МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій

Кафедра комп'ютерних наук та технологій

ДИПЛОМНА БАКАЛАВРСЬКА РОБОТА

на тему

Розробка мобільного додатку закладу
громадського харчування «меню ресторану»
для смартфонів на базі Android

Виконав: студент групи БІТск-1-19
спеціальності
122 «Комп'ютерні науки»
(шифр і назва спеціальності)
Олійник С.С.
(прізвище та ініціали)
Науковий керівник Демківська Т.І.
(прізвище та ініціали)
Рецензент Чупринка В.І.
(прізвище та ініціали)

КИЇВ 2021

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій

Кафедра комп'ютерних наук та технологій

Спеціальність 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри
комп'ютерних наук та технологій
_____ В.Ю.Щербань
(підпис)
«_____» _____ 20 __ р.

ЗАВДАННЯ

НА ДИПЛОМНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Олійнику Сергію Сергійовичу
(прізвище, ім'я, по батькові студента)

1. Тема роботи Розробка мобільного додатку закладу громадського харчування «меню ресторану» для смартфонів на базі Android

Науковий керівник роботи Демківська Тетяна Іванівна, к.т.н., доцент.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом КНУТД від "15" березня 2021 року №75-уч
2. Строк подання студентом дипломної роботи__14.06.2021 р._____
3. Вихідні дані до дипломної роботи (проекту) __Рекомендована література___, офіційна документація Google, Oracle Corporation_____

4. Зміст дипломної роботи (проекту) (перелік питань, які потрібно розробити)_1) аналіз предметної області, 2) моделювання та проектування додатку, 3) програмна реалізація_____

5. Дата видачі завдання 15.03.2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи (проекту)	Терміни виконання етапів	Примітка про виконання
1	Вступ	06.05.2021	
2	Розділ 1 (Аналіз предметної області)	14.05.2021	
3	Розділ 2 (Моделювання та проектування додатку)	22.05.2021	
4	Розділ 3 (Програмна реалізація)	30.05.2021	
5	Висновки	02.06.2021	
6	Оформлення дипломної роботи (проекту) (чистовий варіант)	03.06.2021	
7	Здача дипломної роботи (проекту) на кафедру для рецензування (за 14 днів до захисту)	04.06.2021	
8	Перевірка дипломної роботи (проекту) на наявність ознак плагіату (за10 днів до захисту)	05.06.2021	
9	Подання дипломної роботи (проекту) на затвердження завідувачу кафедри (за 7 днів до захисту)	06.06.2021	

Студент _____ С.С. Олійник
(підпис) (ініціали та прізвище)

Науковий керівник роботи _____ Т.І. Демківська
(підпис) (ініціали та прізвище)

Рецензент _____ В.І. Чупринка
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

У даній бакалаврській дипломній роботі реалізований програмний додаток, який дозволяє користувачеві розміщати замовлення у закладі громадського харчування. Додаток запрограмований для смартфонів та планшетів на базі операційної системи Android. Передбачається встановлення даного додатку на пристрої, які є власністю замовника даного ПЗ.

Додаток має наступну функціональність:

- перегляд та додавання в корзину з каталогу (меню) страв та напоїв;
- формування і перегляд замовлення та його оплата;
- виклик офіціанта;
- перегляд історії замовлень.

Додаток створено на мові програмування Java для системи Android 7.0 і вище.

Дані про позиції меню (назва, ціна) зберігаються у файі реляційної системи керування базами даних SQLite. За обробку і запис даних відповідають компоненти activities

Ключові слова: Лінукс, Андроїд, Розробка ПЗ, Java, Gradle, ООП.

ANNOTATION

A mobile application was implemented in this bachelor’s graduate work, that allows the user to place an order in a restaurant. Application is developed in Java programming language. It is assumed that this application will be installed on devices that are the property of the customer of this software.

The application has the following functions:

- viewing and adding to the basket from the catalog (menu) of drinks and dishes;
- formation and review of the order and its payment;
- call the waiter;
- view order history.

Data on menu items (name, price) are stored in the SQLite database management system file. Activities components are responsible for data processing and writing

Keywords: Linux, Android, Software development, Java, Gradle, OOP.

ЗМІСТ

ВСТУП

Станом на 2021 рік одними з самих найрозповсюдженіших є онлайн-магазини та додатки для дистанційного замовлення продуктів харчування. Більшість популярних інтернет ресурсів надають послуги та товари, які можна замовити дистанційно й оплатити (в тому числі безготівково), мають розроблені додатки для мобільних операційних систем (Android, iOS тощо). Водночас ресторани, кафе та інші заклади громадського харчування не пропонують відвідувачам електронні системи для покращення та модернізації своїх послуг в усередині закладу. Тобто ринок цифрових технологій надає великий обсяг рішень для замовлення товару чи послуги на дистанційній основі, але саме ці фактори не використовуються для клієнтів, які споживають продукти харчування ресторанів або кафе. Тому метою даної дипломної бакалаврської роботи є поліпшення роботи закладів громадського харчування або, навіть, заміна певних працівників обслуговування (офіціантів, барменів), за допомогою використання даної додатку.

Після епохи настільних ПК та ноутбуків відбувся сплеск інтересу до смартфонів і планшетів з боку користувачів даних пристроїв у споживчому сегменті ринку. У 2020 році кількість користувачів операційної системи Android в Україні за даними компанії StatCounter досягла 26,91% відносно усіх доступних на ринку операційних систем (у т.ч. Windows та MacOS). У розрізі мобільних операційних систем Android в Україні має частку ринку 80,48%. Компанії реалізували цю нову тенденцію і почали фокусуватися на мобільній галузі більше ніж будь-коли раніше. В даний момент, швидко зростає промисловість, яка пропонує багато можливостей. Це відкриває величезний ринок з незадоволеним попитом для інженерів програмного забезпечення. Є багато пристроїв на ринку з різними програмним та апаратним забезпеченнями, що ускладнює роботу розробників електронних систем. Вони мають шукати найкраще рішення, як максимально задовольнити попит з встановленими вимогам і якістю. Мета такої розробки для мобільних платформ – замінити настільне програмне забезпечення веб- та мобільними додатками.

Сьогодні більшість людей отримують деякі товари по замовленню через Інтернет. Замовники сьогодні приваблюються не тільки тим, що розміщення замовлення в Інтернеті дуже зручно, але також тому, що продавці надають прозорість в пропонованих категоріях, товарах та цінах на них, а також завдяки надзвичайно спрощеній навігації для пошуку товару або послуги та його (її) замовлення.

Багато людей споживають страви в ресторані, де недостатньо ефективне обслуговування і увага працівників. Щоб використати зростаючу мобільну індустрію, використовується рішення, яке розроблене на мобільні пристрої для операційної системи Android. Електронне меню ресторану є зручним доповненням до паперового, а у перспективі воно зможе замінити паперове. Враховуючи описане, можна вважати, що створення та розповсюдження нової моделі додатків (навіть нової категорії) має перспективу покращити та створити нові можливості як в сфері розробки програмного забезпечення, так і в сфері обслуговування відвідувачів закладів громадського харчування.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка задачі

Основною задачею даної дипломної бакалаврської роботи є спроба автоматизувати на рівні мобільних платформ обслуговування клієнтів в ресторанах та кафе. Передбачається, що додаток буде встановлюватися на мобільні пристрої, які знаходяться у власності закладу. Користувачеві додатку надається можливість переглядати категорії страв та напоїв, які розподілені у традиційному вигляді, так і в паперовому ресторанному меню. Окрім назв самих страв до списку додаються зображення (для поліпшення розуміння клієнтом свого вибору) та ціни. Додаток надає можливість обирати кількість продуктів і надалі проглядати свій чек, в якому вказується сума, на яку клієнт зробив замовлення. Після остаточного ухвалення свого чеку користувач надає запит на виклик офіціанта. Саме така система дозволяє покращити та прискорити обслуговування клієнтів. У існуючій системі присутні деякі проблеми:

- для розміщення будь-яких замовлень працівники ресторанів мають змінювати категорії та страви, відповідно до меню. У цьому методі необхідні час і ручна робота;
- у кожному ресторані потрібні працівники, щоб:
  - прийняти замовлення через систему або особисто;
  - запропонувати клієнтам спосіб оплати й обробити платіж.

На сьогоднішньому ринку робочі ставки з кожним днем збільшуються, що ускладнює пошук працівників, коли це необхідно.

Отже, щоб вирішити це питання розроблений даний додаток спочатку для невеликого бізнесу, як кав'ярні, кафе, ресторану швидкого харчування, але ця система так само може застосовуватися у будь-якій галузі доставки продуктів харчування.

Головною перевагою моєї системи є те, що це значно спрощує процес замовлення для обох пристроїв замовника та ресторану, а також значно полегшує навантаження на ресторану, оскільки весь процес приймання замовлення автоматизований.

