

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
магістр
(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСОБІВ
РОЗРОБЛЕННЯ АГРЕГАТОРІВ НА ПРИКЛАДІ ЗАСТОСУНКУ З
ЛОГІСТИКИ КОРПОРАТИВНОГО ХАРЧУВАННЯ
RESEARCH AND SOFTWARE IMPLEMENTATION OF THE
DESIGNING AGGREGATOR TOOLS ON THE CASE OF THE CORPORATE
CATERING LOGISTICS APPLICATION

Виконав: студент 2 курсу, групи КНТ-220м
Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Системи штучного інтелекту

Шаврін М.В.

(прізвище та ініціали)

Керівник Табунщик Г.В.

(прізвище та ініціали)

Рецензент Шитікова О.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 122 Комп'ютерні науки
 (код і найменування)
 Освітня програма (спеціалізація) Системи штучного інтелекту
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 20__ року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА(КИ)

Шавріна Микити Володимировича
 (прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Дослідження та програмна реалізація засобів розроблення агрегаторів на прикладі застосунку з логістики корпоративного харчування. Research and Software Implementation of the Designing Aggregator Tools on the case of the Corporate Catering Logistics Application
 керівник проекту (роботи) к.т.н., професор Табунщик Галина Володимирівна,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
 затверджені наказом закладу вищої освіти від “05” листопада 2021 року № 435
2. Строк подання студентом проекту (роботи) 01 грудень 2021 року
3. Вихідні дані до проекту (роботи) рекомендована література, технічне завдання
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Розробка архітектури програми. 4. Експлуатація, тестування та експериментальне дослідження програми. 5. Додатки
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Табунщик Г.В., професор		
Нормоконтроль	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання 29 вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	4–7 тижні	Розділ 2
4	Розробка архітектури програми.	8-9 тиждень	Розділ 3
5	Розробка програми.	10-11 тижні	Розділ 4
6	Оформлення пояснювальної записки та документів до неї.	12-13 тиждень	Додатки
7	Нормоконтроль та рецензування.	14–15 тижні	
8	Захист роботи.	16 тиждень	

Студент _____ Шаврін М.В.
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ Табунщик Г.В. _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
119 стр., 2 табл., 42 рис., 2 дод., 24 джерел.

ПРОГРАМНІ ЗАСОБИ, АРХІТЕКТУРА ПЗ, ОПЕРАЦІЙНА СИСТЕМА
ANDROID, МОБІЛЬНИЙ ЗАСТОСУНОК.

Об'єкт дослідження – процес забезпечення логістики корпоративного харчування.

Предмет дослідження – дослідження та реалізація засобів розроблення агрегаторів з логістики корпоративного харчування.

Мета роботи – покращення логістики корпоративного харчування за рахунок використання мобільного застосунка.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мови програмування Java та Kotlin, пристрій під керуванням ОС Android з мінімальною версією системи 5.0, персональний комп'ютер під управлінням операційної системи Ubuntu 18.04, середовище розробки Android Studio 3.

Результати. В роботі розроблено та протестовано мобільний застосунок працюючий на пристроях на базі ОС Android, що надає компанії, замовникам обідів у офіси, зручний інтерфейс для формування даних щодо доступних пропозицій.

Висновки. Були проаналізовані функціонал найбільш відомих сервісів надання послуг віддалених замовлень та проведено дослідження необхідності застосування, що агрегує дані постачальників обідів у офіси. Розроблено архітектуру застосування та розроблено систему "Corliver", що дозволить агрегувати дані від компаній, постачальників обідів.

Галузь використання – малий та середній бізнес.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 119 pages, 2 tables, 42 figures, 2 appendixes, 19 sources.

SOFTWARE, SOFTWARE ARCHITECTURE, ANDROID OPERATING SYSTEM, MOBILE TOOL.

The object of research is the process of providing corporate catering logistics.

Subject of research - research and implementation of tools for the development of aggregators for corporate catering logistics.

The purpose of the work is to improve the logistics of corporate food through the use of mobile applications.

Materials, methods and technical tools: object-oriented programming, programming languages Java and Kotlin, Android-based device with a minimum version of 5.0, personal computer running Ubuntu 18.04, Android Studio 3 development environment.

Results. Based on the analyzed information, the relevant conclusions were drawn from the tools for the organization of aggregator systems and developed a mobile application running on devices based on Android.

Conclusions. The most acceptable means for the organization of the aggregator system in the field of food are analyzed and determined.

Scope - development of systems for data aggregation in the field of corporate nutrition for small and medium-sized businesses.

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК.....	7
ВСТУП.....	8
1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ	9
1.1 Постановка задачі	9
1.2 Огляд існуючих агрегаторів	11
1.3 Висновок до першого розділу	17
2 МАТЕРІАЛИ І МЕТОДИ.....	20
2.1 Обґрунтування вибору архітектурного шаблону для розробки мобільного застосунку.....	20
2.2 Вибір мови програмування для розробки мобільного застосунку	26
2.3 Вибір середовища розробки	28
3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ.....	29
3.1 Проектування архітектури мобільного застосунку.....	30
3.2 Проектування схеми бази даних	34
3.3 Опис клієнтської частини	35
3.4 Опис серверної частини	36
3.5 Висновок до третього розділу	37
4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ.....	38
4.1 Функціональні можливості системи.....	42
4.2 Розробка тестів.....	42
4.3 Висновок до четвертого розділу	47
ВИСНОВКИ	48
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	49
ДОДАТОК А Текст програми	52
ДОДАТОК Б Слайди презентації.....	112

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- ПЗ – Програмне забезпечення;
- ОС – Операційна система;
- MVP – Model-View-Presenter (укр. Модель–Представлення–Пред'явник);
- MVC Model-view-controller (укр. Модель–вигляд–контролер);
- MVVM – Model-View-ViewModel (укр. Модель – Представлення – Представлення модель);
- MVI – Model-View-Intent (укр. Модель – Представлення – Намір)
- DI – Dependency injection (укр. Впровадження залежності);
- HTTP – Hyper Text Transfer Protocol (укр. протокол передачі гіпертексту);
- JSON – JavaScript Object Notation (укр. запис об'єктів JavaScript);
- XML - Extensible Markup Language (укр. розширяєма мова розмітки)
- SDK – Software Development Kit (укр. набір з засобів розробки програмного коду)
- API – Application Programming Interface (укр. програмний інтерфейс програми)
- JVM – Java Virtual Machine (укр. віртуальна машина Java)
- REST - Representational State Transfer (укр. передача репрезентативного стану)
- UML - Unified Modeling Language (укр. уніфікована мова моделювання)
- AJAX - Asynchronous Javascript and XML (укр. асинхронний JavaScript та XML)

ВСТУП

На сьогодні кількість розробників та компаній стрімко ростуть. Лише у галузі інформаційних технологій щорічно з'являється приблизно 50 тисяч нових робочих місць а близько 25% відсотків розробників мають оплачувані обіди у офісах. Ринок замовлень їжі стрімко росте, а популярність агрегаторів їжі збільшується з початку пандемії. Лише у 2020 році кількість замовлень через агрегатори їжі зросло на 75%, а нові компанії, такі як “Ракета” виросли на 500%.

Проте досі ринок корпоративного харчування перебуває у занепаді через відсутність відповідних систем, де компанії можуть замовити обіди для своїх співробітників.

Більш того, за результатами власних досліджень було виявлено, що к кінцю 2022 року приблизно 40-50% компаній планують ввести корпоративне замовлення обідів як один з механізмів повернення розробників до офісів, та використовувати це як бонус задля залучення нових.

Актуальність роботи полягає в аналізі сучасних рішень для створення системи агрегатору з логістики корпоративного харчування, що допоможе компаніям у реалізації планів з корпоративного замовлення обідів для співробітників.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

На сьогодні мобільні застосунки замінюють застарілі програми для персональних комп'ютерів, адже на сьогодні майже усі мають мобільний телефон з собою, та стають більш доступнішим аніж веб сторінки, тому що більшість з них мають функціонал для роботи без інтернету.

Лише за останні 5 років кількість власників мобільних телефонів збільшилась на 1 мільярд (до 5 мільярдів у 2019 році), а у 2018 році 63% населення Землі вже мали мобільні телефони. Протягом наступних років ця кількість зросте, адже за прогнозами у 2022 році кількість користувачів мобільних телефонів зросте до 5.5 мільярдів.

Багато компаній сьогодні відмовляються від звичайних сайтів та програм для персональних комп'ютерів, надаючи перевагу до мобільних застосунків, задля доступніших послуг, що важливо для залучення нових клієнтів. А завдяки крос-платформеній розробці багато стартапів обирають шлях розробки тільки мобільних застосунків.

Мобільні застосунки мають багато переваг. Перш за все вони доступніші ніж веб сторінки, адже встановлюються на мобільні пристрої та завжди доступні для користувача. Більш того, багато з них мають функціонал для роботи без інтернету.

1.1 Постановка задачі

У сучасному світі, котрий постраждав внаслідок пандемії, особливо популярні стали різноманітні сервіси доставки їжі [1].

Лише за 2020 рік ринок доставки їжі в Україні виріс у 6-7 разів та продовжує зростання. Як відзначають у компанії Rocket, навіть після послаблення карантину, ріст користувачів продовжує зростання, адже замовлення їжі додому стало звичним та корисним інструментом замовлення продуктів та блюд з ресторанів.

Проте з послабленням карантину, багато компаній почали повертати своїх співробітників до офісів, адже віддалена робота не тільки знижує ефективність роботи, а й стає причиною вигорання. Багато людей не вміють ефективно розподілити час роботи та страждають на переробітки, а відчуття самотності негативно позначаються на роботі [2].

Тому багато компаній намагаються безболісно повернути своїх працівників у офіси за допомогою різноманітних заохочувань. Однією з головних напрямів у цьому є оплачувані обіди для співробітників. За проведеними власними дослідженнями, близько 70% HR-менеджерів та власників ІТ-компаній планують використовувати цей бонус для своїх співробітників. Проте, наразі на ринку України немає єдиного сервісу котрий міг би об'єднати різні заклади у єдиний сервіс націлений на оптимізацію процесів замовлення обідів у офіси.

Станом на 2021 рік у сфері інформаційних технологій працює більше 250 тис програмістів [3], а за результати власного дослідження у котрому прийняло 200 розробників із різних компаній, 25% вже мають повністю або частково оплачувані обіди (рис. 1.1). Проте взаємодія між компаніями та закладами відбувається у не зручній формі, заповнюючі таблиці замовлень, де легко помилитись, або випадково видалити замовлення колеги.

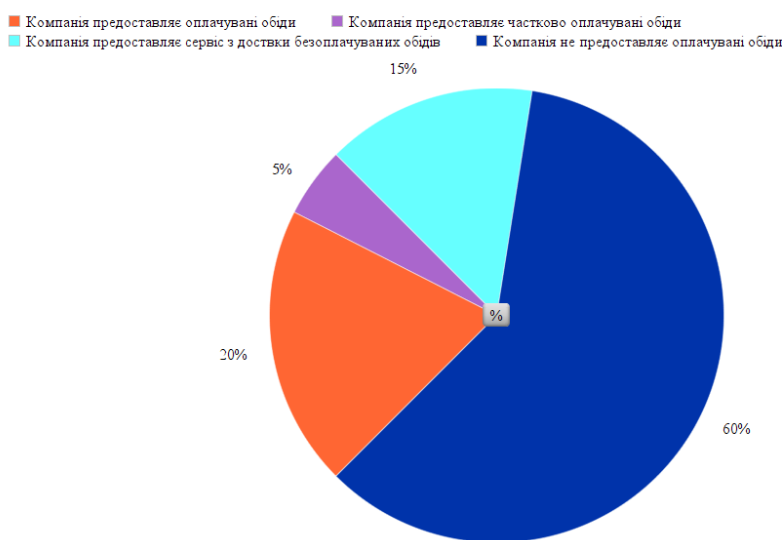


Рисунок 1.1 – Графічне представлення результатів опитування розробників

Отже на сьогодні в Україні замовлення обідів у офіси набуває популярності, проте наразі немає відповідних сервісів для оптимізації процесів замовлення їжі та логістики пов'язаною з цим.

1.2 Огляд існуючих агрегаторів

Агрегатор - сервіс, який збирає дані з різних джерел (найчастіше - інформацію про товари та послуги різних компаній) або ж з'єднує замовників та виконавців послуг. Він є каталогом, де можна знайти потрібні позиції за певними характеристиками і порівняти їх між собою. Якщо людина купує товар чи послугу цьому сайті, то агрегатор отримує від продавця комісію [4].

З'являється все більше гнучких, швидких і зручних систем, які допомагають в роботі закладам харчування. При цьому розвиток інтернетсервісів, а також популярність мобільних застосунків призводить до того, що їх кількість вже перевищує кількість окремо встановлених на комп'ютері програм. З плином часу та розвитком ІТ технологій, звичайна людина все більше має можливості економити свій час не витрачаючи його на речі, які можна робити віддалено і з найменшим втручанням інших людей.

