатися. Для забезпечення максимальної ефективності, маркетингова стратегія має включати в себе банери на сайтах, рекламу в пошукових системах, SMM, поштову розсилку, SEO в просуванні сайту додатка, відстеження контенту та максимально швидке реагування, публікації в ЗМІ і на новинних порталах, а також публікацію POS-матеріалів. Для максимальної ефективності, важливо визначити місцезнаходження лояльної

аудиторії та відповідні канали зв'язку, які найкраще реагують на рекламні матеріали. Комплексна маркетингова стратегія має бути готовою до запуску відразу після релізу програми, щоб забезпечити максимальну успішність проєкту.

Далі проводиться повторне тестування функціональності програми, виявлення та усунення виявлених помилок, оцінка і переосмислення дизайну та технічного завдання, а також створення тестової документації, включаючи тест-кейси. Після усунення знайдених проблем та недоліків додаток може бути підготовлений до випуску стан.

Останній етап – випуск. Після завершення всіх доробок і погодження результатів роботи всіма відділами, додаток можна випустити у світ, починаючи з демоверсії, яка буде завантажена до магазинів додатків. Проте розробка програмного забезпечення не закінчується на цьому етапі. Надалі необхідно забезпечувати постійну технічну підтримку для забезпечення працездатності додатка, а також покращення його функціональності та розширення можливостей.

Провівши аналіз етапів розробки мобільного додатку, можна виділити етапи для розробки додатка для кав'ярні, до першого основного етапу розробки слід віднести підготовчий етап. На цьому етапі проводиться детальний аналіз ніші, вивчається потенційна цільова аудиторія, її потреби та вподобання. Особлива увага приділяється аналізу діяльності конкурентів. Слід виділити основні переваги, які використовують конкуренти, а також проаналізувати їхні недоліки. Крім того, на підготовчому етапі варто визначити головну особливість додатка (оригінальний дизайн, корисна інформація).

До наступного етапу можна віднести створення дизайну додатка. На цьому етапі потрібно, щоб додаток був адаптивним, з динамічними кнопками, картинками і іншими елементами під розмір екрана користувача. Важливо, щоб він не тільки мав добрий вигляд, але і був на інтуїтивному рівні зрозумілим користувачеві. Для розробки буде враховуватися принципи UI/UX дизайну, та буде розроблено зрозумілий для користувача інтерфейс.

Наступний етап – це безпосередньо розробка додатка, а саме розробка архітектури. При створенні мобільного додатку вирішальне значення має забезпечення того, щоб кожен компонент виконував свою функцію. Невелика проблема на етапі створення архітектури мобільного додатка може мати далекосяжні наслідки для життєздатності кінцевого продукту.

Під час обговорення архітектури розробки мобільних додатків архітектори розглядають кращі галузеві практики та стандарти. Розробникам необхідно враховувати всі типи бездротових продуктів в цьому аналізі, щоб додаток легко працювало як зі смартфонами, так і з планшетами.

Після розробки архітектури йде проєктування бази даних. База даних дозволяє зберігати, оновлювати та отримувати інформацію, яка використовується додатком. На етапі проєктування бази даних визначаються сутності, атрибути та зв'язки між ними, що дає змогу створити оптимальну структуру даних, необхідну для ефективної роботи мобільного додатку. Для зберігання бази даних вибрано Firebase, тому що він достатньо потужний, щоб забезпечити швидку роботу з базою даних.

Наступним етапом є написання коду. Це важлива частина процесу розробки, оскільки саме завдяки йому можливо реалізувати функціонал додатка та забезпечити його коректну роботу. Цей етап дозволяє розробникам використовувати мови програмування, які підтримуються обраною платформою, та розробити функціонал додатка, який відповідає потребам користувачів. Крім того, написання коду дозволяє провести тестування додатка для виявлення можливих помилок, що є важливим кроком перед випуском додатка в App Store або Google Play Store.

Тестування є вирішальним етапом у розробці мобільного додатку для кав'ярень. Цей етап містить в собі проведення різних видів тестування, таких як функціональне, візуальне, взаємодійне тестування тощо. Ці тести дозволяють

виявити можливі помилки, які не були помічені розробниками під час написання коду.

Після виявлення помилок, розробники повинні виправити їх та провести повторні тести для перевірки. Цей процес може повторюватись кілька разів, поки додаток не буде повністю протестований та не буде відповідати всім стандартам.

Після тестування проводиться впровадження додатка, необхідно переконатись, що він відповідає всім стандартам платформи, не містить помилок та працює на різних пристроях. Крім того, розробники повинні підготувати всі необхідні матеріали для випуску, такі як знімки з екрану, опис додатка, відео тощо.

Після випуску додатка, розробники повинні продовжувати вдосконалювати його та відповідати на запити та огляди користувачів. Впровадження мобільного додатку є важливим етапом, оскільки саме завдяки йому додаток стає доступним для користувачів та може приносити користь для кав'ярень.

2.2. Дизайн мобільного додатку

Дизайн мобільних додатків – це завдання розробки мобільних додатків. Попри те, що мобільні додатки мають широкий спектр використання, їх об'єднує потреба в оптимальній зручності використання, доступності, взаємодії та загальному досвіді користувача. Одним словом, розв'язання всіх вищезазначених проблем.

Дизайн мобільного додатку є одним з важливих етапів розробки, який містить в собі створення інтерфейсу користувача, вибір кольорової гами, типографіки та іконок, що відповідає функціональності та меті додатка.

Збереження інтуїтивно зрозумілого дизайну програми є однією з найважливіших та найкращих практик, яку слід мати на увазі. Це зумовлено тим, що принципи дизайну інтерфейсу користувача та UX мають одну головну мету - зменшити когнітивне навантаження. Це відноситься до того, наскільки складно

користувачеві доводиться мислити під час використання програми. Змушення користувачів працювати надто важко, щоб виконати заплановану дію, може мати негативні наслідки, такі як заплутаність, перевантаження та недовіра до додатка. Отже, слід забезпечити зрозумілий та простий дизайн програми для максимального комфорту користувачів.

Використання зарекомендованих структур навігації є ще одним способом зробити вашу програму інтуїтивно зрозумілою. Це не тільки зменшує когнітивне навантаження користувачів, але і спрощує та знижує витрати на проектування програми під час розробки.

Наприклад, багато програм для Android розміщують головні навігаційні меню внизу екрана з виділеним активним меню. Очищений та лаконічний дизайн має стати вашим головним пріоритетом під час розробки програми, оскільки це запобігає перевантаженню користувачів занадто великою кількістю інформації, що створює когнітивне перевантаження [24].

На жаль, багато розробників помилково вважають, що заповнення додатка якомога більшою кількістю функцій зробить його більш надійним та привабливим для користувачів. Однак програма, яка обіцяє виконувати занадто багато різних завдань, може спровокувати феномен, відомий як парадокс вибору.

Найкращим емпіричним правилом, яке допоможе вам у цьому, є концепція «один екран, одне завдання».

Згідно зі статистичними даними, середня тривалість концентрації уваги людей знизилася з 12 секунд у 2000 році до 8 секунд у 2017 році та продовжує зменшуватися. У зв'язку з цим виникли два тренди UX/UI – мінімалізм та простота – які стають все більш актуальними для забезпечення успіху програмного забезпечення [33].

Один зі способів зробити свою програму більш мінімалістичною полягає у використанні білого простору. Білий простір дозволяє зменшити кількість елементів інтерфейсу та забезпечити більше "повітря" між інформаційними

блоками. Це полегшує сприйняття інформації та забезпечує більш зручний досвід користування.

Ще один спосіб забезпечити мінімалістичний дизайн полягає у прихованні навігаційних панелей та панелей управління. Замість багатьох кнопок і піктограм, використовуються спеціальні жести, які дозволяють користувачеві інтуїтивно переходити між різними категоріями та функціями програми. Це також допомагає зменшити кількість елементів інтерфейсу та підвищити його естетичність та функціональність.

Програми з динамічним інтерфейсом користувача змінюють свою форму та вигляд відповідно до типу пристрою та інших зовнішніх умов. Це на відміну від динамічного UX, де інтерфейс змінюється залежно від типу взаємодії (а не зовнішнього вигляду) і поведінки користувача (а не зовнішніх умов). Завдяки динамічному інтерфейсу користувача програма ідеально виглядає на різноманітних пристроях, від широких екранів до розумних циферблатів.

Створення динамічного інтерфейсу користувача просте і може включати різноманітні методи проєктування. Одним із таких методів є картковий інтерфейс з адаптивними блоковими картками (маленькими контейнерами для інформації).

Іншим прикладом адаптивності інтерфейсу є автоматичний перехід у темний режим у нічний час. Альтернативою темному режиму є налаштований темний режим і можливість динамічного перемикання кольорів. Іншим прикладом декоративного динамічного інтерфейсу є тінь на елементах програми, яка змінюється відповідно до кута, під яким тримають пристрій [24].

Мобільні додатки є галуззю, що стрімко розвивається. Попри ряду проблем, не пов'язаних із дизайном, які примушують користувачів видаляти додатки, якість загального враження, що вони отримують, значною мірою впливає на рівень утримання додатків.

В результаті аналізу найкращих практик та світових тенденцій, відповідно до прийнятих західних стандартів при розробці дизайну мобільного додатку для

кав'ярні передбачається, що дизайн буде мінімалістичний, з базовою навігацією. Головний екран містить меню, популярні, та нові товари. (Рис. 2.1).