Передбачувані переваги:

1) мінімізація кількісті працівників в ресторанах та кафе;
2) зменшення вартості робочої сили;
3) мінімізація ймовірних помилок.
4) прискорення обслуговування відвідувачів.

Система онлайн-замовлень, яку я пропоную, значно спрощує процес замовлення для замовників і ресторан. Система представляє інтерактивне та найсучасніше меню з усіма доступними варіантами в зручному для використання способах. Клієнт може вибрати один або декілька елементів для розміщення замовлення, який буде приземлятися у кошику. Перед тим, як підтверджувати клієнт може переглянути всі деталі замовлення в кошику. Наприкінці клієнт отримує подробиці підтвердження замовлення. Це дозволяє працівникам ресторану швидко виконувати замовлення, оскільки вони отримані і оброблені ефективно з мінімальними затримками та плутаниною.

1.2 Аналіз предметної області і напрямків дослідження

Сьогодні цифрові технології все більше використовуються людьми в усіх галузях. Багато хто використовує бездротові доступні технології, щоб залишатися в контакті з іншими, незалежно від місця розташування. Зростаюча популярність смартфонів привернули увагу майже всіх. Окрім здійснення та отримання дзвінків, користувачі можуть надсилати та отримувати повідомлення, доступ до Інтернету, медіафайлів, використовувати аудіо- та відеозапис тощо. Смартфони також містять вбудовану клавіатуру, камеру високої роздільної здатності, камеру передньої панелі для відеоконференцій, сенсорний екран тощо. Різні смартфони мають різні функції системи. Мобільний додаток чи просто додаток – це програма, що працює на смартфонах, планшеті або інших мобільних пристроях (як-от Smart-годинники). Програми – це попередньо встановлені або завантажені частини програмного забезпечення, які можуть виконувати певні функції та задовольняти потреби користувача. Програми роблять смартфон більше схожим на портативний комп'ютер завдяки багатоядерним процесорам, великому обсягу пам’яті та функціональній операційній системі. Спочатку

мобільні додатки доступні в інформаційних цілях, включаючи електронну пошту, календар, інформацію про погоду тощо. З вдосконаленням технологій та потреб від користувачів розробники почали створювати додатки для інших задовлення інших потреб. Наприклад, ігри, онлайн-банкінг, відеочат тощо. Програма може показувати дані подібним чином як веб-сайт, а також діяти для завантаження вмісту, який можна використовувати в автономному режимі у випадку, якщо доступ в мережу Інтернет обмежений. На сьогоднішній день на ринку існує безліч додатків для різних операційних систем, зокрема Android, Blackberry та Apple. Android в даний час має максимальну частку ринку.

Android – одна з найпопулярніших мобільних платформ у світі. Побудований на внесках спільноти Linux з відкритим кодом, а також сотні апаратних, програмних та інтегруючих компонентів, Android забезпечує потужну основу розробки для створення мобільних додатків. Використовуючи єдину модель програмування Android дозволяє розгортати додатки для користувачів у широкому діапазоні мобільних пристроїв. Інструмент розробника Android (ADT) – це середовище Android Studio, яке пропонує повний Java IDE: власний набір функцій, які допоможуть розробляти додатки для Android. Програми для Android написані на мові програмування Java. Відкрита платформа розробки Android дозволяє розробникам використовувати сторонні інструменти у своїх додатках, відкривати двері для широкого кола варіантів. Android також забезпечує відмінне середовище тестування з Повний набір інструментів тестування, доступних з ADT.

Існує три типи програм для мобільних пристроїв: локальні, веб-програми та гібриди. Локальні додатки живуть на пристрої. Ці додатки доступні через магазин додатків і розроблені для певної платформи і повністю використовують особливості цього пристрою. Ці програми працювати в автономному режимі і можуть працювати над останніми API цієї платформи. Деякі різні типи локальних додатків включають годинник, калькулятор і т.д. Хоча веб-додатки не являються реальними програмами, замість цього використовуються веб-сайти, призначені для того, щоб мати зовнішній вигляд і симулювати досвід

використання локального додатку. Для цього потрібен браузер, а також засіб передачі даних для запуску. Користувачі спочатку отримують доступ до них через веб-сторінку та в цьому Інтернеті на сторінці вони мають можливість встановити їх на своєму пристрої. Гібридні додатки – це частково прикладна програма, а частково - веб-додаток. Ці програми завантажуються в магазині додатків та скористатися перевагами функцій пристрою. Багато користувачів зацікавлені в використанні локальних програм, оскільки вони можуть бути використані навіть коли немає підключення до Інтернету. Перший Android Mobile був T-Mobile G1, запущений у Сполучених Штатах, який містить вікно сповіщення,яке мало віджети домашнього екрану, що містять Gmail, віджет Google Market і т.д. За даними веб-сайту, кількість додатків доступна в Google Play Store (для платформи Android) становить з грудня 2009 року (16000 додатків) до лютого 2015 року (1,400,000 додатків) зображена. Різні домени додатків для Android включають розваги, освітні цілі, роздрібну торгівлю, фінансові, соціальні, подорожі, освіта, охорона здоров'я тощо.

Сьогодні, коли розвиток апаратного забезпечення мобільного стає кращим, індекс продуктивності набагато перевищує фактичні потреби конфігурації програмного забезпечення. Функції телефону більше залежать від програмного забезпечення. Оскільки операційна система Android стає все більш популярною, програма на базі Android SDK привертає до себе значну увагу. Але зараз частина інтерфейсу програми Android є надто громіздкою, спливаючих реклам надто багато, а функція вказування на рекламу занадто сильно звертає на себе увагу, що створює певні незручності для користувачів.

1.3 Проблеми розробки додатків для Android

Багатофункціональні пристрої. Найпоширенішою проблемою є встановлення властивостей програми для різних пристроїв із різними розмірами екранів, роздільною здатністю і т.д. Існує багато версій кожного пристрою Android і при публікації додатку, існують специфічні для версії деталі. Оскільки це потрібно запускати в різних пристроях, то також це впливає на витрати та бюджетні наслідки. Для кожної версії Android розробник повинен знову

написати код, оскільки існує міграція коду. Проблема доступна для перенесення існуючого коду на нову платформу. Іноді поведінка програми також відрізняється по всій новій платформі.

Тестування додатка Android. Наразі в середовищі розробки недостатньо інструментів для тестування. Існує критична необхідність тестування для платформи Android. Також повинні бути доступні функції налагодження.

Обмежені можливості різних пристроїв: Іноді різні пристрої мають різні можливості з точки зору програмного забезпечення , оскільки деякі браузери мають погану підтримку HTML5.

Емулятори, симулятори. Емулятори – це пристрої, які забезпечують нам апаратне середовище пристроїв Android а симулятори надають нам програмне середовище. Додатки тестуються на емуляторах, яких недостатньо для тестування. Емулятори дуже повільні та забирають багато часу, щоб запустити і запустити.

Інтенсивні дані для додатків. Оскільки мобільні пристрої мають дуже обмежену пам'ять, дуже важко зберігати величезну кількість даних в цьому. Кешування в режимі офлайн не працює належним чином і синхронізація з іншим джерелом даних є складним завданням.

Фрагментація апаратного і програмного забезпечення. Одна кнопка на пристрої Android по-різному працює на іншому пристрої. Тому важко побудувати додаток, який спирається на конкретне обладнання, щоб зробити щось конкретне і важливо враховувати апаратні і програмні відмінності кожного з вживаних на ринку виробників смартфонів.

Оскільки Android – це операційна система з відкритим кодом і можливістю встановлювати ПЗ з неперевірних джерел, це може призвести до появи шкідливих програм, які користувачі можуть встановити помилково або по будь-якій іншій причині, що може призвести до втрати або порушенню даних.

1.4 Особливості Android

Графічний інтерфейс користувача для Android включає в себе такі методи вводу інформації для смартфонів як:

- натискання (tapping);
- гортання (swiping);
- щипок (pinching);
- жести (gestures).

Пристрій за рахунок вібромоторчика надає користувачу принципово інший користувацький досвід у порівнянні з класичними ПК, портативними комп’ютерами тощо.

Також в усіх сучасних моделях смартфонів підтримується технологія мультидотику (multi touch). Саме завдяки цій технології

Висновок

У даному розділі була розглянута постановка задачі дипломної бакалаврської роботи, а також проаналізована предметна область і основні нюанси, які потрібно враховувати розробнику під час проектування та кодування додатків для Android.

2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ ДОДАТКУ

У даному розділі йдеться про функціональні та нефункціональні вимоги до системи. Ми будемо розглядати випадки використання додатків, тому що вони покажуть взаємодію між користувачем і програмою. Важливою частиною для проекту є його планування. Це допоможе зробити структуру проекту, щоб можна було побачити, що зроблено і відстежувати наш прогрес.

