МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

Литвиненко О. Є._____

«____»____2022 р.

ДИПЛОМНИЙ ПРОЄКТ
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ
"БАКАЛАВР"

Тема: Мобільний додаток бонусної карти кав'ярні_____

Виконавець: Жданов В.О._____

Керівник: Нечипорук О.П._____

Нормоконтролер: Тупота Є.В._____

Київ 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп’ютерної та програмної інженерії
Кафедра комп’ютеризованих систем управління
Спеціальність 123 «Комп'ютерна інженерія»
(шифр, найменування)

Освітньо-професійна програма «Системне програмування»
Форма навчання денна

ЗАТВЕРДЖУЮ
Завідувач кафедри
Литвиненко О.Є.
«_____»_____2022 р.

ЗАВДАННЯ
на виконання дипломної роботи (проєкту)
Жданова Владислава Олександровича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема дипломної роботи (проєкту): Мобільний додаток бонусної карти кав’ярні
затверджена наказом ректора від «15» лютого 2022 р. № 251/ст

2. Термін виконання роботи (проєкту): з 16 травня 2022 р. по 19 червня 2022 р.

3. Вихідні дані до роботи (проєкту): інформація про мобільні додатки, інформація про методи розробки програмних систем на базі мобільних ОС, мови програмування високого рівня, *Microsoft Office*

4. Зміст пояснювальної записки:

1) аналіз існуючих систем та методів розробки мобільного ПЗ
2) доцільність розробки програмної системи на базі мобільних ОС
3) проектування та розробка програмної системи для закладів кав’ярні
4) тестування програмного додатку для закладів кав’ярні

5. Перелік обов'язкового графічного (ілюстративного) матеріалу:

1) діаграма активностей
2) структура БД
3) діаграма класів
4) екранна форма інтерфейсу користувача
5) приклад тест-кейсу

Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Ознайомлення з постановкою задачі на дипломний проект	16.05.2022-17.05.2022	
2	Вивчення спеціальної літератури і технічної документації	17.05.2022-18.05.2022	
3	Аналіз основних завдань та цілей проектування	18.05.2022-19.05.2022	
4	Розробка мобільного додатку	20.05.2022-24.05.2022	
5	Проаналізувати сучасні методи розробки програмних систем	25.05.2022-27.05.2022	
6	Підготування розділу 1	27.05.2022-30.05.2022	
7	Підготувати розділ 2	31.05.2022-02.06.2022	
8	Підготувати розділ 3	03.06.2022-05.06.2022	
9	Оформити пояснювальну записку, пройти нормоконтроль та отримати рецензію	05.06.2022-07.06.2022	
10	Підготувати графічний демонстраційний матеріал	07.06.2022-09.06.2022	

7. Дата видачі завдання: “15" лютого 2022 р.

Керівник дипломної роботи (проєкту) _____ Нечипорук О.П.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Жданов В.О.
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи за темою «Мобільний додаток бонусної карти кав'ярні»: 55 сторінок, 18 рисунків, 7 таблиць, 24 використаних джерел.

Дипломна робота складається з трьох частин: аналітичної, теоретично-практичної та практичної. У перших двох частинах проведено аналіз предметної області, здійснено постановку задачі та опис об’єкту. У другій частині також описана розробка програмної системи. В практичній частині описані реалізація та тестування програмного додатку.

Областю застосування є підприємства малого та середнього бізнесу, які працюють у сфері послуг.

МОБІЛЬНИЙ ДОДАТОК, СЕРВЕР, БД, ANDROID

Об’єкт дослідження дипломної роботи – обмін даними між сервером та програмним додатком.

Предмет дослідження дипломної роботи – програмна система з обміну даними на базі клієнт-серверної архітектури через мережу Інтернет.

Мета дипломної роботи – створення сприятливого для кінцевого користувача мобільного додатку для використання у закладах кав’ярні.

Методи дослідження - опрацювання необхідного літературно-довідкового матеріалу, методи проектування, розробки та тестування програмної системи для обслуговування користувачів, огляд сучасних технологій та методів обміну інформацією для створення програмних систем.

Проведено аналіз доцільності розробки програмної системи; розглянуто методи обміну інформацією; досліджено підходи до результативного проектування, розробки та тестування програмної системи для програмної системи обслуговування користувачів; під час виконання тестування було отримано результати виконання функціональної роботи та проаналізовано зручність використання створеної програмної системи.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

ЦП – центральний процесор

ОС – операційна система

ПЗ – програмне забезпечення

ООП – об’єктно-орієнтоване програмування

ОЗП – оперативний запам’ятовуючий пристрій

БД – база даних

СКБД – система керування базою даних

ЖЦ – життєвий цикл

IDE – Integrated Development Environment (інтегроване середовище розробки)

SDK – software development kit (набір засобів розробки програмного забезпечення)

ВСТУП

Смартфони зробили наше життя набагато комфортніше та швидше. У міру постійного розвитку технологій, що лежать в основі смартфонів, індустрія мобільних програм також розвивається. За даними Statista, з 2021 року кількість користувачів мобільних телефонів у світі вже перевищила 7 мільярдів і ця тенденція буде тільки зростати (рис. 1). Розробка мобільних програм стала одним з основних побічних явищ вибухового зростання попиту на смартфони.

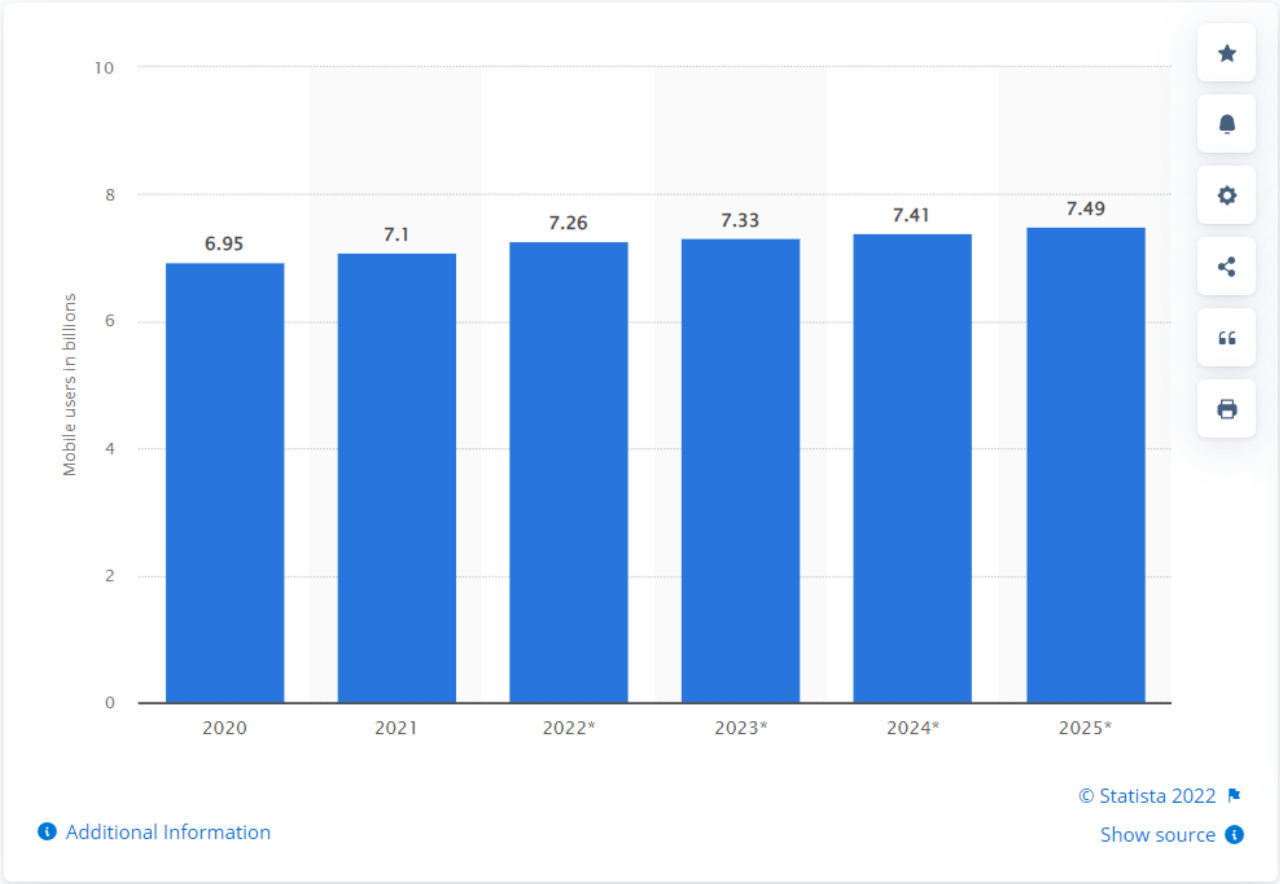


Рисунок 1 – Світова статистика кількості користувачів мобільних телефонів

Смартфон - визначення з Вікіпедії. У перекладі з англійської, «смарт» означає «розумний» або «кмітливий». Значення слова «смартфон» - це розумний телефон з набором додаткових функцій.

На відміну від звичайних мобільних телефонів, смартфони мають більш розвинену операційну систему (Android, iOS, Windows), для якої розробники можуть писати програмний код, реалізуючи нові ідеї (рис. 2). Адже після установки стороннього ПЗ можливості апарату стають ще ширше.

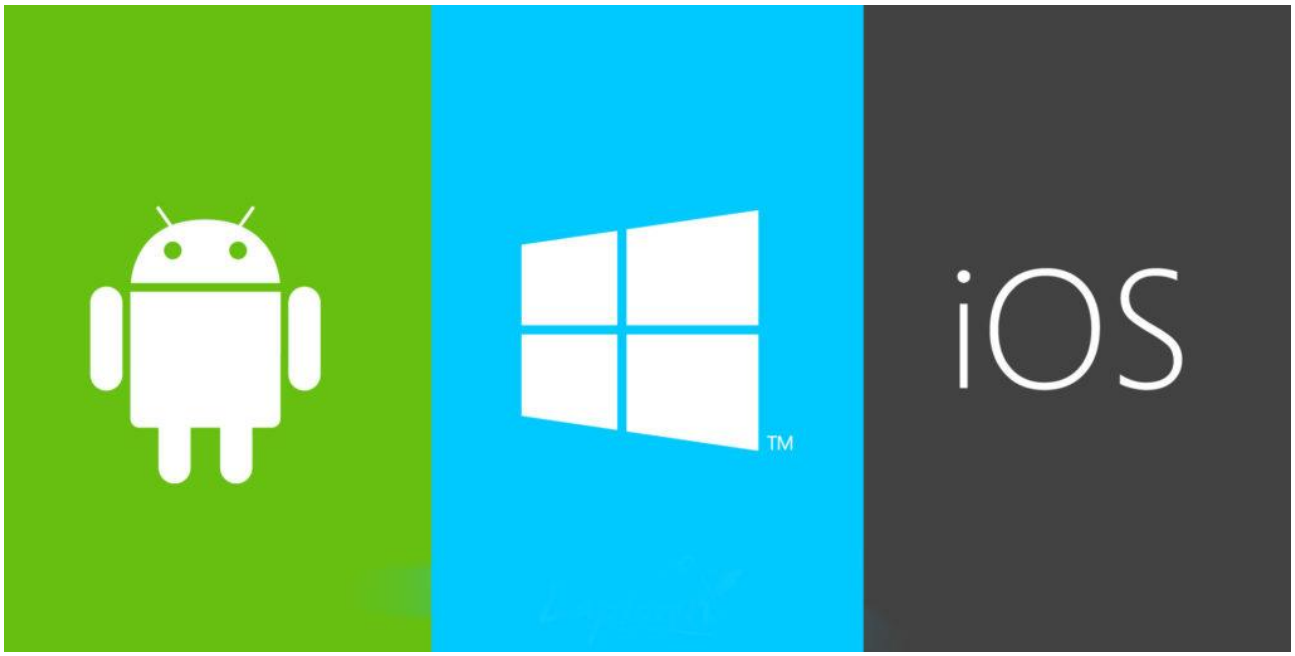


Рисунок 2 – Основні операційні системи смартфонів

Функції смартфона включають в себе не тільки можливість здійснювати дзвінки, як це було у випадку з мобільним телефоном. Вони мають на увазі набагато більше. Пристрій можна використовувати для серфінгу в інтернеті, GPS-навігації, спілкування в соціальних мережах, навчання, читання книг, мобільної фотографії, а також обробки відео. Мобільний гаджет може вести підрахунок пройдених кроків і стежити за станом артеріального тиску. Він може бути використаний для занять спортом, включаючи біг, спортивну ходьбу і стрибки зі скакалкою.

Функції і можливості апарату обмежені тільки уявою користувача і набором встановлених додатків. Додатково нові програми завжди можна встановити з магазину додатків, куди розробники постійно їх додають.

Актуальність обраної теми обумовлюється тим, тенденція збільшення кількості користувачів мобільними телефонами на базі ОС Android, iOS зростає з кожним днем. Більшості стає зрозуміло, що за допомогою смартфона можна отримати доступ до необмеженої інформації. За рахунок цього ринок мобільних програм є перспективною сферою, в якій вже працює велика кількість людей. Тому зараз створюється багато застосунків для різних сфер діяльності, в тому числі і для сфери послуг. Але розробники, в свою чергу, стикаються з проблемами вибору інструментів розробки та отримання максимальної користі від взаємодії з додатком.

Мета - провести дослідження ринку розробки мобільних додатків, аналіз сфери послуг, методів та забезпечення, які потрібні при розробці мобільних додатків для відповідної сфери діяльності, визначити основні тенденції. Сформувати навички самостійної творчої роботи та закріпити теоретичні знання, отримані в процесі навчання, а також суміжних загальнотеоретичних і спеціальних дисциплін. Використати набуті навички технічного мислення.

Завдання – розробити зручний і зрозумілий мобільний застосунок для кав'ярні, який отримує дані з БД, що знаходиться на сервері, з можливістю створення облікового запису користувача та програми лояльності. Успішно застосувати обрані методи та інструменти розробки мобільного програмного забезпечення.

Даний проект направлений на дослідження предметної області, яка стосується можливості організації роботи в закладах загального харчування, а саме кав'ярні з можливістю зручного обслуговування клієнтів через мережу Інтернет.

Для досягнення поставленої мети я використав такі методи дослідження: порівняльний аналіз і вивчення літератури, експериментальне моделювання та розрахунки.

Цей проект може бути реалізований в закладі загального користування після проведення додаткових дій та аналізу.

РОЗДІЛ 1
АНАЛІЗ ПРИНЦИПІВ РОЗРОБКИ МОБІЛЬНИХ ЗАСТОСУНКІВ НА РІЗНИХ ПЛАТФОРМАХ

1.1 Платформи для розробки мобільних додатків

Дуже поширене питання, яке ставить собі кожен розробник, яку платформу обрати для розробки мобільного додатку? У цьому технологічно керованому суспільстві кожна людина використовує смартфон, так як мобільний телефон все більше стає невід'ємною частиною життя.

Розробка мобільних програм грає вирішальну роль у створенні впізнаваності бренду для різних бізнес-одиниць. Тепер, коли мова йде про розробку мобільних додатків, на ринку домінують в основному дві основні платформи: Android та iOS. Обидва вони гарні по-своєму, і важко обрати лише один.

Також можна створити свою програму для обох платформ, але тоді, потрібно взяти до уваги той факт, що існуючий бюджет буде розтягнутий, і загальна вартість проекту значно збільшиться. Але першим кроком завжди має бути розуміння різниці між Android та iOS . У цьому розділі необхідно зрозуміти різницю між обома платформами, склавши докладний звіт про порівняльний аналіз.

Apple iOS — це операційна система назва якої перекладається як "операційна система для iPhone", але сьогодні вона встановлюється і на планшети iPad, але в будь-якому випадку ставиться тільки на мобільні девайси компанії Apple.