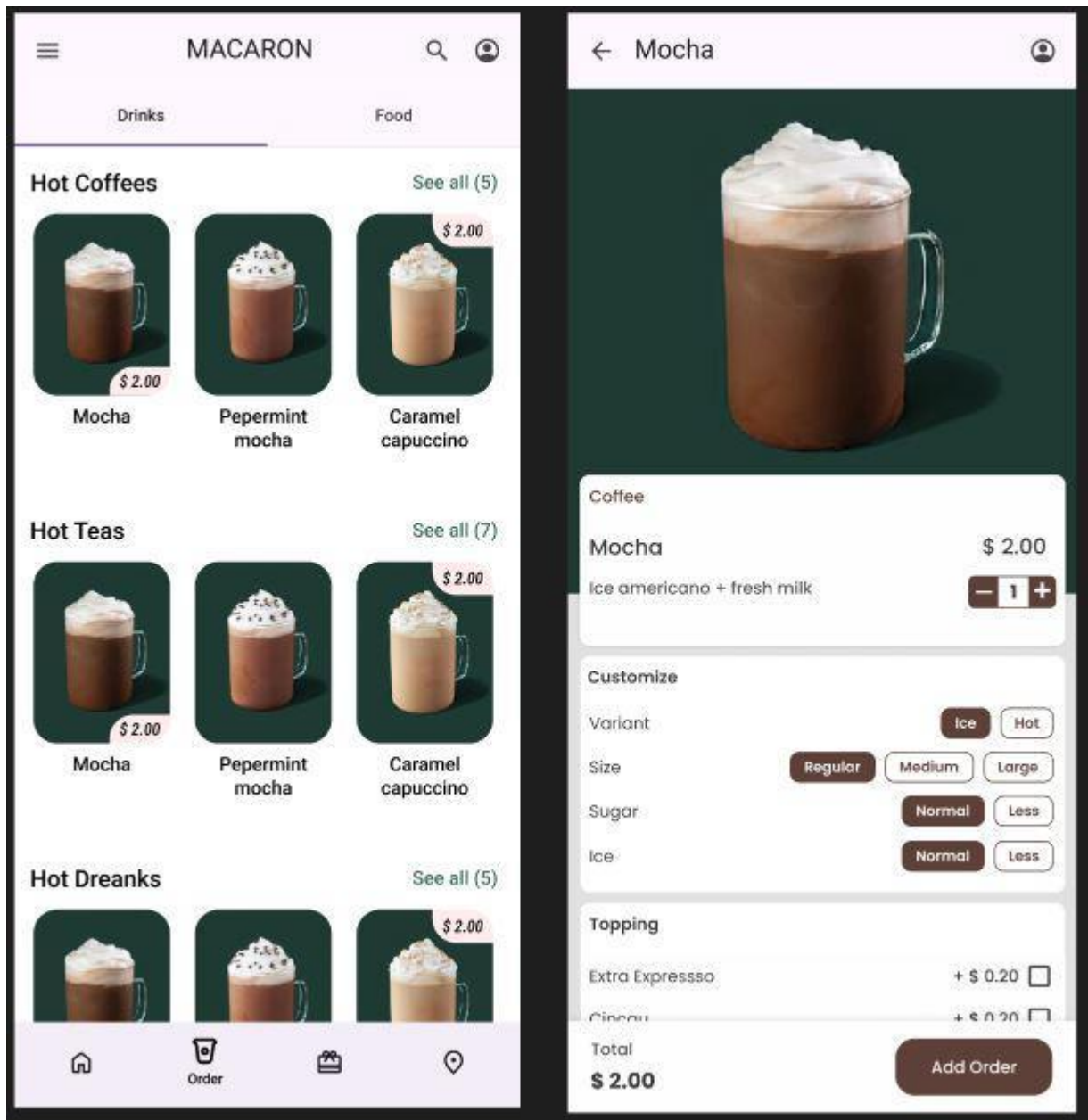


Рис. 2.1 Головний екран та екран вибору кави додатка

Меню представлене у вигляді горизонтального списку зліва зверху, яке відкривається при натисканні на нього. На головному екрані знаходяться також кнопки для швидкого доступу до інших розділів додатка, таких як налаштування профілю користувача та історія замовлень, акції, та місцезнаходження кав'ярні.

Екран деталей кави відкривається при кліку на конкретний продукт. Цей екран містить детальніше опис кави, фотографії, її складові та вартість. Крім того, можна додати можливість змінити складові та замовити додаткові інгредієнти.

Основна мета дизайну мобільного додатку для кав'ярні - забезпечити зручний та простий спосіб замовлення кави вдома або в офісі, а також збільшити лояльність клієнтів до кав'ярні.

Щодо можливостей додатка, можна розглянути можливість онлайн-оплати, бонусну систему та можливість відстежувати стан свого замовлення в режимі реального часу (Додаток А).

2.3 Архітектура додатку

Архітектура мобільного додатку може бути визначена залежно від його функціональності та масштабів. У загальному випадку, мобільний додаток складається з трьох основних компонентів: клієнтської частини, серверної частини та бази даних.

Клієнтська частина відповідає за інтерфейс користувача та взаємодію з користувачем. Цей компонент може містити елементи, такі як екрани, кнопки, меню, форми введення, інформаційні повідомлення та інші.

Серверна частина забезпечує функціональність додатка, що потребує обробки на сервері. Цей компонент може містити логіку додатка, функції та процедури обробки запитів, базу даних, сховища даних, API та інші елементи. База даних забезпечує зберігання та обробку даних, які використовуються в мобільному додатку. Цей компонент може містити таблиці, зв'язки між ними, зберігання інформації про користувачів, історію взаємодії з додатком та інші.

Крім того, можуть використовуватися додаткові компоненти, такі як бібліотеки, фреймворки та сервіси, що допомагають забезпечити функціональність додатка та поліпшити його продуктивність та безпеку.

При проєктуванні архітектури мобільного додатку необхідно враховувати його масштаб, функціональність, безпеку, продуктивність та інші вимоги. Важливо вибрати оптимальні рішення для кожного компонента та забезпечити їх взаємодію для досягнення мети додатка [32].

Під час розробки будь-якої мобільного додатка чи вебпрограми необхідно докладати максимум зусиль для того, щоб переконатися у якісному зібранні кожного компонента. Навіть найдрібніші проблеми, що можуть виникнути на етапі створення архітектури мобільного додатку, можуть суттєво підірвати якість кінцевого продукту. Зокрема, якщо бажаєте досягти успіху, необхідно ретельно й ефективно забезпечити високу якість архітектури мобільної програми.

Архітектура мобільного додатку – це складний набір структурних елементів та їх інтерфейсів, що формують систему та визначають їх поведінку в межах усіх структурних елементів. Це можна розглядати як скелет програми, від якого залежить уся робота мобільного додатку. Ігноруючи важливий елемент у процесі розробки архітектури мобільного додатку, можна на шляху до успіху поставити під загрозу свій проєкт. Складність побудови високоякісної архітектури залежить від розміру програми, але правильно розроблена архітектура може значно зекономити час, енергію та витрати у майбутньому [19].

У випадку відсутності архітектури програми можуть виникнути різні негативні наслідки, такі як високі витрати на розробку та тривалі процеси, труднощі з тестуванням та обслуговуванням програми, підвищені ризики помилок тощо.

Однак відповідна архітектура мобільних додатків може мати визначальне значення, сприяючи повному зміцненню та прискоренню процесів розробки програмного забезпечення та забезпеченню чіткого потоку даних, що спростить масштабування програми в майбутньому. Правильна архітектура може також забезпечити більшу гнучкість та дозволити використовувати методологію Agile для розробки програмного забезпечення. Це своєю чергою спрощує процеси

тестування та робить подальше обслуговування додатків більш ефективним та оптимізованим. Отже, правильна архітектура мобільного додатку може значно зекономити час та кошти бізнесу.

Початковою передумовою для створення успішної мобільної програми є розроблення правильної архітектури, яка містить низку важливих характеристик. Одна з них - незалежність від зовнішніх ресурсів, що виключає можливість залежності від певних бібліотек та фреймворків. Відповідно до цього принципу, проекти на iOS і Android, що демонструють успіх, не використовують фреймворки або подібні засоби, а засновані на ручній роботі розробників. Крім того, діаграма архітектури програми не повинна бути доступна операційним системам та базам даних інших програм, оскільки вона повинна функціонувати незалежно від будь-яких сторонніх впливів.

У контексті мобільних додатків, хороша архітектура має декілька ключових переваг. Перш за все, вона зробить процес розробки та підтримки програми більш ефективним і простішим. Крім того, програми з правильною архітектурою мають більшу гнучкість та зручність внесення змін, а також підвищену стійкість до помилок [32].

Наприклад, Clean Architecture є однією з найефективніших архітектур для масштабних проєктів з значним бюджетом, оскільки вона дозволяє розділити програму на логічні компоненти, що спрощує процес розробки, тестування та підтримки.

Даний вид архітектури є універсальним та надає можливість встановлювати різноманітні плагіни та швидко виправляти помилки. Проте, слід мати на увазі, що створення даної архітектури за допомогою фреймворків не є рекомендованим підходом. Програмний код слід розробляти відповідно до вимог системи, виконуючи всі необхідні етапи процесу розробки з початку до кінця (Рис 2.2).

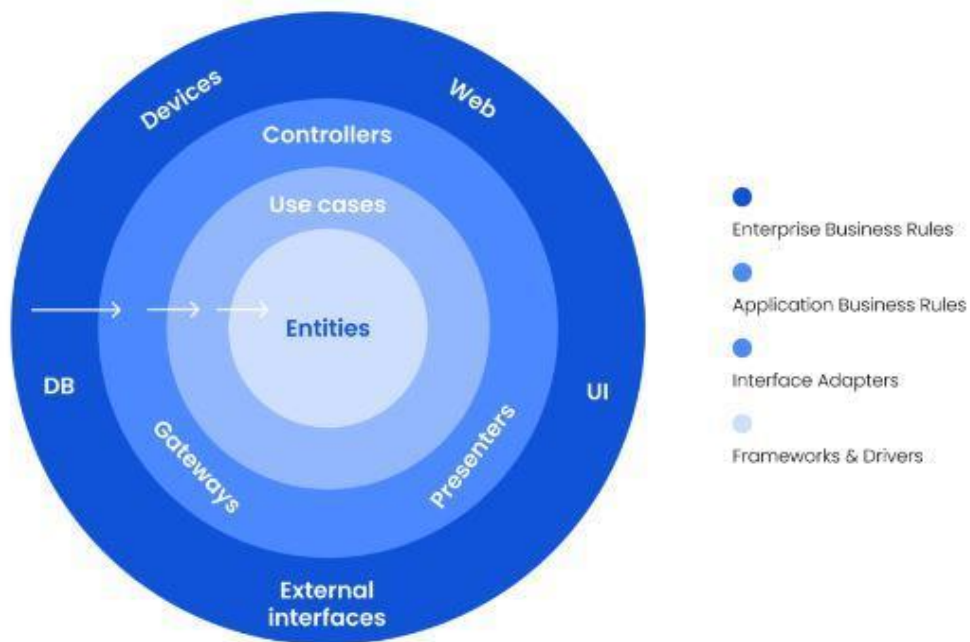


Рис. 2.2. Чиста архітектура мобільних додатків [19]

Кожен рівень мобільного додатку Clean Architecture є автономним і незалежним від інших програм та компонентів. Кожен рівень містить ключовий фрагмент, що містить логіку програми та важливі об'єкти. Всі рівні пов'язані між собою за правилом залежності, згідно з яким у вихідному коді всі залежності мають бути вказані лише внутрішньо. Наприклад, код внутрішнього кола не може згадувати сутності зовнішнього кола, такі як функції, класи, змінні тощо [32].

У сучасних реаліях Clean Architecture є найбільш надійним рішенням для розробки мобільних додатків. Вона дозволяє програмі бути незалежною від фреймворків, баз даних та інших зовнішніх залежностей. Переходи між рівнями в мобільних додатках Android Clean Architecture здійснюються через два інтерфейси: один для запиту та інший для відповіді. Це необхідно, щоб забезпечити відокремленість внутрішнього рівня від зовнішнього та забезпечити правило залежності, проте дозволяти передавати дані між рівнями (Рис. 2.3.).