Розробка інтуїтивно зрозумілих і простих програм для мобільних пристроїв може бути складним завданням. Платформа Android використовується на пристроях, які залежать від розміру екрана, площі, а також мови та додатків здатних підтримувати ці варіанти так, щоб якість користувацького досвіду була стабільною на всіх підтримуваних пристроях. Сама природа мобільних додатків відмінно впливає на дизайн інтерфейсу користувача. Розмір екрана для мобільних пристроїв, як правило, набагато менше, ніж звичайні користувачі екрану знайомий з настільними додатками. Інформація повинна бути стиснута, користувальницький інтерфейс спрощений, таким чином досвід користувача не погіршується. Мобільні пристрої також вводять ряд функцій і можливості, які недоступні або не зустрічаються на настільних комп'ютерах. Різні датчики такі як акселерометр, GPS, гіроскоп та сенсорні екрани є стандартними для мобільних пристроїв. Ці датчики відкривають світ можливостей у тому, як користувачі можуть взаємодіяти з програмою у багатьох способах, які неможливі з настільними комп'ютерами.

2.1 Огляд використаних у розробці інструментів

Мову програмування, середовище для розробки, SDK тощо для програміста можна порівняти з інструментом та будматеріалами для будівельника. Залежно від конкретного проекту програміст обирає для його реалізації певні інструменти та рішення: мову програмування, середовище для розробки, бібліотеки, фреймворки тощо. У даному розділі розглянуті усі необхідні для розробки додатку рішення та інструменти, а також наведений короткий огляд платформи Android з точки зору програміста.

Java для Android

Java – це популярна крос-платформна об’єктно-орієнтована мова програмування з суворою типізацією. Java випущена 1995 року компанією Sun Microsystems. Вихідний код Java компілюється у байт-код, який виконується у віртуальній машині Java (JVM). Java має продуману та повну реалізацію об’єктно-орієнтованого програмування, а з такими мовами як C, C++, C# та JavaScript її поріднює спільний C-подібний синтаксис.

Філософія Java. Java створювалася з опорою на наступні принципи:

- повна підтримка об’єктно-орієнтованого програмування;
- кросплатформність;
- вбудована підтримка мережевих протоколів;
- безпечне виконання коду з віддалених пристроїв;
- простота програмування.

Java має статичну типізацію і суворий синтаксис. Ця специфікація чітко виділяє помилки, які можуть і повинні бути виявлені під час компіляції, і ті, які трапляються під час виконання. Час компіляції зазвичай складається перекладу програм в автономне байтове представлення коду. Заходи під час виконання включають завантаження та зв'язування класів, необхідних для виконання програма, необов'язкове створення машинного коду та динамічна оптимізація програма та фактичне виконання програми. Мова програмування Java є високорівневою мовою. Це включає управління автоматичним сховищем, зазвичай за допомогою збирача сміття, щоб уникнути проблеми безпеки явного вилучення (як в C б або C + + delete). Високопродуктивні збирачі сміття можуть мати обмежені паузи стосовно підтримки системного програмування та застосування в режимі реального часу. Мова не включає в себе будь-які небезпечні конструкції, такі як прямий доступ до масиву без перевірки індексу, оскільки такі небезпечні конструкції призведуть до того, що програма поводиться неузгоджено.

Компіляція програми для Android відбувається за таким самим шляхом, як перетворення файлів Java у байт-код. Коли додаток (що складається з Bytecode)

встановлюється на пристрій, під час інсталяції відбувається також інший крок. Байт-код додатка перетворюється в машинний код, оптимізований для цього пристрою Android, що покращує продуктивність виконання програми. Цей процес відомий під назвою Ahead of Time (AoT) і став можливим завдяки віртуальній машині Android Runtime (ART). Компіляція AoT відбувається лише в Android KitKat (4.4) і вище, але також забезпечує зворотну сумісність. Попередні версії Android покладалися на іншу віртуальну машину, відома як Dalvik. Як і в APT, Дальвік вніс зміни в байт-код Java, перетворивши його в певну форму, яка зробила різноманітні зміни ефективності, щоб оптимізувати додаток для видів малопотужних пристроїв. Спочатку Android був розроблений. Проте, на відміну від APT, Дальвік не склав Bytecode в машинний код до виконання runtime - використовуючи підхід, відомий під назвою Just in Time compilation (JIT). Цей процес набагато ближчий до того, що використовується у віртуальних середовищах Java на ПК. Android Froyo (2.2) побачив поліпшення, коли Dalvik отримав можливість профілювати додаток під час роботи для часто використовуваних частин Dalvik Bytecode. Далі ці вказівки були назавжди переведені на машинний код Dalvik, щоб підвищити швидкість програми. У будь-якому випадку збірки додатків Android - це перетворення Bytecode, що робить Java, написаний для Android, менш однорідним. Зміни до Bytecode обмежують переносимість додатка. Інший спосіб Android відрізняється від Java з наявністю стандартних бібліотек. Java є такою портативною, оскільки вона спирається на стандартизовану колекцію бібліотек, яку можна використовувати на різних платформах, таких як мережеві бібліотеки та бібліотеки інтерфейсу користувача. Android пропонує підмножину того, що забезпечує Java, і те, що вона надає, є лише для Android.

Android широко використовує Java прийняття парадигм об'єктно-орієнтованого програмування і він розроблений, щоб обійти концепції інкапсуляції, успадкування та поліморфізму. Використовується все це під час створення додатків. Всі об'єкти в Android успадковуються від класу Object у певній формі, спираючись на свої функції, щоб забезпечити спеціальну

поведінку та функції. Деякі об'єкти, доступні через API Android вказують на певну ієрархію, яка успадковує кожен об'єкт, і всі вони, зрештою, успадковують один від одного.

2.2 Об’єктно-орієнтована парадигма програмування

Об’єктно-орієнтоване програмування (ООП) – парадигма програмуваня, у якій основними концепціями є поняття об’єкті і класів.

Об’єкт – це сутність, екземпляр класу, у котрому можна посилати повідомлення і яка може на них реагувати використовуючи свої дані. Дані об’єкта приховані від основного коду програми. Приховання даних називається інкапсуляцією. Наявність інкапсуляції достатньо для об’єктності мови програмування, але це ще не означає його об’єкної орієнтованості – для цього необхідна наявність наслідування. Третім надзвичайно важливим критерієм об’єктної орієнтованості є поліморфізм. Поліморфізм – це можливість з однаковою специфікацією мати різну реалізацію.

Також виділяють ще одну важливу, але не основну характеристику ООП – абстрагування. Абстрагування це спосіб виділити набір значимих характеристик об’єкту виключаючи з розгляду незначимі характеристики. Відповідно, абстракція – це набір усіх таких значимих характеристик.

Інкапсуляція – це властивість системи, яка дозволяє об’єднувати дані і методи, що працюють з ними у класі, і приховувати деталі реалізації від користувача.

Наслідування – це властивість системи, яка дозволяє описати новий клас на основі вже існуючого з частково або повністю запозиченою функціональністю. Клас, з якого здійснюється наслідування, називається базовим, батьківським класом. Новий клас – потомком, спадкоємцем або похідним класом.

Поліморфізм – це властивість системи використовувати об’єкти з однаковим інтерфейсом без інформації про тип і внутрішню структуру об’єкта. На рис. 2.1 наведено схематичне зображення принципів ООП.

Рисунок 2.1. Схематичне зображення принципів ООП.

Реалізацією інкапсуляції у Java є 4 види модифікаторів доступу: public, protected, default, private.

Public – рівень передбачає доступ до компоненту з цим модифікатором із екземпляру будь-якого класу і будь-якого пакета.

Protected – рівень передбачає доступ до компоненту з цим модифікатором з екземплярів рідного класу і класів-потомків незалежно від того, в якому пакеті вони знаходяться.

Default – рівень передбачає доступ до компоненту з даним модифікатором з екземплярів будь-яких класів, що знаходяться в одному пакеті з цим класом.

Private – рівень передбачає доступ до компоненту з даним модифікатором тільки з-під цього класу.

Приклад:

```
public class Human {
    public String name;
    protected String surname;
    private int age;
    int birthdayYear;
}
```

public String name; — ім’я, яке доступне з будь-якого місця у додатку.

protected String surname; — прізвище доступне з рідного класу та його потомків.

private int age; — вік доступний тільки к рамках класу Human.

int birthdayYear; — хоч і не вказується явний модифікатор доступу, система розуміє його як default, дата народження буде доступна всьому пакету, в якому знаходиться клас Human.

2.3 Android

У цьому параграфі ми розглянемо версії Android та середовище розгортання. Перша версія операційної системи Android була опублікована у 2008 році. Android стабільно кожні 2-3 роки отримує оновлення. Кожна версія принесла нові можливості користувачам і розробникам. Розробники Android повинні враховувати версію Android їх додаток буде працювати. Нашою метою є розробка програми, яка буде працювати на різних версіях операційної системи Android. Рішення розроблювати вказане ПЗ на базі ОС Андроїд 9.0 Nougat було викликано двома головними факторами: особисте використання мобільних девайсів на основі саме цієї операційної системи а також тим, що велика кількість смартфонів різних виробників (Samsung, Huawei, Xiaomi, Realme тощо) об’єднується навколо Android версій 9 та 10. У табл. 2.3 наведені дані про частку ринку основних існуючих версій Android за 2020 р.