З системою **Android OS** сьогодні має справу практично кожен. ОС Android була розроблена спочатку під пристрої з сенсорним екраном, такі як планшети і смартфони. Сьогодні на базі цієї системи працюють навіть наручні годинники та телевізори. Вона створена на основі Linux kernel компанією Google і є однією з найпопулярніших мобільних ОС на сьогоднішній день. Була розроблена спеціально для інтуїтивного управління жестами через інтерфейс. І перший комерційний пристрій на базі ОС Android з'явився у 2008 році. На сьогоднішній день найбільш актуальна дванадцята версія системи – Android 12.

Завантаження нових додатків на iOS можливо тільки через AppStore, на відміну від системи Android, де можлива пряма установка apk-файлу.

Оскільки у світі налічується понад 7 мільярдів мобільних користувачів, не можна ігнорувати надзвичайну важливість наявності мобільного додатка для будь-якого бізнесу. Але знову ж таки, приймаючи це важливе рішення, спочатку потрібно вибрати між Android та iOS.

Слід брати до уваги різні фактори, але три основні, які слід враховувати – це компетенції, пов'язані з кожною платформою, процес розробки програми та цільова аудиторія для програми, яка розробляється. Необхідно докладно обговорити ці фактори разом з усіма іншими факторами, сформуючи структурований порівняльний аналіз і навести відповідний приклад .

1.2 Основні фактори, які необхідно враховувати перед вибором платформи для розробки програм.

1.2.1 Ринкова частка

Згідно з останніми даними Statista, на червень 2021 Android охоплює більше 80% ринку серед усіх ОС. На другому місці, звичайно ж, iOS, але з великим відривом вона займає трохи більше 18% ринку. Однак розрив виглядає досить великим, трохи скорочується залежно від регіону чи країни. Наприклад, в Індії Android займає колосальні 92% ринку, тоді як у США лідирує iOS із 59%

від загальної частки ринку. Весь сценарій залежить від аудиторії. Додаткові демографічні дані грають вирішальну роль, коли виникає ситуація, коли потрібно запустити свою програму в такому місці, як Індія, де Android – безперечний чемпіон. В цей час в Україні за спостереженнями Viber цей показник такий: Android – 84%, iOS – 15%.

1.2.2 Демографія

Демографічні дані допомагають скласти детальний порівняльний аналіз програми. Залежно від вибору та переваг операційної системи серед аудиторії потрібно створити додаток на цій ОС. Спробуємо розбити демографічні дані на основі статистики Asonalex і PR Newswire .

Демографія Android:

- Популярний серед вікової групи від 18 до 34 років;
- 10% – чоловіки;
- 12% ідентифіковано як інтроверти;
- Середня зарплата становить близько 37 тисяч доларів;
- 24% із більшою ймовірністю заробляють від 50 до 100 тисяч доларів на рік;
- У середньому, витрати на технології становлять близько 50 доларів на місяць;
- 29% схильні заощаджувати.

Демографія iOS:

- 18% із більшою ймовірністю будуть жінки;
- 29% старше 35 років;
- 14% ідентифіковані як екстраверти;
- 37-38% мають диплом про закінчення навчання;
- На 26% частіше витрачають гроші;

- Середня зарплата становить близько 53 тисячі доларів, а 67% становлять понад 200 тисяч доларів на рік;
- Витрати на технології становлять близько 100 доларів на місяць.

Ці тенденції відображають повну картину уподобань користувача. Однією з найважливіших відмінностей є те, що купівельна спроможність користувачів iOS більша, ніж користувачів Android. Цей фактор дуже важливий при виборі платформи для створення програм.

Друга за важливістю тенденція – поведінка користувачів. Користувачі Apple трохи сковзнули у бік екстравертів. Ці статистичні дані свідчать про лояльний культ яблука серед користувачів.

Нарешті, з наведеного вище обговорення очевидно, що клієнти iOS дуже лояльні до бренду, тоді як пристрої Android доступні за ціною. З іншого боку, користувачів Android набагато більше, ніж користувачів iOS. Але навіть якщо користувачів iOS менше, ніж користувачів Android, рівень доходу у користувачів iOS вищий. А коли рівень доходів вищий, витрати автоматично стають вищими.

Тож, хоча й ринкова частка Android набагато більше, ніж у iOS, але в той же час клієнтська база продуктів Apple більше, ніж у користувачів Android. Також їх платоспроможність більша, ніж у користувачів Android.

1.2.3 Апаратне та програмне забезпечення

Дуже важливий фактор, що впливає на рішення, - це мова програмування, яка допоможе створити програму. У платформах iOS та Android використовуються окремі мови програмування. Коли мова йде про Android, найчастіше використовуються мови Java та Kotlin. Крім них, ще однією мовою, яка теж не рідко використовується серед програмістів, є C++.

Java – один із найстаріших та перевірених методів створення якісної програми. Це популярна мова програмування для андроїд, яка у 2019 році

увійшла до ТОП 5 найкращих та функціональних мов. Проте в той же час вона досить складна. Він працює на кшталт ООП. Це означає, що він працює з класами, винятками, конструкторами, за якими потрібно ретельно стежити та продумувати логіку.

У 2014 році вона була визнана компанією Google як офіційна мова андроїд програмування, що значно полегшило життя розробникам. Процес розробки програм спрощується за рахунок візуального UI-редактора, функції автодоповнення коду та інших можливостей.

Що стосується функціональності, Java відноситься до однієї з найпотужніших мов. Вона дозволяє реалізувати найскладніші завдання та проекти, тому користується такою популярністю. Тут використовуються як Java класи, так і XML-файли.

Kotlin – це відносно молода мова, яка з'явилася у 2017 році. Але за такий короткий проміжок вона змогла завоювати любов багатьох програмістів, і в 2019 була визнана компанією Google кращою мовою програмування для андроїд, відсунувши Java на друге місце.

Kotlin увібрала у собі найкращі якості інших мов. Вона має масу переваг, серед яких автоматичне визначення типів даних, функції-розширення та інші. Вона проста у розумінні, освоєнні та доступна для кожного. Створено Kotlin на основі Java, тому переходити до нової мови з Java дуже просто.

Мова легко інтегрується з багатьма фреймворками, проста для вивчення, її код відкритий. Завдяки зрозумілому синтаксису підвищується продуктивність при створенні додатків, а послідовність легко дотримуватись. Короткість, легка читабельність та лаконічність – основні якості Kotlin.

Але при цьому вона теж має недоліки. Швидкість компіляції коду нестабільна, може проходити як швидко, так і зі значними затримками. Мова нова, тому не так багато навчального матеріалу, спільнота ще не розвинена ґрунтовно, тому в ході виникнення складнощів або питань розробникам нерідко

доводиться самостійно шукати шляхи вирішення. І все ж, Kotlin по праву вважається однією із найкращих мов в Android програмуванні.

Розробникам на iOS надається кілька варіантів: Objective-C, Swift та C++. Кожен з них має свої переваги та недоліки, підходять для виконання певних завдань. Всі вони відносяться до ООП, де виконуються основні принципи розробки, серед яких є групування подібних завдань до класів.

Objective-C – це мова програмування iOS, яка була розроблена ще у 80-х роках минулого століття. Довгий час вона була найпопулярнішою мовою розробки компанії Apple. З'явилася шляхом схрещування C та Smalltalk.

Переваги: код підтримується, регулярно оновлюється, при цьому внесення змін є простим. Багато документації та технічної літератури, велика спільнота, де люди допомагають одне одному пізнавати цю мову. Вона дуже схожа на будь-яку іншу із сімейства C, потрібно лише вивчити синтаксис. Крім того, Objective-C можна застосовувати всередині проектів, написаних на Swift, тому що вони сумісні.

Недоліки: складнощі з розумінням для новачків, складна у вивченні. Продуктивність не надто швидка, а сам процес складання уповільнюється, якщо відбувається взаємодія з файлами Swift.

При всіх недоліках Objective-C є дуже популярною мовою програмування на iOS. Вона використовується для великих проектів та інтернет магазинів.

Swift – це сучасна мова програмування для iOS, яка використовується більшістю розробників. Вона з'явилася у 2014 році, і увібрала у себе найкращі якості мов C та Objective-C. Головна особливість цієї мови у тому, що у етапі програмування значно зменшується кількість помилок з допомогою суворої типізації об'єктів. Додано велику кількість сучасних функцій: замикання, дженерики та інші. Процес програмування під iOS став більш гнучким та захоплюючим.

Переваги: вона дуже швидка, навігація по файлах зрозуміла і спрощена, повністю побудована на C, тому легко читається. Зовні максимально нагадує

англійську мову, її синтаксис спрощений і зрозумілий. Розмір коду зменшується за рахунок лаконічності. Є можливість використовувати шаблони, легко поєднується з Objective-C та має відмінний рівень безпеки. Використання динамічних бібліотек дозволяє програмам працювати швидше та стабільніше.

Недоліки: через те, що Swift – це молода мова програмування iOS, постійно відбуваються оновлення та зміни, тому доводиться ретельно стежити за готовими програмами. Залишається ще багато розробників, які використовують Objective-C, оскільки він більш стабільний.

Swift підходить для проектів різної складності, від інтернет-магазинів до банківських сервісів.

Інструменти розробки програмного забезпечення. Насамперед, потрібно обговорити апаратне забезпечення для обох платформ. Для розробки програм під iOS знадобиться Mac або будь-яке обладнання, орієнтоване на Apple. Неможливо розробити програму для iOS на іншому обладнанні, такому як Windows або Linux і т. д. З іншого боку, розробка програми для Android можлива на декількох пристроях. Можна почати з будь-якого обладнання, наприклад Mac, Windows та Linux. На відміну від iOS, операційна система не викликає занепокоєння при розробці програми для Android.

IDE: Спочатку IDE, що використовується для розробки під Android, була Eclipse. Розробники створювали додатки для Android на Eclipse, яка була аж ніяк не кращою IDE, тому що мала безліч недоліків та обмежень. Пізніше Google представила свою SDK Android Studio і всі розробники переключилися на неї. Вона була просунутою і безумовно краще, ніж попередня IDE. З іншого боку, для додатків iOS потрібно використовувати Xcode, яка є відмінною IDE для перетворення загального досвіду розробки Apple. Отже, згідно вибору мови програмування, потрібно звикнути до інструменту, який допоможе створити відповідну програму.

По-перше, для створення програми для iOS необхідно мати пристрій Apple, тоді як будь-який пристрій/ОС можна використовувати для створення

програми під Android. По-друге, Android Studio від Google використовується для розробки Android, а Xcode - для розробки програм під iOS. Якщо розробник тільки починає вивчати розробку мобільних додатків, йому краще використовувати більш нові мови, такі як Kotlin та Swift.

1.2.4 Витрати на розробку мобільних додатків

Найкраща частина розробки мобільних програм - це те, що можна почати з форми безкоштовно. Коли справа стосується цін, такої різниці між Android та Apple немає. Причина в тому, що і Apple, і Android надають безкоштовні інструменти та документацію для розробки програм. За допомогою обох цих платформ можна безперешкодно виконувати тестування та налагодження. Етап кодування не потребує жодних витрат. Вид витрат з'являється, коли потрібно завантажити свій витвір на ринок.

Необхідно заплатити певну ціну, щоб завантажити розроблений додаток у Google Play Store (якщо це додаток для Android) та Apple App Store (якщо ця програма для iOS). Для iOS ви повинні платити 199 доларів на рік, щоб завантажити свою програму в магазин додатків. Для підприємств ця ціна ще вища. З іншого боку, завантаження програми в Google Play Store коштує всього 25 доларів на все життя. Розробник повинен визнати цей факт, тому що якщо монетизація не спрацює належним чином, не буде повернення на свої інвестиції.

Плюси та мінуси обох планів завантаження. Сума, що стягується в магазині Google Play, набагато менше, ніж у магазині Apple. Деякі користувачі можуть вважати це перевагою для розробників Android. Але історія трохи інша. Наявність номінальної плати за все життя може призвести до накопичення безлічі поганих та фіктивних програм у магазині Google Play. Це, безумовно, не буде хорошим знаком для розробників Android, оскільки вони завантажують свою програму в таке місце, де вже є багато сміття. З іншого боку, коли справа доходить до Apple App Store, якість є головною проблемою.

Ще одна річ, на яку слід звернути увагу, полягає в тому, що оскільки існує періодична плата, яку потрібно платити щороку, існує більш висока ймовірність того, що програма працює добре і підтримує мінімальний стандарт, проходячи кілька процесів доопрацювання та перегляду щороку. Люди схильні купувати продукцію, орієнтовану на якість. З цієї причини покупки саме більше в Apple App Store, ніж у Google Play Store .

Тож, Apple Store коштує 99 доларів на рік, а Google Play Store - 25 доларів за все життя для завантаження програми. Очевидно, що завантаження Android-додатків набагато економніше, але Apple забезпечує першокласну якість та безпеку. Люди витрачають більше на покупку програм для iOS, ніж на програми для Android .

1.2.5 Прийняття в магазині додатків

Процес перевірки програми iOS в Apple App Store триває довше, ніж публікація Android-програми в Google Play Store. Причина, через яку Apple вимагає більше часу, дуже проста. Процес перевірки програми для iOS - непросте завдання, оскільки вона перевіряється командою досвідчених експертів Apple. Що стосується програм Android, то цей процес менш складний і вимагає менше часу. На відміну від Apple App Store, для додатків Android процедура в основному заснована на автоматичних тестах.

Google доручає розробникам обробляти деякі помилки, а надсилання кількох версій може здійснюватись протягом одного дня. Програмісти повинні приділяти першочергову увагу при відправці програми, оскільки існує високий ризик того, що після такого тривалого очікування програма буде відхилена для пристроїв iOS, а для Android програма може бути завантажена з помилками та представлена клієнтам.

Тож, App Store вимагає більше часу на перевірку та публікацію програми, тоді як Google Play Store займає порівняно менше часу, щоб завершити процес.

Це означає, що опублікувати програму в магазині програм досить складно, оскільки вона проходить через кілька рівнів сканування.

1.2.6 Принципи дизайну

Одна з основних причин лояльності Apple до бренду серед користувачів iOS – акуратний та чистий дизайн програми. Але Google також створила кілька правил для оптимального дизайну, який впадає у вічі. Material Design в Android, завдяки приємному ефекту паперу та чорнила допомагає в створенні красивого інтерфейса. Коротше кажучи, існують певні шаблони та рекомендації для програм iOS та Android .

Загалом обидва пропонують унікальний дизайн, та вибір суб'єктивний.

1.2.7 Варіанти монетизації додатків

Для отримання доходу за допомогою створеної програми знадобиться рішення про вибір методу для кожної платформи. У той час, як користувачам Android вдається ігнорувати рекламу, якщо вони хочуть зосередитися на контенті, клієнти iOS відхиляють рекламу всередині додатків. Щоб досягти успіху з цим варіантом потрібно з'ясувати свою цільову аудиторію та переконати її купити ваш продукт за допомогою ефективних стратегій продажу та маркетингу. Можна вибрати платні або преміальні додатки для iOS і дешевші програми для Android з великим упором на дохід від реклами.

Тобто, для Android основна частина доходу надходить від реклами. Для iOS основний дохід надходить від покупок у додатку.

1.3 Висновки до розділу

Таблиця 1 - Порівняльна таблиця, в якій зібрано усі основні моменти проведеного аналізу

Ключові аспекти	**Android**	**iOS**
Мова розробки	Java, Kotlin	Objective-C, Swift
IDE	Android Studio	Xcode
Цільова аудиторія	Менш цінна	Більш цінна
Філософія дизайну	Особливі вимоги	Гнучкий
Можливість налаштування	Легко налаштувати	Обмежена налаштованість
Складність розробки	Висока	Середня
Час розробки	Залежить від складності	Залежить від складності
Прийняття в магазині додатків	Короткий процес огляду програми	Тривалий процес перевірки програми (приблизно 7 днів)

Врешті-решт, можна сказати, що слід перейти до розробки додатків для iOS, якщо є обладнання для iOS і є бажання слідувати менш складному процесу розробки та отримувати більше доходу на користувача. Вибір за

Android, якщо є бажання задовольнити ширшу аудиторію в більшості країн, що розвиваються, з простим налаштуванням додатків і меншим часом очікування виходу на ринок. Зрештою, вибір платформи для розробки мобільних програм повністю залежить від бюджету, часу і ринкових вимог.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНО-АПАРАТНОЇ СИСТЕМИ

2.1 Архітектура та реалізація

Програмний додаток для ОС Android складається з набору активностей (Activities), кожній з яких відповідає графічне зображення програми. **Клас Activity** – важливий компонент програми Android, а спосіб запуску й об'єднання дій є основною частиною моделі додатків платформи. На відміну від парадигм програмування, в яких програми запускаються за допомогою методу main(), система Android ініціює код у екземплярі Activity, викликаючи конкретні методи зворотного виклику, які відповідають певним етапам її життєвого циклу. Кожна активність представлена у проекті класом, який реалізовано мовою Kotlin, що у однойменному файлі з розширенням .kt (рис. 3).