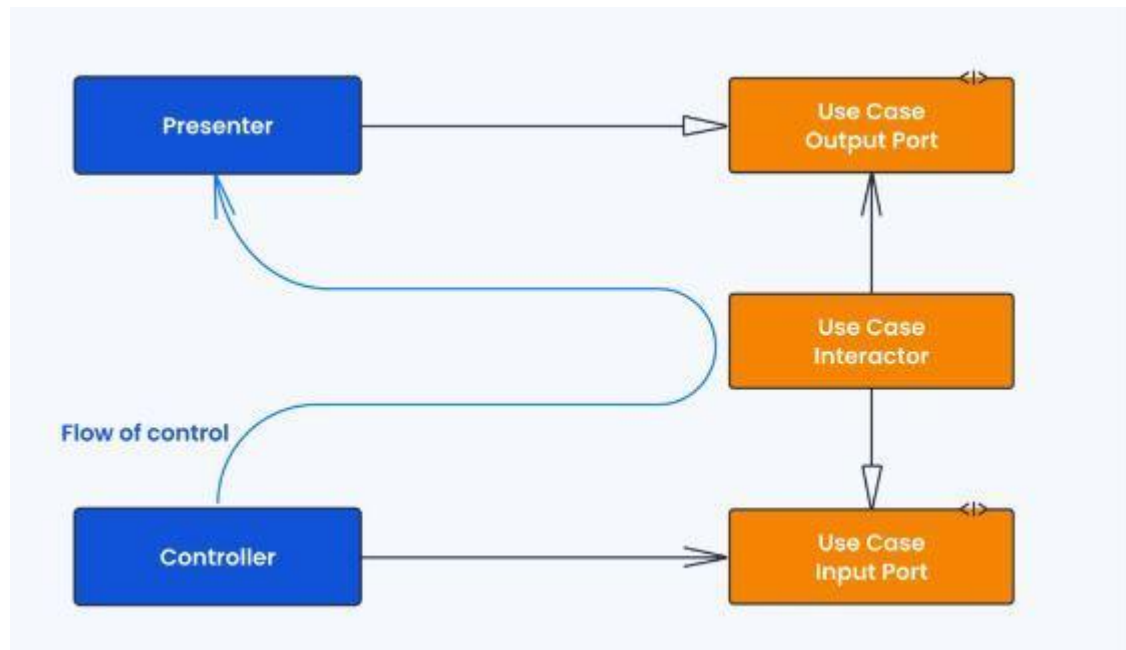


Рис. 2.3 Залежність архітектури в ОС Android [32]

У межах архітектури мобільного додатку Android використовується принцип інверсії залежностей (за літерою D від аббревіатури SOLID), щоб забезпечити спрямованість залежності у зворотний потік даних. Іншими словами, замість того, щоб випадки використання прямо залежали від презентатора (що порушує правило залежності), вони залежать від інтерфейсу на своєму рівні, і презентатор повинен реалізувати цей інтерфейс [19].

У процесі розробки високоякісної архітектури мобільного додатку важливо забезпечувати синхронізацію всіх процесів та можливостей додатку. Однак, запитання щодо того, які функції будуть дійсно вимагатися в майбутньому додатку, залишається відкритим. Отже, включення всіх ідей до першої версії не є оптимальним рішенням, оскільки це може призвести до збільшення часу розробки, перевантаження інтерфейсу та затримки в запуску продукту.

Добре розроблена архітектура є важливою для всіх типів додатків, які можуть бути складними чи простими. Використання ефективної архітектури в

мобільних додатках може заощадити час, зусилля та кошти. Програми з надійною архітектурою можуть бути легше налаштовані, перевірені та налагоджені.

Щоб оцінити якість архітектури вашого додатка, варто використовувати контрольний список наступних критеріїв:

- Ефективність: додаток повинен виконувати поставлені завдання та виконувати функції в будь-яких умовах. Система повинна бути ефективною, надійною та справлятися з усіма навантаженнями.
- Гнучкість: обране рішення має бути легко змінити, помилок має бути мінімум. Ви повинні мати можливість змінювати окремі елементи без фатальних наслідків для інших елементів.
- Розширюваність: програма повинна бути готовою до додавання нових функцій.
- Масштабованість: час, необхідний для розробки та оновлення, повинен зменшуватись. Надійна архітектура дозволяє направляти розробку в кілька паралельних потоків.
- Можливість тестування: архітектура для мобільних додатків повинна бути легко тестована, що означає, що кількість помилок зменшується, а надійність зростає [32].

При розробці програмного забезпечення, особливо у контексті мобільних додатків, першочерговою задачею є розробка багаторівневої архітектури, що передбачає розбиття програми на окремі рівні з різними функціональними призначеннями.

Один з таких рівнів – це рівень даних, на якому забезпечується безпека обслуговування даних, включаючи доступ до них, утиліти та сервісні інструменти. Саме на цьому рівні здійснюється інтеграція з базами даних та іншими джерелами даних, і він є основою, на якій можна далі розвивати бізнес-логіку та особливості презентації [30].

Окрім того, важливо вибрати шаблони для основних операцій, які визначатимуть спосіб роботи програми. При цьому слід мати на увазі гнучкість

дизайну, що дозволить відповідати змінам у вимогах бізнесу та очікуваннях клієнтів, не потребуючи змін в усій програмі.

Наступний рівень – рівень бізнес-логіки. Для створення мобільних додатків на платформі рівня бізнес-логіки виконується складна робота з керування, що включає окремі завдання щодо кешування, перевірки та інших технічних операцій. Цей рівень є місцем для розробки складних бізнес-процедур та визначення корпоративної політики. Зокрема, на рівні бізнес-логіки розташовані фасади програми з базовим робочим процесом, компонентами та сутностями.

Таким чином, рівень бізнес-логіки є ключовим компонентом мобільних додатків, на якому зосереджена увага на всіх аспектах, пов'язаних з логікою та бізнесом.

Останній рівень – це рівень презентації. Важливим кроком на рівні презентації є визначення профілю користувача, що дозволить відтворити на екрані зображення та розташування елементів, відповідно до вимог та очікувань користувачів. Особлива увага приділяється формату даних, що вводяться користувачем, який повинен відповідати вимогам та не допускати введення недійсних даних. Таким чином, рівень презентації визначає формат відображення даних та інтерфейсу, що забезпечує зручний та ефективний доступ до функціонала мобільного додатку [19].

Отже, провівши аналіз архітектури, можна розробити архітектуру. Архітектура мобільного додатку для кав'ярні буде базуватися на Clean Architecture – принципі розробки програмного забезпечення, що дозволяє розділити додаток на логічні шари та розмістити їх у відповідних модулях. Це дозволяє забезпечити високу міцність та гнучкість системи, забезпечити легкість тестування та розширення функціональності (Рис 2.4).

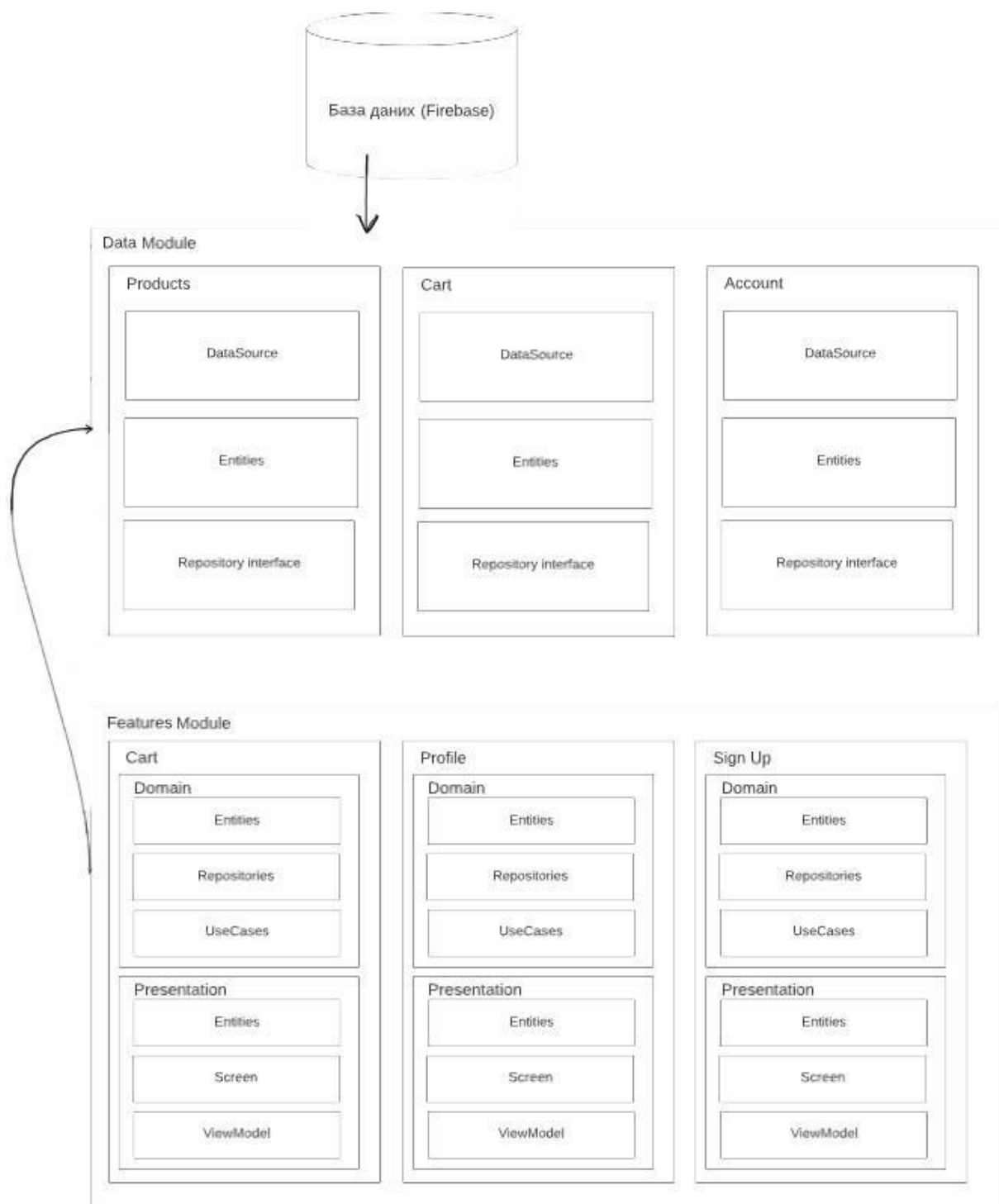


Рис. 2.4. Базова архітектура мобільного додатку для кав'ярні

Додаток складатиметься з основних модулів: Data, Features, Navigation, Core та App. В модулі Data будуть розміщені пакети програми, а саме: продукти, кошик акаунт тощо. Кожен пакет матиме свою відповідну структуру: у пакетах

знаходитимуться класи, що відповідають за доступ до даних про напої, їжу, користувача тощо.

Модуль Features буде складатися з під модулів продукти, кошик, профіль тощо. Під модулі вже діляться на певні пакети, де кожен пакет також матиме свою відповідну структуру. Пакет Domain відповідає за бізнес-логіку роботи під модуля. Наприклад під модуль «Продукти» опрацьовує як буде отримано ці продукти, за допомогою певної вибірки. Пакет Presentation міститиме екрани для відображення користувачу та певну UI-логіку.

Модуль Navigation відповідатиме за навігацію по мобільному додатку.

Модуль App – це ключовий модуль, який буде збирати всі модулі в єдиний проєкт.