Таблиця 1.3 Частка ринку різних версій Android

ВЕРСІЯ	**API**	**Частка ринку, %**
4.0 – 4.4	16-19	1,56%
7.0-7.1	21-22	10,65%
8.0-8.1	23	13,03%
9.0	24-25	21.71%
10.0	26-28	41,61%
Інші версії	-	11,44%

Порівняння Android та iOS

Головним конкурентом Android є операційна система для мобільних платформ від Apple Inc. - IOS. Одним з найбільш очевидних переваг вибору Android для iOS є те, що Android пропонує безліч додаткових можливостей для розгортання обладнання. Екосистема Android величезна, з пристроями, які відповідають будь-якому смаку та бюджету. Для тих, хто хоче кращого флагмана в класі, Samsung Galaxy S21 є кращим пристроєм на ринку, а середня пропозиція від таких як брендів як Honor і OnePlus обіцяє високу продуктивність, не порушуючи банків. В той час як Apple має обмежений модельний ряд пристроїв, які мають сімейство Android, все ще існує чітке розповсюдження варіантів. Між звичайними, Plus та SE варіантами iPhone охоплює практично всі розміри, які ви могли б хотіти, як і різні члени сім'ї iPad. Apple може запропонувати різні пристрої в залежності від потреб покупця, але iOS все ще суттєво більш обмежена, ніж Android з точки зору апаратного забезпечення.

Виробники зазвичай не приділяють увагу питанням безпеки і захисту інформації. Перелік вразливостей, експлоїтів та інших недоліків безпеки, виявлених в ОС Android, досить великий як і список додатків, що виявляються зловмисним програмним забезпеченням, розміщених у магазині Google Play.

iOS має кращу репутацію, але і тут не все ідеально. Недавні помітні недоліки, включаючи помилку HomeKit і вразлвості Meltdown/Spectre. В цілому, проблем, пов’язаних з безпекою даних на пристроях Apple значно менше, ніж на Android. Це сприяє швидшому розгортанню програмного забезпечення. Apple може надавати оновлення на всі свої телефони безпосередньо, тоді як користувачі Android повинні зачекати, доки їх виробник телефонів не впровадив версію оновлення Google, яка працює з власною підсистемою Android.

2.4 Android Studio

Для нашого проекту ми будемо використовувати Android Studio як офіційну IDE для розробки на Android. Android Studio пропонує середовище з логічним структуруванням. Графічний інтерфейс може бути написаний у формі файлу XML. Додаток під час розробки буде виконуватися в емуляторі а також

протестований на фізичному пристрої, підключеному до комп'ютера. Android Studio пропонує безліч функцій і робить розробку додатків більш зручною.

В якості IDE функція Android Studio полягає в тому, щоб забезпечити інтерфейс для розробника і програмувати і керувати проектами. У той же час Android Studio надає доступ до Android SDK або 'Software Development Kit'. Подумайте про це як про розширення коду Java, що дозволяє плавно працювати на пристроях Android і скористатися перевагами рідного апаратного забезпечення. Java як крос-платформна мова програмування застосовується для створення програмних продуктів на різних програмно-апаратних платформах а, для того, щоб ці додатки виконувалися на Android, потрібна SDK для Android. Android Studio має мету надати для розробника повний комплект інструментів для цього. Android Studio також дозволяє запускати код або через емулятор, або через пристрій, підключений до комп’ютера. Після цього також можна налагодити програму під час роботи, отримати зворотній зв'язок, пояснюючий причину проблеми та інше, щоб розробник мав змогу швидше вирішити проблему.

Середовище виконує наступні функції в поточній стабільній версії:

- графічна підтримка Gradle;
- спеціалізований рефакторинг і швидкі виправлення для Android;
- інструменти Lint для досягнення продуктивності, зручності використання, сумісності версій та інших задач;
- можливість інтеграції ProGuard;
- шаблони для створення спільних конструкцій та компонентів;
- редактор макету з широкими можливостями, який дозволяє користувачам перетягувати й вставляти компоненти інтерфейсу користувача, можливість попереднього перегляду макетів на кількох екранах конфігурацій;
- підтримка створення додатків Android Wear додатків;
- підтримка Google Cloud Platform, що дозволяє інтегрувати Cloud Messaging з "Firebase" (раніше "Google Cloud Messaging") та Google App Engine;

- віртуальний пристрій Android (емулятор) для запуску та налагодження програм у Android-студії.

Це офіційна IDE (інтегроване середовище розробки) для платформи Android, розроблена компанією Google і використовується для більшості програм. Android Studio була вперше оголошена на конференції Google I / O в 2013 році і була випущена широкій публіці в 2014 році після різних бета-версій. До його випуску розробка Android здійснювалася переважно через Eclipse IDE, що є більш загальним Java IDE, який також підтримує безліч інших мов програмування. Android Studio спрощує процес розробки ПЗ (програмного забезпечення) порівняно з непрофесійним середовищами, які направлені не тільки на розробку додатків та програм саме під цю операційну систему.

2.5 Gradle

Протягом багатьох років розробка програм мала прості вимоги до компіляції та упаковки програмного забезпечення. Але ландшафт сучасного програмного забезпечення змінився і тому існують потреби в автоматизації розробки. Сьогодні проекти залучають великі та різноманітні стеки програмного забезпечення, що включають декілька мов програмування та застосування широкого спектру стратегій тестування. З підвищення гнучкої практики, програми повинні підтримувати ранню інтеграцію коду, а також часто і доступне тестування та виробниче середовище.

Gradle – система автоматичного збирання проекту. Це наступний еволюційний крок у створенні інструментів на базі JVM. Він опирається досвід таких інструментів автоматичного збирання як Ant і Maven і використовує їх напрацювання. Gradle використовує предметно-орієнтовану мову (DSL, domain-specified language) замість XML. Оскільки Gradle - це рідний JVM, це дозволяє писати власну логіку на Java або Groovy. У світі Java є неймовірно велика кількість бібліотек та структур. Управління залежностями використовується для автоматичного завантаження цих компонентів з репозиторіїв і їх додавання до вашого коду програми. Gradle не тільки легко інтегрувати у проект. Також Gradle сумісний з існуючою інфраструктурою управління залежностями (наприклад,

Maven і Ivy). Здатність Gradle управляти залежностями не обмежується зовнішніми бібліотеками. Оскільки ваш проект зростає в розмірах та складності, ви хочете організувати його код в модулі з чітко визначеними обов'язками. Gradle забезпечує потужну підтримку для визначення та організації багатопроектних збірок, а також для моделювання залежностей між проектами.

Gradle чи Marven?

Припустим, потрібно скопіювати файл у певне місце за умовою, що ви створюєте версію вашого випуску. Щоб визначити версію, ви перевіряєте рядок в метаданих, що описують ваш проект. Якщо це відповідає певній нумерації (наприклад, 1.0-RELEASE), ви скопіюєте файл з точки А в пункт В. З зовнішньої точки зору це може звучати як тривіальне завдання. Якщо ти маєш покластися XML - мова складання багатьох традиційних інструментів, що виражають цю просту логіку , стає проблемним процесом. Відповіддю може бути додавання функцій скриптів через нестандартні механізми розширення. Ви закінчуєте змішування коду сценаріїв з XML або за допомогою зовнішніх сценаріїв з вашої логіки побудови.

Ось ще один приклад. Maven слідує парадигмі convention over configuration, ввівши стандартизований макет проекту та побудувавши життєвий цикл для проектів Java. Це чудовий підхід, якщо ви хочете забезпечити уніфіковану структуру програми для greenfield-проекту, який не містить жодних обмежень, що накладаються попередньою роботою. Однією зі строгих умов Maven є те, що один проект повинен виробляти один артефакт, такий як файл JAR. Для цього потрібно створити два окремі проекти незважаючи на те, що ви можете зробити це з обхідним рішенням, ви не можете відчути, що ваш процес збірки потрібно адаптувати до інструмента, а не інструмент до вашого процесу побудови.

Це лише деякі з проблем, з якими розробники могли зіткнутися. Часто доводиться жертвувати нефункціональними вимогами до моделювання вашого підприємства області автоматизації.

2.6 Система керування базами даних SQLite

Для зберігання, оброблення та додавання даних у ході виконання додатку необхідно застосовувати певне сховище даних. У розробці даного додатку було вирішено використати полегшену реляційну систему керування базами даних SQLite. Систему керування базами даних реалізовано у вигляді бібліотеки, котру можна підключити до проекту. SQLite не потребує окремого процесу (серверу) і компонується разом з основним кодом додатку. Це досить швидкий та зручний спосіб збереження та опрацювання даних у додатку, котрий не вимагає окремого серверу БД, не навантажує мобільний пристрій і дозволяє в короткі терміни обробляти та записувати інформацію в БД.