Серед найбільш відомих сервісів надання послуг віддалених замовлень в закладах харчування можна виділити: Grubhub, Notch, eatfirst. В даному пункті буду розглянуті лише декілька, найбільш відомих кожний в своєму напрямлені.

1.2.1 Сервіс Grubhub

Grubhub - сервіс по доставці їжі у Англії та США. Окрім замовлення їжі до дому, вони надають функціонал по замовленню обідів у офіси від компаній-партнерів. Серед клієнтів цього сервісу є такі компанії як HP, IBM, appzen та інші. Більш того вона має приблизно 10 тис. працівників.

Сервіс дозволяє обрати ресторан за вподобою, використовуючи фільтрацію за національністю кухні та за тематикою закладів (м'ясні, веганські та інші).

Станом на 2021 рік бізнес-модель стартапу передбачає надання двох видів сервісу:

- для ресторанів - можливість пошуку нових клієнтів, власне програмне забезпечення по веденню бізнесу по доставці їжі;
- для компаній - можливість як маленьким так і великим компаніям оплачувати повністю або частково вартість страв у ресторанах для своїх співробітників через версію ПО Grubhub.

Користувачський інтерфейс застосунку представлений на рисунку 1.2.

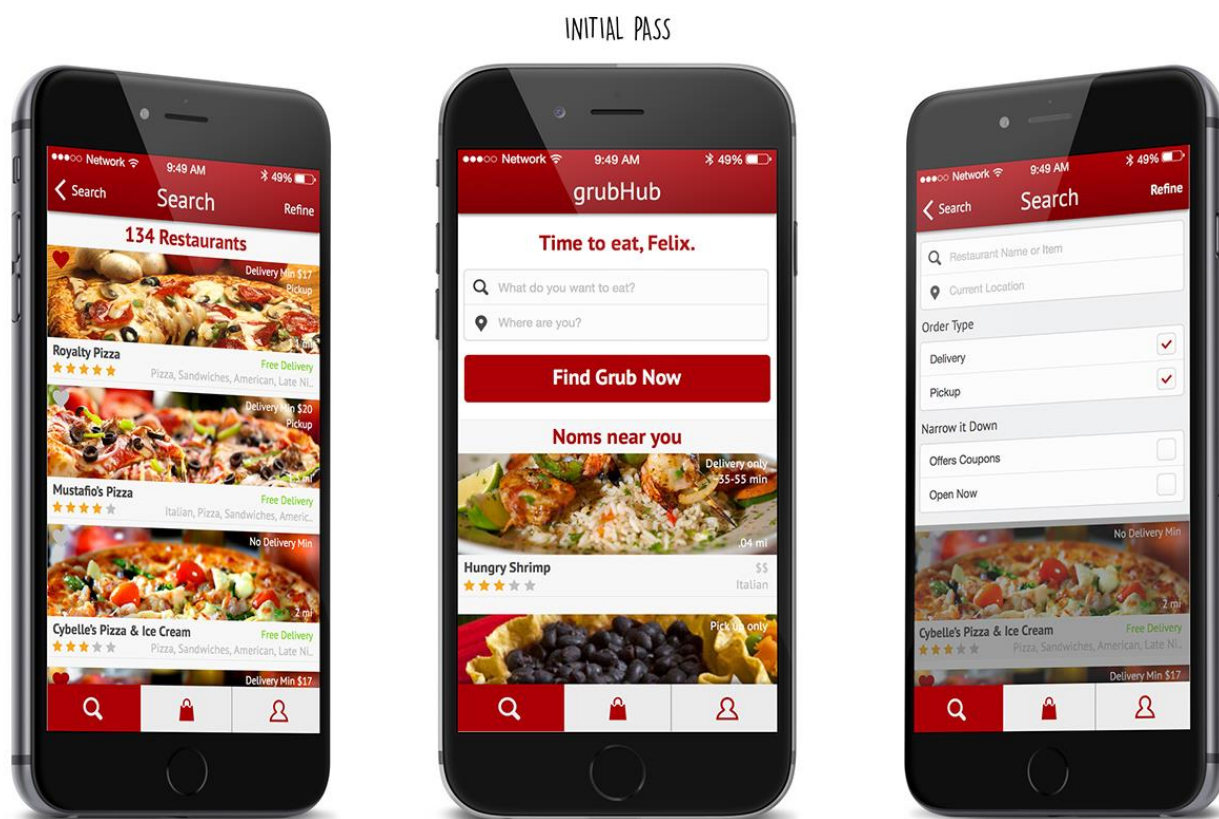


Рисунок 1.2 – Приклад інтерфейсу сервісу Grubhub [21]

Сервіс має багато переваг:

- мобільний та веб за стосунки;

- корпоративна версія сервісу;
- власний штат кур'єрів;
- фільтрація ресторанів за типами кухні;
- можливість залишати відгуки.

Серед недоліків:

- відсутність механізму відслідковування статусу замовлення;
- відсутність зворотнього зв'язку між клієнтами і закладами;
- відсутність функціоналу, який би «радив» заклади найближчі до місця знаходження клієнта на основі його вподобань;
- сервіс не працює з закладами України.

1.2.2 Сервіс Notch

Notch - сервіс по замовленню їжі з місцевих фермерських ринків. Компанія заснована у 2015 та працює на території Канади. Окрім індивідуального замовлення їжі, сервіс надає функціонал з замовлення свіжих фруктів та овочів у офіси задля забезпечення смачних та корисних перекусів для співробітників.

Також вони надають сервіс із замовлення їжі у якості бібліотек, котрі можна інтегрувати з веб сторінками ресторану, якщо заклад вже має власну логістику.

Сервіс підтримує веб версію, мобільні застосунки для платформ iOS та Android, а також програма для персональних комп'ютерів.

Серед переваг:

- великий вибір продуктів з місцевих ферм;
- приємний та зрозумілий дизайн;
- відстеження місцезнаходження замовлення;
- засоби комунікації з кур'єрами та закладами.

Серед недоліків:

- не працює в Україні;

- відсутність функціоналу, який би «радив» заклади найближчі до місця знаходження клієнта на основі його вподобань;
- розділення закладів за кухнею та типами їжі.

Користувачський інтерфейс застосунку представлений на рисунку 1.3.

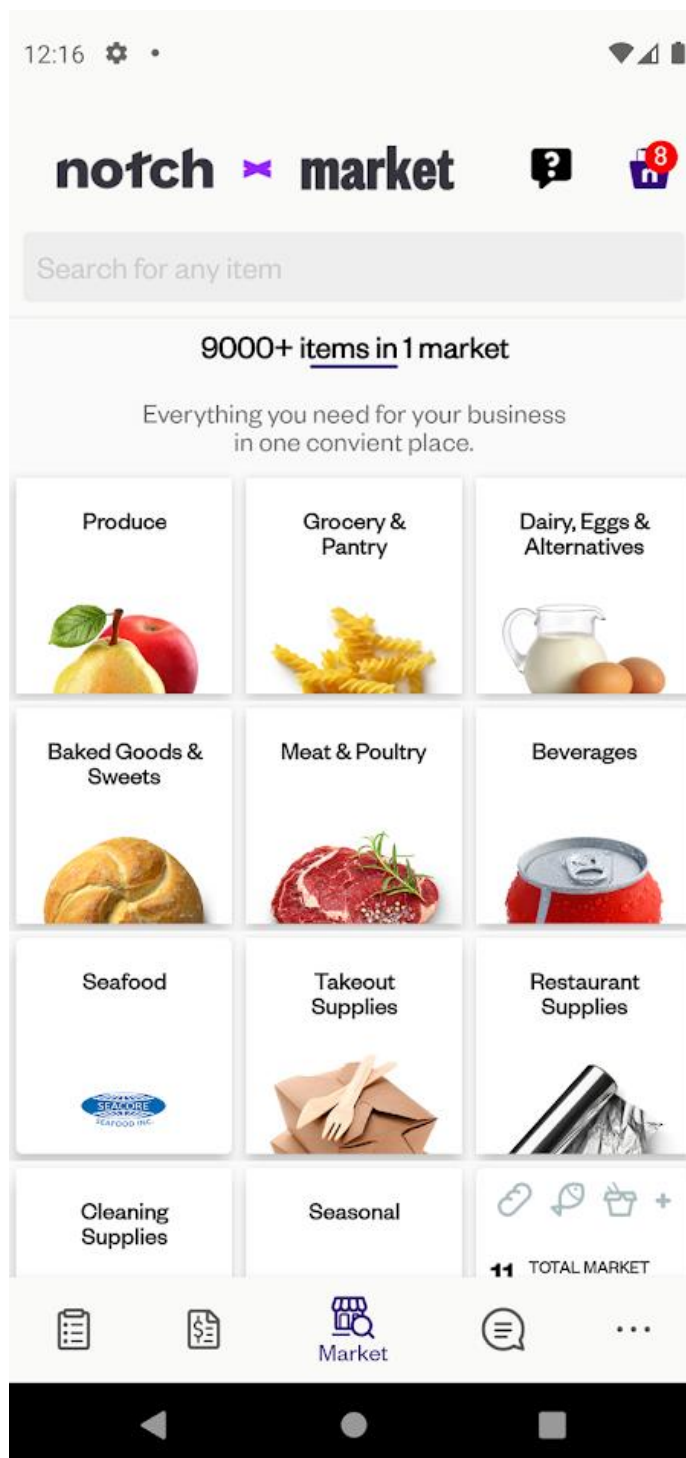


Рисунок 1.3 – Приклад інтерфейсу сервісу Notch [22]

1.2.3 Сервіс EatFirst

EatFirst - сервіс з кейтерінгу здорового харчування, який працює у Австралії, Германії, Англії та Нідерландах. Сервіс працює з такими компаніями як Amazon, Facebook, Microsoft та Sony (рис. 1.4).

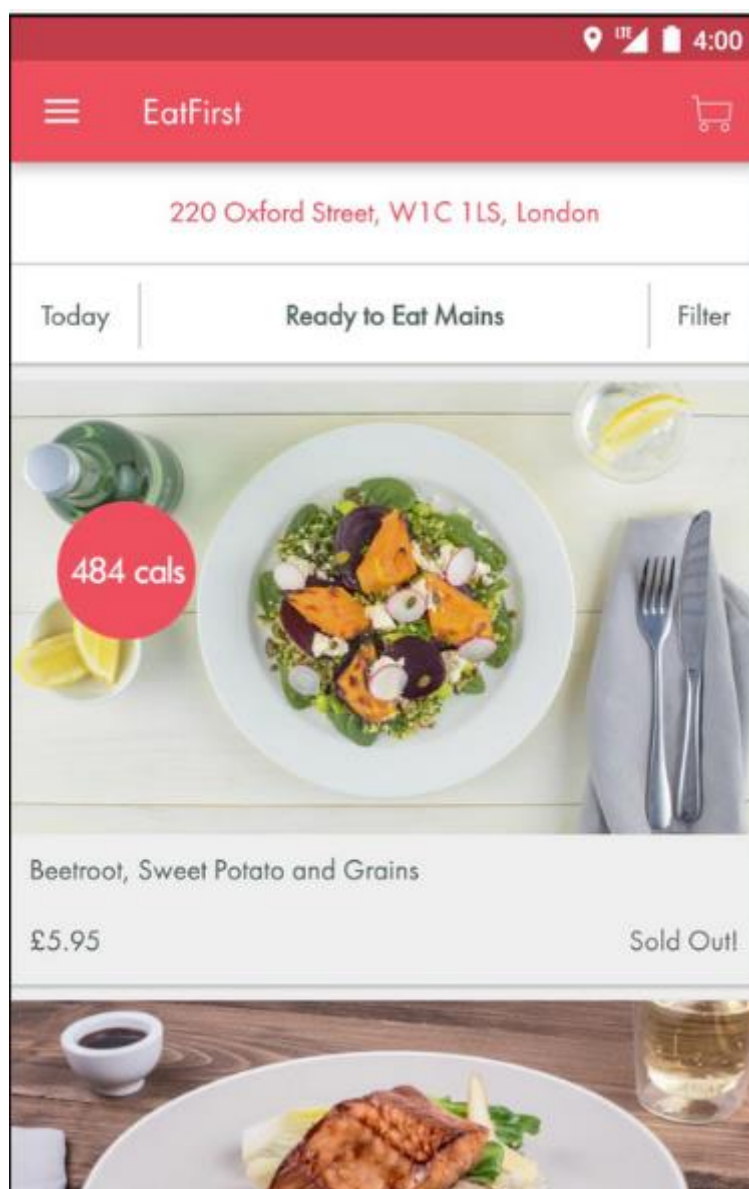


Рисунок 1.4 – Приклад інтерфейсу сервісу EatFirst [23]

Основний напрямок компанії - організація фуршетів та організація їжі для перекусу в офісах.

Також з початку пандемії вони ввели функціонал замовлення обідів для співробітників до дому.

Сервіс має такі переваги:

- мобільний та веб за стосунки;
- корпоративна версія сервісу;
- напрямок у сфері здорового харчування.