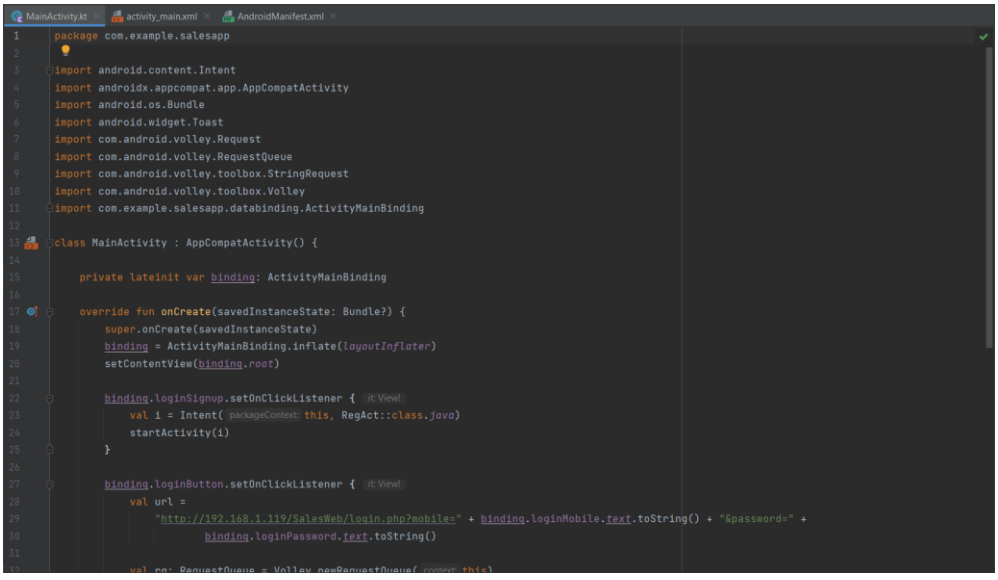


Рисунок 3 – Вигляд основної активності програми в Android Studio – MainActivity.kt

Кожній активності відповідає xml-файл (рис. 4). У вигляді xml-коду в ньому описано розташування об'єктів, що створюють візуалізацію додатку. Коли активність запускається система Android розпізнає розмір екрану мобільного пристрою та приводить контент, що виводиться, відповідно до розмітки, описаної xml-файлом. Тому виходить, що та сама активність буде виглядати однаково незалежно від діагоналі екрану.

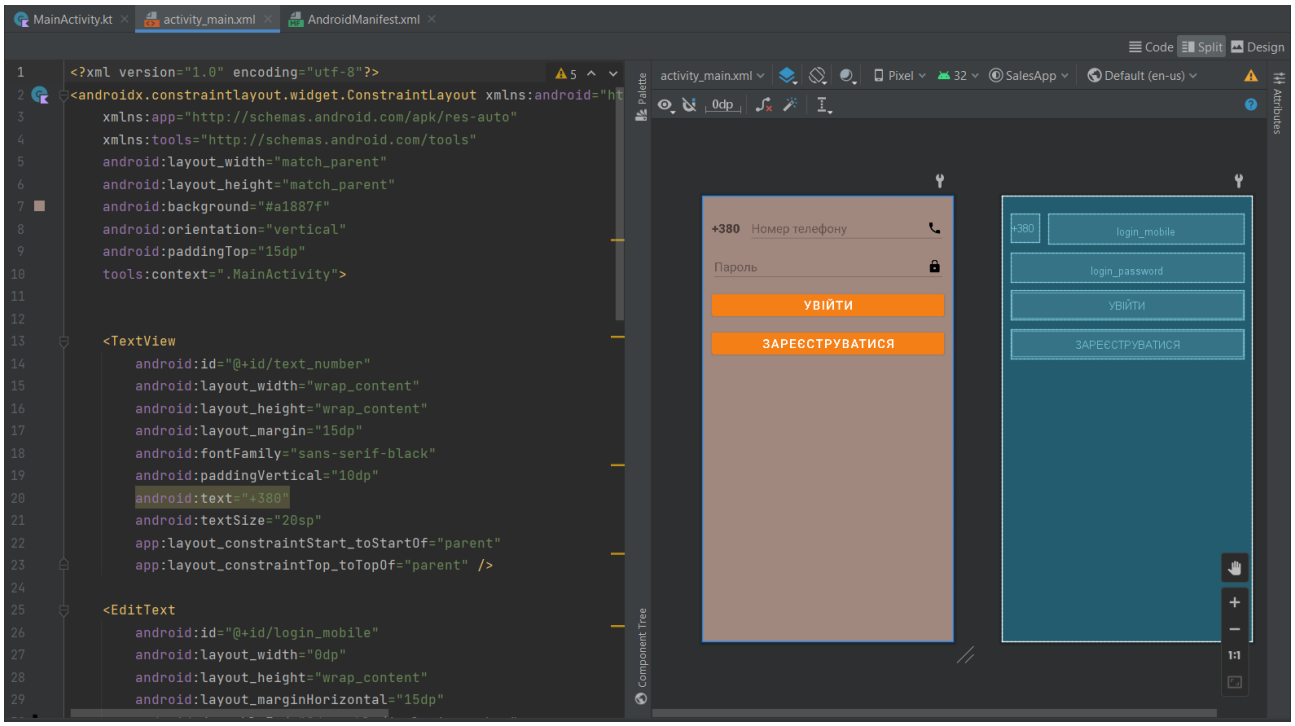


Рисунок 4 – Вигляд xml-файлу основної активності програми в Android Studio – activity_main.xml

Також, реалізована програма містить файл AndroidManifest.xml (рис. 5), що знаходиться у корені вихідного набору проекту. Файл маніфесту описує важливу інформацію про програму для інструментів складання Android, операційної системи Android та Google Play.

Вищеописані складові проекту отримуються автоматично та першочергово з базовими параметрами при створенні самого проекту в Android

Studio. Тоді їх необхідно доповнювати, а також створювати нові активності залежно від потреб розробника. В той час як в самому файлі AndroidManifest.xml більшість основних елементів додається під час реалізації програми (особливо при використанні шаблонів коду).

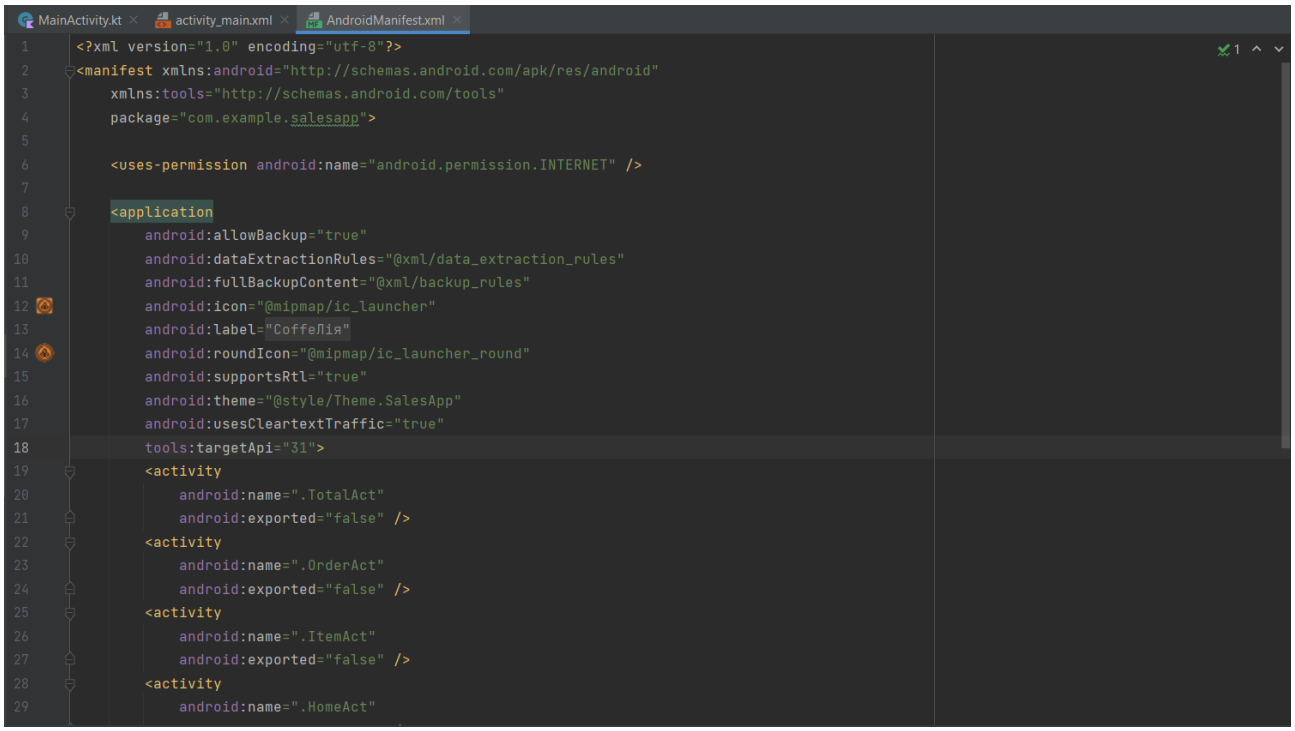


Рисунок 5 – Вигляд xml-файлу маніфесту в Android Studio - AndroidManifest.xml

Android-додаток працює у зв’язці з віртуальним сервером, тому проект реалізований на базі **«клієнт-серверної архітектури»** (рис. 6). Даний термін поняття комплексне:

Клієнт – пристрій користувача, який надсилає запит на сервер для отримання даних або для виконання деякого набору системних дій.

Сервер – являється спеціальним системним обладнанням, що призначене для вирішення кола завдань відповідальних за процес виконання заданих

програмних кодів. Сервер виконує роботи, такі як сервісне обслуговування на клієнтські запити, надання користувачам доступу до певних системних ресурсів, зберігання необхідних даних в БД або окремо.

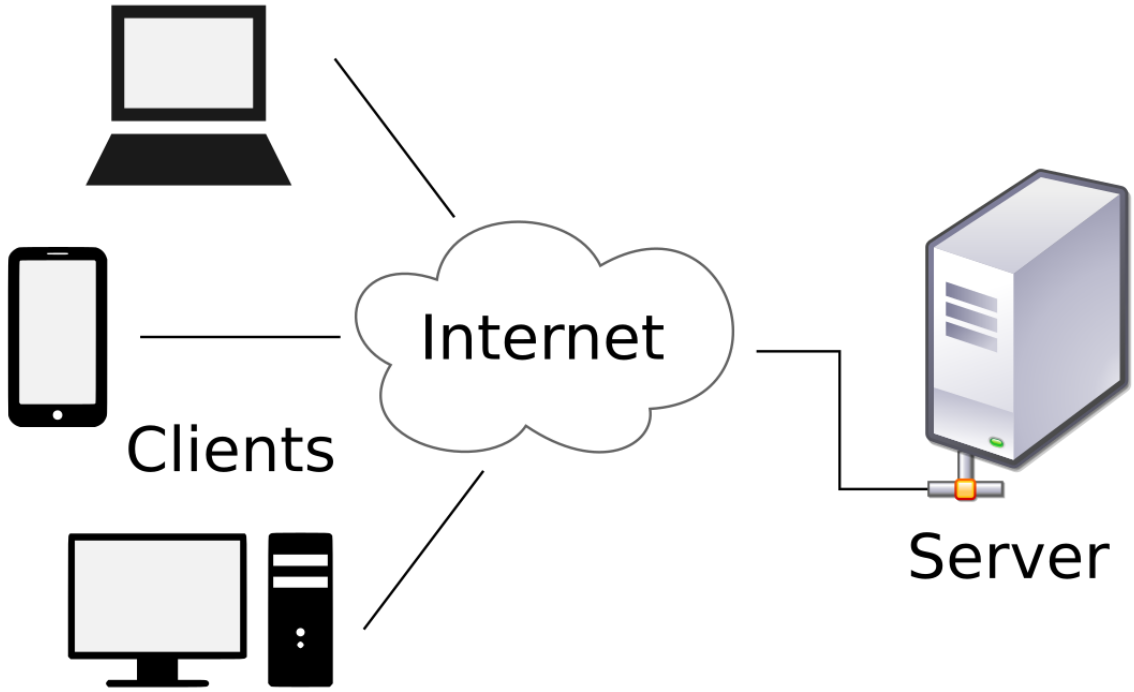


Рисунок 6 – Вигляд клієнт-серверної архітектури

Головною особливістю такої моделі можна вважати те, що користувач надсилає деякий запит на сервер, який потім системно обробляється і клієнту повертається кінцевий результат. Сервер може одночасно обслуговувати декількох клієнтів. Клієнтом виступає саме мобільний додаток.

В якості сервера використовується **WampServer** — платформа веб-розробки на Windows, яка дозволяє створювати динамічні веб-додатки з веб-сервером Apache, інтерпретатором скриптів PHP, БД MySQL та MariaDB (рис. 7).

Wamp діє як віртуальний сервер на комп'ютері. Під час реалізації проекту це дозволяє без жодних наслідків перевірити всі його функції, оскільки сервер локалізований на комп'ютері і не підключений до Інтернету. По-перше, це означає, що не потрібно чекати, поки файли будуть завантажені, по-друге – це значно полегшує створення резервних копій при необхідності. Однак, щоб фактично запустити сервер, для нього були створені віртуальний хостинг та домен. Для можливості підключення клієнтів не тільки в локальній мережі до WampServer, були проведені налаштування конфігураційних файлів Apache «httpd.conf» та «httpd-vhosts.conf».

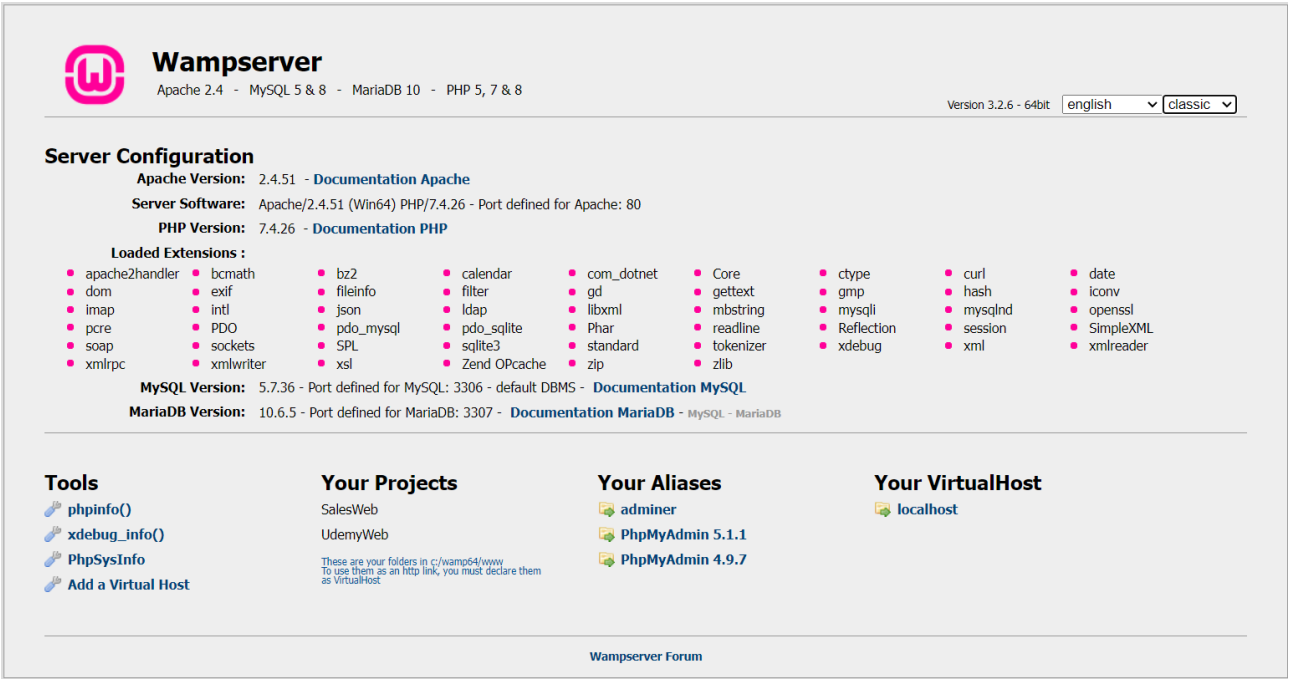


Рисунок 7 – Головна сторінка WampServer

Програма працює з реляційною **СКБД MySQL**. MySQL – це популярний сервер БД, який є в наборі у багатьох різних ПЗ. Частина від назви SQL позначається як мова структурованих запитів - Structured Query Language, яку MySQL використовує для комунікації з іншими програмами. Більш того,

MySQL має власні розширені функції мови SQL для забезпечення користувачів додатковим функціоналом.