2.7 Логічна модель БД

Логічне проектування – це вивчення предметної області додатку, яке дозволяє визначити інформаційні зв’язки системи. На етапі логічного проектування визначаються об’єкти (сутності) та взаємозв’язки між ними, формується логічна модель бази даних.

Сутність – це клас однотипних об’єктів, інформація про які повинна бути врахована у моделі. Наприклад, сутність «Рахунок».

Екземпляр сутності – це конкретний представник даної сутності. Наприклад, «Рахунок за пасту «Карбонара»».

Атрибут сутності – це іменована характеристика, яка є певною властивістю сутності. Наприклад, сутність «Історія замовлень» такі атрибут як «Подія», «Номер стола», «Дата і час».

Зв’язок – це деяка асоціація між двома сутностями. Наприклад, сутність «Страви і напої» має зв’язок з сутністю «Рахунок» і може виражатися наступною фразою: «Страва і напій замовляється відвідувачем і додається в рахунок».

Логічна модель не прив’язана до конкретної програмно-апаратної реалізації СКБД і несе виключно інформативну та демонстраційну функції майбутньої БД. На рис. 2.2 наведена логічна модель БД, яка містить дані про доступні у закладі страви та напої й оброблює та зберігає замовлення.

Рисунок 2.2 Логічна модель бази даних

Перелік сутностей моделі:

1. Страви і напої
2. Рахунок
3. Історія замовлень

Перелік атрибутів до кожної з сутностей:

1. Страви і напої:
    a. Назва страви.
    b. Ціна.
2. Рахунок:
    a. Назва страви.
    b. Ціна.
3. Історія замовлень:
    a. Подія.
    b. Номер стола.
    c. Дата і час.

2.8 Функціональні вимоги

Функціональні вимоги вказують на функції, які повинен виконувати додаток. У табл. 2.1 наведені функціональні вимоги нашої заявки. Функціональні вимоги вказують на конкретні результати системи. Це слід протиставляти нефункціональним вимогам, які вказують на загальні характеристики, такі як вартість та надійність. Функціональні вимоги керують архітектурою програми системи, нефункціональними вимогами керує технічна архітектура системи.

Таблиця 2.1. Функціональні вимоги.

Вимога	**Опис**
Відкрити список категорій страв	Користувач може продивитися категорію страв і вибрати потрібну
Додати у корзину	Додати при натисканні страву з певної категорії до корзини
Проглянути корзину	Перейти до кошика ,де знаходиться список вибраних страв
Очистити картину	Прибрати усі вибори страв із корзини
Підтвердити замовлення	При підтвердженні списку у кошику , замовлення відправляється до кухні
Проглянути чек	Проглянути список всього замовленого та ціну, котру треба сплатити
Підтвердити чек	При натисканні персонал отримує сповіщення про готовність клієнтом оплатити замовлення
Викликати офіціанта	При натисканні персонал отримує сповіщення, що клієнт потребує працівника ресторану
Проглянути історію	Переглянути список усіх підтверджених дій

2.9 Нефункціональні вимоги

В інженерних системах та інженерних вимогах нефункціональна вимога (NFR) – це вимога, яка визначає критерії, які можуть використовуватися для оцінки роботи системи, а не конкретної поведінки. Вони протиставляються функціональним вимогам, що визначають певну поведінку або функції. План впровадження функціональних вимог детально описаний у проекті системи. План впровадження нефункціональних вимог детально описаний в архітектурі

системи, оскільки вони, як правило, є архітектурно значимими вимогами. Загалом, функціональні вимоги визначають те, що система повинна робити, а нефункціональні вимоги визначають, як система повинна виконувати ці задачі. Функціональні вимоги зазвичай включають індивідуальну дію або частину системи, можливо явно в сенсі математичної функції, функції вводу, виводу, обробки . На противагу цьому, нефункціональні вимоги описують загальну властивість системи в цілому або конкретного аспекту, а не конкретну функцію. Загальні властивості системи зазвичай позначають різницю між тим чи успішно реалізувався проект розвитку, чи ні. У табл. 2.2. наведені нефункціональні вимоги для даної дипломної бакалаврської роботи.

Таблиця 2.2. Нефункціональні вимоги

Вимога	**Опис**
Графічний інтерфейс користувача	Графічний інтерфейс користувача створений без використання зайвих технологій та ускладнених функцій. Спроектовано для максимально інтуїтивно зрозумілого використання додатку.
Платформа	Додаток доступний на систему Andoid, Підтримуються версія Android 7.1 і пізніше.
Оцінка затратності апаратних ресурсів	Додаток не потребує великих обчислювальних потужностей від апарату та не займає багато місця на накопичувачі.

Висновок

У даному розділі були розглянуті такі рішення та інструменти як-от: мова програмування Java, об’єктно-орієнтований підхід, середовище розробки Android Studio, систему автоматичного збирання Gradle та система керування базами даних SQLite. Усі ці рішення та інструменти застосовуються для розробки додатку в рамках бакалаврської дипломної роботи.

Приділено увагу моделюванню БД, конкретизації функціональних і нефункціональних вимог для додатку.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

Мета: створення додатку, який покликаний автоматизувати процес розміщення замовлення відвідувачем закладу.

Функціональність додатку:

- навігація по асортименту страв і напоїв, доступних у закладі;
- формування корзини замовлення;
- розміщення замовлення;
- створення запиту на оплату чеку готівкою або безготівковим способом (за умови підключеного платіжного шлюзу);
- виклик офіціанта;
- перегляд історії замовлень.

Для збереження і обробки інформації про замовлення використовується реляційна БД SQLite.

Для візуалізації застосовані діаграми активності, діаграми процесів використання (Use Case Diagram) та їх опис. Приклад використання – це набір сценаріїв, що описують взаємодію між користувачем та системою.

3.1 Структура вихідного коду

Основна функціональність та деякі властивості графічного інтерфейсу реалізовані у вигляді класів, до яких можна отримати доступ з будь-якого місця у коді додатку.

Перелік класів та їх короткий опис:

- AskForCheckActivity – клас, який обробляє запит на формування рахунку для оплати;
- CartActivity – обробник корзини;
- DatabaseHandler – підключення та ініціалізація бази даних;
- Food, FoodActivity та FoodAdapter, – класи, в яких реалізована функціональність перегляду та взаємодії с меню доступних у закладі страв і напоїв;

- History, FoodHistoryAdapter – клас, в якому реалізована функціональність перегляду історії замовлень;
- MainActivity, Menu, MenuAdapter, MyOrdersActivity – класи, які відповідають за коректну роботу UI головного екрану, меню та замовлень.

3.2 Фізична модель бази даних

Фізична модель БД визначає спосіб розміщення даних у середовищі зберігання та способи доступу до цих даних, котрі підтримуються на фізичному рівні. Відрізняється від концептуально та логічної моделей детальним описом структури БД з використанням конкретних, чітко визначених властивостей і специфікацій для обраної системи управління базами даних. Фізична модель будується виключно на основі логічної й чітко слідує її структурі: таблиці відповідає сутність, стовпцю – атрибут.

У табл. 3.1 описана відповідність сутностей логічної моделі і таблиць фізичної для БД додатку.

Таблиця 3.1 Відповідність сутностей таблицям

Сутність	**Таблиця**
Страви і напої	Food
Рахунок	Bill
Історія замовлень	History

Як було сказано вище, фізична модель детально описує реалізацію бази баних у конкретній СКБД. Кожному стовпцю повинен відповідати тип даних, які він буде зберігати. У табл. 3.2, 3.3., 3.4 наведені стовпці для таблиць Food, Bill і History відповідно з вказанням типу даних, сумісного з SQLite.

Таблиця 3.2 Таблиця Food

Ім’я стовпця	**Тип**	**Примітка**
key_c	Int	Первісний ключ
food_name_c	String	
food_cost_c	Double	

Таблиця 3.3 Таблиця Bill

Ім’я стовпця	**Тип**	**Примітка**
key_b	Int	Первісний ключ
food_name_c	Int	Зовнішній ключ
food_name_b	String	
food_cost_b	Double	

Таблиця 3.4 Таблиця History

Ім’я стовпця	**Тип**	**Примітка**
key_h	Int	Первісний ключ
key_b	Int	Зовнішній ключ
food_event	String	
table_number	Int	
food_cost_h	Double	
date	Text	

Таблиці бази даних Food та Bill й Bill та History пов'язані між собою типом зв'язку «один-до-багатьох».

На рис. наведена схема фізичної моделі БД додатку з вказаними для SQLite типами даних та типами зв’язків, а у лістингу коду 1 наведена частина програмного коду, яка відповідає за створення таблиць.