Отже, Clean Architecture є ефективним вибором для розробки мобільного додатку, оскільки він забезпечує чітке розмежування відповідальності між різними рівнями абстракції, забезпечує належний зв'язок між ними та покращує можливості тестування та розширення додатка. Крім того, розбиття додатка на модулі є необхідним для підтримки масштабованості та полегшення процесу розробки.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ДОДАТКУ

3.1 Моделювання моделі бази даних

Розробка бази даних повинна починатись з її моделювання. На цьому етапі необхідно визначити структуру бази, включаючи склад таблиць, їхню структуру та логічні зв'язки. Структура реляційної таблиці визначається складом стовпців, їх послідовністю, типом даних кожного стовпця та їхнім розміром, а також ключем таблиці.

Для додатку для кав'ярні було розроблено схему бази даних, використовуючи програмне забезпечення LucidChart (Рис 3.1).

«LucidChart – це орієнтоване на веб програмне забезпечення, призначене для створення та редагування діаграм. LucidChart дозволяє користувачам об'єднуватися та працювати разом над створенням блок-схем, організаційних діаграм, шаблонів вебсайтів, UML-діаграм, ментальних карт, прототипів програмного забезпечення та багатьох інших типів діаграм. LucidChart надає зручний користувацький Drag&Drop-інтерфейс та можливості одночасного співробітництва декількох користувачів» [38]. Програма надає можливості експорту у векторні PDF-файли та растрові JPEG- та PNG-файли. LucidChart може працювати на більшості операційних систем, таких як Microsoft Windows, Mac OS X та Linux [26].

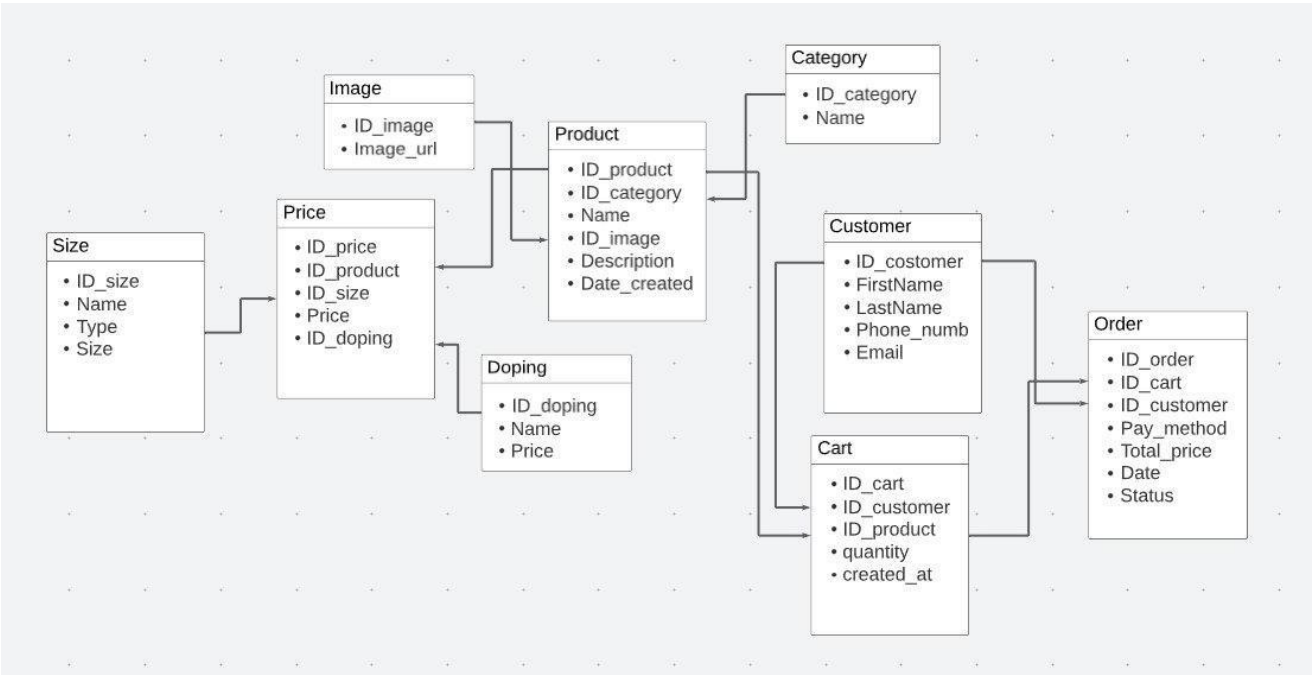


Рис. 3.1. Схема бази даних додатка для кав'ярні

Таблиця продуктів містить поля: Id_product, id_category, id_image, назва товару, опис та дата додавання (Табл. 3.1).

Таблиця 3.1

Таблиця Product

Product
ID_product
ID_category
ID_image
Name
Description
Date_created

Щодо ціни продукту, створена окрема таблицю для цін, яка містить такі поля, як Id_price, Id_product, розмір, добавки, ціна, ці поля дозволяють зберігати різні ціни для продуктів в залежності від розміру та додавання різних добавок

(Табл. 3.2). Таким чином, для кожного продукту можна мати багато записів в таблиці цін, які відповідають різним варіантам розміру продукту та добавок.

Якщо ціна залежить від розміру чашки, вам потрібно буде створити окремий запис ціни для кожного розміру чашки. Якщо ціна залежить від додавання додатків, вам потрібно буде створити окремий запис ціни для кожного можливого варіанту додатків.

Наприклад, якщо кавовий напій може бути з молоком або без молока, вам потрібно мати два записи в таблиці цін: один для кавового напою без молока та один для кавового напою з молоком.

Також гарним варіантом є створення окремої таблиці для допінгу, яка містить інформацію про кожен доступний допінг та його ціну. У цьому можна створити запис ціни для кожного можливого варіанту комбінації розміру продукту та допінгу.

Таблиці 3.2 – 3.3.

Таблиці Ціна та Розмір

Price	Size
ID_prize	ID_size
ID_product	Name
ID_size	Type
ID_doping	Size
Price	

Поля для таблиці допінгу:

- Id_doping - унікальний ідентифікатор допінгу
- Назва допінгу - назва допінгу (наприклад, молоко, цукор, сироп)
- Ціна - ціна допінгу

Таблиці цін має наступні поля:

- Id_price - унікальний ідентифікатор ціни
- Id_product - ідентифікатор продукту, до якого відноситься ціна
- Id_size - розмір продукту, до якого відноситься ціна (наприклад, мала, середня, велика)
- Ціна продукту - ціна продукту без додатків та з урахуванням розміру чашки
- Id_doping - ідентифікатор додатку, до якого відноситься ціна

Таким чином, для кожного продукту може бути декілька записів в таблиці цін, які вказують ціну продукту з різними розмірами чашок та додатками.

Коли кав'ярня продає не тільки каву, але і їжу, то доцільно, використовувати механізм управління різними типами продуктів в базі даних.

Один зі способів вирішення цього завдання полягає у створенні багатопольової таблиці продуктів, де деякі поля будуть заповнені для продуктів, які потребують цих даних, а для інших продуктів ці поля можуть залишатися порожніми.

Наприклад, для кави поля "Розмір" та "Допінг" є необхідними, тоді як для їжі, такі поля явно не мають сенсу. Розбиття таблиці продуктів на таблиці напоїв та їжі може бути доцільним, якщо кав'ярня має значну кількість різних типів продуктів і якщо вони відрізняються за характеристиками, такими як готовка, склад, калорії тощо.

Якщо кав'ярня пропонує переважно напої, то можна розбити таблицю на "напої" і "додатки до напоїв". В іншому випадку, якщо меню містить переважно їжу, можна створити таблиці "перші страви", "другі страви" та "десерти".

Проте, якщо кав'ярня має невелику кількість продуктів і вони не мають значної різноманітності, то можна зберегти одну таблицю "продукти", як в нашому випадку.

Створення таблиці «Розмір» доцільне для зберігання інформації про розмір напоїв та їжі в грамах або мілілітрах (Табл. 3.3).

Це дає можливість легко внести зміни до меню, наприклад, додати новий розмір, змінити розмір існуючого продукту або додати новий продукт з різними розмірами.

Також, це дозволить проводити аналіз продажів в залежності від розміру продукту та робити висновки про популярність різних розмірів.

Таблиці 3.4 – 3.5.

Таблиці Замовлення та Кошик

Order	Cart
ID_order	ID_cart
ID_cart	ID_customer
ID_customer	ID_product
Pay_method	Quantity
Total_price	Created_at
Date	
Status	

Таблиця «Замовлення» зазвичай використовується для зберігання інформації про замовлення, які роблять клієнти (Табл. 3.4). Така таблиця містить наступні поля:

Id – Унікальний ідентифікатор замовлення

Id_клієнта – Унікальний ідентифікатор клієнта, який зробив замовлення

Date – Дата та час, коли було зроблено замовлення

Pay_method – Тип оплати (готівка, карта)

Status – Статус замовлення, наприклад "нове", "в обробці", "відправлене", "виконане"

Total_price – Загальна сума замовлення

Таблиця «Кошик» зберігає інформацію про ті продукти, які клієнт обрав для покупки, але ще не підтвердив своє замовлення (Табл. 3.5). Таблиця «Замовлення» ж зберігає інформацію про оформлені замовлення, тобто про ті продукти, які клієнт обрав для покупки та підтвердив своє замовлення, а також про інформацію про клієнта, який зробив замовлення, дату та час замовлення, адресу доставки (якщо така є), інформацію про оплату тощо.

Отже, основна різниця між таблицею «Кошик» і «Замовлення» полягає у тому, що перша таблиця зберігає тимчасову інформацію про те, що клієнт планує купити, а друга таблиця зберігає остаточну інформацію про замовлення, яке було підтверджено та готове до обробки.

Додавання ідентифікатора таблиці «Кошик» в таблицю «Замовлення» є дуже важливим елементом для відстеження, який продукт з кошика був доданий до конкретного замовлення.

Один з можливих способів зробити це – додати поле "id_cart" (ідентифікатор кошика) в таблицю «Замовлення», яке буде посилатися на id кошика з таблиці "Кошик". Таким чином, можна буде відслідковувати, які продукти з кошика були додані до кожного замовлення.

У таблиці «Кошик» ми також використовуємо зовнішні ключі для посилання на інші таблиці, наприклад, id_user посилається на таблицю «Користувачів», а id_product та size_id посилаються на таблицю «Продуктів» і «Розмірів» відповідно. Також в таблиці ми зберігаємо кількість одиниць продукту, що додані до кошика, а також інформацію про розмір та додаткові складові, якщо вони є.

Таблиця «Користувач» містить інформацію про користувачів системи. Вона є однією з основних таблиць в системі кав'ярні, оскільки вона містить основні дані про клієнтів, які використовують цю систему. Це дозволяє переглядати історію замовлень користувача, запроваджувати програму лояльності та систему бонусів.

Якщо ж користувач не реєструється, то дозволяється замовлення без вказання контактних даних, але в такому випадку не буде можливості

відстежувати історію замовлень цього користувача. Тобто, в таблиці замовлень в такому випадку буде передаватися, що `id_client` відсутнє.