Серед недоліків:

- відсутність механізму відслідковування статусу замовлення;
- відсутність зворотнього зв'язку між клієнтами і закладами;
- відсутність функціоналу, який би «радив» заклади найближчі до місця знаходження клієнта на основі його вподобань;
- сервіс не працює з закладами України.

Результати порівняння існуючих агрегаторів приведені у таблиці 1.1.

Таблиця 1.1 - Порівняння існуючих агрегаторів

Назва	Переваги	Недоліки
1	2	3
Grubhub	- мобільний та веб за стосунки; - корпоративна версія сервісу; - власний штат кур'єрів; - фільтрація ресторанів за типами кухні; - можливість залишати відгуки.	- відсутність механізму відслідковування статусу замовлення; - відсутність зворотнього зв'язку між клієнтами і закладами; - відсутність функціоналу, який би «радив» заклади найближчі до місця знаходження клієнта на основі його вподобань; - сервіс не працює з закладами України.

Продовження табл.1.1

1	2	3
Notch	- великий вибір продуктів з місцевих ферм; - приємний та зрозумілий дизайн; - відстеження місцезнаходження замовлення; - засоби комунікації з кур'єрами та закладами.	- не працює в Україні; - відсутність функціоналу, який би «радив» заклади найближчі до місця знаходження клієнта на основі його вподобань; - розділення закладів за кухнею та типами їжі.
Eatfirst	- мобільний та веб застосунки; - корпоративна версія сервісу; - напрямок у сфері здорового харчування.	- відсутність механізму відслідковування статусу замовлення; - відсутність зворотнього зв'язку між клієнтами і закладами; - відсутність функціоналу, який би «радив» заклади найближчі до місця знаходження клієнта на основі його вподобань; - сервіс не працює з закладами України.

1.3 Висновок до першого розділу

В данному розділі було створено огляд аналогів сервісів із подібним функціоналом. Було порівняно функціональність цих продуктів та виявлені системи зі схожим функціоналом, проте вони не охоплюють цілі та функції запропоновані в дипломному проекті.

Також на основі отриманих та проаналізованих даних була побудована Канвас бізнес-моделі проекту (рис 1.5).



Рисунок 1.5 – Канва бізнес-моделі

2 МАТЕРІАЛИ І МЕТОДИ

Перш ніж приступити до розробки програми необхідно проаналізувати майбутню структуру проекту та спроектувати майбутню архітектуру мобільного застосунку та схему бази даних задля ефективної розробки та реалізації застосунку.

При розробці програми необхідно продумати її архітектуру, обрати мову розробки. На сьогодні склались певні норми розробки програм, проте досі не існує єдиного стандарту, а велика кількість шаблонів та інструментів дозволяють розробляти застосунки у різні шляхи. Нижче зроблено їх аналіз задля ефективної розробці системи з доставки замовлень.

2.1 Обґрунтування вибору архітектурного шаблону для розробки мобільного застосунку

Із ростом проекту, з'являється необхідність розбивання проекту на частини за шаблонами проектування, котрі допомагають спростити розуміння та покращує комунікацію між усіма членами команди та, посилаючись на стардартні конструкції знижає кількість помилок.

Розглядаючи архітектурні патерни проектування при розробці застосунків під ОС Андроїд, слід виділити такі популярні як MVC (Model-View-Controller), MVP (Model-View-Presenter), MVVM (Model-View-ViewModel), MVI (Model-View-Intent).

2.1.1 Паттерн MVC

Для початку необхідно розглянути базовий паттерн MVC. Концепція цього патерну була описана Трюгве Реєнскаугом у 1978 році [5]. Вперше він був реалізований для мови SmallTalk, а кінцева версія паттерну була оприлюднена у 1988 році у журналі “Technology Object” [6]. Перед

розробниками була необхідність винайти архітектурне рішення, котре дозволило би відділити графічний інтерфейс від бізнес логіки, а бізнес логіку від даних. На сьогодні цей патерн є одним із найпопулярнішим у багатьох технологіях, дав розвиток новим шаблонам і став у основі великої кількості програмних рішень. У класичному варіанті, цей шаблон поділяється на три частини: Модель (Model), Представлення (View), Контролер (Controller). Схема шаблону вказана на рис. 2.1.

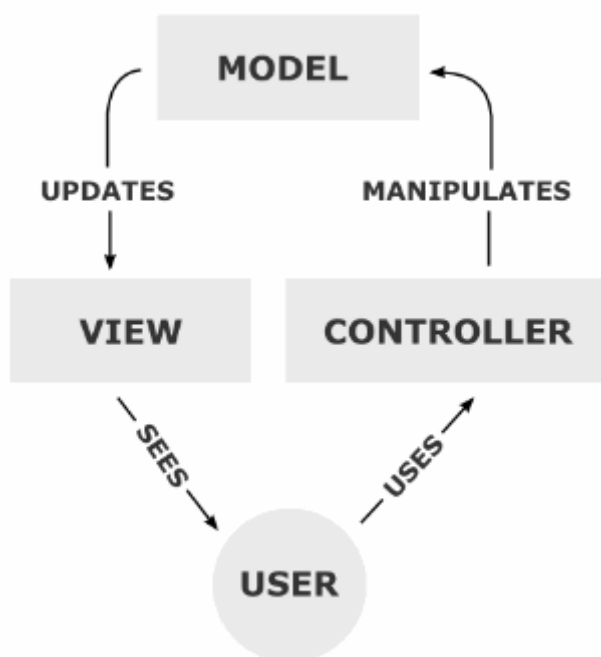


Рисунок 2.1 – Схема MVC [7]

Під моделлю, зазвичай, мають на увазі основний функціонал бізнес логіки, котра є повністю незалежною від інших частин програми. Вона має наступними ознаками:

- модель - бізнес логіка програми;
- модель знає лише про себе та не має доступу та знань про контролери чи представлення;
- іноді модель є лише шар даних, наприклад база даних;
- іноді під моделлю мають на увазі клас-менеджер роботи з даними, набір об'єктів чи просто логіка програми.

Представлення є тією частиною паттерна, котра відповідає за відображення даних з моделі у користувацькому інтерфейсі. Але вона не впливає на модель та її дані. Вона лише читає їх.

Найголовніша ознака представлення є реалізація відображення даних із моделі у користувацькому інтерфейсі (або у різних його інтерпретаціях), наприклад HTML-сторінка, або вікно програми у мобільному застосунку.

Контролер відстежує зміни пов'язані з діями користувача та відповідно цим змінам він впливає на модель.

Відокремлення моделі від представлення дозволяє перевикористовувати різні компоненти для відображення інформації. Зміна даних моделі через контролер буде впливати на один або декілька візуальних компонентів та автоматично виправить інформацію відповідно до змін.

У світі андроїда цей підхід недостатньо відповідає особливостям системи. Базовий компонент відображення користувацького інтерфейсу Activity не можна трактувати як представлення. Це шар відображення, а тому для правильного будування паттерну MVC треба виділити елементи інтерфейсу у Activity та для них вже зробити свої контролери, що призведе до значного ускладнення розуміння коду, адже окрім відображення треба буде ще й пам'ятати про життєвий цикл, та ускладнить тестування цього коду.

Рішення цієї проблеми полягає у створенні абстрактного інтерфейсу для представлення. Таким чином контролер буде працювати за нею, а не самим представленням.

Усе це і є головна ідея MVP.

2.1.2 Паттерн MVP

MVP є похідним паттерном від MVC, але на відміну від нього, пред'явник комунікує з представленням через інтерфейс, та представлення нічого не знає про модель. З моделлю працює тільки пред'явник, а на основі отриманих даних він впливає на представлення. У свою чергу, представлення,

реагуючі на дії користувача, впливає на пред'явника. Схема шаблону вказана на рис. 2.2.

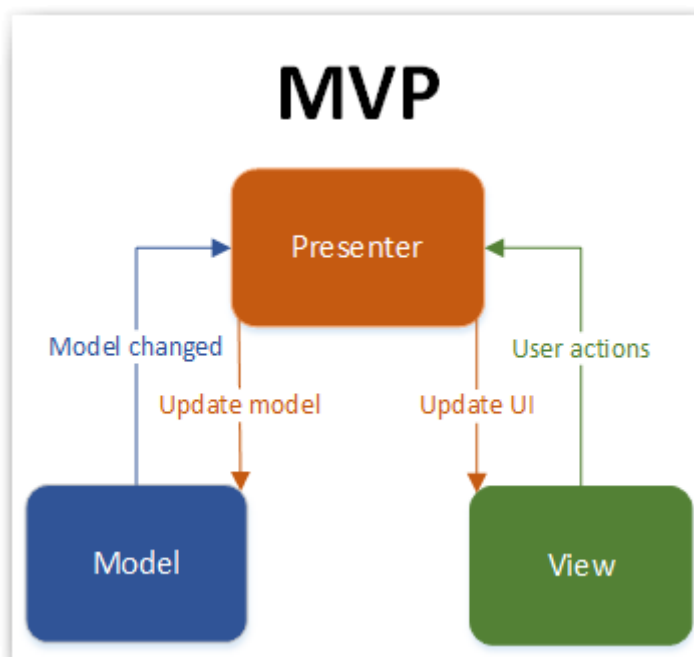


Рисунок 2.2 – Схема MVP [8]

Розмежувачі таким чином шари та комунікації через інтерфейси, досягається велика ступінь перевикористовування пред'явника та спрощується його тестування.

Цей патерн був одним з найпопулярніших у світі Андроїд розробки. Навіть були написані фреймворки (Моху [9], Mosby [10]) для написання застосунків на основі нього, котрі спрощували будовання шаблону у застосунку.

Проте у цього підходу є недолік. Через багату кількість інтерфейсів та їх перевикористання, з'являється необхідність підтримувати інтерфейси представлень, що ускладнює роботу з підтримки та додає зайвий шаблонний код.

2.1.3 Паттерн MVVM

Питаючись спростити шаблони, Джон Госсман у 2005 році представив шаблон MVVM, як модифікація шаблону MVP, де інтерфейси взаємодії представлення та презентера було замінено на зв'язування (біндінг) даних.

Представлення підписується на зміни даних, та коли вони міняються у моделі, представлення міняє їх у себе. Модель представлення набагато легше тестувати, завдяки цьому, а код стає більш незалежним. Схема шаблону вказана на рис. 2.3.

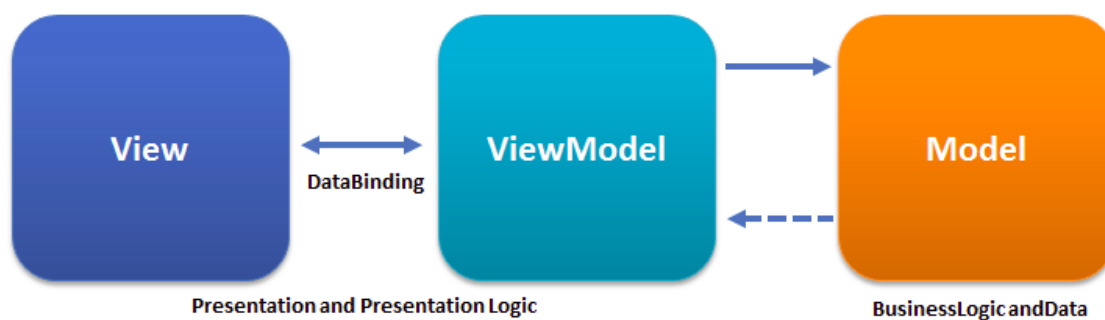


Рисунок 2.3 – Схема MVVM [11]

У 2017 році, корпорація Google випустила набір бібліотек під назвою “Android Architecture Components”, де представила базовий клас ViewModel для спрощення створення застосунків за цим архітектурним патерном. Разом з ними було представлено класи LiveData та Lifecycle, котрі разом із класом ViewModel створили неперевершений механізм організації архітектури застосунків, котрий вмів не тільки розділяти логіку, а й ще зав’язувався на життєвому циклі Андроїд застосунків. Цю трійку класів було дуже тепло прийнято спільнотою розробників і більшість нових проектів під Андроїд почали розроблятися згідно патерну MVVM, за допомогою цих архітектурних компонентів.

Також було представлено бібліотеку для зв’язування ViewModel із користувацьким інтерфейсом: DataBinding. Проте, не дивлячись на те що він

мав би бути частиною MVVM, має багато недоліків і для складних проектів вимагав писати складний код у розмітці екрану, що погіршило налагодження коду та його тестування. Тому багато розробників відмовились від його використання.

2.1.4 Паттерн MVI

На сьогодні використання патерну MVVM разом із бібліотеками архітектурних компонентів є найпоширюваним рішенням при розробці Андроїд застосунків. Проте, разом із цим, набуває популярність патерн MVI, котрий заснований на парадигмі реактивного програмування, та базується на бібліотеці RxJava. Цей паттерн популяризує у своїх докладах компанія Badoo.