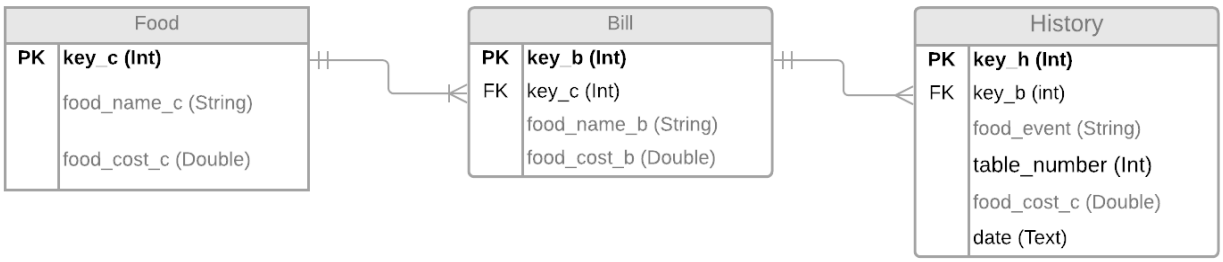

Рисунок 3.1 Фізична модель бази даних

Лістинг 1. Реалізація БД додатку.

```
public databaseCreator(Context context) {
    super(context, "ordersManager", null, 1);
}

// Створення Таблиць
```

```
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL("CREATE TABLE " + "cart" + "("
            + "id_c" + " INTEGER PRIMARY KEY," + "food_name_c" + " TEXT,"
            + "food_cost_c" + " TEXT" + ")");
    db.execSQL("CREATE TABLE " + "bill" + "("
            + "id_b" + " INTEGER PRIMARY KEY," + "food_name_b" + " TEXT,"
            + "food_cost_b" + " TEXT" + ")");
    db.execSQL("CREATE TABLE " + "history" + "("
            + "id_h" + " INTEGER PRIMARY KEY," + "food_event" + " TEXT,"
            + "table_number" + " TEXT," + "date" + " TEXT"  + ")");
}
```

3.3 Діаграма варіантів використання (Use Case Diagram)

Діаграма варіантів використання показує взаємозв'язок між акторами та випадками використання. Два основних компоненти use case діаграм є використання випадків та дійових осіб.

Дійова особа представляє користувача або іншу систему, яка буде взаємодіяти з вашою системою моделювання Прикладом використання є зовнішній вигляд системи, що представляє певну дію, яку може використати користувач для виконання завдання. Використання випадків проходить практично в кожному проекті. Вони допомагають виявити вимоги та планування проекту. На початковому етапі розробки програмного забезпечення необхідно визначити більшість випадків використання, але, в залежності від того,як продовжуватиметься проект, зміни можуть стати помітними.

На рис. 3.2 наведена діаграма використання додатку даної бакалаврської дипломної роботи.

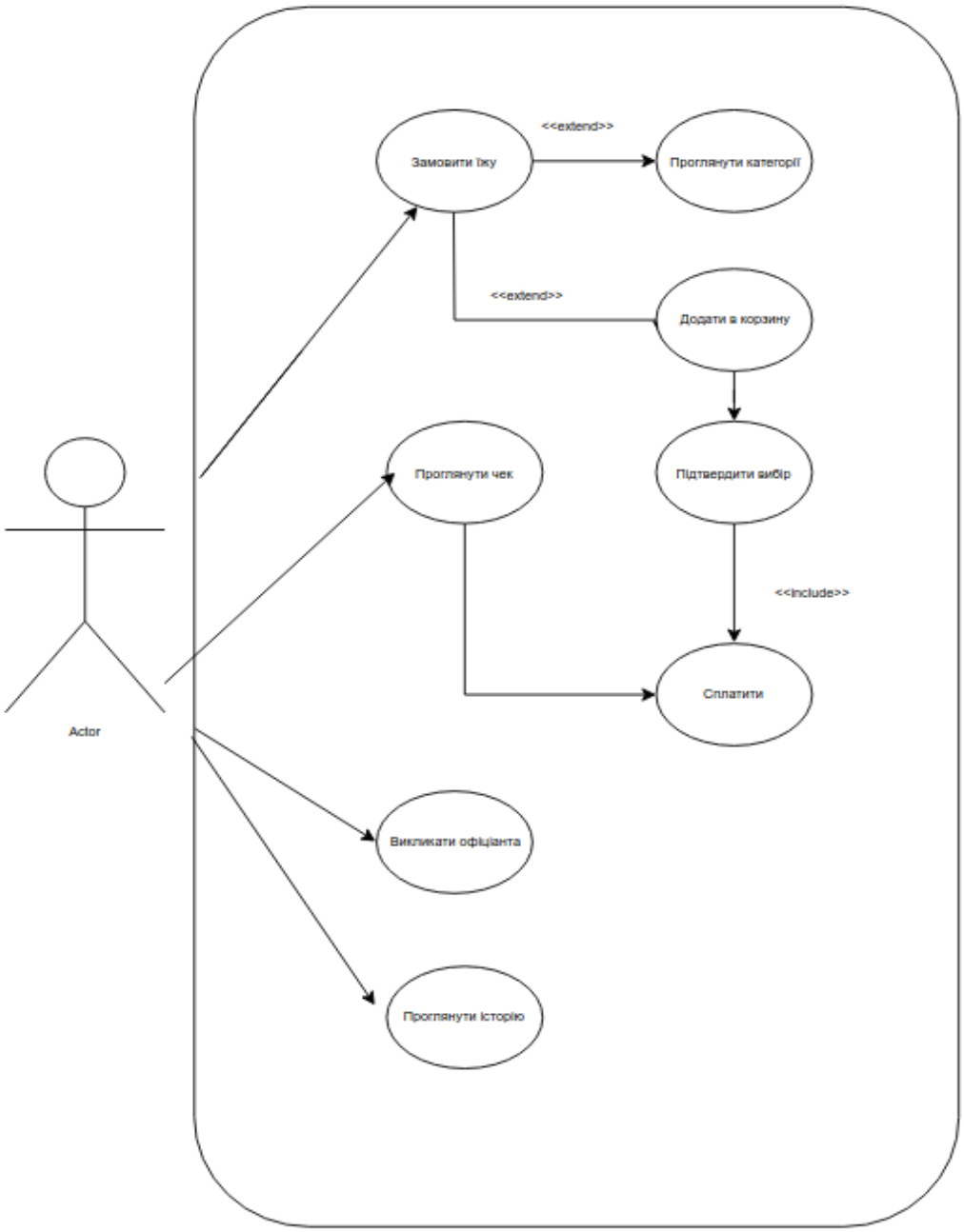

Рисунок 3.2 Діаграма варіантів використання

У табл. 3.5 - 3.10 наведені пояснення до кожного випадків використання.

Таблиця 3.5 Замовлення

USE CASE 1	Замовлення їжі
Особа	Клієнт
Основний потік	1. Користувач вибирає основну кнопку “Order Food” 2. Відкривається меню по категоріям страв та напоїв

Таблиця 3.5 Додавання в корзину

USE CASE 2	Додати в корзину
Особа	Клієнт
Основний потік	1. Користувач вибирає страву,або напій (позицію) 2. Позиція додається в корзину і користувач може перейти до корзини чи продовжити свій вибір

Таблиця 3.6. Розміщення замовлення

USE CASE 3	Розміщення замовлення
Особа	Клієнт
Основний потік	1. Користувач підтверджує вибір і переходить до корзини 2. Користувач розміщує замовлення
Альтернативний потік №1	1. Користувач повертається до каталогу і додає у корзину інші позиції
Альтернативний потік №2	1. Користувач видаляє зміст корзини.

Таблиця 3.7. Оплата чеку

USE CASE 4	Оплата чеку
Особа	Клієнт
Умова	Розміщене у системі замовлення
Основний потік	1. Користувач підтверджує вибір і сплату чеку 2. Користувач підтверджує корзину

Таблиця 3.8 Перегляд чеку

USE CASE 5	Переглянути чек
Особа	Клієнт
Основний потік	1. Користувач переглядає свій чек з деталями здійсненної оплати

Таблиця 3.9 Виклик офіціанта

USE CASE 6	Викликати офіціанта
Особа	Клієнт
Основний потік	1. Користувач надає запит на виклик офіціанта

Таблиця 3.10 Перегляд історії

USE CASE 7	Переглянути історію
Особа	Клієнт
Основний потік	1.Користувач бачить список дій,які виконувались,а саме,враховуються наступні дії: - розміщення замовлення - оплата чеку - виклик офіціантя

3.4 Діаграма активності

Діаграми активності описують поведінку робочого процесу системи. Діаграми активності схожі на діаграми стану, оскільки діяльність - це стан справ. Діаграми описують стан показуючи послідовність виконаних дій. Діаграми активності можуть відображати види діяльності умовними або паралельними.

Коли використовувати? Основною причиною використання діаграм діяльності є моделювання робочого процесу за розробленою системою. Діаграми активності також корисні для:

- аналізу ситуації використання, описуючи, які дії повинні відбуватися і коли вони повинні виникати;
- описування складного послідовного алгоритму;
- моделювання додатків з паралельними процесами.

На рис. 3.3 наведена діаграма активностей додатку, що розробляється.