Для адміністрування СКБД MySQL використовується графічний веб-інтерфейс **phpMyAdmin** (рис. 8). Він дозволяє через браузерний інтерфейс здійснювати адміністрування сервера MySQL, в тому числі запускати запити SQL, а також переглядати та редагувати вміст таблиць БД.

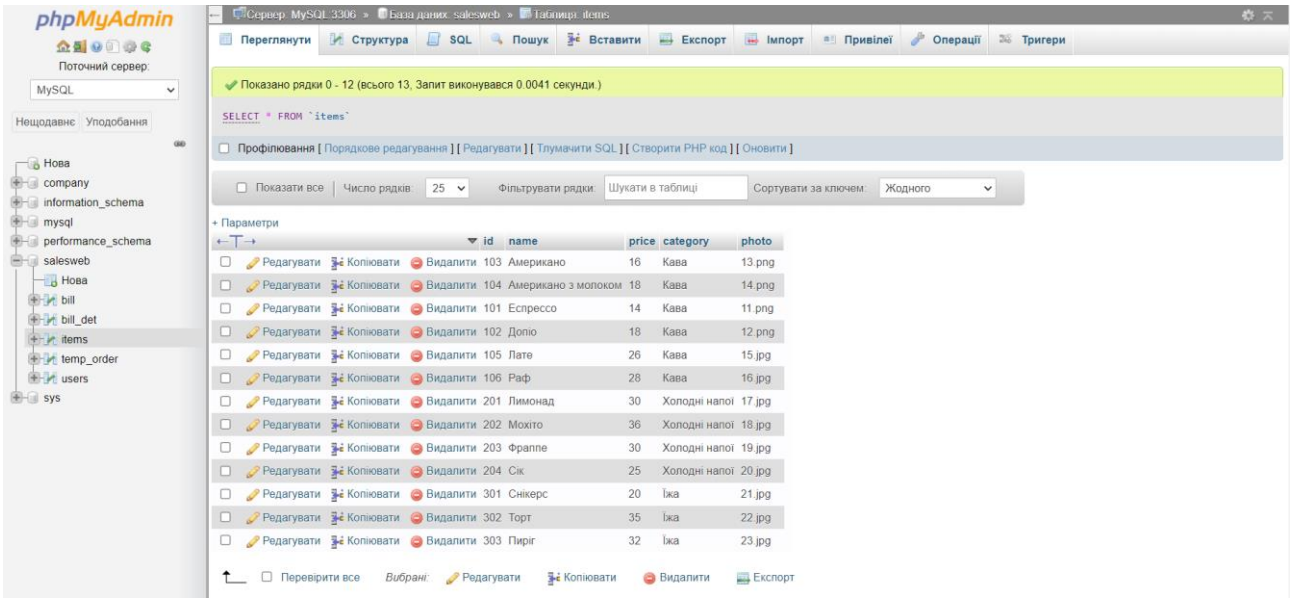


Рисунок 8 – Вигляд веб-додатку «phpMyAdmin» у вікні браузера

Виконання запитів клієнта до сервера виконується через створені в інтерпретаторі **Apache NetBeans** скрипти керування запитами на мові програмування **PHP** (рис.9). NetBeans - це вільне інтегроване середовище розробки (IDE) для різних мов програмування , в тому числі PHP, JavaScript, HTML5. NetBeans IDE розроблена для таких платформ як Windows, Linux, FreeBSD та Solaris. Також при необхідності можна зібрати NetBeans для інших платформ.

За якістю та можливостями NetBeans змагається з найкращими IDE, орієнтованими на веб-серверне програмування, підтримуючи такі методи, як рефакторинг, виділення синтаксичних конструкцій кольором, профілювання, автодоповнення мовних конструкцій, шаблони коду та інше.

Головна сфера застосування PHP - написання скриптів, які працюють на сервері. Тобто, PHP здатний генерувати динамічні сторінки, обробляти дані форм, а також надсилати та приймати запити.

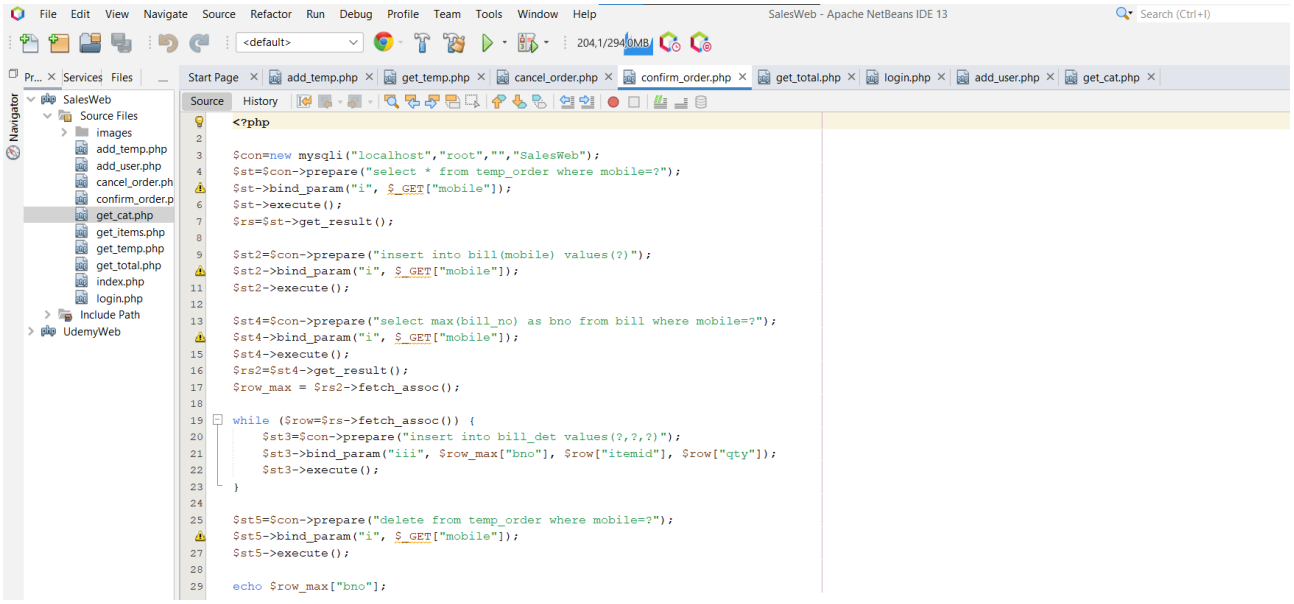


Рисунок 9 – Вигляд IDE «Apache NetBeans» для програмування на мові PHP

В області розробки ПЗ використовується спеціальна мова графічного опису об'єктного моделювання - мова UML (Unified Modeling Language). При описі роботи програми створюється абстрактна Модель системи або підсистеми, яка називається UML-моделлю. На стадії опису роботи програми для наочного представлення роботи окремих функцій програми наводиться діаграма компонентів.

Так як основними елементами програми для ОС Android є активності, то схема роботи реалізованої програми представляє собою схему зв'язків між активностями. На рисунку 10 показана діаграма активностей UML, що демонструє алгоритм роботи всієї розроблюваної програми.

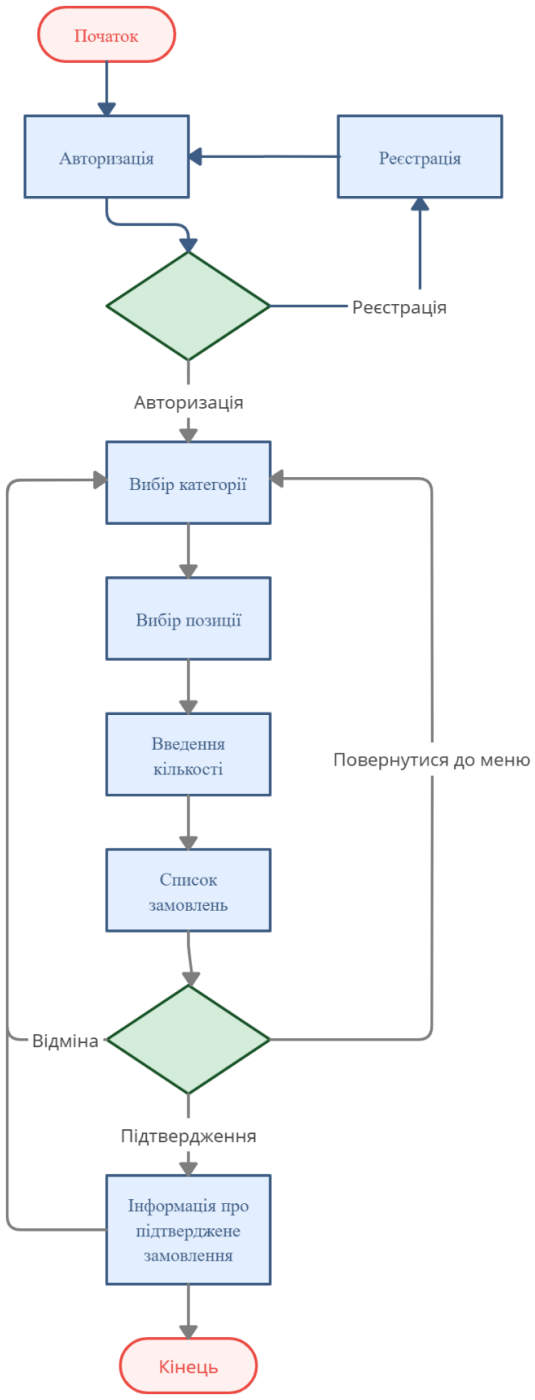


Рисунок 10 – Покрокова схема роботи всієї програми

2.2 Життєвий цикл Android

Activity lifecycle використовується для контролю **ЖЦ** Android-додатку (рис. 11). Ці цикли жорстко контролюються системою, залежать від користувача, ресурсів тощо. Наприклад, ми запускаємо додаток бонусної карти кав'ярні. Система приймає рішення про запуск програми. Хоч кінцевий результат залежить від системи, вона все одно керується певними заданими правилами. Це дає можливість визначити, чи можна завантажити, призупинити додаток чи зовсім припинити його роботу. В тому випадку, якщо користувач в даний момент працює з певним вікном, система надає пріоритет відповідній активності. Коли вікно невидиме і система вирішує зупинити роботу програми, для залучення додаткових ресурсів, тоді буде припинено роботу тієї частини програми, що має більш низький пріоритет. В системі Android кількість ресурсів більш обмежена, тому система виконує більш жорсткий контроль роботи додатків.

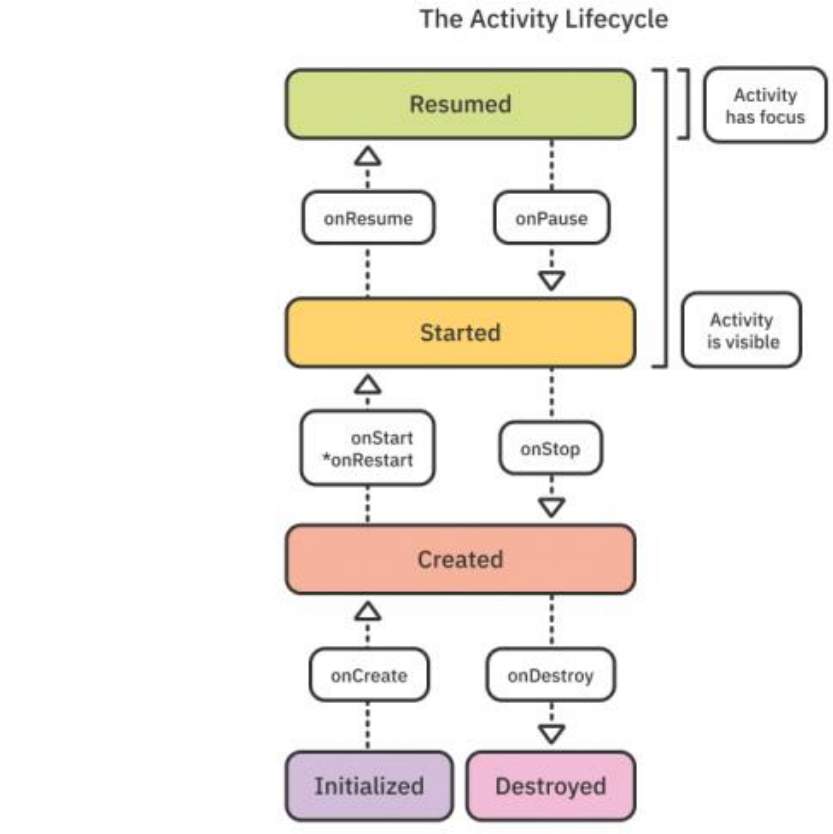


Рисунок 11 – Життєвий цикл додатка на Android

Основні методи ЖЦ Android-додатків:

- onCreate() – метод, що викликається при створенні або перезавантаженні додатку;
- onStart() – викликається безпосередньо перед тим, як додаток стає видимим для користувача і в той же час супроводжується викликом методу onResume()
- onRestart() – викликається в випадку, якщо вікно повертається в пріоритетний режим одразу після виклику onStop(). Тобто викликається після того, як активність була зупинена та знову була запущена користувачем.
- onResume() – викликається після методу onStart(), навіть в той час коли вікно активності працює в пріоритетному режимі і спостерігається користувачем. У цей момент користувач може взаємодіяти зі створеним вікном.
- onPause() – коли користувач переходить до взаємодії з новим додатком, система викликає цей метод для переривання вікна. Можна сказати, що активність згортається та зберігає незафіксовані дані. Разом з тим звільняються ресурси та зупиняється відтворення відео, аудіо та анімацій.
- onStop() – викликається тоді, коли вікно стає не в полі зору користувача. Це відбувається при знищенні додатку, або якщо був запущений інший, який перекрив вікно поточного додатку.
- onDestroy() – метод, який викликається в випадку закінчення роботи додатку, після виклику методу finish() або в тому випадку, коли система сама знищує даний екземпляр активності для того, щоб звільнити ресурси.

2.3 Обґрунтування вибору технологій розробки

2.3.1 Мова програмування Kotlin

Оскільки платформа під яку ведеться розробка Android, а IDE Android Studio, то мова програмування обрана відповідна. Надалі вибір був між Java та Kotlin.

І Java і Kotlin – статично типізовані мови програмування. Вони звісно підтримують ОПП, але також і процедурне програмування. Основний код програм на Java та Kotlin пишеться в функції main, якій передається масив аргументів командного рядка. Важлива особливість полягає в тому, що одна і та ж програма може бути частково написана на Kotlin, частково на Java. Програми написані на Kotlin можуть використовувати всі наявні Java-бібліотеки, і навпаки.

Загалом Kotlin і Java дуже схожі, але Kotlin надає ряд важливих переваг:

Гнучкий та простий синтаксис

Прості функції і структури оголошуються одним рядком. Додавання data-анотації до класу робить активацію автоматичної генерації різних шаблонів.