MVI відносно новий патерн, котрий базується на принципах MVC та TEA (The Elm Architecture) з поєднанням парадигми реактивного програмування. У цьому патерні всі елементи користувацького інтерфейсу поділяються на ті що створюють події (Observable), та ті, що тільки щось відображають (Consumer). Також існують елементи, які роблять і те і інше. Такий елемент одночасно є і творцем (Producer) і приймачем (Receiver), а в парадигмі реактивного програмування він називається Subject. При такому підході - все є потік, і кожен потік починається з того моменту, як будь-якої producer, починає створювати інформацію, а закінчується на будь-якому receiver, який, в свою чергу, інформацію приймає. Як результат, застосунок можна розглядати як потоки даних. Потоки даних - головна ідея парадигми реактивного програмування і RxJava зокрема.

Проте у MVI є дуже вагомий мінус. У Андроїді він базується на бібліотеці RxJava та парадигми реактивного програмування, що робить це рішення обмежувачим. З розвитком мови програмування Kotlin з її сильним механізмом многопоточного програмування, та складності реалізації самого патерну, робить її не дуже привабливим патерном для розробки застосунків. Схема шаблону вказана на рис. 2.1.

Вище наведені основні архітектурні патерни при проектуванні системи для застосунків під ОС Android. Вони мають як недоліки, так і переваги перед іншими. У таблиці 2.1 наведено порівняння цих патернів.

Таблиця 2.1 – Порівняння архітектурних патернів

Характеристика	MVC	MVP	MVVM	MVI
Підтримка системою Android	-	+-	+	-
Робота з життєвим циклом Android	-	-	+	+-
Підтримка спільнотою розробників під Android	-	+	+	+-
Простота у реалізації	+-	+	+-	-
Простота у використанні у системі Android	-	+-	+-	+-

Зважаючи на таблицю 2.1 можна зробити висновки що MVVM є найбільш прийнятним при виборі архітектурного патерну при розробці застосунків под ОС Android.

2.2 Вибір мови програмування для розробки мобільного застосунку

На сьогодні, при розробці ПЗ для пристроїв на базі ОС Андроїд виділяють дві основні мови: Java та Kotlin.

2.2.1 Java

Java - об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи [13].

При розробці застосунків під ОС Андроїд можна зіткнутись з проблемою, що остання версія Java на якій можна розробляти застосунки – Java 8, що є застарілою версією. Більш того, Java 8 підтримується лише на останніх версіях ОС Android. Якщо потрібно зворотна сумісність із більш

старими версіями (наприклад термінали на ОС Android використовують більш застарілі версії системи), потрібно використовувати Java 7.

Але за багатолітню історію ОС Android біло написано велику кількість бібліотек саме на цій мові, що робить її невід’ємною частиною усіх програм.

2.2.2 Kotlin

Kotlin - статично типізована мова програмування, що працює поверх JVM і розробляється компанією JetBrains з 2010 року, а публічно представлена у 2011 році.

Автори ставили перед собою ціль створити лаконічнішу та типобезпечнішу мову, ніж Java [14]. Також розробники ставили своєю ціллю зробити зворотну сумісність із Java, щоб код на цих мовах міг безпечно викликати один одного. Також ця мова запровадила багато потужних механізмів многопоточності, котрі працюють набагато ефективніше ніж потоки у Java.

Також, на основі досвіду Java, Kotlin впровадив більш ефективніший механізм оновлень версії мови, через що незалежно від версії ОС Android можливо використовувати останню версію Kotlin.

2.2.3 Порівняння мов програмування

Отже Kotlin є більш сучасною мовою, більш лаконічна та типобезпечніша за Java. Більше того, наразі актуальна версія Java є 12’та, але остання версія ОС Андроїд працює лише з 8-ою версією, котра була випущена у 2014 році, що робить неможливим користування останніми оновленнями мови, а Kotlin, у той час, завдяки кращому механізму зворотній сумісності, можна використовувати останню версію мови без ускладнень. Через це Kotlin набрав більшу популярність при розробці ПЗ під ОС Андроїд. Більше того,

Kotlin має зворотну сумісність із Java, тобто можливо викликати код Java у Kotlin, та навпаки [15].

У таблиці 2.2 наведено порівняння цих мов у контексті розробки застосунків під ОС Android.

Отже, виходячи із результатів аналізу цих двох мов, Kotlin є більш пріоритетною мовою при розробці застосунків під ОС Android.

Таблиця 2.2 – Порівняння Java та Kotlin за функціоналом

Функціонал	Java (8)	Kotlin
Лаконічність	-	+
Зворотна сумісність	-	+
Офіційна підтримка Google	+	+
Підтримка спільнотою	+-	+
Продуктивність роботи	+-	+

2.3 Вибір середовища розробки

Середовище розробки Android Studio 3.6 було обрано для роботи у рамках дипломної роботи з метою реалізації мобільного застосунку.

Головні переваги цього середовища [16]:

- офіційна підтримка Google;
- швидкий та багатий на функціонал емулятор;
- розумний редактор коду;
- готові шаблони компонентів Android API;
- широкий функціонал тестування коду;
- графічний редактор для створення користувацького інтерфейсу;
- підтримка систем контролю кода;
- підтримка усіх додаткових засобів для розробки від Google;
- підтримка мов програмування як Java, так і Kotlin.

3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

Система замовлень буде складатись з мобільного застосунку та серверної частини.

На сьогодні найпопулярніші формати з обміну даних є JSON та XML.

XML - запропонований консорціумом World Wide Web Consortium (W3C) стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками [17]. Він був створений у 1998 році як заміна SGML розробленою у 1960-х роках. Стандарт XML визначає набір базових лексичних та синтаксичних правил для побудови мови описання інформації шляхом застосування простих тегів. Цей формат достатньо гнучкий для того, аби бути придатним для застосування в різних галузях. Іншими словами, запропонований стандарт визначає метамову, на основі якої шляхом запровадження обмежень на структуру та зміст документів визначаються специфічні, предметно-орієнтовані мови розмітки даних. Ці обмеження описуються мовами схем, такими як XML Schema (XSD), DTD або RELAX NG [17].

JSON - текстовий формат з обміну даними створений у 2009 як частина мови JavaScript, проте на сьогоднішній день роботу з цим форматом підтримують майже усі сучасні мови програмування. Його популярність пояснюється простим форматом, котрий легко як читати розробнику, так й легкість створювання парсерів JSON схеми. За рахунок своєї лаконічності в порівнянні з XML, формат JSON може бути більш придатним для серіалізації складних структур. Якщо говорити про веб-застосунки, у такому ключі він доречний у задачах обміну даними як між браузером і сервером (AJAX), так і між самими серверами (програмні HTTP-інтерфейси). Формат JSON так само добре підходить для зберігання складних динамічних структур у реляційних базах даних або файловому кеші [18].

Через простоту використання та через велике поширення було обрано використання JSON Schema як стандарт обміну даними між сервером та мобільним застосунком.

3.1 Проектування архітектури мобільного застосунку

Перш за все необхідно продумати та спроектувати структуру мобільного застосунку.

Основні екрани будуть:

- реєстрація;
- авторизація;
- основна сторінка;
- сторінка інформації меню компанії;
- екран страви з можливістю додання її до замовлення;
- екран замовлення;
- екран історії замовлень.

На основі сформованих вимог, була спроектована блок-схема функціоналу системи (рис. 3.1).

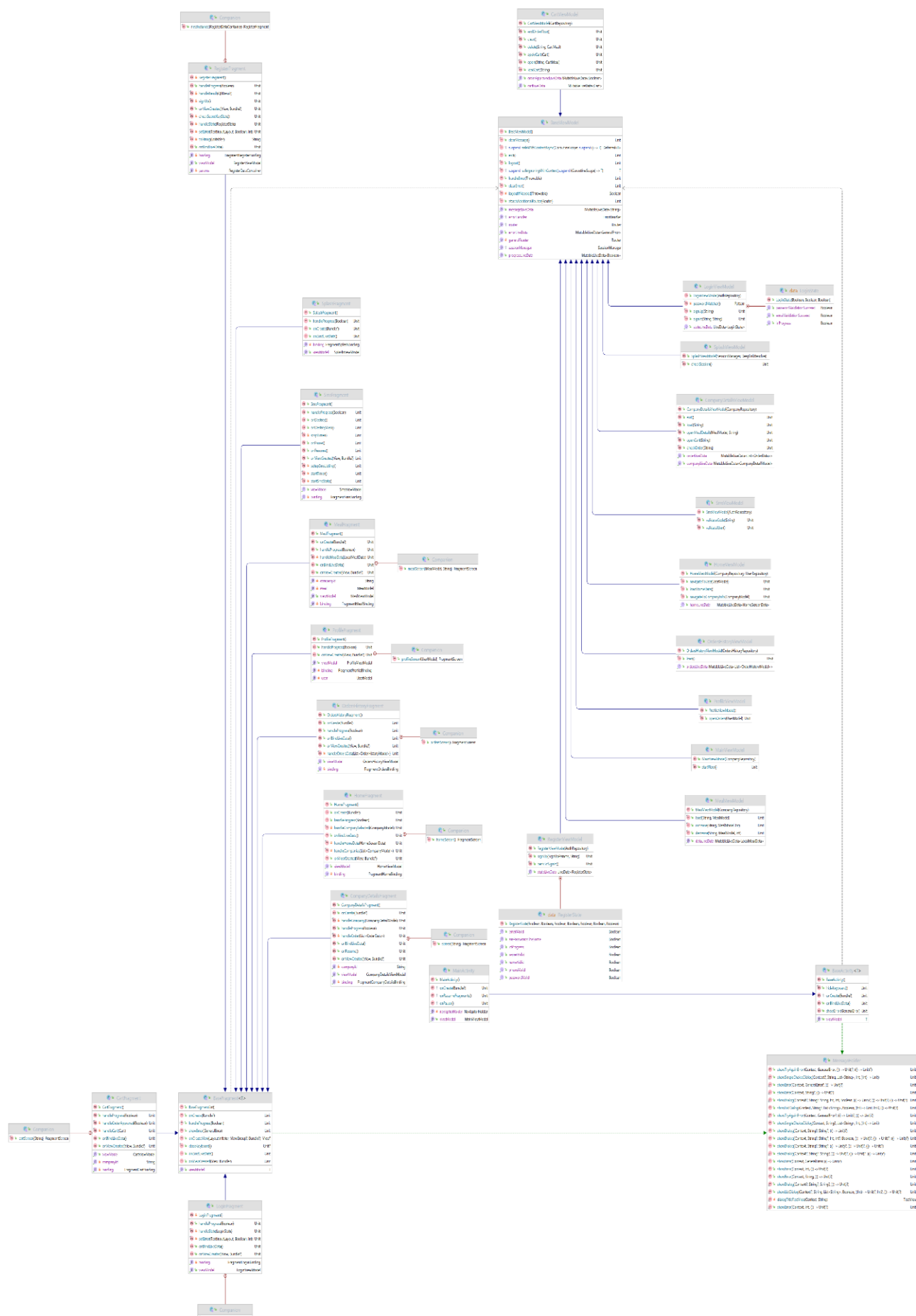


Рисунок 3.2 – UML діаграма класів основного функціоналу

Далі була розроблена UML діаграма класів моделей (рис. 3.3).

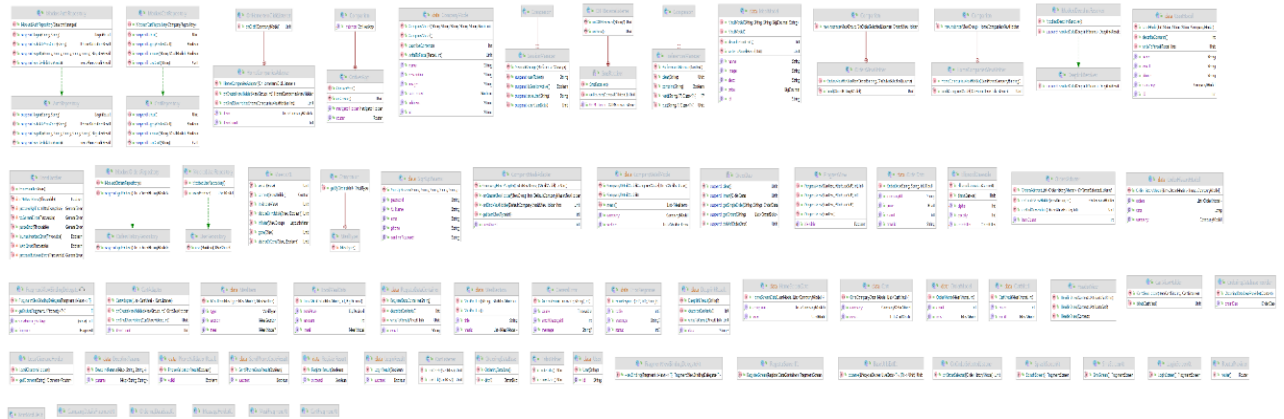


Рисунок 3.3 – UML діаграма класів моделей

На завершення була створена UML діаграма валідації даних (рис. 3.4).