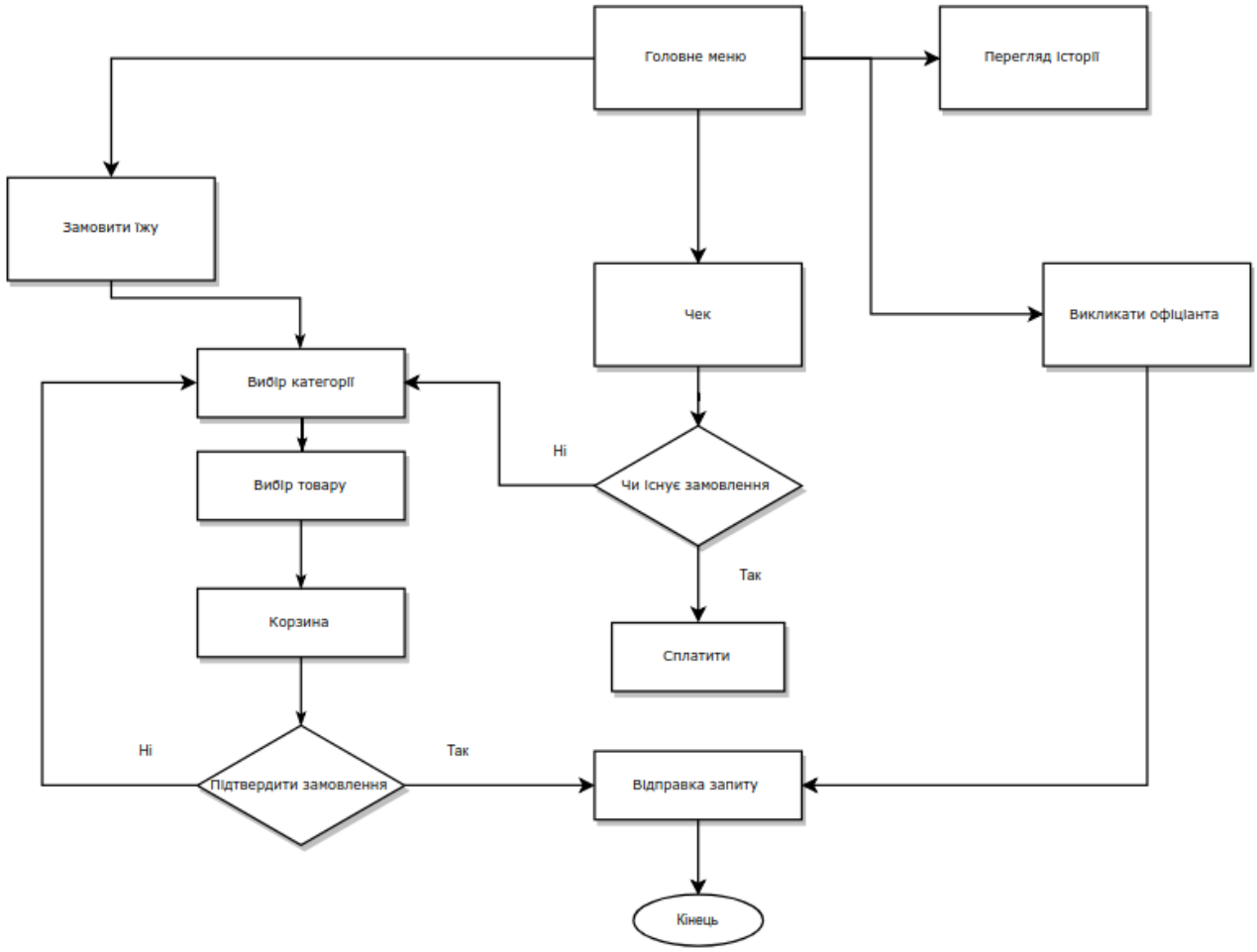

Рисунок 3.3. Діаграма активності

3.5 Інструкція користувача

В даному підрозділі за допомогою ілюстрацій (знімків екрану) наводиться інструкція для користувача додатку.

Для того, щоб розмістити замовлення у закладі, у головному екрані додатку торкніться кнопки «Order Food» (рис. 3.4). Далі оберіть категорію страв та/або напоїв (рис. 3.5).

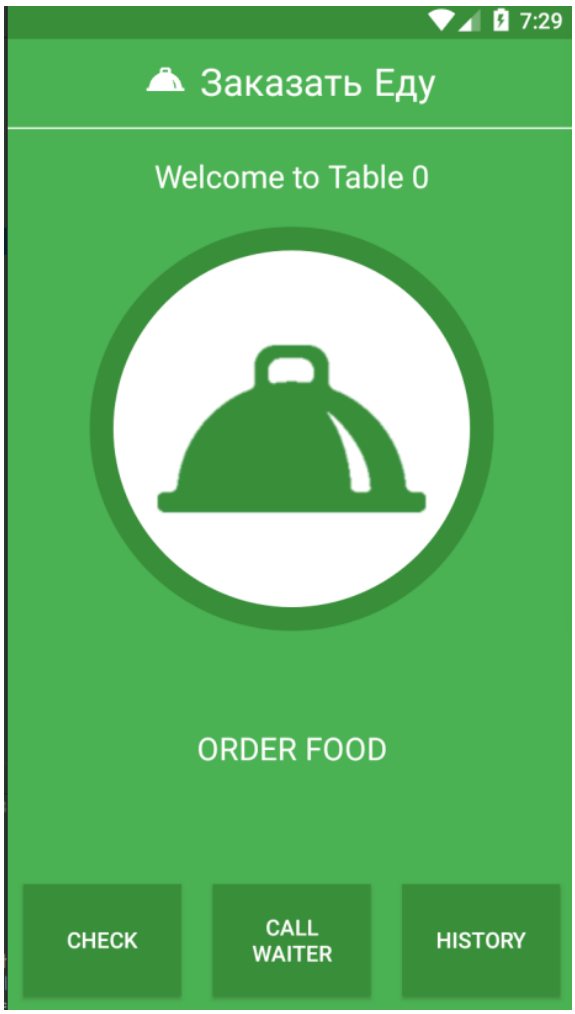

Рисунок 3.4

Рисунок 3.5

Оберіть страву або напій і торкніться піктограми ⊕ навпроти назви страви (напою) (рис. 3.6). Торкніться кнопки «Назад», щоб повернутися до списку категорій і додати у корзину інші позиції, або натисніть на піктограму корзини ( ), щоб перейти до корзини (рис. 3.6). Для того, щоб розмістити замовлення, торкніться кнопки «Order». Зверніть увагу, що кнопка «Empty cart» очищає всю корзину від обраних Вами позицій. Підтвердіть замовлення, торкнувшись кнопки «Yes» (рис. 3.7, 3.8).

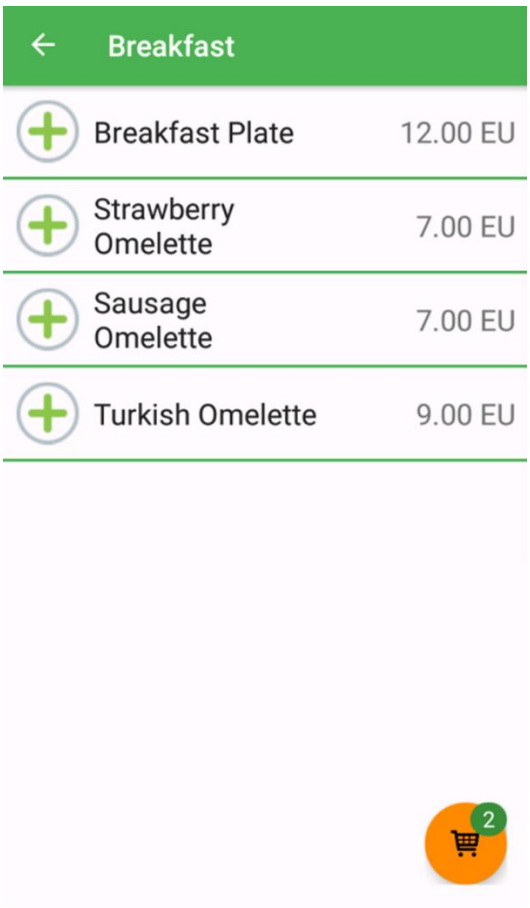

Рисунок 3.6

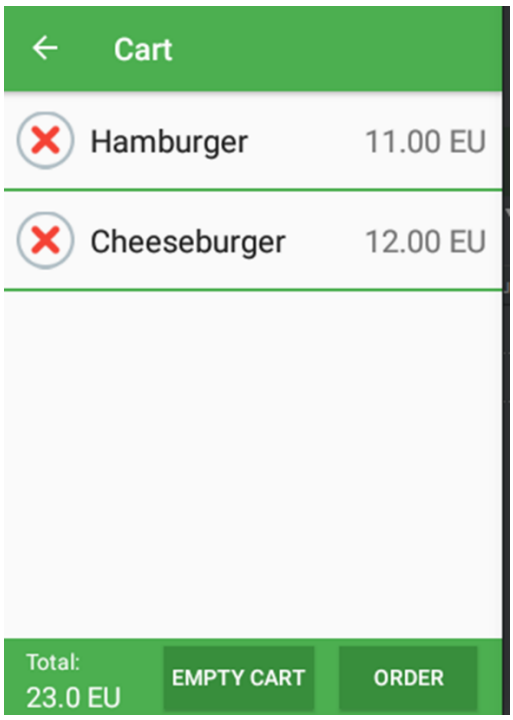

Рисунок 3.7

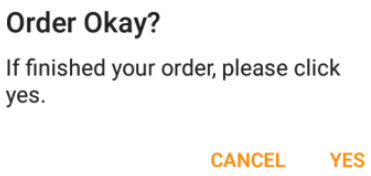

Рисунок 3.8

Для оплати розміщеного замовлення з головного екрану торкніться кнопки «Check» (рис. 3.8).