3.2. Реалізація додатка

Реалізація додатку починається з створення проєкту та налаштування його в середовищі Android Studio. До налаштування належить мінімальна та цільова версії Android SDK, це потрібно, для того щоб забезпечити сумісність додатку з різними пристроями Android та забезпечити працездатність належним чином з конкретною версією операційної системи Android. Для додатку вибрано мінімальну версію SDK – API 24: Android 7.0, оскільки вона покриває майже 95% користувачів, та легко підтримується розробниками.

Після налаштування створюються ключові модулі, в яких буде проводитися написання коду згідно практики Clean Code. Розбивання проєкту на модулі дозволяє логічно групувати функціональність за окремими компонентами. Кожен модуль відповідальний за конкретну частину додатку, таку як авторизація, меню, замовлення тощо (Рис.3.2). Це полегшує розробку, тестування та управління проєктом. Також розбиття проєкту на модулі згідно Clean Code допомагає забезпечити більшу чистоту, модульність, гнучкість та підтримку кодової бази, що полегшує розробку та підтримку мобільного додатку в майбутньому.

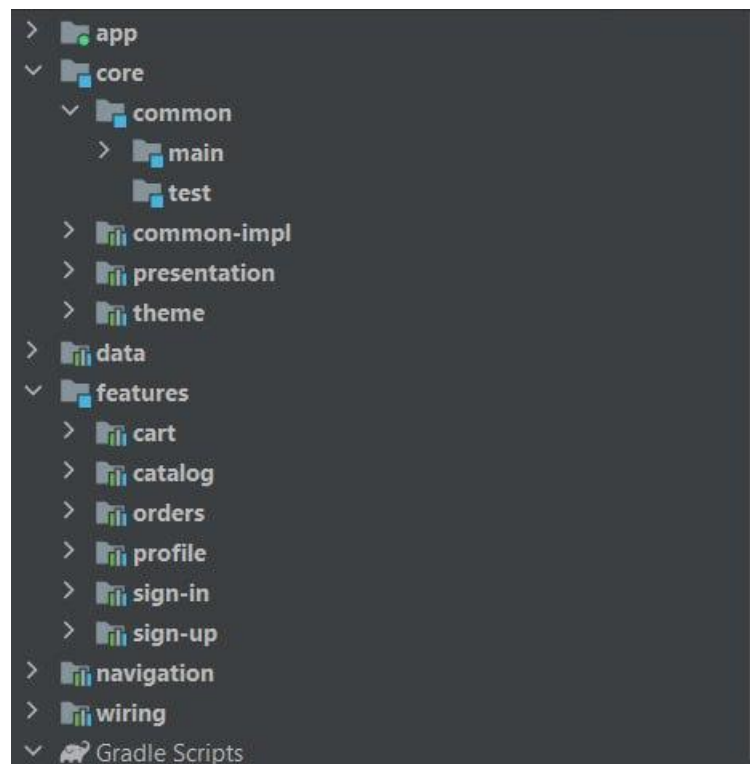


Рис. 3.2. Модулі додатку для кав'ярні

Далі потрібно підключити всі залежності та бібліотеки, які використовуються в додатку. Залежності додаються в файл **build.gradle** кожного модуля. Для цього використовується система збірки Gradle, яка автоматично завантажує необхідні бібліотеки з Інтернету та включає їх до проєкту (Рис. 3.3).

```

Libs = [
    Android : [
        CoreKtx          : "androidx.core:core-ktx:1.9.0",
        FragmentKtx      : "androidx.fragment:fragment-ktx:1.5.5",
        AppCompat        : "androidx.appcompat:appcompat:1.6.0",
        LifecycleRuntimeKtx : "androidx.lifecycle:lifecycle-runtime-ktx:2.5.1",
        LifecycleViewModelKtx: "androidx.lifecycle:lifecycle-viewmodel-ktx:2.5.1",
        ConstraintLayout  : "androidx.constraintlayout:constraintlayout:2.1.4",
        RecyclerView      : "androidx.recyclerview:recyclerview:1.2.1",
        NavigationFragments : "androidx.navigation:navigation-fragment-ktx:$navVersion",
        NavigationUI       : "androidx.navigation:navigation-ui-ktx:$navVersion",
    ],
    Google : [
        Material          : "com.google.android.material:material:1.7.0",
        Hilt               : "com.google.dagger:hilt-android:$hiltVersion",
        HiltCompiler       : "com.google.dagger:hilt-compiler:$hiltVersion",
        FirebaseDatabase  : "com.google.firebase:firebase-database-ktx:20.1.0",
        FirebaseAuth       : "com.google.firebase:firebase-auth-ktx:21.1.0"
    ],
    JetBrains: [
        CoroutinesCore     : "org.jetbrains.kotlinx:kotlinx-coroutines-core:1.6.4",
        KotlinSerialization: "org.jetbrains.kotlinx:kotlinx-serialization-json:1.4.1"
    ],
    Testing : [
        JUnit: "junit:junit:4.13.2",
    ],
]

```

Рис. 3.3. Залежності та бібліотеки проєкту

Провівши підготовчі кроки можна починати створювати користувацький інтерфейс за допомогою файлів макетів XML (Рис. 3.4). Використання файлів макетів XML дозволяє розділити презентаційну частину додатку (макет) від логіки його роботи. Це покращує структуру проєкту і дозволяє розробникам спеціалізуватися на своїй області відповідальності. Файли макетів XML надають можливість створювати адаптивний інтерфейс, який може адаптуватися до різних розмірів екрану та орієнтацій пристрою.

Після створення макетів додається логіка користувацького інтерфейсу. За допомогою View Binding можна зручно звертатися до елементів інтерфейсу без явного виклику `findViewById()`. Це спрощує взаємодію з елементами і дозволяє

виконувати різні операції, такі як обробка натискання кнопки або введення даних користувачем (Рис. 3.5).

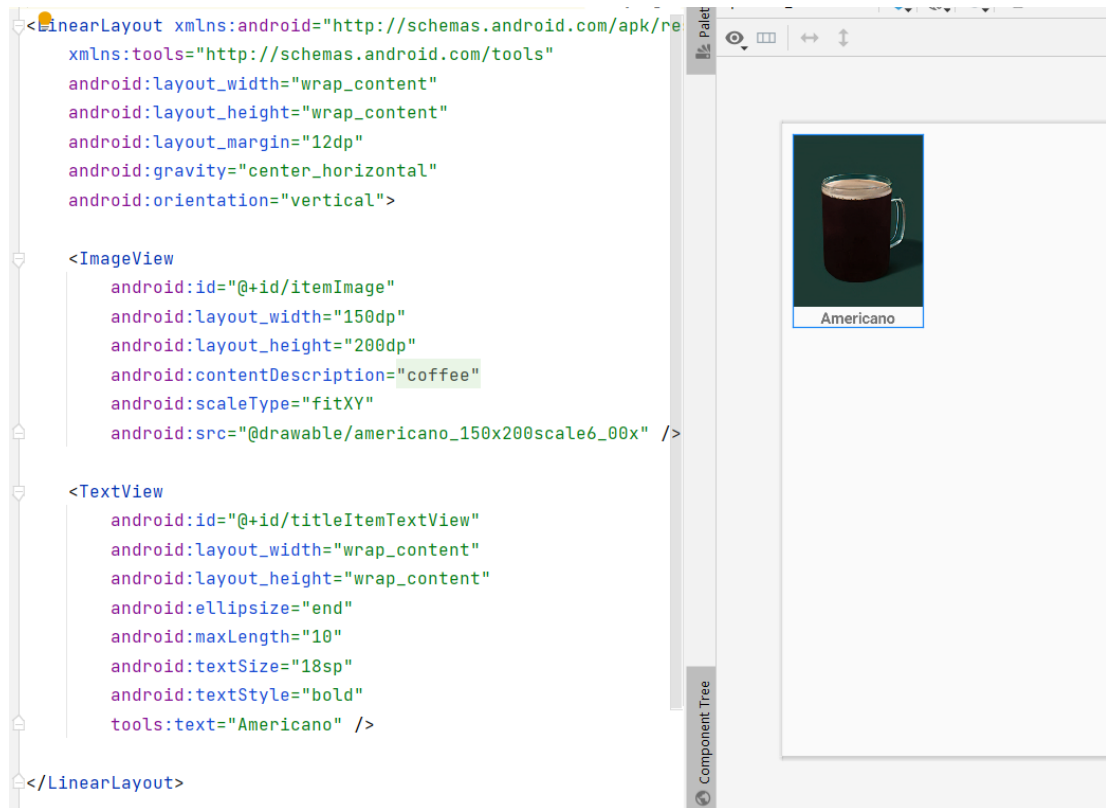


Рис. 3.4. Файл макету `Product_item.xml`

```

17  @AndroidEntryPoint
18  class HomeFragment : Fragment(), SectionAdapter.OnSectionClickListener,
19      ProductAdapter.OnProductClickListener {
20
21      private lateinit var binding: FragmentHomeBinding
22      private val viewModel by viewModels<HomeViewModel>()
23
24      override fun onCreateView(
25          inflater: LayoutInflater,
26          container: ViewGroup?,
27          savedInstanceState: Bundle?
28      ): View {
29          binding = FragmentHomeBinding.inflate(inflater, container, attachToParent: false)
30          return binding.root
31      }
32
33      override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
34          super.onViewCreated(view, savedInstanceState)
35          val adapter = SectionAdapter()
36          viewModel.homeState.observe(viewLifecycleOwner) { it: HomeState!
37              adapter.submitList(it.sectionList)
38          }
39          binding.parentRecyclerView.adapter = adapter
40
41          adapter.setSectionListener(this)
42          adapter.setInnerListener(this)
43      }
44
45      override fun onClick(product: Product) {
46          viewModel.launchDetailProduct(productId = product.id)
47      }

```

Рис. 3.5. Файл логіки користувацького інтерфейсу головного екрану

Після написання всіх файлів макетів XML та їх логіки в Fragment реалізовується логіка отримання даних з Firebase. Дані отримуються в модулі data, де знаходяться основні класи для роботи з даними. В даному проєкті використовується патерн Repository (Рис. 3.6). Патерн Repository використовується в програмуванні для розділення логіки доступу до даних від решти додатку, завдяки створенню єдиного місця для отримання та зберігання даних, приховуючи деталі роботи з базою даних або зовнішніми джерелами даних. Використання патерна дозволяє забезпечити чистоту, модульність та ефективність в роботі з даними в додатку, спрощуючи розробку, тестування та підтримку.

```

15 class RealProductsDataRepository @Inject constructor(
16     private val productsDataSource: ProductsDataSource,
17     private val lazyFlowSubjectFactory: LazyFlowSubjectFactory,
18 ) : ProductsDataRepository {
19
20     private val updateNotifierFlow = MutableStateFlow(OnChange(Unit))
21
22     override suspend fun getProductById(id: Long): ProductDataEntity {
23         return productsDataSource.getProductById(id)
24     }
25
26     @ExperimentalCoroutinesApi
27     override fun getProducts(filter: ProductDataFilter): Flow<Container<List<ProductDataEntity>>> {
28         return updateNotifierFlow.flatMapLatest { it: OnChange<Unit> } {
29             lazyFlowSubjectFactory.create {
30                 delay( timeMillis: 1000)
31                 productsDataSource.getProducts(filter) ^create
32             }.listen()
33         }
34     }
35
36
37     override suspend fun getAllCategories(): List<String> {
38         return productsDataSource.getAllCategories()
39     }
40

```

Рис. 3.6. Клас ProductRepository додатка

Вкінці проводиться тестування додатка, де спочатку пишуться тести і потім запускаються а локальній машині. Для цього використовується фреймворк для тестування, такий як JUnit, для створення тестових сценаріїв, виклику методів та перевірки очікуваних результатів. Тести перевіряти правильність роботи логіки додатку. Також пишуться інтеграційні тести, які перевіряють взаємодію між компонентами додатка, такі як користувацький інтерфейс, база даних та зовнішні API. В проєкті використано фреймворк Espresso, для автоматизованого тестування реального взаємодії користувача з додатком та перевірки правильності відображення даних.