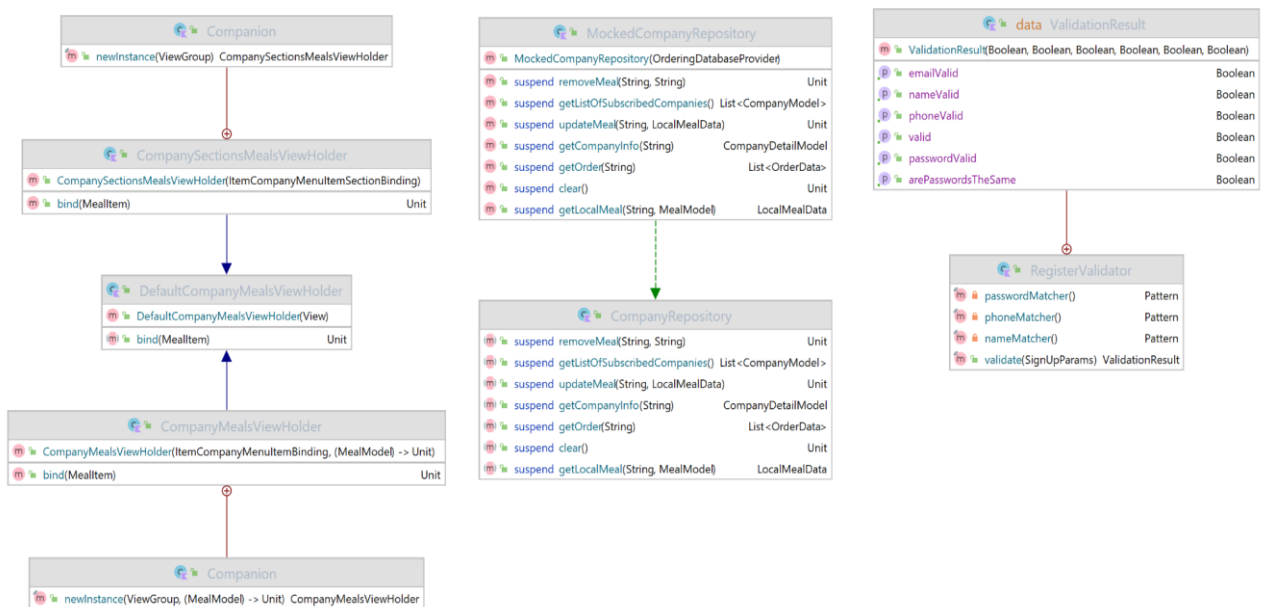


Рисунок 3.4 – UML діаграма класів валідації даних

3.2 Проектування схеми бази даних

При розробці застосунку з замовленням їжі необхідно спроектувати модель бази даних з взаємодією між замовником та сервісом з доставки.

Перш за все можна виділити основні сутності:

- модель користувача;
- модель замовлення;
- модель транзакції;
- модель страви.

На рис. 3.5 можна побачити схему бази даних.

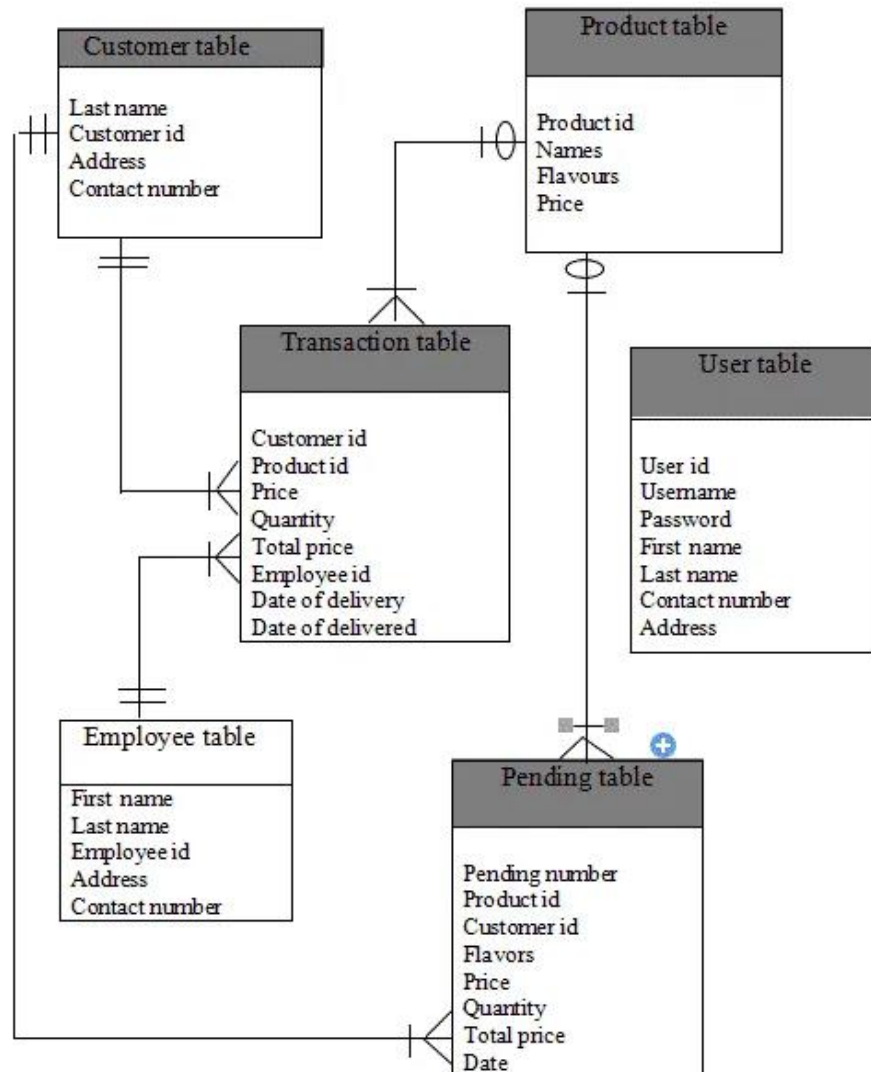


Рисунок 3.5 – Схема бази даних

3.3 Опис клієнтської частини

Клієнтська частина представлена у вигляді мобільного застосунку для пристроїв на базі операційної системи Android де користувачі можуть створити замовлення.

Взаємодія з серверною частиною буде виконуватись завдяки стандартній бібліотеці OkHttp а робота с JSON буде через стандартну бібліотеку Gson.

OkHttp - сучасна стандартна бібліотека для роботи з запитами у Android. Вона підтримує як стандартні REST запити, так і роботу з сокетамі.

Також вона має інтеграцію з бібліотекою Gson, завдяки чому запити можуть повертати готові моделі для роботи з даними.

Також мобільний застосунок матиме функціонал збереження (кешування) даних. Збереження даних буде зроблено з допомогою інтегрованої у Android SDK бази даних.

На сьогодні існує 3 основні бази даних для використання у мобільних застосунках на базі Android: Room, Realm та ObjectBox [20].

Room - реляційна база даних побудована на основі SQLite. Вона підтримує усі можливості SQL та працює з готовими моделями. Також в неї багата підтримка мови програмування Kotlin, як наприклад підтримка suspend функцій так data класів, що поліпшує розробку, а завдяки підтримки компонентів життєвого циклу, вона може уникнути витoki пам'яті при змінах стану застосунку. Також, на сьогодні це єдина офіційна база даних для роботи у Android SDK. Більш того, вона найлегша з усіх існуючих.

Realm – NoSQL база даних для зберігання об'єктів. Вона більш проста у використанні ніж Room та працює швидше, проте має недоліки у вигляді відсутності підтримки Kotlin та поганою підтримкою багатопоточного коду.

ObjectBox – також NoSQL база даних, котра по продуктивності найефективніша (порівняння швидкості роботи зображено на рис.), проте вона також має недоліки у вигляді відсутності підтримки мови програмування Kotlin

а також ряду стандартних бібліотек для роботи з життєвим циклом, що може призвести до витоку пам'яті.

На рисунку 3.6 наведено порівняння швидкості роботи баз даних.

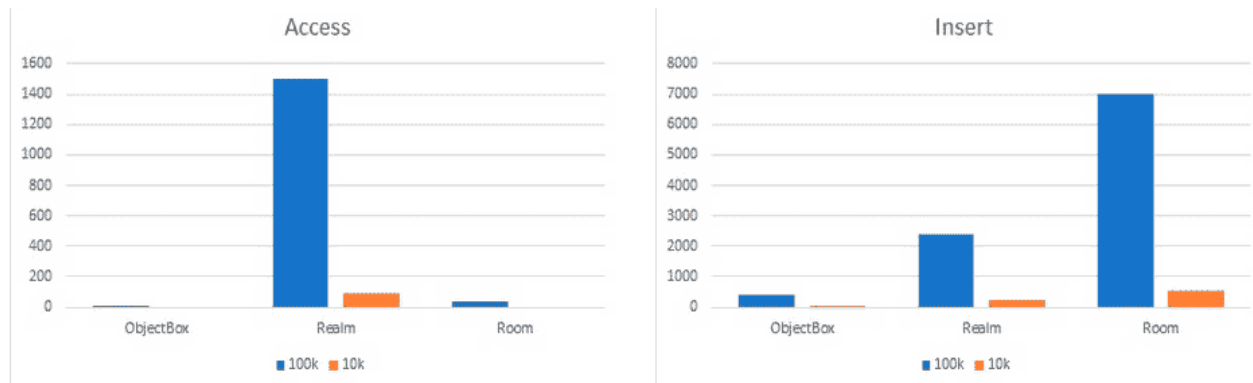


Рисунок 3.6 – Порівняння швидкості роботи баз даних[20]

Підключення кожної бібліотеки також збільшує кінцевий розмір застосунку. Нижче наведено скільки пам'яті потребує кожна з них:

- Realm — 3-4 Мб;
- ObjectBox — 1-1,5 Мб;
- Room — 50 Кб.

Не дивлячись на більш ефективну роботу бібліотек Realm та ObjectBox, для проекту було обрано базу даних Room через її підтримку Kotlin, життєвого циклу, а також завдяки офіційній підтримці Google. Більш того, вона забезпечить економію використання пам'яті, що позитивно буде сприйнято користувачем.

3.4 Опис серверної частини

Для серверної частини буде використана безкоштовна відкрита Grubhub API [19] для створення агрегаторів їжі. Цей сервіс надає функціонал з готових REST API запитів котрі дозволяють створити повноцінний сервіс з

замовленням їжі та з інтеграціями у системи компаній. Також цей сервіс має докладну документацію.

Ми будемо використовувати функціонал з реєстрації та авторизації, вибору компанії, страв та створення замовлення.

3.5 Висновок до третього розділу

У цьому розділі було розглянуті аспекти створення системи з замовлення обідів, а саме архітектура системи, архітектура взаємодії між клієнтською частиною та сервером, опис клієнтської та серверної частин.

Також була спроектована блок-схема роботи мобільного застосунку, побудована UML проекту та спроектована база даних для створення замовлень.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

Для нормального функціонування програми необхідні наступні умови до мобільного пристрою:

- версія ОС Android 5.0 та вище;
- оперативна пам'ять 512 Мб та більше;
- наявність виходу до мережі Інтернет;
- вільний простір на жорсткому диску не менше, ніж 8 Мб.

Після встановлення застосунку, користувачу необхідно зареєструватись або авторизуватись, якщо він вже має обліковий запис (рис. 4.1).

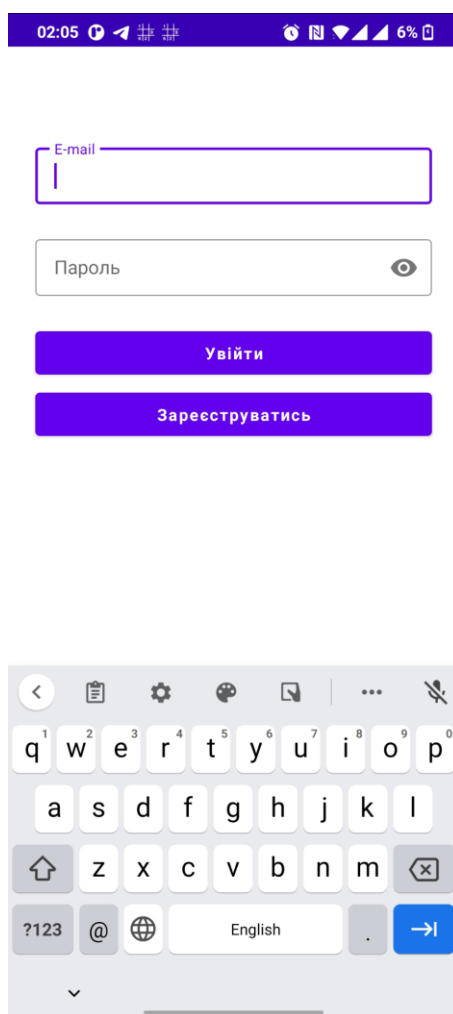


Рисунок 4.1 – Екран авторизації

Після авторизації користувач отримує можливість зробити замовлення. Перш за все він може обрати компанію до вподоби, через натискання необхідної. Після чого користувач потрапляє на екран де може обрати страву (рис. 4.2).