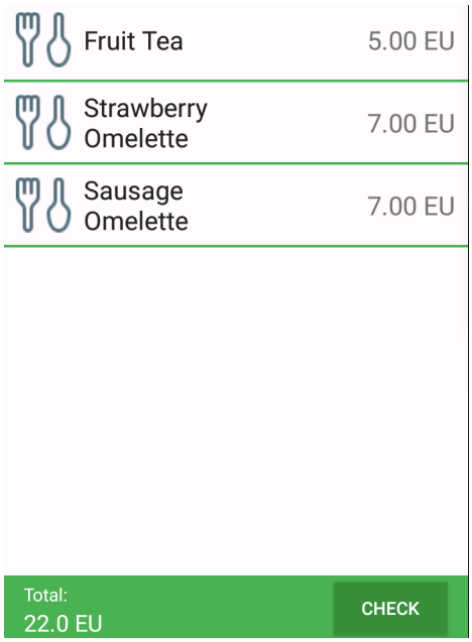

Рисунок 3.8

Для виклику офіціанта з головного екрану натисніть на кнопку «Call waiter» (рис. 3.9, 3.10).

Рисунок 3.9

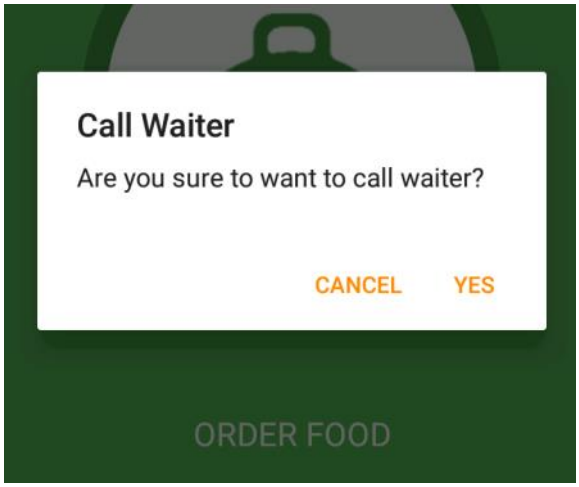

Рисунок 3.10

Для перегляду історії дій у додатку з головного екрану натисніть на кнопку «History» (рис. 3.11).

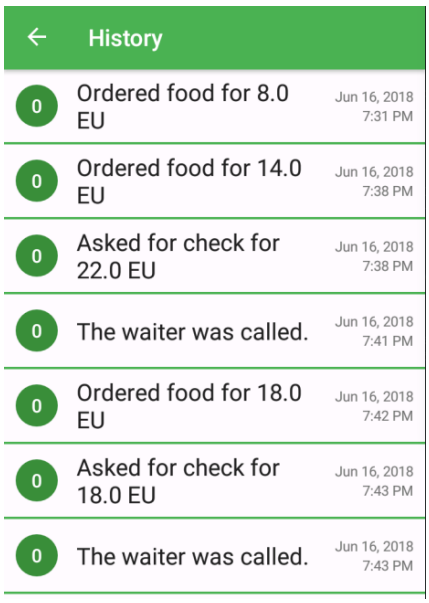

Рисунок 3.11

ВИСНОВКИ

У даній бакалаврській дипломній роботі проведено ґрунтовне дослідження ситуації на ринку мобільних додатків загалом і предметної області зокрема. Викладені основні теоретичні та практичні основи програмування мобільних додатків для операційної системи Android з використанням сучасних мов програмування та об’єктно-орієнтованої парадигми. Результатом практичної роботи в рамках даної бакалаврської дипломної роботи було розроблено додаток, головною метою якого є спрощення та автоматизація процесів обслуговування відвідувачів закладів громадського харчування.

Додаток розроблено у середовищі Android Studio на мові програмування Java. Передбачається, що додаток буде встановлюватися на планшети або смартфони під замовлення закладу. Реалізовано наступну функціональність:

- навігація по асортименту страв і напоїв, доступних у закладі;
- формування корзини замовлення;
- розміщення замовлення;
- створення запиту на оплату чеку готівкою або безготівковим способом (за умови підключеного платіжного шлюзу);
- виклик офіціанта;
- перегляд історії замовлень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Operating System Market Share Ukraine [Електронний ресурс] // StatCounter.com. – 2021. – Режим доступу до ресурсу: https://gs.statcounter.com/os-market-share/all/ukraine/2020.
2. Mobile Operating System Market Share Ukraine [Електронний ресурс] // StatCounter.com. – 2021. – Режим доступу до ресурсу: https://gs.statcounter.com/os-market-share/mobile/ukraine/2020.
3. Android Developers [Електронний ресурс] // developers.android.com. – 2020. – Режим доступу до ресурсу: https://developer.android.com/guide.
4. Clean Coder [Електронний ресурс] // blog.cleancoder.com. – 2012. – Режим доступу до ресурсу: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html.
5. Bloch J. Effective Java, Third Edition / Joshua Bloch., 2017.
6. Kathuria A. Challenges in Android Application Development: A Case Study / A. Kathuria, A. Gupta. // International Journal of Computer Science and Mobile Computing. – 2015. – С. 294–299.
7. Head First Object-Oriented Analysis Design / Brett D. McLaughlin, Gary Pollice & David West., – 2006
8. Setzer A. Java as a Functional Programming Language / Anton Setzer. // Lecture Notes in Computer Science. – 2013.
9. Гослинг Д., Джой, Б. Стил Г., та ін. Язык программирования Java SE 8. Подробное описание, 5-е издание / [Д. Гослинг, Б. Джой, Г. Стил та ін.]. – М: Вильямс, 2015. – 672 с.
10. Nudelman G. Android Design Patterns / Greg Nudelman., 2013.
11. Шилтд Г. Java. Полное руководство. 8-е издание / Герберт Шилдт – М: Вильямс, 2012. – 1104 с.
12. Muschko B. Gradle in Action 1st Edition / Benjamin Muschko., 2012.
13. SQLite Учебное пособие [Електронний ресурс] – Режим доступу до ресурсу: http://www.w3big.com/ru/sqlite/default.html.

14. Учебник по SQLite [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://coderlessons.com/tutorials/bazy-dannykh/vyuchit-sqlite/uchebnik-po-sqlite.

15. SQLite Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.sqlite.org/docs.html.

16. Kreibich J. A. Using SQLite / Jay A. Kreibich., 2010.

17. Burd B. Java Programming for Android Developers For Dummies / Barry Burd., 2016. – 456 с.

18. Späth P. Learn Java for Android Development: Migrating Java SE Programming Skills to Mobile Development / P. Späth, J. Friesen., 2020. – 850 с.

19. Horton J. Android Programming for Beginners: Build in-depth, full-featured Android apps starting from zero programming experience, 3rd Edition / John Horton., 2021. – 742 с. – (3). – (9781800566446, 1800566441).

20. Kunal D. A. Gradle Essentials / Dabir Abhinandan Kunal., 2015. – 176 с. – (1783982373).

21. Berglund T. Building and Testing with Gradle: Understanding Next-Generation Builds / T. Berglund, M. McCullough., 2011. – 116 с.

22. Pelgrims K. Gradle for Android / Kevin Pelgrims., 2015. – 172 с.

23. Комлев Н. Объектно-ориентированное программирование / Николай Комлев., 2020.

24. Гуськова О. Объектно-ориентированное программирование в Java / Ольга Гуськова., 2019.

25. Машнин Т. Объектно-ориентированное программирование на Java. Платформа Java SE / Тимур Машнин., 2021. – (5041888833).

26. Java: The Complete Reference, Eleventh Edition: Herbert Schildt - 2018.

27. Professional Android: Reto Meier, Ian Lake – 2018

28. Пирская Л. Разработка мобильных приложений в среде Android Studio / Любовь Пирская., 2021.

29. Dundar O. Expert Android Studio / O. Dundar, M. Yener., 2016.

30. AMC College. Android Studio: Apps Development / AMC College.