Класи даних

Kotlin використовує спеціальні класи, які спеціально призначені для зберігання даних. Вони генерують різні шаблони, такі як: equals(), hashCode(), toString() і т.д. Для порівняння нижче вказано код написаний на мові Java:

```
class Flower {
private String title;
private Country country;
public String getTitle() {
```

```
return title;
}
public void setTitle(String title) {
this.title = title;
}
public Country getCountry() {
return country;
}
public void setCountry (Country country) {
this.country = country;
}
}
```

Код виконуючий таку ж функцію написаний на мові Kotlin:

```
data class Flower(var title:String,var country:Country)
```

Типізація

У мові Kotlin немає необхідності явно вказувати тип змінної.

```
fun main (args: Array ) { //Неявне визначення
val text = 10
println (text)
}

fun main (args: Array ) { // Явне визначення
val text: Int = 10
println (text)
```

}

Розумні приведення типів

Компілятор для Kotlin вважається дуже розумним, коли мова заходить про приведення типів. У більшості випадків просто не потрібно явно вказувати оператори приведення, оскільки мова має оператор is, який сам виконує всю роботу:

```
fun demo (x: Any) {
if (x is String) { i.print (x.length) // x автоматично приводиться до типу String
}
}
```

Функціональне програмування

Мова Kotlin створена під функціональне програмування. Вона надає велику кількість корисних можливостей, такі як: функції вищого порядку, лямбда-вирази, перевантаження операторів і ледачі обчислення логічних виразів. До прикладу, робота зі списками:

```
fun main (args: Array ) {
val nums = arrayListOf (15, -5, 11, -39)
val nonNegNum = nums.filter {it >= 0} // обирає числа, які вище нуля
println (nonNegNums) // Результат: 15, 11
}
```

Функції вищого порядку - це функції, які в якості аргументів приймають інші функції та повертають функції. Приклад:

```
fun alpha (func: () -> Unit) {}
```

У ньому func - це ім'я аргументу, а () -> Unit - це тип функції. Зрозуміло, що func буде функцією, яка не приймає аргументів та нічого не повертає.

Лямбда-вирази - функції, які не оголошуються, а передаються у вигляді виразів. Наприклад:

val sum: (Int, Int) -> Int = {x, y -> x + y}

Оголошується змінна sum, яка бере в себе два числа, вираховує їх суму і приймає результат, приведений до цілого числа. Для виклику досить просто написати sum(2, 2).

Порівняння швидкості Java і Kotlin

Перша збірка Kotlin-коду може займати приблизно на 15-20% часу більше, ніж це може відбутися на Java. Але тим часом інкрементна збірка Kotlin відбувається з вищою швидкістю, ніж у Java. Судячи з цього можна сказати, що мови приблизно однакові за швидкістю компіляції.

Мова програмування Kotlin - це наступний етап розвитку Java, з якої він повністю сумісний. Але ряд вагомих переваг над більш застарілою мовою Java робить його відмінним та сучасним інструментом для розробки мобільних додатків.

2.3.2 СКБД MySQL

Вибір СКБД являє собою складне завдання з великою кількістю параметрів і це один із важливих етапів розробки БД. Обраний програмний продукт має задовольняти як поточні, так і майбутні потреби програми. Для даного програмного додатку в якості СКБД була обрана MySQL.

MySQL являється рішенням для малих і середніх програмних продуктів. Зазвичай MySQL використовують як веб-сервер, до якого можуть звертатися локальні або віддалені клієнти, проте в збірку дистрибутиву MySQL входить бібліотека внутрішнього сервера, що в свою чергу дозволяє додавати MySQL

інтегровано в автономні програми, в такі як в даному випадку – веб-сервер WAMP.

СКБД MySQL є гнучкою системою, яка забезпечується підтримкою великої кількості типів таблиць: MyISAM – підтримують повнотекстовий пошук, InnoDB – підтримують транзакції рівня окремих записів. Через те що MySQL має відкриту архітектуру і GPL-ліцензуванню, в ньому все ще з'являються нові типи таблиць.

До основних переваг СКБД MySQL можна віднести:

- підтримка деякої кількості одночасних запитів, багатопоточність;
- записи фіксованої та змінної довжини;
- оптимізація зв'язків з можливістю приєднання багатьох даних за час виконання одного проходу;
- підтримка спеціальних та ключових полів;
- всі дані зберігаються у форматі ISO8859_1;
- гнучка і зрозуміла система привілеїв та паролів;
- швидка система пам'яті заснована на потоках;
- підтримка рядків змінної довжини, міток часу та чисел довжиною від 1 до 4 байт форматів int, double, float, fixed;
- просте управління таблицею, яке включає додавання та видалення ключів і полів.
- будь-які операції роботи з рядками не залежать від регістру символів у оброблюваних рядках;

2.4 Структура БД

Використовувана БД спроектована на основі реляційної моделі. Заздалегідь було проведено аналіз сутностей в ході реалізації функцій додатку для кав'ярні. Проведена нормалізація таблиць даних та визначені первинні ключі. На (рис. 12) відображена схема БД бонусної карти кав'ярні для платформи Android з описом таблиць. Потоки даних у аналізованій діаграмі поширюються від БД до інших елементів за допомогою SQL запитів створених у PHP скриптах.

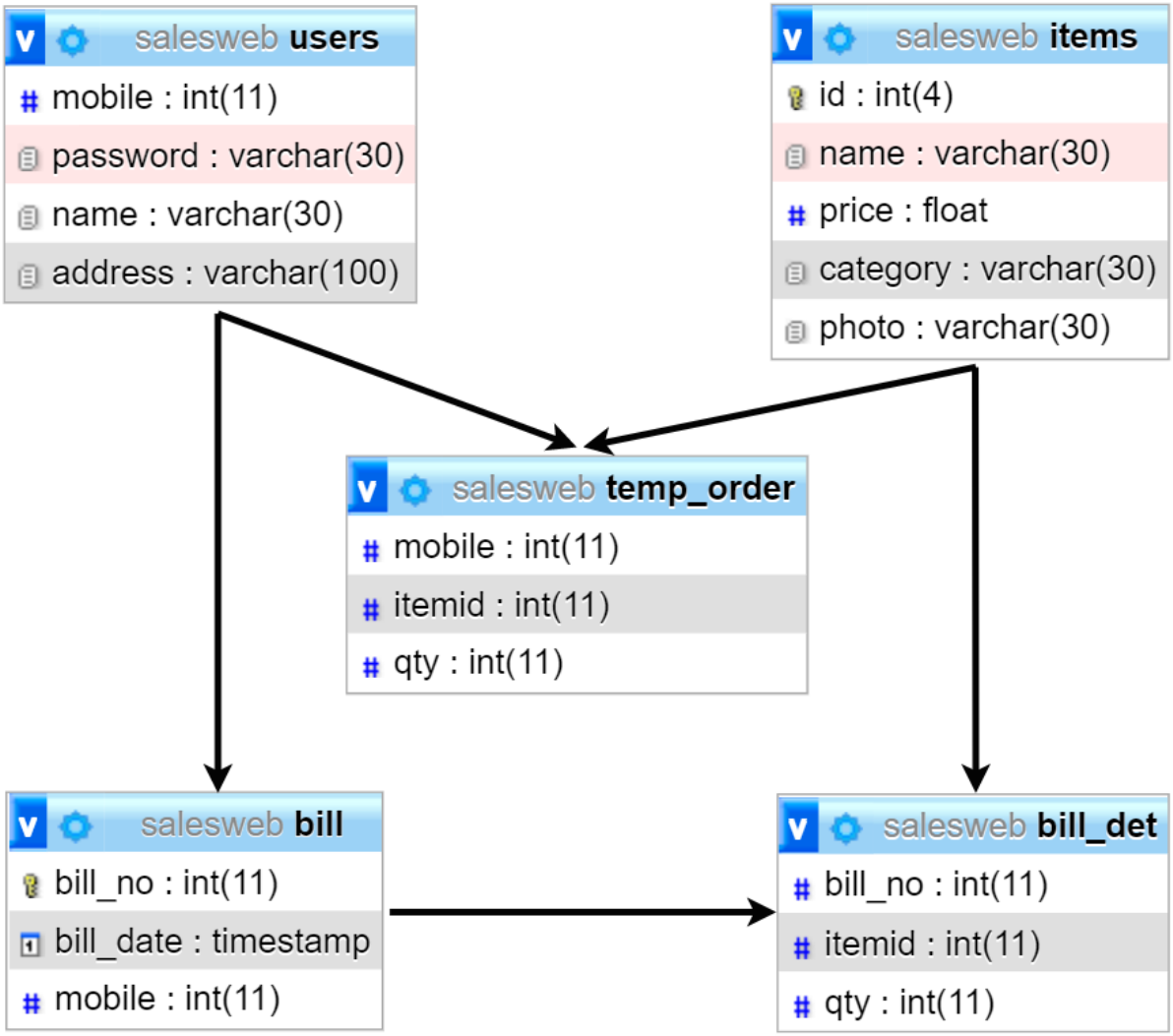


Рисунок 12 – Структура БД

Таблиця 2 – users

Назва	Тип	Опис
mobile	int	Мобільний номер користувача
password	string	Пароль для авторизації та реєстрації
name	string	Ім'я користувача
address	string	Пошта для реєстрації

Таблиця 3 – items

Назва	Тип	Опис
Id	int	Ідентифікатор
name	string	Назва позиції
price	float	Ціна позиції
category	string	Категорія позиції
photo	string	Посилання на фото позиції

Таблиця 4 – bill_det

Назва	Тип	Опис
bill_no	int	Номер замовлення
itemid	int	Ідентифікатор позиції
qty	int	Кількість одиниць позиції

Таблиця 5 – bill

Назва	Тип	Опис
bill_no	int	Номер замовлення (Ідентифікатор)
bill_date	string	Дата та час замовлення

mobile	string	Мобільний номер користувача

Таблиця 6 – temp_order

Назва	Тип	Опис
mobile	int	Мобільний номер користувача
itemid	int	Ідентифікатор позиції
qty	int	Кількість одиниць позиції

2.5 Перелік модулів

Функціонально, додаток складається з наведених нижче модулів (Активностей). Android-додаток здатен складатися з декількох активностей і може перемикатися між ними під час виконання програми.

- **MainActivity** – головна активність містить два об’єкта EditText та дві кнопки. Вона призначена для авторизації користувача та переходу до активностей регістрації (RegAct) та меню вибору категорії (HomeAct). Під час авторизації перевіряє правильність введення даних користувачем за допомогою скрипта «login.php», який перевіряє наявність користувача в таблиці «users».

- **RegAct** – активність регістрації користувача та занесення інформації про нього до БД. Перевіряє коректність введених даних та викликає скрипт «add_user.php» для занесення інформації про користувача в таблицю «users».

- **HomeAct** – активність вибору категорії містить список елементів залежно від кількості категорій, які дозволяють уточнити групу позицій що цікавить користувача. Після вибору певної категорії відбувається перехід у наступну активність. У параметрах передається уточнена інформація по обраній групі. Активність використовує шаблон форматування ListView. Робота

з БД відбувається за допомогою скрипту get_cat.php беручи вміст таблиці «items».

- **ItemAct** – активність вибору об'єкта інтересу (позиції) містить список об'єктів інтересу, які задовольняють обраній раніше категорії. Для цього активність, отримавши в параметрах уточнену інформацію про об'єкти інтересу, формує запит до БД та на підставі отриманої інформації створює динамічний перелік об'єктів за допомогою адаптера "ItemAdapter". Робота з БД відбувається за допомогою скрипту «get_items.php» беручі вміст таблиці «items».

- **QtyFragment** – фрагментна активність викликається після натискання на кнопку певної позиції та створюється поверх минулої активності. Має тільки два об’єкти в структурі: EditText та Button. Потрібна для того щоб отримати від користувача кількісне значення обраної раніше позиції та передати отримані параметри в БД. Запит до БД надається через скрипт «add_temp.php» для створення в таблиці «temp_order» інформації про тимчасове замовлення з прив’язкою до номера користувача.

- **OrderAct** – активність яка отримує дані про тимчасове замовлення з БД через скрипт «get_temp.php» та заносить їх у список. Також вона має панель меню з певними параметрами для подальшої дії. При натисканні кнопки «Відмінити» викликається скрипт «cancel_order.php», який в свою чергу отримує номер користувача та видаляє відповідне тимчасове замовлення з БД. При обранні кнопки «Повернутися до меню», відбувається перехід до активності «HomeAct», тоді користувач може знову обрати категорію та додати нову позицію до замовлення. Коли натискається кнопка «Підтвердити» викликається скрипт «confirm_order.php», після чого в таблиці «bill» та «bill_no» вноситься інформація про підтверджене замовлення та разом з тим видаляється тимчасове. Одразу з цим активність передає параметри замовлення в наступну.

- **TotalAct** – кінцева активність, що рахує загальну суму замовлення та виводить її разом з додатковою інформацією на екран. Нижче всього є кнопка яка повертає користувача назад до вибору категорії (HomeAct).

Кожен PHP-скрипт виконує відповідні SQL-запити та повертає певні дані з БД до активності, яка його викликала.

Вихід із кожної активності здійснюється за допомогою кнопки «Назад».

Схема активностей розробленого додатка представлена нижче на рисунку 13 у вигляді діаграми класів UML.

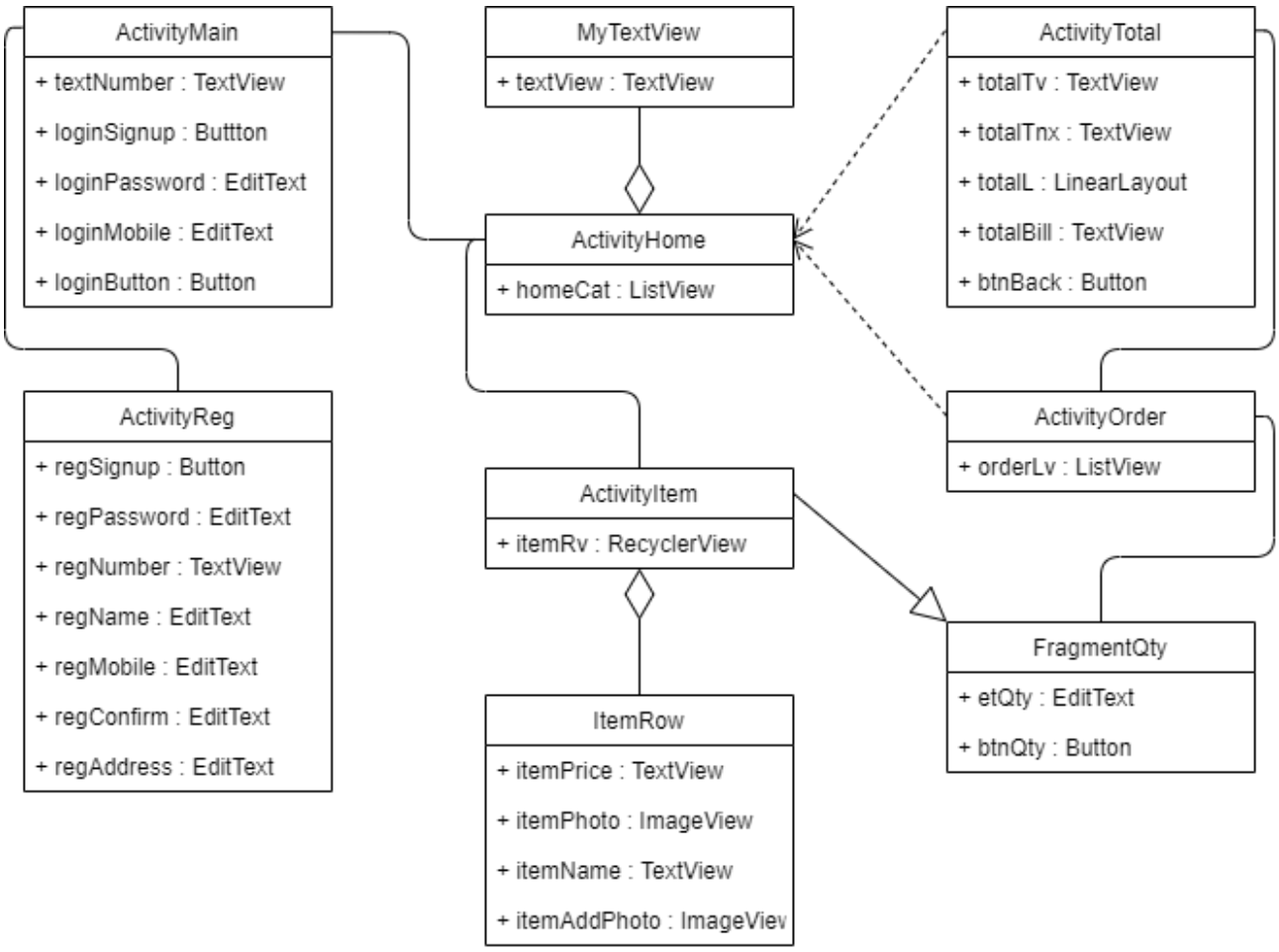


Рисунок 13 – Активності програми і зв'язки між ними

2.6 Висновки до розділу

1) В межах даного розділу була розглянута архітектура програмного додатку та визначено що програмний додаток складається з набору активностей.