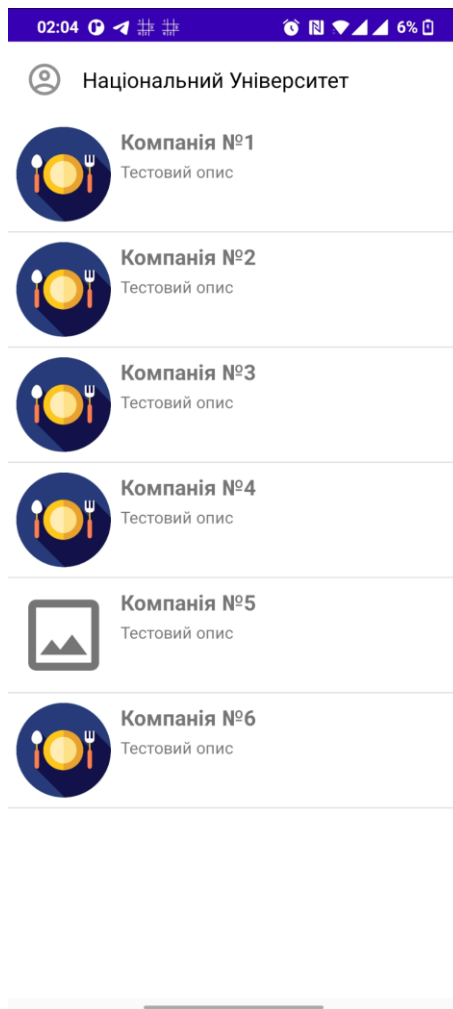


Рисунок 4.2 – Екран списку компаній

Обрав страву зі списку користувач може відкрити детальний екран страви де зможе додати її до кошику (рис. 4.3).

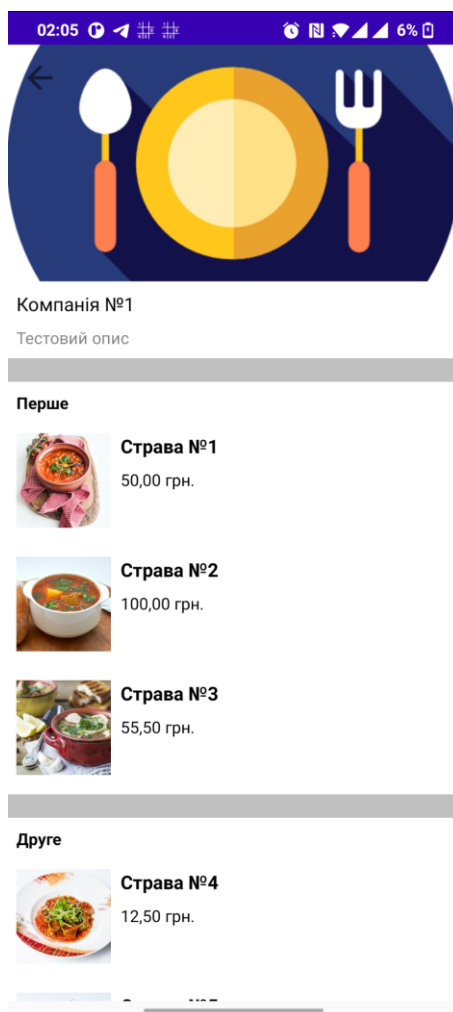


Рисунок 4.3 – Екран доступних до замовлення страв

Натискаючи кнопки + та - користувач додає або видаляє страви з корзини. На цьому екрані користувач також може побачити загальну ціну замовлення (рис. 4.4).



Рисунок 4.4 – Екран додавання страви у кошик

Після додавання хоч однієї страви у кошик у користувача з'являється можливість відкрити кошик та підтвердити замовлення, або відредагувати його.

На екрані кошику користувачу достатньо підтвердити замовлення, після чого воно відправиться компанії з доставці обідів (рис. 4.5).

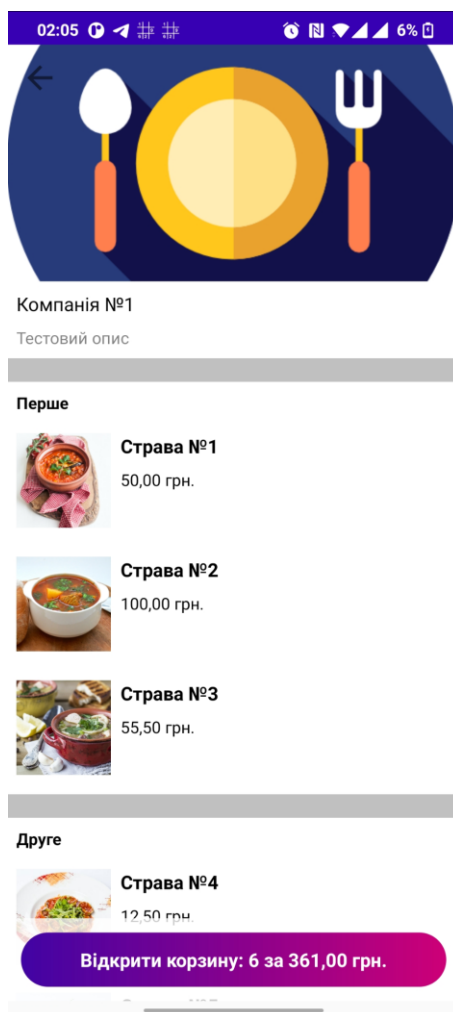


Рисунок 4.5 – Екран доступних до замовлення страв з заповненою корзиною

4.1 Функціональні можливості системи

Система складається з функціоналу замовлення обіду.

Користувач має можливість зареєструватись у системі, обрати компанію для замовлення, переглянути меню та замовити обід. Також користувач має можливість переглянути історію замовлень.

4.2 Розробка тестів

Тестування є однією із найважливіших частин розробки програм. Нажаль, у більшості своєму, у багатьох аутсорс компаніях написання тестів недостатньо часу, а тому дуже часто зустрічаються помилки. Написання тестів

необхідне при розробці якісного ПЗ, особливо у великих проектах із великою кількістю людей. Іноді внесення навіть невеликих змін можуть негативно відобразитись на інших компонентах програми, що на перший погляд можна непомітити. Тести допомагають розробникам перевіряти не тільки працездатність коду, але й його правильність. Проте слід пам'ятати, що тести - це також частина програми, яку потрібно підтримувати та правити.

Існує три види тестів у Android: unit-тести, інтеграційні тести, UI тести.

4.2.1 Unit-тести

Unit-тести - це тести JVM коду без залежностей компонентів Android. Вони пишуться за допомогою бібліотеки JUnit, котра має відповідні класи та методи задля запровадження тестів. Unit-тести використовуються, при тестуванні якогось загального коду та класів-утиліт.

У нашому проекті є логіка корзини та метод котрий рахує загальну ціну замовлення (рис. 4.6).

```
fun getTotalMealsPrice(meals: List<CartMeal>): BigDecimal {  
    val total = meals.map { it.meal.price.multiply(BigDecimal(it.count)) }  
    val sum = total.sumOf { it }  
    return sum  
}
```

Рисунок 4.6 – Метод розрахунку загальної ціни

Цей метод може здаватись простим та зрозумілим, проте коли є логіка із грошима, необхідно використовувати клас `BigDecimal` із стандартної бібліотеки `java.math`. Він працює не як звичайні примітиви, а тому можна випадково зробити помилку при роботі з ними.

Задля гарантування вірної роботи методу можна написати Unit-тест, котрий буде перевіряти правильність цього методу.

Щоб підключити її у проект, необхідно додати відповідну залежність у файл конфігурації (рис. 4.7).

```
testImplementation 'junit:junit:4.12'
androidTestImplementation 'androidx.test.ext:junit:1.1.1'
```

Рисунок 4.7 – Підключення бібліотеки JUnit

Після цього, у відповідній папці androidTest створити клас з анотацією @RunWith (рис. 4.8).

```
@RunWith(AndroidJUnit4::class)
class CartUnitTest {
```

Рисунок 4.8 – Створення тестового класу

Для написання тесту створюється відповідний метод з анотацією @Test де програмно описується його логіка та порівнюється з очікуваним результатом. (рис. 4.9).

Як можна зауважити, структура тестового методу можна поділити на три головні етапи:

- ініціалізація тестових даних;
- виклик методу що тестується;
- перевірка на відповідність.

Для запуску цих тестів необхідний підключений пристрій, що неможливо при використанні автоматизованих систем. Для уникнення цієї проблем необхідно написання інтеграційних тестів котрі емулюють логіку пристрою.

```

@Test
fun totalCartSum_isCorrect() {
    val cart = Cart(
        CompanyDetailModel(),
        meals = listOf(
            CartMeal(MealModel(price = BigDecimal( val: 2.5)), count: 2), //5
            CartMeal(MealModel(price = BigDecimal( val: 5)), count: 3), //15
            CartMeal(MealModel(price = BigDecimal( val: 10)), count: 4), //40
            CartMeal(MealModel(price = BigDecimal( val: 25)), count: 5), //125
            CartMeal(MealModel(price = BigDecimal( val: 52)), count: 6) //312
        )
    )

    val correctPrice = BigDecimal( val: 497)

    val result = NumberUtils.getTotalMealsPrice(cart.meals)

    assertEquals(result, correctPrice)
}

```

Рисунок 4.9 – Створення тестового методу

4.2.2 Інтеграційні тести

Інтеграційні тести – тести, що можуть емулювати платформи-залежні методи. Для написання таких тестів, у котрих необхідно використовувати методи Android SDK, використовується бібліотека Robolectric [24]. Для його використання, перш за все, бібліотеку необхідно додати у файл конфігурації проекту (рис. 4.10).

```
testImplementation 'org.robolectric:robolectric:4.3'
```

Рисунок 4.10 – Підключення бібліотеки Robolectric

Та позначити необхідний клас анотацією `@RunWith` з використанням `RobolectricTestRunner`, після чого, методи Unit-тестів у цьому класі матимуть

доступ до методів Android і використання таких тестів стає можливим у CI/CD (рис. 4.11).

```
@RunWith(RobolectricTestRunner::class)
@Config(sdk = [21], application = CorliverApp::class)
class CartIntegrationTest {
```

Рисунок 4.11 – Створення тестового класу для інтеграційних тестів

Проте бібліотека Robolectric має й недоліки. Через те що розробники не можуть вчасно вносити зміни у платформу тестування свіжих версій Android SDK може призвести до того що тести не будуть працювати з ними.

4.2.3 UI - тести

UI-тести потрібні для тестування користувацького інтерфейса та його реакції на емульовані дії користувача. Для цього використовується бібліотека Espresso. Вона допомагає відтворити поведінку користувача та перевірити інтерфейс.

Підключається вона так само як і інші бібліотеки для тестування: додаванням відповідної залежності у файл конфігурації (рис. 4.12)

```
androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
```

Рисунок 4.12 – Підключення бібліотеки Espresso

Завдяки класам ActivityTestRule можна відтворити логіку певного екрану, а завдяки методам роботи з цим правилом можна отримати доступ до UI компонентів та відтворити дії користувача.

Таким чином можливо написати тести на користувацький інтерфейс та емулювати поведінку користувача.

4.3 Висновок до четвертого розділу

У цьому розділі були розглянуті та описані експлуатаційні можливості системи та побудовані тести для тестування системи задля її правильної роботи. Також з наведеними прикладами у вигляді рисунків описані головні функції системи згідно функціональних вимог за розділом 1.

ВИСНОВКИ

Під час роботи над виконання дипломної роботи магістра було проаналізовані різні системи з створення замовлень обідів, було сформовані функціональні вимоги для системи, котрі були реалізовані у застосунку.

Для успішного виконання завдання було поставлено і вирішено такі завдання:

- ознайомлення та аналіз предметної області;
 - аналіз існуючих інструментів при розробці та проектуванні систем.
- Обрана мова Kotlin та відповідні бібліотеки для реалізації компонентів системи. Як середовище розробки була обрана Android Studio;
- розроблення системи з замовлення обідів у офіси, що відповідає сучасним архітектурним підходам.