Після завершення реалізації мобільного додатку для кав'ярні перевіряється чи додаток відповідає всім функціональним вимогам, вимогам користувачів, та чи працює стабільно, без помилок.

3.3. Використання додатка

Додаток для кав'ярні забезпечує користувачам зручний спосіб замовлення та отримання послуг кав'ярні. Для того щоб користуватися додатком, користувач переходить на головну сторінку, там він має можливість переглянути популярні напої або їжу. Він також може натиснути на напої, та переглянути детальний опис, та далі зробити замовлення, вибравши додаткові добавки до напою, або їжі (Рис 3.7).

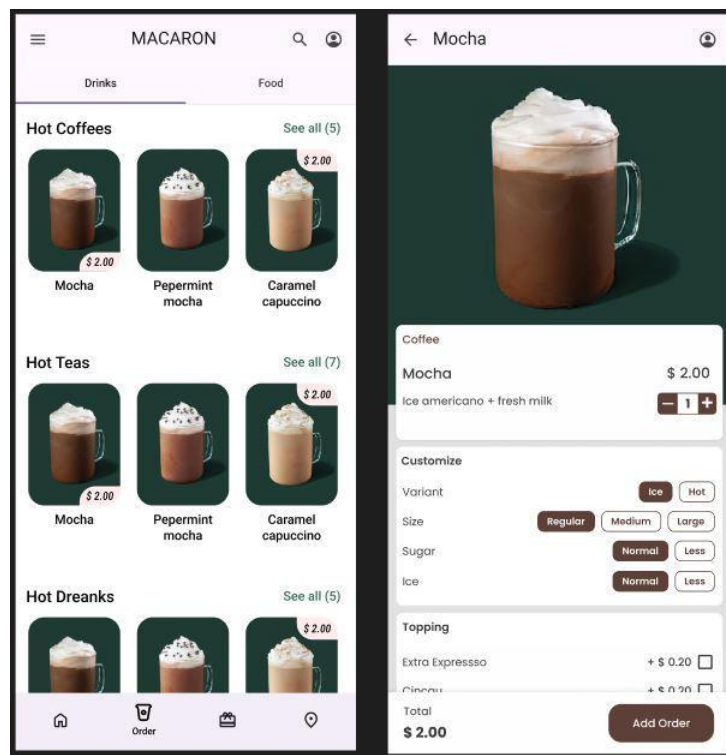


Рис. 3.7 Головний екран та екран вибору кави

Після вибору, користувач переходить в корзину, де має можливість продовжити замовлення та вибрати ще напої або їжу, або оплатити своє замовлення (Рис. 3.8).

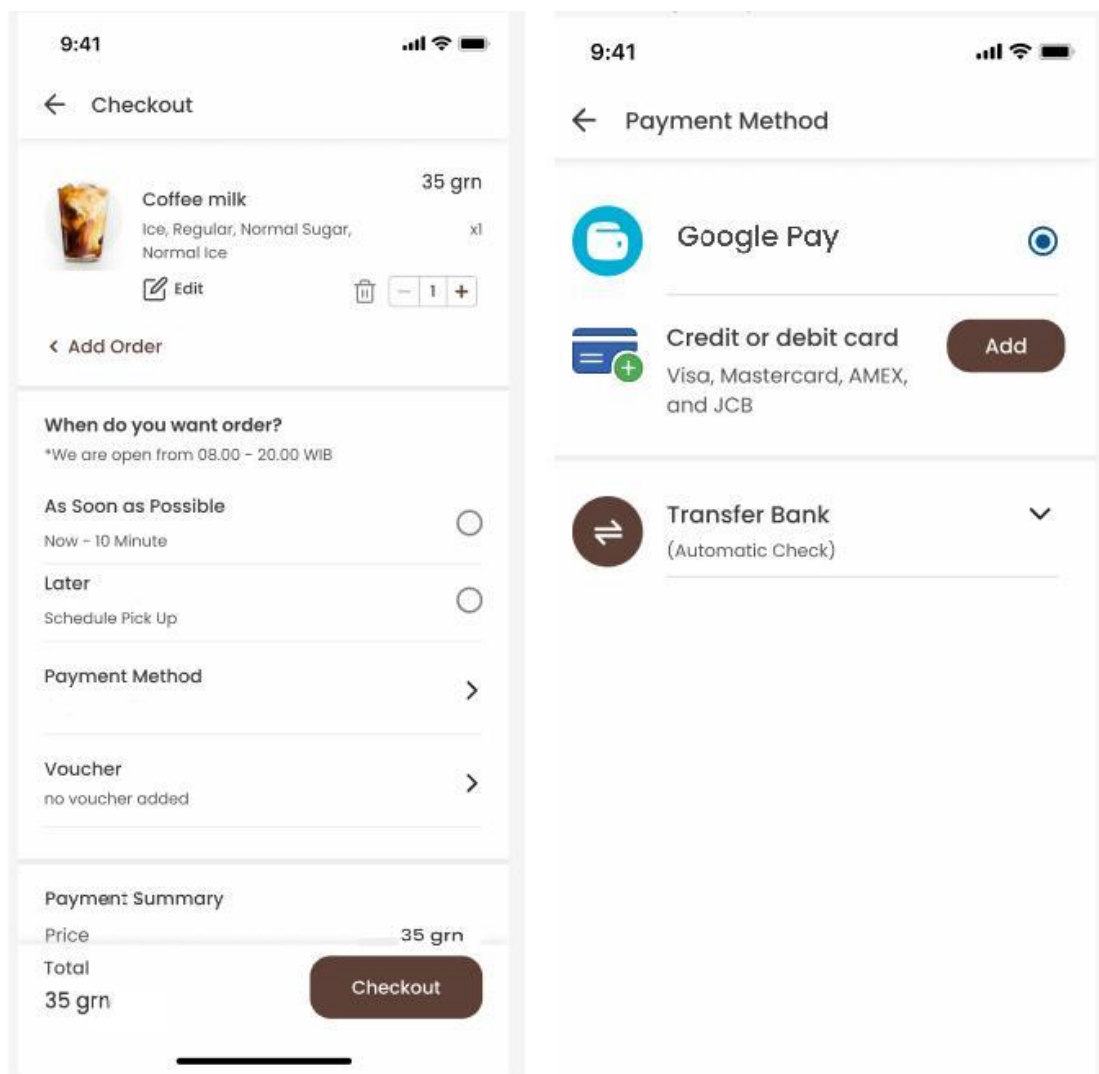


Рис. 3.8 Екран кошика та вибору типу оплати

Після успішної оплати користувачу приходить підтвердження в додаток, після якого він може забрати своє замовлення (Рис. 3.9).

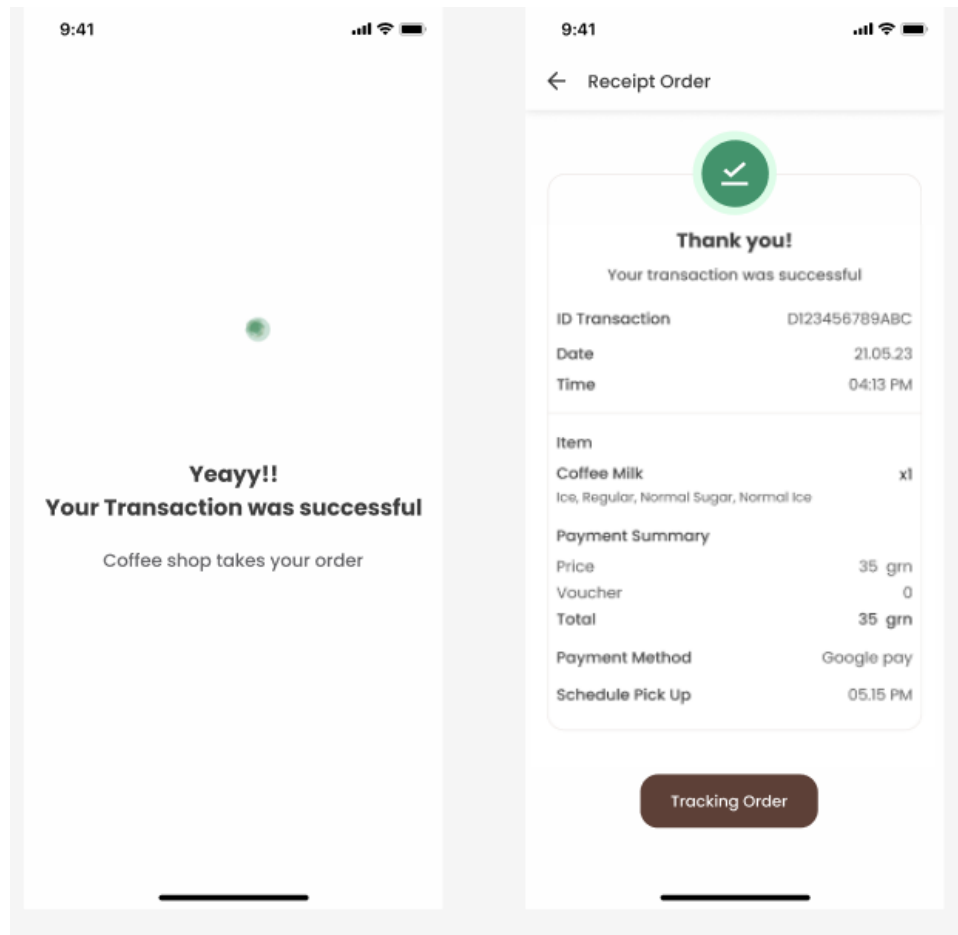


Рис. 3.9 Екран підтвердження оплати та замовлення

Також користувач має змогу зареєструватися в додатку, де матиме можливість отримувати особливі бонуси та переваги. Зареєстровані користувачі можуть брати участь у бонусній програмі, де вони заробляють бонусні бали з кожним замовленням і можуть обмінювати їх на знижки або безкоштовні продукти. Крім того, вони отримують доступ до ексклюзивних пропозицій та акцій, які можуть включати спеціальні знижки або безкоштовні додатки до замовлення.