2) Розглянута структура БД з описом таблиць представлених в даній БД та її взаємодія з програмним додатком.

3) Наведені аргументи на користь обраних технологій розробки.

4) Наведено перелік застосованих методів розробки, та те для чого вони застосовувались. Розглянуто програмну реалізацію всього проекту загалом.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ДОДАТКУ НА ПЛАТФОРМІ ANDROID

3.1. Системні вимоги

Для опису характеристик, яким має відповідати смартфон, на який буде встановлено додаток, було визначено такі характеристики:

- ОС Android версії 5.1 KitKat (API 22) і вище;
- ЦП з тактовою частотою 2.0 ГГц або більше;
- ОЗП – 1024 мегабайт або більше;
- вільне місце у пам’яті внутрішнього накопичувача 30 мегабайт для встановлення додатку.

Характеристики, яким має відповідати комп’ютерна система, на який буде встановлено WampServer 3.2.6:

- WampServer 3.2.6 не можна встановлювати поверх попередніх версій програми;
- Версія Windows має бути вищою, ніж Windows XP і Windows Server 2003;
- У Windows повинні бути встановлені пакети Microsoft Visual C++ Redistributable Package 2012 і 2015, які необхідні для роботи компонентів, що входять до складу збірки WampServer 3.2.6;
- Рекомендується встановлювати WampServer 3.2.6 в папку "wamp" у корені диска "C: wamp".

3.2 Опис функціональності

При запуску програми відображається екран авторизації користувача, на якому користувач повинен ввести номер телефону та пароль для переходу до Меню (рис. 14). В тому випадку, коли у користувача немає акаунту він переходить до реєстрації через кнопку «Зареєструватися» (рис. 15). При неправильному введені даних (на екранах з авторизацією та реєстрацією) з'являється повідомлення, яке вказує на це (рис. 15).

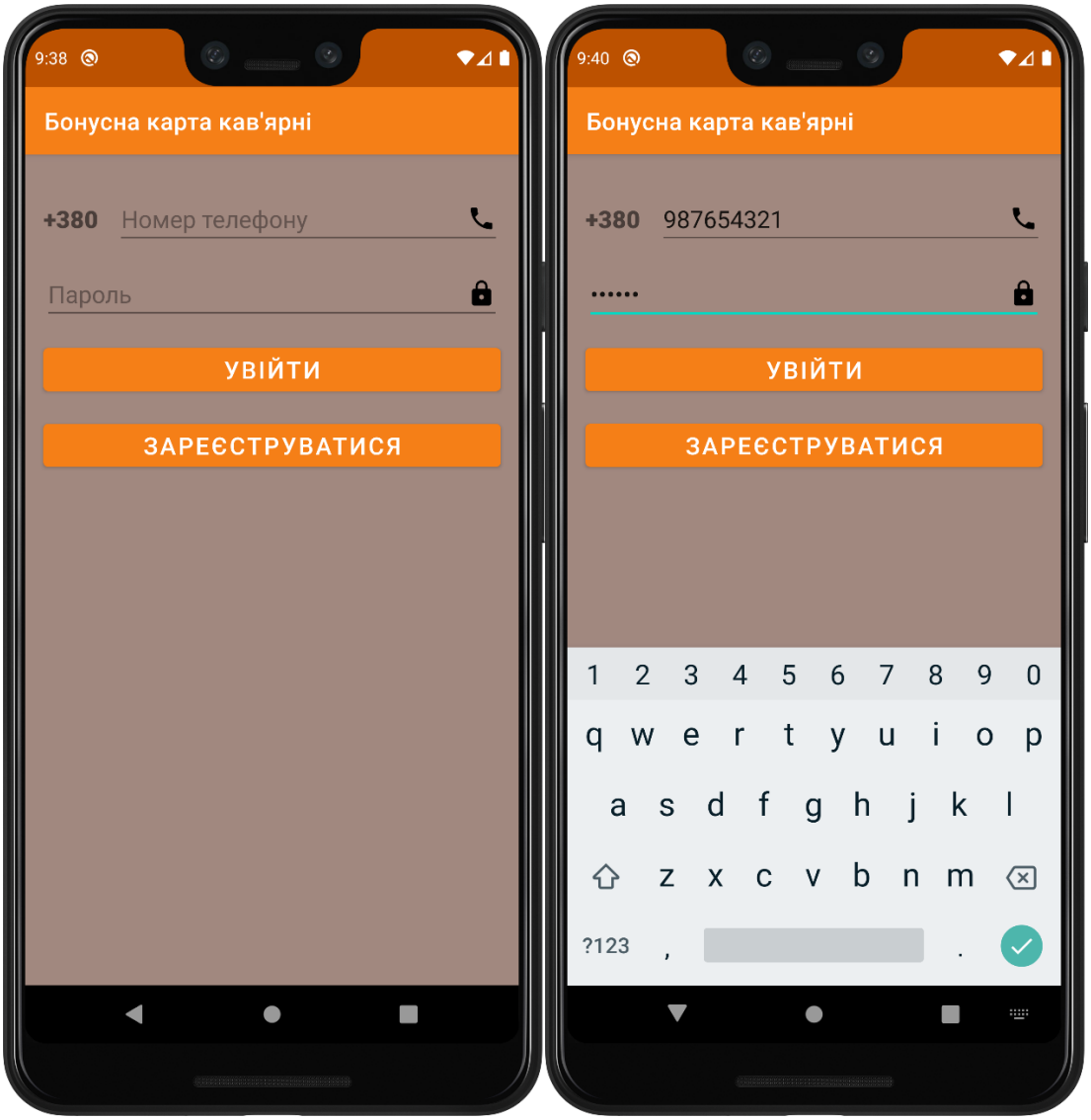


Рисунок 14 – Основна активність

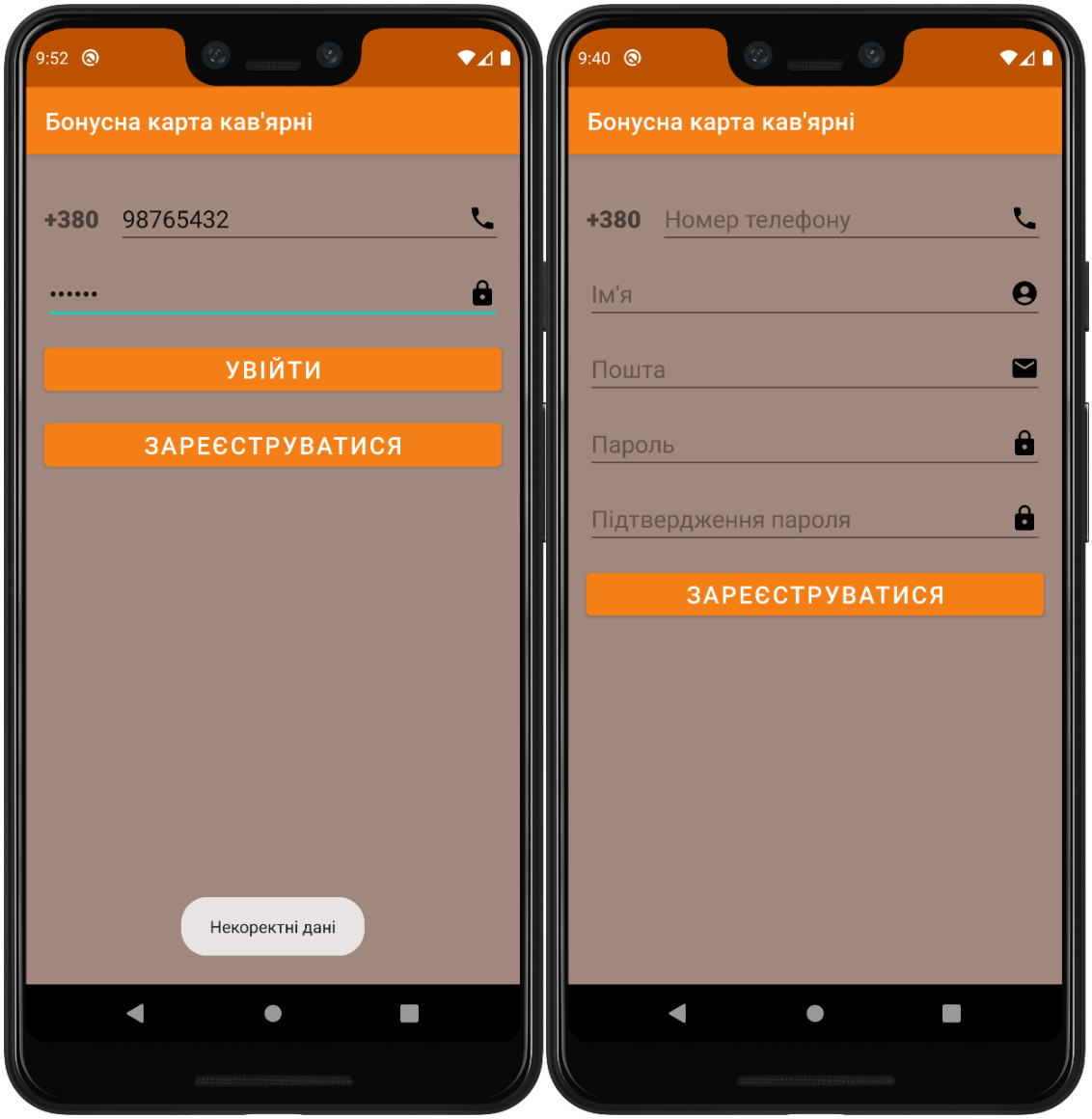


Рисунок 15 – Введення неправильних даних (ліворуч) та екран реєстрації (праворуч)

При натисканні на кнопку «Увійти» користувачеві надається список вибору категорій завантажений з БД. Після натискання певної категорії додаток переходить на іншу активність та показує вже список позицій залежно від обраної категорії із зазначенням назви, вартості та кнопкою «Додати», при цьому теж завантажуючи його з БД. Ці дії у відповідному порядку показані нижче на рисунку 16.

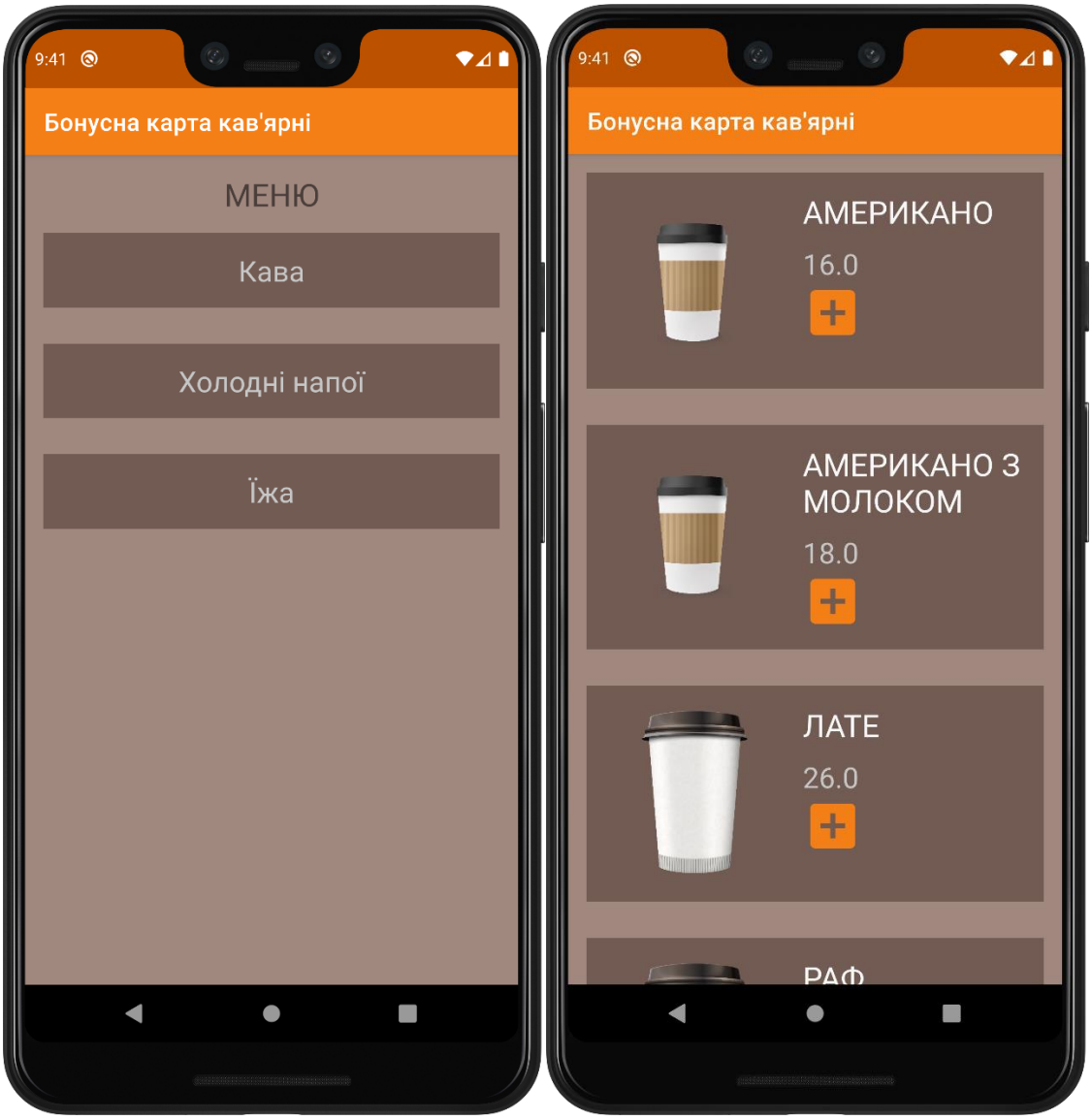


Рисунок 16 – Вибір категорій

Визначившись з позицією, користувачу потрібно натиснути кнопку «Додати». Після чого на екрані з’являється фрагмент, який дає користувачу вказати кількість одиниць обраної позиції (рис. 17). Після чого користувач отримує список вмісту його замовлення (рис. 17).