Розроблена система “Corliver” відповідає усім сформованим функціональним вимогам, має зрозумілий користувацький інтерфейс, малий розмір та відповідає стандартам безпеки ОС Android. Завдяки цьому бізнес може організувати замовлення їжі для своїх підлеглих, що покращить їх відношення до компанії та підвищить інтерес у поверненні до офісів після закінчення карантинних обмежень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1 Итоги-2020. Как в пандемию развивались службы доставки [Электрон. ресурс]. – Режим доступа: <https://ain.ua/ru/2020/12/18/itogi-2021-kak-razvivalis-sluzhby-dostavki/>

2 Как удалённая работа влияет на эффективность? Два новых исследования [Электрон. ресурс]. – Режим доступа: <https://upr.ru/article/kak-udalyennaya-rabota-vliyaet-na-effektivnost/>

3 Ринок праці 2021 [Електрон. ресурс]. – Режим доступу: https://dou.ua/lenta/articles/jobs-and-trends-2021/?from=tg&utm_source=telegram&utm_medium=social

4 Что такое агрегатом [Электрон. ресурс]. – Режим доступа: <https://secretmag.ru/enciklopediya/chto-takoe-agregator-obyasnyаем-prostymi-slovami.htm>

5 MVC [Electronic resource]. – Access mode: <http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html>

6 A cookbook for using the model-view controller user interface paradigm in Smalltalk-80 [Electronic resource]. – Access mode: <https://dl.acm.org/doi/10.5555/50757.50759>

7 Model-View-Controller [Электрон. ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Model-View-Controller>

8 Android MVP пример для начинающих. Без библиотек и интерфейсов [Электрон. ресурс]. – Режим доступа: <https://startandroid.ru/ru/blog/493-mvp-dlja-nachinajuschih-bez-bibliotek-i-interfejsov.html>

9 Moxy is MVP library for Android [Electronic resource]. – Access mode: <https://github.com/Arello-Mobile/Moxy>

10 A Model-View-Presenter. Model-View-Intent library for modern Android apps. Mosby [Electronic resource]. – Access mode: <https://github.com/sockeqwe/mosby>

11 Model-View-ViewModel [Электрон. ресурс]. – Режим доступа:
<https://ru.wikipedia.org/wiki/Model-View-ViewModel>

12 Model-View-Intent(MVI) in Android [Electronic resource]. – Access mode:
<https://www.novatec-gmbh.de/en/blog/mvi-in-android/>

13 Java [Электрон. ресурс]. – Режим доступа:
<https://uk.wikipedia.org/wiki/Java>

14 Kotlin [Электрон. ресурс]. – Режим доступа:
<https://uk.wikipedia.org/wiki/Kotlin>

15 Java vs Kotlin: It's Time To Expand Android Development [Electronic resource]. – Access mode: <https://hackernoon.com/java-vs-kotlin-its-time-to-expand-android-development-f08e3d6a72b6>

16 Everything you need to build on Android [Electronic resource]. – Access mode: <https://developer.android.com/studio/features>

17 XML [Электрон. ресурс]. – Режим доступа:
<https://uk.wikipedia.org/wiki/XML>

18 JSON [Электрон. ресурс]. – Режим доступа:
<https://uk.wikipedia.org/wiki/JSON>

19 Grubhub [Electronic resource]. – Access mode:
<https://enterprise.grubhub.com/>

20 Порівняння Realm, ObectBox і Room [Электрон. ресурс]. – Режим доступа: <https://vseznayko.com/dovidnik/porivnyannya-realm-obectbox-i-room>

21 GrubHub Design [Электрон. ресурс]. – Режим доступа:
<https://www.behance.net/gallery/27827503/GrubHub-Redesign>

22 Notch [Электрон. ресурс]. – Режим доступа:
https://play.google.com/store/apps/details?id=com.vendorhero.storefront&hl=en_US&gl=US

23 Eatfirst [Электрон. ресурс]. – Режим доступа:
<https://apkpure.com/eatfirst-fresh-food-delivery/com.eatfirst.android>

24 Robolectric: test-drive your Android code [Electronic Resource]. – Access mode: <http://robolectric.org/>

ДОДАТОК А
Текст програми

A.1 Файли ресурсів застосунку

A.1.1 Файл AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.coriliver.android">

    <application
        android:name=".CorliverApp"
        android:allowBackup="false"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@style/Theme.Coriliver">

        <activity
            android:name=".activity.MainActivity"
            android:screenOrientation="portrait"
            >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

</manifest>
```

A.1.2 Файл build.gradle

```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
    id 'kotlin-kapt'
    id 'kotlin-parcelize'
}

android {
    compileSdkVersion 30
    buildToolsVersion "30.0.3"

    defaultConfig {
        applicationId "com.coriliver.android"
        minSdkVersion 21
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
}
```

```

    }
}
compileOptions {
    sourceCompatibility JavaVersion.VERSION_1_8
    targetCompatibility JavaVersion.VERSION_1_8
}
kotlinOptions {
    jvmTarget = '1.8'
}
buildFeatures {
    viewBinding true
}
}

dependencies {

    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk8:1.5.30"
    implementation 'androidx.core:core-ktx:1.6.0'
    implementation 'androidx.appcompat:appcompat:1.3.1'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.1'
    implementation 'com.google.android.gms:play-services-auth:19.2.0'
    implementation 'com.google.android.gms:play-services-auth-api-phone:17.5.1'

    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-core:1.5.2'
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-android:1.5.2'
    implementation 'androidx.browser:browser:1.3.0'

    implementation "androidx.lifecycle:lifecycle-common-java8:2.3.1"
    implementation 'androidx.lifecycle:lifecycle-extensions:2.2.0'
    implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.3.1'

    implementation "org.koin:koin-android:2.0.1"
    implementation "org.koin:koin-androidx-viewmodel:2.0.1"

    implementation 'com.github.terrakok:cicerone:7.1'

    implementation "com.squareup.okhttp3:okhttp:4.9.1"
    implementation "com.squareup.okhttp3:logging-interceptor:4.9.1"
    implementation "com.squareup.retrofit2:retrofit:2.9.0"
    implementation "com.squareup.retrofit2:adapter-rxjava2:2.8.1"
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
    implementation 'com.google.code.gson:gson:2.8.8'

    implementation 'androidx.room:room-runtime:2.3.0'
    implementation 'androidx.room:room-ktx:2.3.0'
    kapt 'androidx.room:room-compiler:2.3.0'

    implementation 'com.github.bumptech.glide:glide:4.12.0'
    kapt 'com.github.bumptech.glide:compiler:4.12.0'

    implementation("io.github.g00fy2.quickie:quickie-bundled:1.2.3")
    implementation ('com.alimuzaffar.lib:pinentryedittext:2.0.6') {
        exclude group: 'androidx.appcompat', module: 'appcompat'
    }

    testImplementation 'org.robolectric:robolectric:4.3'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
}

```

A.1.3 Файл activity_main.xml

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".activity.MainActivity">

    <FrameLayout
        android:id="@+id/main_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

A.1.4 Файл fragment_home.xml

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/white">

    <androidx.appcompat.widget.AppCompatImageView
        android:id="@+id/home_profile_image"
        android:layout_width="32dp"
        android:layout_height="32dp"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:alpha="0.8"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/outline_account_circle" />

    <androidx.appcompat.widget.AppCompatTextView
        android:id="@+id/home_company_title"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:lines="1"
        android:textSize="18sp"
        android:textColor="@color/black"
        app:layout_constraintBottom_toBottomOf="@+id/home_profile_image"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/home_profile_image"
        app:layout_constraintTop_toTopOf="@+id/home_profile_image"
        tools:text="Some cool value" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/home_companies"
        android:layout_width="match_parent"
```

```

        android:layout_height="0dp"
        android:layout_marginTop="16dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/home_profile_image" />

<com.coriliver.android.custom.ProgressView
    android:id="@+id/progressView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:visibility="gone"
    tools:visibility="visible"
/>

</androidx.constraintlayout.widget.ConstraintLayout>

```

A.1.5 Файл fragment_cart.xml

```

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <androidx.core.widget.NestedScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <androidx.appcompat.widget.AppCompatImageView
                android:id="@+id/back"
                android:layout_width="32dp"
                android:layout_height="32dp"
                android:layout_margin="12dp"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent"
                app:srcCompat="@drawable/arrow_back" />

            <androidx.appcompat.widget.AppCompatTextView
                android:id="@+id/cart_top_title"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textSize="24sp"
                android:textStyle="bold"
                android:layout_margin="12dp"
                android:textColor="@color/black"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintTop_toBottomOf="@+id/back"
                tools:text="Заказ из Test Delivery" />

            <androidx.appcompat.widget.AppCompatTextView
                android:id="@+id/cart_order_title"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textSize="16sp"
                android:layout_margin="12dp"

```

```

        android:textColor="@color/black"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/cart_top_title"
        tools:text="Заказ" />

<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/cart_order_list"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="12dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/cart_order_title"
    />

<androidx.appcompat.widget.AppCompatTextView
    android:id="@+id/cart_order_total_sum"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="16sp"
    android:layout_margin="12dp"
    android:textColor="@color/black"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/cart_order_list"
    tools:text="Заказ" />

<androidx.appcompat.widget.AppCompatTextView
    android:id="@+id/cart_confirm"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textColor="@color/white"
    android:textSize="16sp"
    android:textStyle="bold"
    android:gravity="center"
    android:text="@string/accept_order"
    android:background="@drawable/bg_open_cart"
    android:layout_margin="12dp"
    android:padding="12dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/cart_order_total_sum"
    />
</androidx.constraintlayout.widget.ConstraintLayout>

</androidx.core.widget.NestedScrollView>

<com.coriliver.android.custom.ProgressView
    android:id="@+id/progressView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:visibility="gone"
    tools:visibility="visible"
    />

</FrameLayout>

```

A.1.6 Файл fragment_company_details.xml

```

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"

```

```

xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent">

<androidx.core.widget.NestedScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <androidx.appcompat.widget.AppCompatImageView
            android:id="@+id/company_image"
            android:layout_width="0dp"
            android:layout_height="200dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <androidx.appcompat.widget.AppCompatImageView
            android:id="@+id/back"
            android:layout_width="32dp"
            android:layout_height="32dp"
            android:layout_margin="12dp"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:srcCompat="@drawable/arrow_back" />

        <androidx.appcompat.widget.AppCompatTextView
            android:id="@+id/company_name"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:textColor="@color/black"
            android:textSize="16sp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/company_image"
            tools:text="Some cool name" />

        <androidx.appcompat.widget.AppCompatTextView
            android:id="@+id/company_description"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:alpha="0.5"
            android:textColor="@color/black"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/company_name"
            tools:text="Some cool description" />

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/company_menu"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:nestedScrollingEnabled="false"

```

```

        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/company_description"
    />

</androidx.constraintlayout.widget.ConstraintLayout>

</androidx.core.widget.NestedScrollView>

<FrameLayout
    android:id="@+id/open_cart_layout"
    android:visibility="gone"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/bg_open_cart_layout"
    android:layout_gravity="bottom"
>

    <androidx.appcompat.widget.AppCompatTextView
        android:id="@+id/open_cart"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="@color/white"
        android:textSize="16sp"
        android:textStyle="bold"
        android:gravity="center"
        tools:text="Open Cart: 1 for 123.23 UAH"
        android:background="@drawable/bg_open_cart"
        android:layout_margin="12dp"
        android:padding="12dp"
    />

</FrameLayout>

<com.coriliver.android.custom.ProgressView
    android:id="@+id/progressView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:visibility="gone"
    tools:visibility="visible"
/>

</FrameLayout>

```

A.1.7 Файл fragment_login.xml

```

<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:tools="http://schemas.android.com/tools">

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/login_email"
        style="@style/LoginTextInputLayoutStyle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="80dp"
        android:layout_marginEnd="24dp"
    >

```

```

        android:hint="@string/login_hint_email"
        android:theme="@style/LoginTextInputLayoutStyle"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/login_email_input"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textEmailAddress" />

    </com.google.android.material.textfield.TextInputLayout>

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/login_password"
        style="@style/LoginTextInputLayoutStyle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="24dp"
        android:layout_marginEnd="24dp"
        android:hint="@string/login_hint_password"
        android:theme="@style/LoginTextInputLayoutStyle"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/login_email"
        app:passwordToggleEnabled="true">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/login_password_input"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textPassword" />

    </com.google.android.material.textfield.TextInputLayout>

    <com.google.android.material.button.MaterialButton
        android:id="@+id/login_sign_in_button"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="24dp"
        android:layout_marginEnd="24dp"
        android:text="@string/login_button_sign_in"
        android:textAllCaps="false"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/login_password" />

    <com.google.android.material.button.MaterialButton
        android:id="@+id/login_sign_up_button"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="4dp"
        android:layout_marginEnd="24dp"
        android:text="@string/login_button_sign_up"
        android:textAllCaps="false"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/login_sign_in_button" />

```

```

<com.coriliver.android.custom.ProgressView
    android:id="@+id/progressView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:visibility="gone"
    tools:visibility="visible"
/>
</androidx.constraintlayout.widget.ConstraintLayout>

```

A.1.8 Файл fragment_meal.xml

```

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.core.widget.NestedScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <androidx.appcompat.widget.AppCompatImageView
                android:id="@+id/meal_image"
                android:layout_width="0dp"
                android:layout_height="200dp"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent" />

            <androidx.appcompat.widget.AppCompatImageView
                android:id="@+id/back"
                android:layout_width="32dp"
                android:layout_height="32dp"
                android:layout_margin="12dp"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent"
                app:srcCompat="@drawable/arrow_back" />

            <androidx.appcompat.widget.AppCompatTextView
                android:id="@+id/meal_name"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_marginStart="8dp"
                android:layout_marginTop="8dp"
                android:layout_marginEnd="8dp"
                android:textColor="@color/black"
                android:textSize="18sp"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toBottomOf="@+id/meal_image"
                tools:text="Some cool name" />

            <androidx.appcompat.widget.AppCompatTextView
                android:id="@+id/meal_price"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_marginStart="8dp"