ВИСНОВКИ

У випускній кваліфікаційній (дипломній) роботі був розроблений повністю функціональний мобільний додаток для кав'ярні на платформі Android.

В першому розділі було проведено аналіз та сучасний стан кав'ярень в Україні та проведено аналіз актуальності створення додатка. За останні роки, не дивлячись на війну та кризу в Україні спостерігається збільшення кількості відкриттів нових кав'ярень по всій країні. Кавова культура в Україні розквітає, а власники кав'ярень ставлять особливий акцент на якість кави, стильний дизайн та затишну атмосферу. Кав'ярні в Україні також активно впроваджують інноваційні технології, такі як мобільні додатки для замовлення та оплати, що робить відвідування кав'ярень ще зручнішим та сучасним. Також було проведено аналіз існуючих додатків, щоб зрозуміти, що потрібно покращити, та дізнатися їхні можливості, функціонал, дизайн та користувацький досвід.

Проведено аналіз сучасних інструментів розробки, оскільки це має велике значення для розробники програмного забезпечення. Він дозволяє оцінити інструменти, які можуть забезпечити ефективну та продуктивну розробку додатків.

В другому розділі була розглянута технологія проектування додатку. Були ідентифіковані етапи створення додатку, що включали налаштування проєкту, створення користувацького інтерфейсу, розробку дизайну та вибір архітектури додатку. Вибір Clean Architecture та розділення проєкту на багатомодульність надає структурований підхід до розробки, що дозволяє підтримувати високу якість коду, забезпечувати легку зміну та розширення функціоналу. Також розділення проєкту на модулі дозволяє розбити додаток на логічні компоненти, сприяючи кращій організації, перевикористанню коду та покращенню розширюваності. Вибір мінімалістичного дизайну надає естетичний та сучасний вигляд додатку,

зосереджуючись на простоті, чистоті та зручності використання. Він підкреслює основні функції та забезпечує зосередженість користувача на важливій інформації.

У третьому розділі була проведена програмна реалізація додатку. Були використані обрані інструменти розробки, зокрема мова Kotlin та середовище розробки Android Studio. Була виконана реалізація логіки отримання даних з Firebase, оскільки це забезпечує можливість зберігати та синхронізувати дані між пристроями та сервером, та дозволяє зручно керувати інформацією про кав'ярню, меню, замовлення та інші дані, що залучаються в додаток. Також було виконано створення користувацького інтерфейсу за допомогою файлів макетів XML, та написання тестів для перевірки працездатності додатку та виявлення помилок.

Розробка мобільного додатку дозволила забезпечити зручну платформу для замовлення та оплати їжі та напоїв. Додаток відповідає актуальним потребам клієнтів і сприяє поліпшенню їхнього досвіду від відвідування кав'ярні. Подальші розвиток та вдосконалення додатка можуть включати розширення функціональності та впровадження нових технологій для забезпечення ще більшого комфорту та задоволення користувачів. При розробці отримано ключові навички, що дозволить бути більш професійним в подальшій діяльності розробника програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Війна дає сміливість для мрії: як хлопці з Ірпеня та Бородянки відкрили кав'ярню у Львові. *LIGA*. URL: <https://life.liga.net/istoriyi/article/voyna-daet-smelost-dlya-mechty-kak-rebyata-iz-irpenya-i-borodyanki-otkryli-kofeynyu-vo-lvove> (дата звернення: 11.03.2023).
2. В Україні кав'ярень відкривається більше, ніж м'ясних та рибних магазинів разом узятих. *AgroNews*. URL: <https://agronews.ua/news/v-ukraini-novykh-kav-iaren-vidkryvaet-sia-bil-she-nizh-m-iasnykh-ta-rybnykh-mahazyniv-razom-uziatykh/> (дата звернення: 12.04.2023).
3. Громик О. Аналіз сучасного стану кавової індустрії в Україні. *Актуальні проблеми розвитку готельно-ресторанного бізнесу*, м. Луцьк, 9 трав. 2022 р. Луцьк, 2022. С. 250–265.
4. Державна служба статистики України. *Сайт державної служби статистики України*. URL: <https://ukrstat.gov.ua/> (дата звернення: 21.05.2023).
5. Додаток bacara coffee | бакара кава. *Додаток Bacara Coffee | Бакара Кава*. URL: <https://bacara.coffee/> (дата звернення: 18.04.2023).
6. Економічна правда. Які цифрові послуги доступні для підприємців у Дії. *Економічна правда*. URL: <https://www.epravda.com.ua/columns/2023/03/2/697617/> (дата звернення: 21.05.2023).
7. Кава, генератори та розширення бізнесу під час війни. *Суспільні новини*. URL: <https://suspilne.media/365442-peretvorili-kavarni-na-neoficijni-punkti-nezlamnosti-intervu-zi-spivvlasnikom-merezi-zakladiv-u-cernivcah/> (дата звернення: 08.04.2023).
8. Кадова культура в Україні: три хвили, розвиток альтернативи та світове визнання | Na chasi. *Na chasi*. URL:

<https://nachasi.com/specprojects/2021/10/04/coffee-culture-in-ukraine/> (дата звернення: 08.03.2023).

9. Мірер П. Мобільні додатки воєнного стану. Застосунки, що будуть корисними під час війни. *Суспільні новини*. URL: <https://suspilne.media/239406-mobilni-dodatki-voennogo-stanu-zastosunki-so-budut-korisnimi-pid-cas-vijni/> (дата звернення: 11.04.2023).

10. На Хмельниччині відкрили першу інклюзивну кав'ярню | Хмельницька обласна військова адміністрація – Офіційне інтернет-представництво. *Хмельницька обласна військова адміністрація – Офіційне інтернет-представництво*. URL: <https://www.adm-km.gov.ua/?p=122800> (дата звернення: 18.04.2023).

11. Полева Я. В. Цифрові технології у формуванні маркетингових стратегій бізнесу : thesis. 2021. URL: <http://dspace.puet.edu.ua/handle/123456789/11840> (дата звернення: 13.04.2023).

12. Ресторанний бізнес під час війни: де найбільш інтенсивно відкриваються нові заклади. *24 Канал*. URL: https://24tv.ua/business/restoranniy-biznes-pid-chas-viyni-vidkrittya-restoraniv-ukrayini_n2151721 (дата звернення: 19.03.2023).

13. Ресторанний бізнес під час війни та блекаутів: як змінився середній чек за рік. *UBR.ua*. URL: <https://ubr.ua/business-practice/restorannyi-biznes-vo-vremya-voyny-i-blekautov-kak-izmenilsya-sredniy-chek-za-god> (дата звернення: 19.03.2023).

14. Ринок HoReCa під час війни: поточний стан та ключові тенденції. *VoxUkraine* | «Вокс Україна» – більше ніж найкраща аналітика про Україну. URL: <https://voxukraine.org/rynok-horeca-pid-chas-vijny-potochnyj-stan-ta-klyuchovi-tendentsiyi/> (дата звернення: 21.03.2023).

15. Силівейстр В. Кав'ярні третьої хвилі: що купують частіше та чому – Poster. *Poster POS – програма автоматизації HoReCa: система обліку на*

планиеті. URL: <https://joinposter.com/ua/post/kavyarni-tretoy-khvyli> (дата звернення: 16.03.2023).

16. Споживання кави в Україні зростає швидше, ніж в інших країнах європи - аналітики pro-consulting. delo.ua | pro capital group. *Головна | Pro Capital Group*. URL: https://pro-capital.ua/ua/press_center/expert/potreblenie-kofe-v-ukraine-rastet-bystree,-chem-v-drugix-stranax-evropy-analitiki-pro-consulting.-delo.ua/ (дата звернення: 03.03.2023).

17. Фокус Р. Кав'ярні та "холостяцька" їжа. Якими будуть ресторани після війни. *ФОКУС*. URL: <https://focus.ua/uk/lifestyle/513144-kofeyni-i-holostyackaya-eda-kakimi-budut-restorany-posle-voyny> (дата звернення: 02.04.2023).

18. Як український ресторанний бізнес адаптується до викликів війни: дослідження. *Бізнес. Район. Ділові новини*. URL: <https://business.rayon.in.ua/news/547964-yak-ukrainskiy-restoranni-biznes-adaptuetsya-do-viklikiv-viyni-doslidzhennya> (дата звернення: 21.05.2023).

19. Alexander I. Architecture in jetpack compose – MVP, MVVM, & MVI. *Medium*. URL: <https://medium.com/mobile-at-octopus-energy/architecture-in-jetpack-compose-mvp-mvvm-mvi-17d8170a13fd> (date of access: 21.05.2023).

20. Android - secure & reliable mobile operating system. *Android*. URL: <https://www.android.com/> (date of access: 21.05.2023).

21. Android Studio: переваги та особливості. *QualityAssuranceGroup*. URL: <https://qagroup.com.ua/publications/android-studio-perevagy-ta-osoblyvosti/> (дата звернення: 09.04.2023).

22. Android studio & app tools - android developers. *Android Developers*. URL: <https://developer.android.com/studio> (date of access: 21.05.2023).

23. App stores list. *Business of Apps*. URL: <https://www.businessofapps.com/guide/app-stores-list/> (date of access: 08.04.2023).

24. Best practices for mobile app design. *Wix Playground LIVE*. URL: https://www.wix.com/playground/post/best-practices-for-mobile-app-design?utm_source=google&utm_medium=cpc&utm_campaign=13708482660^124757113432&experiment_id=^530755701272^_DSA&gclid=CjwKCAjw5pShBhB_EiwAvmnNV_YYyB3WupJFCEVleJUggMDnUnf8XzwsN4iu1knkJHpCbXpxLJ89uBoCQTcQAvD_BwE (date of access: 21.05.2023).
25. Case study - aroma kava | dinarys by andrew D | readymag. *Dinarys*. URL: <https://dinarys.com/aroma-kava/3/> (date of access: 18.04.2023).
26. Eisenberg A., Melton J. SQL standardization. *ACM SIGMOD Record*. 2000. Vol. 29, no. 1. P. 63–67. URL: <https://doi.org/10.1145/344788.344819> (date of access: 16.04.2023).
27. Firebase. *Firebase*. URL: <https://firebase.google.com/> (date of access: 16.03.2023).
28. Google maps platform | google for developers. *Google for Developers*. URL: <https://developers.google.com/maps> (date of access: 12.05.2023).
29. Google play console. *Android Developers*. URL: <https://developer.android.com/distribute/console> (date of access: 17.04.2023).
30. Kantamani S. P. Detailed guide on android clean architecture. *Medium*. URL: <https://medium.com/android-dev-hacks/detailed-guide-on-android-clean-architecture-9eab262a9011> (date of access: 21.04.2023).
31. Menon K. What is kotlin? Guide to kotlin programming language | simplilearn. *Simplilearn.com*. URL: <https://www.simplilearn.com/tutorials/kotlin-tutorial/what-is-kotlin-programming-language> (date of access: 01.03.2023).
32. Mobile app architecture: best practices in 2023 | intellectsoft. *Intellectsoft Blog*. URL: <https://www.intellectsoft.net/blog/mobile-app-architecture/> (date of access: 28.04.2023).