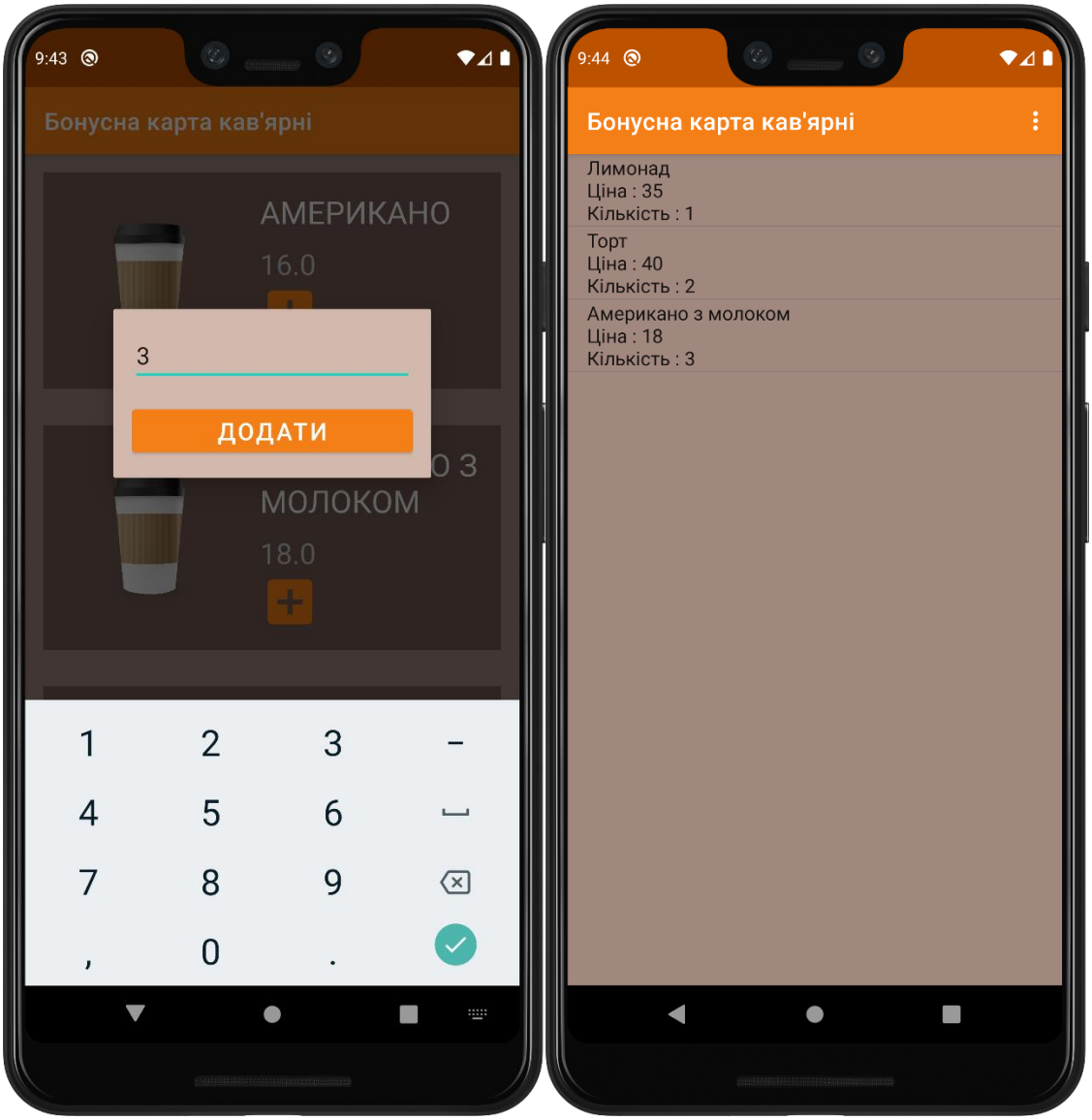


Рисунок 17 – Введення кількості одиниць обраної позиції (ліворуч), список вмісту замовлення (праворуч)

Після в панелі меню користувач має обрати наступну дію з можливих (рис. 18): «Відмінити» - замовлення видаляється з БД та запускається екран з вибором категорій; «Підтвердити» - в БД створюється кінцеве замовлення з усіх позицій які обрав користувач та підраховується загальна сума. Після чого відбувається перехід на наступний екран де користувач бачить загальну суму та номер свого замовлення. Також є кнопка, при натисканні якої користувач

повертається до Меню (рис. 18); «Повернутися до меню» - повернення до меню.

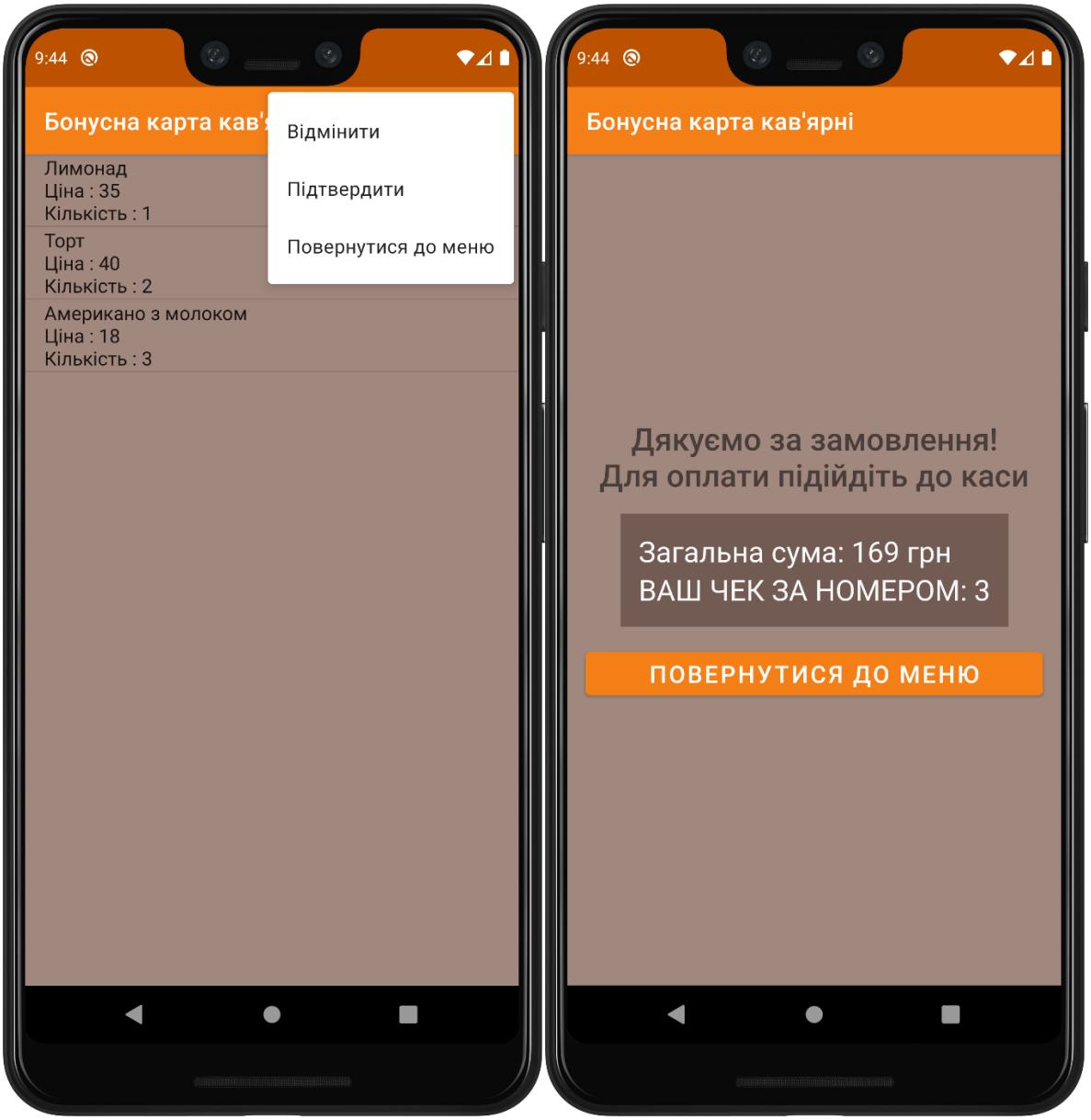


Рисунок 18 – Панель меню (ліворуч), кінцевий екран (праворуч)

3.3 Тестування

Кінцевим етапом при розробці будь-якого ПЗ є тестування. **Тестування** – один з найважливіших етапів у ЖЦ розробки ПЗ, який може гарантувати належну якість реалізованого програмного продукту.

Для цього був використаний емулятор смартфона Google Pixel 3 XL та особистий смартфон Meizu M3s, які в свою чергу мають різні версії Android: 10 (API 29) та 5.1 (API 22) відповідно. Протестоване ПЗ послідовно запускалося на цих емуляторах, при цьому аналізувалася поведінка та залежно від результатів аналізу були внесені зміни в програмний код.

Тестування буває ручне та автоматизоване.

Під час виконання ручного тестування сам процес представляється у вигляді тест- кейсів. Тест-кейс це процес, який виконує певний набір дій або умов, які необхідні для перевірки коректності функціональної роботи програмної системи.

Набір тест-кейсів – це поширена практика контролю якості ПЗ. У реалізації тест-кейсу це виглядає наступним чином: в тому випадку, коли отриманий результат відповідає очікуваному, тест вважається пройденим і до колонки з даними про результат записується стан тесту, як «пройдено», в іншій ситуації ставиться запис «не пройдено». Тестова ситуація зазвичай складається з таких частин:

1) передумова – послідовність кроків, які приводять програмну систему до стану, при якому можна починати тестування.

2) опис тестового випадку – список дій при яких можливо проаналізувати необхідний розробнику функціонал програми.

3) постумова – дії, при яких система повертається до початкового стану перед початком тестування.

Тестування даного ПЗ проведено набором тест-кейсів, які перевіряють коректність роботи основних функцій програмної системи в залежності від різних умов. В таблиці 3.1 розглянуто тестовий випадок проходження реєстрації та авторизації користувача у різних випадках введення даних через використання інтерфейсу програми.

Таблиця 7 – Тестовий випадок перевірки реєстрації та авторизації користувача шляхом введення різних даних у відповідні поля на головному екрані та екрані реєстрації.

Передумова		
Відкрити головний екран програми та екран реєстрації. Перевірити появу екранної клавіатури при натисненні на поля введення. Перевірити коректну роботу інших функціональних кнопок додатку		
Подія	Очікуваний результат	Фактичний результат
Тестовий випадок		
Ввести існуючі дані у поля для авторизації в додатку на головному екрані користувацького інтерфейсу програми та натиснути кнопку «Увійти».	Перехід на екран вибору категорії без додаткових повідомлень на екрані.	Пройдено.
У полі для введення номеру ввести неіснуючий мобільний номер користувача та правильно введений пароль на головному екрані користувацького інтерфейсу програми, натиснути кнопку «Увійти».	З’являється спливаюче повідомлення з текстом «Некоректні дані».	Пройдено.
У полі для введення паролю ввести неіснуючий пароль користувача та правильно введений мобільний номер на головному екрані користувацького	З’являється спливаюче повідомлення з текстом «Некоректні дані».	Пройдено.

інтерфейсу програми, натиснути кнопку «Увійти».		

Продовження таблиці 7

Ввести коректні дані у поля для реєстрації в додатку на екрані реєстрації та натиснути кнопку «Зареєструватися».	Перехід на головний екран користувацького інтерфейсу програми, з’являється спливаюче повідомлення з текстом «Аккаунт створений».	Пройдено.
У полі для введення номеру ввести існуючий у БД мобільний номер користувача та коректні дані в інших полях на екрані реєстрації, натиснути кнопку «Зареєструватися».	З’являється спливаюче повідомлення з текстом «Такий номер вже існує».	Пройдено.
У полях для введення паролів ввести різні значення та коректні дані в інших полях на екрані реєстрації, натиснути кнопку «Зареєструватися».	З’являється спливаюче повідомлення з текстом «Пароль не співпадає».	Пройдено.
Постумова		
При успішній реєстрації переходимо до головного екрану програми.	Відкриття головного екрану мобільного додатку.	Пройдено.
При успішній авторизації переходимо до екрану вибору категорії.	Відкриття екрану вибору категорії.	Пройдено.

3.4 Можливі варіанти розвитку програмного додатку

Можливими шляхами розвитку програми є:

- інтеграція програми зі сторонніми БД;
- реалізація двокомпонентної БД (локальної і серверної);
- реалізація функції обирання користувачем розміру позиції за допомогою «RadioButton»;
- реалізація функції онлайн оплати замовлення;
- реалізація інтерфейсу на різних мовах.

3.5 Висновки до розділу

1) У даному розділі було проаналізовано результати роботи додатку, продемонстровано вигляд його інтерфейсу та показана взаємодія з користувачем через цей інтерфейс.

2) Було описано вимоги та процес тестування ПЗ, для переконання в коректності роботи програмного додатку бонусної карти кав’ярні.

3) Було проведено ручне тестування за планом тест-кейсу функціональної роботи мобільного додатку на аналіз коректної взаємодії з БД, чим було продемонстровано коректну роботу програмного продукту.

4) Розглянуто можливі варіанти розвитку програмного продукту в майбутньому та способи їх реалізації.

ВИСНОВОК

Піднімаючи питання про темпи збільшення користувачів смартфонів в Україні, не можна не сказати про актуальність розробки програмного додатку, відповідного до тематики сфери послуг і реалізованого у відповідності з новітніми інформаційними технологіями. Частка мобільних пристроїв з ОС Android в Україні становить 84%. Тому орієнтуючись на переважаючу більшість користувачів доцільно вести розробку застосунку саме на платформі Android.

В рамках дипломної проекту були досягнуто перераховані нижче результати.

1. Було проведено аналіз принципів розробки на мобільних платформах, в результаті чого було виявлено, що 84% смартфонів в Україні створені на базі ОС Android. Враховуючи це, дана платформа була обрана для розробки мобільного додатку бонусної карти кав'ярні.

2. Було створено програмний продукт, який реалізує наступні функції:

- використання клієнтського додатку зі зручним інтерфейсом, який задовольняє потреби користувача при замовленні в закладі кав'ярні;
- зручне керування зовнішньою БД та отримання з неї потрібної інформації за запитом системи;
- використання веб-сервера для отримання даних в будь-який момент при умові наявності інтернету;
- реєстрація користувача у системі та прив'язка аккаунту до дій всередині системи.

3. Представлений результат реалізації програмного продукту та тестування його за різних умов використання, чим було підтверджено його

працездатність як на стандартних емуляторах, взятих із SDK Android, так і на реальному пристрої на базі ОС Android.

СПИСОК ПОСИЛАНЬ ВИКОРИСТАНИХ ІНТЕРНЕТ ДЖЕРЕЛ

1. https://www.statista.com/statistics/218984/number-of-global-mobile-users-since-2010/
2. https://htmler.ru/2020/12/29/aktualnost-razrabotki-mobilnyh-prilozhenij-pod-android/
3. https://www.hse.ru/edu/vkr/153008195
4. https://keepwarning.com/293-Prilozhenie-dlq-Android-ili-iOS-kakoe-vi-bi-razrabotali-Otchet-o-sravnitelxnom-analize
5. https://uk.wikipedia.org/wiki/%D0%A1%D0%BC%D0%B0%D1%80%D1%82%D1%84%D0%BE%D0%BD
6. https://hddrecover.ru/total/vse-o-smartfone-chto-eto-takoe-i-kak-on-rabotaet-plyusy-i-minusy-telefona/
7. https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/
8. https://wezom.com.ua/blog/prilozheniya-dlya-android
9. https://wezom.com.ua/blog/prilozheniya-dlya-ios
10. https://www.azoft.ru/blog/ios-vs-android/
11. https://developer.android.com/guide/components/activities/intro-activities
12. https://www.makeuseof.com/tag/how-to-set-up-your-own-wampserver/

13. https://wiki.gentoo.org/wiki/MySQL/Startup_Guide/ru

14. https://www.raywenderlich.com/21382977-android-lifecycle

15. https://studopedia.ru/21_90335_obruntuvannya-viboru-tehnologii-rozrobki-programnogo-seredovishcha-ta-movi-programuvannya.html

16. https://dspace.nau.edu.ua/bitstream/NAU/53008/1

17. https://jak.bono.odessa.ua/articles/zhittevij-cikl-dodatki-na-android.php

18. https://ela.kpi.ua/bitstream/123456789/28744/1/

19. http://ipkey.com.ua/uk/faq/912-android.html

20. http://www.programmer.dp.ua/download/tlumachniy-slovnik-z-informatiki.pdf

21. http://eprints.library.odeku.edu.ua/3772/1/Romashov_rozrobka_webservisa_M_2018.pdf.pdf