33. Mobile app design | InVision. *Collaborate better | InVision*. URL: <https://www.invisionapp.com/defined/mobile-app-design> (date of access: 21.03.2023).
34. Prostir.coffee by bonsite. *AppAdvice*. URL: <https://appadvice.com/app/prostir-coffee/1446636349> (date of access: 24.03.2023).
35. Singla S. What is the operating system market share? - scaler topics. *Scaler Topics*. URL: <https://www.scaler.com/topics/operating-system-market-share/> (date of access: 01.05.2023).
36. SQL Підручник. Уроки для початківців. W3Schools українською. *HTML CSS JavaScript PHP. W3Schools українською. Уроки для початківців*. URL: <https://w3schoolsua.github.io/sql/index.html#gsc.tab=0> (дата звернення: 12.04.2023).
37. Ukrinform. Першу в Європі роботизовану кав'ярню відкрили у Дніпрі. *Укрінформ - актуальні новини України та світу*. URL: <https://www.ukrinform.ua/rubric-technology/2673679-persu-v-evropi-robotizovanu-kavarnu-vidkrili-u-dnipri.html> (дата звернення: 11.03.2023).
38. What is Lucidchart learning center. *Lucidchart*. URL: <https://www.lucidchart.com/pages/tour> (date of access: 22.04.2023).
39. Why you need a website for your coffee shop (and how to make it) - hot cup factory. *Hot Cup Factory*. URL: <https://hotcupfactory.com/blogs/news/why-you-need-a-website-for-your-coffee-shop-and-how-to-make-it> (date of access: 21.05.2023).
40. Савчук А., Омельчук А. Аналіз сучасного стану кавової індустрії в Україні. *Цифрова трансформація фінансової системи України та країн V-4 в умовах євроінтеграції. Digital transformation of the financial system of Ukraine and V4 countries in the context of european integration*: III Міжнар. науково-практ. інтернет-конф., м. Дубляни - Ірпінь, 26 квіт. 2023 р. Дубляни, 2023. С. 264–267.

ДОДАТКИ

ДОДАТОК А

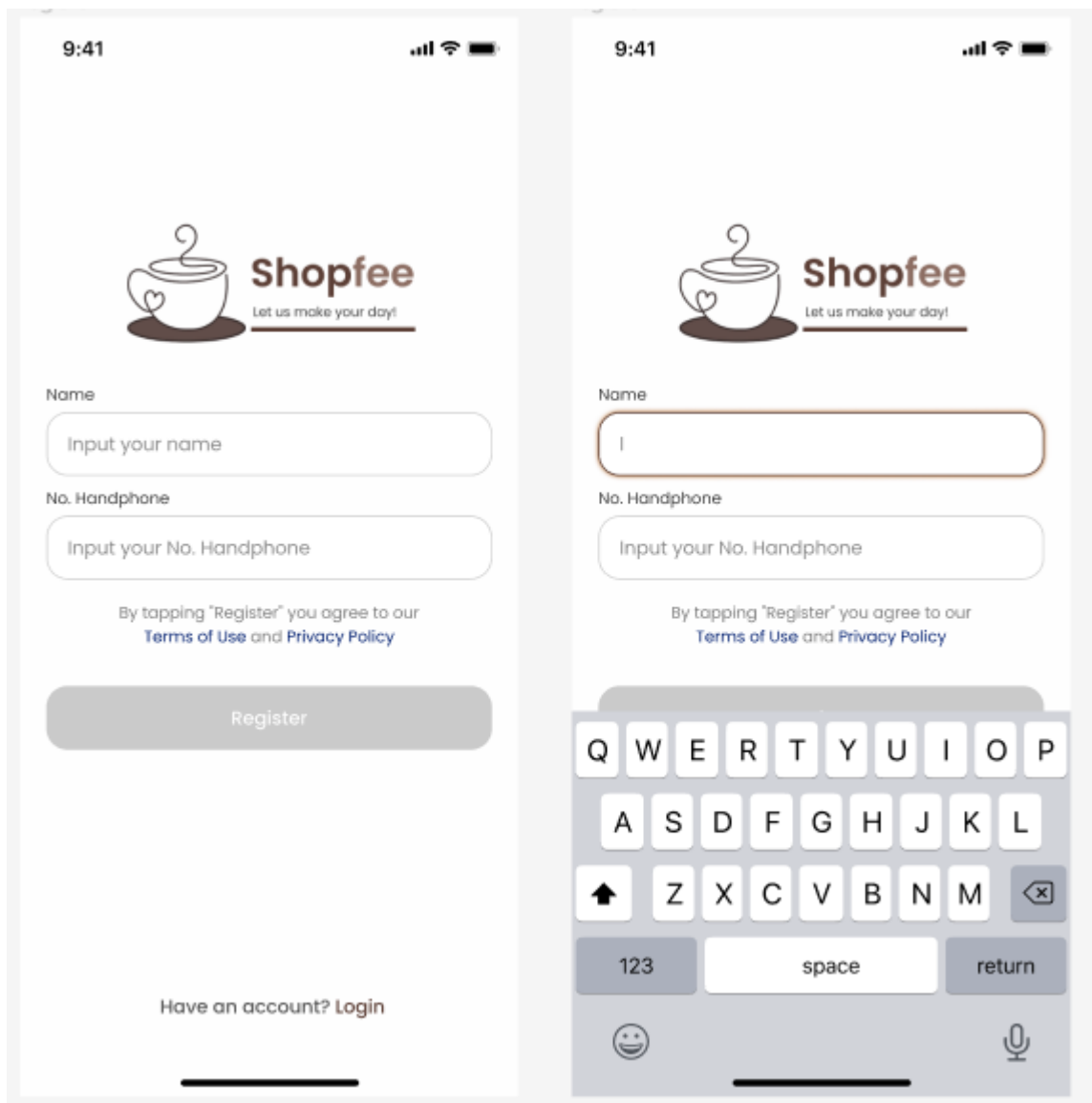


Рис. 1.1. Сторінка реєстрації

ДОДАТОК Б

```

@Composable
fun LoginScreen(
    viewModel: LoginViewModel,
    navController: NavController
) {
    val viewState = viewModel.viewState.observeAsState(LoginViewState())

    with(viewState.value) {
        LazyColumn(contentPadding = PaddingValues(start = 16.dp, end = 16.dp)) {
            item {
                Text(
                    modifier = Modifier.padding(top = 32.dp),
                    text = when (loginSubState) {
                        LoginSubState.SignIn -> stringResource(R.string.sign_in_title)
                        LoginSubState.SignUp -> stringResource(R.string.sign_up_title)
                        LoginSubState.Forgot -> stringResource(R.string.forgot_title)
                    },
                    style = TextStyle(
                        color = AppTheme.colors.headerTextColor,
                        fontWeight = FontWeight.Bold,
                        fontSize = 50.sp
                    )
                )
            }
            item {
                Row(modifier = Modifier.padding(top = 16.dp)) {
                    Text(
                        text = when (loginSubState) {
                            LoginSubState.SignIn -> stringResource(R.string.sign_in_text)
                            LoginSubState.SignUp -> stringResource(R.string.sign_up_text)
                            else -> ""
                        },
                        style = TextStyle(
                            color = AppTheme.colors.primaryTextColor,
                            fontWeight = FontWeight.Normal,
                            fontSize = 14.sp
                        )
                    )
                    Spacer(modifier = Modifier.width(4.dp))
                }
            }
        }
    }
}

```

```

        if (loginSubState != LoginSubState.Forgot) {
            Text(
                modifier = Modifier.clickable {
                    when (loginSubState) {
                        LoginSubState.SignIn ->
viewModel.obtainEvent(LoginEvent.SignUpClicked)
                        LoginSubState.SignUp ->
viewModel.obtainEvent(LoginEvent.SignInClicked)
                        LoginSubState.Forgot -> Unit
                    }
                },
                text = when (loginSubState) {
                    LoginSubState.SignIn -> stringResource(R.string.sign_in_action)
                    LoginSubState.SignUp -> stringResource(R.string.sign_up_action)
                    else -> ""
                },
                style = TextStyle(
                    color = AppTheme.colors.actionTextColor,
                    fontWeight = FontWeight.Medium,
                    fontSize = 14.sp
                )
            )
        }
    }
}

item {
    when (loginSubState) {
        LoginSubState.SignIn -> SignInView(
            viewState = this@with,
            onLoginFieldChange = {
                viewModel.obtainEvent(LoginEvent.EmailChanged(it))
            },
            onPasswordFieldChange = {
                viewModel.obtainEvent(LoginEvent.PasswordChanged(it))
            },
            onCheckChanged = {
                viewModel.obtainEvent(LoginEvent.CheckBoxChanged(it))
            },
            onForgetClick = {

```

```

        viewModel.obtainEvent(LoginEvent.ForgetClicked)
    },
    onButtonClick = {
        viewModel.obtainEvent(LoginEvent.OnLoginButtonClicked)
    }
)
LoginSubState.SignUp -> SignUpView()
LoginSubState.Forgot -> ForgotView()
}
}
}
}

LaunchedEffect(key1 = viewState.value.loginAction, block = {
    when (val action = viewState.value.loginAction) {
        is LoginAction.OpenDashBoard -> {
            navController.navigate("${NavigationTree.Main.name}/${action.username}")
        }
        popUpTo(NavigationTree.Login.name)
    }
})

DisposableEffect(key1 = Unit, effect = {
    onDispose {
        viewModel.obtainEvent(LoginEvent.LoginActionInvoked)
    }
})
}

enum class LoginSubState {
    SignIn, SignUp, Forgot
}

sealed class LoginAction {
    data class OpenDashBoard(val username: String) : LoginAction()
    object None : LoginAction()
}

data class LoginViewState(
    val loginSubState: LoginSubState = LoginSubState.SignIn,

```

```

    val emailValue: String = "",
    val passwordValue: String = "",
    val fullNameValue: String = "",
    val rememberMeChecked: Boolean = false,
    val isProgress: Boolean = false,
    val isLoginPassed: Boolean = false,
    val loginAction: LoginAction = LoginAction.None
)

@AndroidEntryPoint
class HomeFragment : Fragment(), SectionAdapter.OnSectionClickListener,
    ProductAdapter.OnProductClickListener {

    private lateinit var binding: FragmentHomeBinding
    private val viewModel by viewModels<HomeViewModel>()

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentHomeBinding.inflate(inflater, container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val adapter = SectionAdapter()
        viewModel.homeState.observe(viewLifecycleOwner) {
            adapter.submitList(it.sectionList)
        }
        binding.parentRecyclerView.adapter = adapter
        adapter.setSectionListener(this)
        adapter.setInnerListener(this)
    }

    override fun onClick(product: Product) {
        viewModel.launchDetailProduct(productId = product.id)
    }

```

```
}  
  
override fun onSectionClick(sectionEntities: SectionEntities) {  
    viewModel.launchProductMore(sectionEntities.id)  
}  
}
```