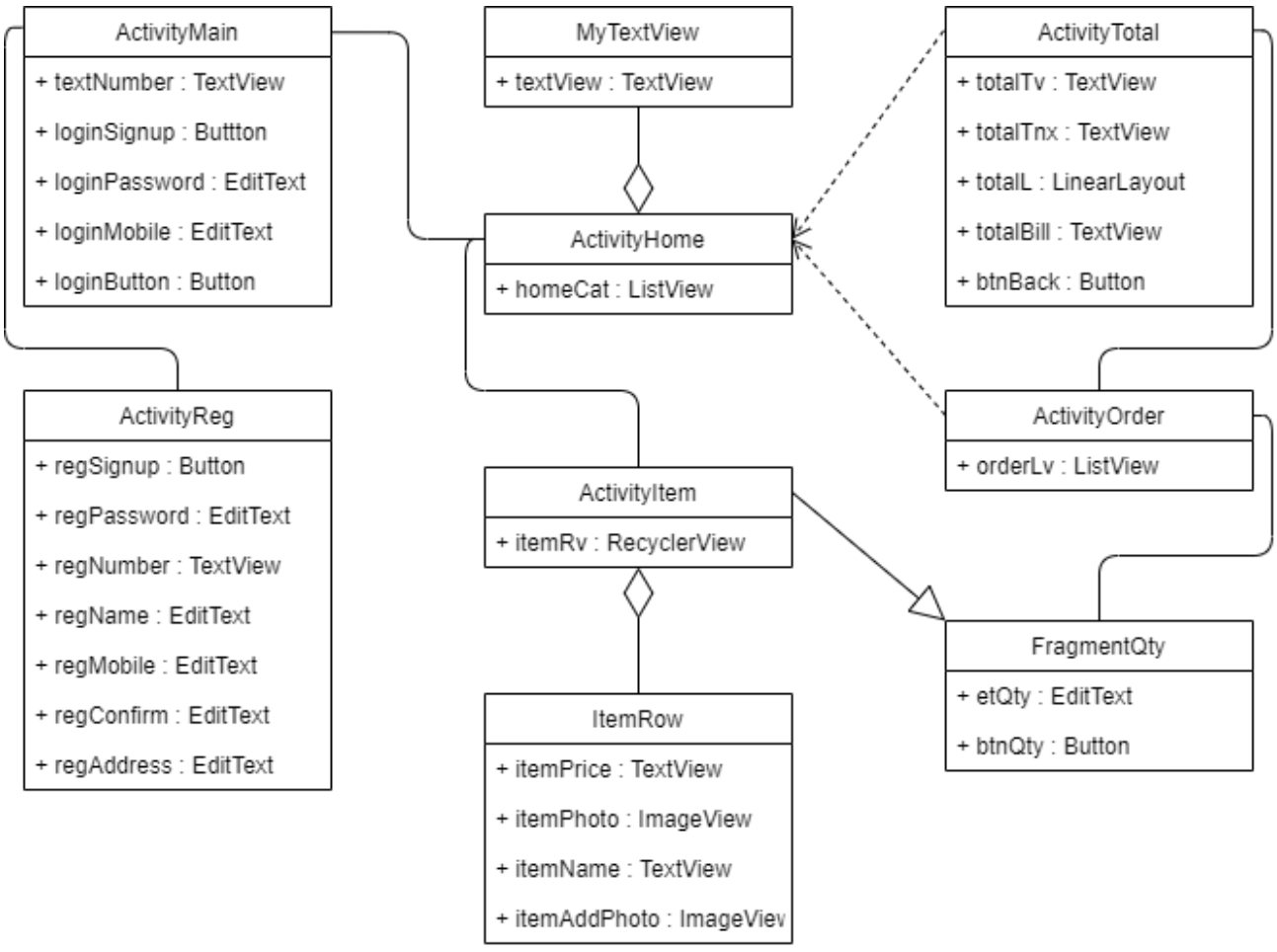
22. http://inmad.vntu.edu.ua/portal/static/86B4CE19-2274-4422-862D-DD2CAF2DFF4C.pdf

23. https://conf.ztu.edu.ua/wp-content/uploads/2018/05/5-1.pdf

24. https://www.xenonstack.com/blog/kotlin-andriod

ДОДАТОК А

Діаграма класів



ДОДАТОК Б

Код програмного додатку

MainActivity.kt

```
package com.example.salesapp

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Toast
import com.android.volley.Request
import com.android.volley.RequestQueue
import com.android.volley.toolbox.StringRequest
import com.android.volley.toolbox.Volley
import com.example.salesapp.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.loginSignup.setOnClickListener {
            val i = Intent(this, RegAct::class.java)
            startActivity(i)
        }

        binding.loginButton.setOnClickListener {
            val url =
                "http://192.168.1.119/SalesWeb/login.php?mobile=" +
binding.loginMobile.text.toString() + "&password=" +
                    binding.loginPassword.text.toString()

            val rq: RequestQueue = Volley.newRequestQueue(this)
            val sr = StringRequest(Request.Method.GET, url, { response ->
                if (response.equals("0"))
                    Toast.makeText(this, "Некоректні дані", Toast.LENGTH_LONG).show()
                else {
                    UserInfo.mobile = binding.loginMobile.text.toString()
```

```
                val i = Intent(this, HomeAct::class.java)
                startActivity(i)
            }

        }, { error ->
            Toast.makeText(this, error.message, Toast.LENGTH_LONG).show()
        })

        rq.add(sr)
    }
  }
}
```

RegAct.kt

```
package com.example.salesapp

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Toast
import com.android.volley.Request
import com.android.volley.RequestQueue
import com.android.volley.toolbox.StringRequest
import com.android.volley.toolbox.Volley
import com.example.salesapp.databinding.ActivityRegBinding

class RegAct : AppCompatActivity() {

    private lateinit var binding: ActivityRegBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityRegBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.regSignup.setOnClickListener {
            if (binding.regPassword.text.toString() == binding.regConfirm.text.toString()) {
                val url =
                    "http://192.168.1.119/SalesWeb/add_user.php?mobile=" +
binding.regMobile.text.toString() + "&password=" +
                            binding.regPassword.text.toString() + "&name=" +
binding.regName.text.toString() + "&address=" + binding.regAddress.text.toString()

                val rq: RequestQueue = Volley.newRequestQueue(this)
                val sr = StringRequest(Request.Method.GET, url, { response ->
                    if (response.equals("0"))
                        Toast.makeText(this, "Такий номер вже існує", Toast.LENGTH_LONG).show()
                    else {
```

```
                Toast.makeText(this, "Аккаунт створений", Toast.LENGTH_LONG).show()
                UserInfo.mobile = binding.regMobile.text.toString()
                val i = Intent(this, HomeAct::class.java)
                startActivity(i)
            }

        }, { error ->
            Toast.makeText(this, error.message, Toast.LENGTH_LONG).show()
        })

        rq.add(sr)

    } else {
        Toast.makeText(this, "Пароль не співпадає", Toast.LENGTH_LONG).show()
    }
  }
 }
}
```

HomeAct.kt

```
package com.example.salesapp

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.ArrayAdapter
import android.widget.Toast
import com.android.volley.Request
import com.android.volley.RequestQueue
import com.android.volley.toolbox.JsonArrayRequest
import com.android.volley.toolbox.Volley
import com.example.salesapp.databinding.ActivityHomeBinding

class HomeAct : AppCompatActivity() {

    private lateinit var binding: ActivityHomeBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityHomeBinding.inflate(layoutInflater)
        setContentView(binding.root)

        val url = "http://192.168.1.119/SalesWeb/get_cat.php"
        val list = ArrayList<String>()
        val rq: RequestQueue = Volley.newRequestQueue(this)
        val jar = JsonArrayRequest(Request.Method.GET, url, null, { response ->
            for (x in 0 until response.length())
                list.add(response.getJSONObject(x).getString("category"))
```

```
            val adp = ArrayAdapter(this, R.layout.my_textview, list)
            binding.homeCat.adapter = adp
        }, { error ->
            Toast.makeText(this, error.message, Toast.LENGTH_LONG).show()
        })
        rq.add(jar)

        binding.homeCat.setOnItemClickListener { _, _, i, _ ->
            val cat: String = list[i]
            val obj = Intent(this, ItemAct::class.java)
            obj.putExtra("cat", cat)
            startActivity(obj)
        }
    }
}
```

ItemAct.kt

```
package com.example.salesapp

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Toast
import androidx.recyclerview.widget.LinearLayoutManager
import com.android.volley.Request
import com.android.volley.RequestQueue
import com.android.volley.toolbox.JsonArrayRequest
import com.android.volley.toolbox.Volley
import com.example.salesapp.databinding.ActivityItemBinding

class ItemAct : AppCompatActivity() {

    private lateinit var binding: ActivityItemBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityItemBinding.inflate(layoutInflater)
        setContentView(binding.root)

        val cat: String? = intent.getStringExtra("cat")
        val url = "http://192.168.1.119/SalesWeb/get_items.php?category=$cat"
        val list = ArrayList<Item>()

        val rq: RequestQueue = Volley.newRequestQueue(this)
        val jar = JsonArrayRequest(Request.Method.GET, url, null, { response ->
            for (x in 0 until response.length())
                list.add(
                    Item(
```

```
                response.getJSONObject(x).getInt("id"),
                response.getJSONObject(x).getString("name"),
                response.getJSONObject(x).getDouble("price"),
                response.getJSONObject(x).getString("photo")
            )
        )

        val adp = ItemAdapter(this, list)
        binding.itemRv.layoutManager = LinearLayoutManager(this)
        binding.itemRv.adapter = adp
    }, { error ->
        Toast.makeText(this, error.message, Toast.LENGTH_LONG).show()
    })
    rq.add(jar)
  }
}
```

Item.kt

```
package com.example.salesapp

class Item(var id: Int, var name: String, var price: Double, var photo: String)
```

ItemAdapter.kt

```
package com.example.salesapp

import android.app.Activity
import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.squareup.picasso.Picasso

class ItemAdapter(private val context: Context, private var list: ArrayList<Item>) :
  RecyclerView.Adapter<RecyclerView.ViewHolder>() {
  override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
RecyclerView.ViewHolder {
    val v: View = LayoutInflater.from(context).inflate(R.layout.item_row, parent, false)
    return ItemHolder(v)
  }

  override fun onBindViewHolder(holder: RecyclerView.ViewHolder, position: Int) {
    (holder as ItemHolder).bind(
      list[position].name,
```

```
            list[position].price,
            list[position].photo,
            list[position].id
        )
    }

    override fun getItemCount(): Int {
        return list.size
    }

    class ItemHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {

        private var itemName: TextView = itemView.findViewById(R.id.item_name)
        private var itemPrice: TextView = itemView.findViewById(R.id.item_price)
        private var itemPhoto: ImageView = itemView.findViewById(R.id.item_photo)
        private var itemAddPhoto: ImageView = itemView.findViewById(R.id.item_add_photo)

        fun bind(n: String, p: Double, u: String, item_id: Int) {
            itemName.text = n
            itemPrice.text = p.toString()
            val web = "http://192.168.1.119/SalesWeb/images/$u"
            web.replace(" ", "%20")
            Picasso.get().load(web).into(itemPhoto)

            itemAddPhoto.setOnClickListener {
                UserInfo.itemId = item_id
                val obj = QtyFragment()
                val manager = (itemView.context as Activity).fragmentManager
                obj.show(manager, "Qty")
            }
        }
    }
}
```

QtyFragment.kt

```
package com.example.salesapp

import android.app.DialogFragment
import android.content.Intent
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import com.android.volley.Request
import com.android.volley.RequestQueue
```

```
import com.android.volley.toolbox.StringRequest
import com.android.volley.toolbox.Volley


class QtyFragment : DialogFragment() {

    @Deprecated("Deprecated in Java")
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        val v = inflater.inflate(R.layout.fragment_qty, container, false)

        val et = v.findViewById<EditText>(R.id.et_qty)
        val btn = v.findViewById<Button>(R.id.btn_qty)

        btn.setOnClickListener {
            val url = "http://192.168.1.119/SalesWeb/add_temp.php?mobile=" + UserInfo.mobile +
                    "&itemid=" + UserInfo.itemId + "&qty=" + et.text.toString()

            val rq: RequestQueue = Volley.newRequestQueue(activity)
            val sr = StringRequest(Request.Method.GET, url, {
                val i = Intent(activity, OrderAct::class.java)
                startActivity(i)
            }, { error ->
                Toast.makeText(activity, error.message, Toast.LENGTH_LONG).show()
            })
            rq.add(sr)
        }

        return v
    }

}
```

OrderAct.kt

```
package com.example.salesapp

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import android.widget.ArrayAdapter
import android.widget.ListView
import android.widget.Toast
import com.android.volley.Request
```

```
import com.android.volley.RequestQueue
import com.android.volley.toolbox.JsonArrayRequest
import com.android.volley.toolbox.StringRequest
import com.android.volley.toolbox.Volley

class OrderAct : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_order)

        val url = "http://192.168.1.119/SalesWeb/get_temp.php?mobile=" + UserInfo.mobile
        val orderLv: ListView = findViewById(R.id.order_lv)

        val list = ArrayList<String>()
        val rq: RequestQueue = Volley.newRequestQueue(this)
        val jar = JsonArrayRequest(Request.Method.GET, url, null, { response ->
            for (x in 0 until response.length())
                list.add(
                    response.getJSONObject(x).getString("name") + "\n" +
                            "Ціна : " + response.getJSONObject(x).getString("price") + "\n" +
                            "Кількість : " + response.getJSONObject(x).getString("qty")
                )

            val adp = ArrayAdapter(this, android.R.layout.simple_list_item_1, list)
            orderLv.adapter = adp
        }, { error ->
            Toast.makeText(this, error.message, Toast.LENGTH_LONG).show()
        })
        rq.add(jar)
    }

    override fun onCreateOptionsMenu(menu: Menu?): Boolean {

        menuInflater.inflate(R.menu.my_menu, menu)
        return super.onCreateOptionsMenu(menu)
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {

        if (item.itemId == R.id.item_menu) {
            val i = Intent(this, HomeAct::class.java)
            startActivity(i)
        }

        if (item.itemId == R.id.item_cancel) {
            val url = "http://192.168.1.119/SalesWeb/cancel_order.php?mobile=" + UserInfo.mobile

            val rq: RequestQueue = Volley.newRequestQueue(this)
            val sr = StringRequest(Request.Method.GET, url, {
                val i = Intent(this, HomeAct::class.java)
                startActivity(i)
```

```
            }, { error ->
                Toast.makeText(this, error.message, Toast.LENGTH_LONG).show()
            })
            rq.add(sr)
        }

        if (item.itemId == R.id.item_confirm) {
            val url = "http://192.168.1.119/SalesWeb/confirm_order.php?mobile=" + UserInfo.mobile

            val rq: RequestQueue = Volley.newRequestQueue(this)
            val sr = StringRequest(Request.Method.GET, url, { response ->
                val i = Intent(this, TotalAct::class.java)
                i.putExtra("bno", response)
                startActivity(i)
            }, { error ->
                Toast.makeText(this, error.message, Toast.LENGTH_LONG).show()
            })
            rq.add(sr)
        }

        return super.onOptionsItemSelected(item)
    }

}
```

TotalAct.kt

```
package com.example.salesapp

import android.annotation.SuppressLint
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.TextView
import android.widget.Toast
import com.android.volley.Request
import com.android.volley.RequestQueue
import com.android.volley.toolbox.StringRequest
import com.android.volley.toolbox.Volley

class TotalAct : AppCompatActivity() {
    @SuppressLint("SetTextI18n")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_total)

        val url =
            "http://192.168.1.119/SalesWeb/get_total.php?bill_no=" + intent.getStringExtra("bno")
```

```
        val totalTv: TextView = findViewById(R.id.total_tv)

        val rq: RequestQueue = Volley.newRequestQueue(this)
        val sr = StringRequest(Request.Method.GET, url, { response ->
            totalTv.text = "Загальна сума: $response грн"
        }, { error ->
            Toast.makeText(this, error.message, Toast.LENGTH_LONG).show()
        })
        rq.add(sr)

        val btnBack: Button = findViewById(R.id.btn_back)
        btnBack.setOnClickListener {
            val i = Intent(this, HomeAct::class.java)
            startActivity(i)
        }
    }
}
```

UserInfo.kt

```
package com.example.salesapp

class UserInfo {
    companion object {
        var mobile: String = ""
        var itemId: Int = 0
    }
}
```