```

```

        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:textColor="@color/black"
        android:textSize="16sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/meal_name"
        tools:text="123.0" />

<androidx.appcompat.widget.AppCompatTextView
    android:id="@+id/meal_description"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:alpha="0.5"
    android:textColor="@color/black"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/meal_price"
    tools:text="Some cool description" />

<LinearLayout
    android:id="@+id/meal_counter_layout"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:gravity="center"
    android:layout_marginTop="18dp"
    app:layout_constraintTop_toBottomOf="@id/meal_description"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent">

    <androidx.appcompat.widget.AppCompatImageView
        android:id="@+id/menu_remove"
        android:layout_width="32dp"
        android:layout_height="32dp"
        android:background="@drawable/bg_meal_button"
        android:padding="4dp"
        app:srcCompat="@drawable/ic_remove_black"
        app:tint="@color/white" />

    <androidx.appcompat.widget.AppCompatTextView
        android:id="@+id/menu_counter"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/black"
        android:textSize="18sp"
        android:text="0"
        android:layout_marginStart="32dp"
        android:layout_marginEnd="32dp"
        />

    <androidx.appcompat.widget.AppCompatImageView
        android:id="@+id/menu_add"
        android:layout_width="32dp"
        android:layout_height="32dp"
        android:background="@drawable/bg_meal_button"
        app:tint="@color/white"
        app:srcCompat="@drawable/ic_add_black"
        android:padding="4dp"

```

```

        />

    </LinearLayout>

    <androidx.appcompat.widget.AppCompatTextView
        android:id="@+id/total_price"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="24dp"
        android:layout_marginEnd="8dp"
        android:textColor="@color/black"
        android:textSize="18sp"
        android:gravity="center"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/meal_counter_layout"
        tools:text="Загалом: 250.50 грн." />

    </androidx.constraintlayout.widget.ConstraintLayout>

</androidx.core.widget.NestedScrollView>

<com.coriliver.android.custom.ProgressView
    android:id="@+id/progressView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:visibility="gone"
    tools:visibility="visible"
/>

</FrameLayout>

```

A.1.9 Файл fragment_orders.xml

```

<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <androidx.appcompat.widget.AppCompatImageView
        android:id="@+id/back"
        android:layout_width="24dp"
        android:layout_height="24dp"
        android:layout_marginStart="12dp"
        app:layout_constraintBottom_toBottomOf="@+id/order_history_title"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/order_history_title"
        app:srcCompat="@drawable/arrow_back"
        app:tint="@color/black" />

    <androidx.appcompat.widget.AppCompatTextView
        android:id="@+id/order_history_title"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="8dp"
        android:lines="1"

```

```

        android:text="@string/orders_history"
        android:textColor="@color/black"
        android:textSize="24sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/back"
        app:layout_constraintTop_toTopOf="parent" />

<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/order_history_list"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_marginTop="16dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/order_history_title" />

</androidx.constraintlayout.widget.ConstraintLayout>

```

A.1.10 Файл fragment_profile.xml

```

<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <View
        android:id="@+id/top_radial_view"
        android:layout_width="match_parent"
        android:layout_height="80dp"
        android:background="@drawable/bg_radial"
        app:layout_constraintTop_toTopOf="parent" />

    <androidx.appcompat.widget.AppCompatImageView
        android:id="@+id/back"
        android:layout_width="24dp"
        android:layout_height="24dp"
        android:layout_marginStart="12dp"
        android:layout_marginTop="12dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/top_radial_view"
        app:srcCompat="@drawable/arrow_back"
        app:tint="@color/white" />

    <LinearLayout
        android:id="@+id/profile_main"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="12dp"
        android:gravity="center"
        android:paddingStart="12dp"
        android:paddingEnd="12dp"
        android:orientation="vertical"
        app:layout_constraintTop_toBottomOf="@+id/top_radial_view">

        <androidx.appcompat.widget.AppCompatTextView
            android:id="@+id/profile_name"
            android:layout_width="match_parent"

```

```

        android:layout_height="wrap_content"
        android:lines="1"
        android:textColor="@color/black"
        android:textSize="18sp"
        android:textStyle="bold"
        tools:text="Tester Name" />

<androidx.appcompat.widget.AppCompatTextView
    android:id="@+id/profile_phone"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:lines="1"
    android:textColor="@color/black"
    android:textSize="16sp"
    tools:text="+3 8050 111 11 11" />

<androidx.appcompat.widget.AppCompatTextView
    android:id="@+id/profile_email"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:lines="1"
    android:textColor="@color/black"
    android:textSize="16sp"
    tools:text="test@test.test" />

</LinearLayout>

<LinearLayout
    android:id="@+id/profile_orders"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="12dp"
    android:gravity="center"
    android:paddingStart="12dp"
    android:paddingEnd="12dp"
    android:paddingTop="12dp"
    android:paddingBottom="12dp"
    android:orientation="horizontal"
    android:background="@drawable/bg_profile_item_bg"
    app:layout_constraintTop_toBottomOf="@+id/profile_main">

    <androidx.appcompat.widget.AppCompatTextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:lines="1"
        android:layout_weight="1"
        android:textColor="@color/black"
        android:textSize="18sp"
        android:text="@string/my_orders" />

    <androidx.appcompat.widget.AppCompatImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@drawable/ic_navigate_next"
        app:tint="@color/black" />

</LinearLayout>

<com.coriliver.android.custom.ProgressView
    android:id="@+id/progressView"
    android:layout_width="match_parent"

```

```

    android:layout_height="match_parent"
    android:visibility="gone"
    tools:visibility="gone"
  />

```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

A.1.11 Файл fragment_register.xml

```

<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:tools="http://schemas.android.com/tools">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_marginBottom="16dp"
        app:layout_constraintBottom_toTopOf="@+id/register_sign_in_button"
        app:layout_constraintTop_toTopOf="parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <com.google.android.material.textfield.TextInputLayout
                android:id="@+id/register_name"
                style="@style/LoginTextInputLayoutStyle"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginStart="24dp"
                android:layout_marginTop="24dp"
                android:layout_marginEnd="24dp"
                android:hint="@string/register_hint_full_name"
                android:theme="@style/LoginTextInputLayoutStyle">

                <com.google.android.material.textfield.TextInputEditText
                    android:id="@+id/register_name_input"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:inputType="textPersonName" />

            </com.google.android.material.textfield.TextInputLayout>

            <com.google.android.material.textfield.TextInputLayout
                android:id="@+id/register_email"
                style="@style/LoginTextInputLayoutStyle"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginStart="24dp"
                android:layout_marginTop="8dp"
                android:layout_marginEnd="24dp"
                android:hint="@string/login_hint_email"
                android:theme="@style/LoginTextInputLayoutStyle">

                <com.google.android.material.textfield.TextInputEditText
                    android:id="@+id/register_email_input"

```

```

        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textEmailAddress" />

</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/register_password"
    style="@style/LoginTextInputLayoutStyle"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="24dp"
    android:hint="@string/login_hint_password"
    android:theme="@style/LoginTextInputLayoutStyle"
    app:passwordToggleEnabled="true">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/register_password_input"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPassword" />

</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/register_confirm_password"
    style="@style/LoginTextInputLayoutStyle"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="24dp"
    android:hint="@string/login_hint_confirm_password"
    android:theme="@style/LoginTextInputLayoutStyle"
    app:passwordToggleEnabled="true">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/register_confirm_password_input"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPassword" />

</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/register_phone"
    style="@style/LoginTextInputLayoutStyle"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="24dp"
    android:hint="@string/register_hint_phone"
    android:theme="@style/LoginTextInputLayoutStyle">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/register_phone_input"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="phone" />

```

```

        </com.google.android.material.textfield.TextInputLayout>

        <com.google.android.material.button.MaterialButton
            android:id="@+id/register_scan_qr_button"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginStart="24dp"
            android:layout_marginTop="24dp"
            android:layout_marginEnd="24dp"
            android:text="@string/login_scan_qr_button"
            android:textAllCaps="false" />
    </LinearLayout>

</ScrollView>

<com.google.android.material.button.MaterialButton
    android:id="@+id/register_sign_in_button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginEnd="24dp"
    android:layout_marginBottom="16dp"
    app:layout_constraintBottom_toBottomOf="parent"
    android:text="@string/register_back_to_sign_in_button"
    android:textAllCaps="false" />

<com.coriliver.android.custom.ProgressView
    android:id="@+id/progressView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:visibility="gone"
    tools:visibility="visible"
/>

</androidx.constraintlayout.widget.ConstraintLayout>

```

A.1.12 Файл fragment_sms.xml

```

<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.appcompat.widget.AppCompatTextView
        android:id="@+id/sms_title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="32dp"
        android:layout_marginEnd="16dp"
        android:gravity="center"
        android:text="@string/we_have_sent_you_an_sms_with_verification_code"
        android:textColor="@color/black"
        android:textSize="24sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ProgressBar

```

```

        android:id="@+id/sms_progress"
        style="@style/InScreenProgressBar"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:layout_marginTop="16dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/sms_title" />

<LinearLayout
    android:id="@+id/sms_code_layout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="24dp"
    android:layout_marginStart="16dp"
    android:layout_marginEnd="16dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/sms_progress">

    <com.alimuzaffar.lib.pin.PinEntryEditText
        android:id="@+id/txt_pin_entry"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:cursorVisible="false"
        android:digits="1234567890"
        android:inputType="number"
        android:maxLength="5"
        android:textIsSelectable="false"
        tools:text="12345"
        app:pinTextBottomPadding="18dp"
        app:pinBackgroundDrawable="@drawable/bg_sms_pin"
        app:pinBackgroundIsSquare="true"
        android:textSize="28sp" />

</LinearLayout>

<com.google.android.material.button.MaterialButton
    android:id="@+id/sms_button_apply"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginTop="24dp"
    android:layout_marginEnd="16dp"
    android:text="@string/apply"
    android:textAllCaps="false"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/sms_code_layout" />

<androidx.appcompat.widget.AppCompatTextView
    android:id="@+id/sms_button_resend"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:textSize="14sp"
    android:textColor="@color/resend_code_color"
    app:layout_constraintEnd_toEndOf="@+id/sms_button_apply"
    app:layout_constraintTop_toBottomOf="@+id/sms_button_apply"
    tools:text="Resend code in: 0:50" />

</androidx.constraintlayout.widget.ConstraintLayout>

```

A.1.13 Файл fragment_splash.xml

```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bg_radial">

    <androidx.appcompat.widget.AppCompatTextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:fontFamily="sans-serif-medium"
        android:gravity="center"
        android:shadowColor="#000000"
        android:shadowDx="1"
        android:shadowDy="1"
        android:shadowRadius="1"
        android:text="Corliver"
        android:textColor="@color/white"
        android:textSize="48sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

A.1.14 Файл item_cart.xml

```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingBottom="8dp"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <androidx.appcompat.widget.AppCompatImageView
        android:id="@+id/meal_image"
        android:layout_width="80dp"
        android:layout_height="80dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        android:layout_margin="8dp"
        />

    <androidx.appcompat.widget.AppCompatTextView
        android:id="@+id/meal_name"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_margin="8dp"
        android:textColor="@color/black"
        android:textSize="16sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@id/meal_image"
        app:layout_constraintTop_toTopOf="parent"
```

```

        tools:text="Very" />

<androidx.appcompat.widget.AppCompatTextView
    android:id="@+id/meal_price"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_margin="8dp"
    android:textColor="@color/black"
    android:textSize="14sp"
    app:layout_constraintEnd_toStartOf="@+id/meal_count"
    app:layout_constraintStart_toEndOf="@id/meal_image"
    app:layout_constraintTop_toBottomOf="@id/meal_name"
    tools:text="Всего: 123.50 грн" />

<androidx.appcompat.widget.AppCompatTextView
    android:id="@+id/meal_count"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_margin="8dp"
    android:textColor="@color/black"
    android:textSize="24sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="@+id/meal_delete"
    app:layout_constraintEnd_toStartOf="@+id/meal_delete"
    app:layout_constraintTop_toTopOf="@+id/meal_delete"
    tools:text="4" />

<androidx.appcompat.widget.AppCompatImageView
    android:id="@+id/meal_delete"
    android:layout_width="40dp"
    android:layout_height="40dp"
    android:layout_margin="8dp"
    android:padding="8dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/meal_name"
    app:srcCompat="@android:drawable/ic_delete" />

</androidx.constraintlayout.widget.ConstraintLayout>

```

A.1.15 Файл item_company_menu_item.xml

```

<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingBottom="8dp"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <androidx.appcompat.widget.AppCompatImageView
        android:id="@+id/meal_image"
        android:layout_width="80dp"
        android:layout_height="80dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        android:layout_margin="8dp"
        />

    <androidx.appcompat.widget.AppCompatTextView
        android:id="@+id/meal_name"

```

```

        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:textColor="@color/black"
        android:textSize="16sp"
        android:textStyle="bold"
        tools:text="Very\nLong\nName\of\ndish"
        app:layout_constraintStart_toEndOf="@id/meal_image"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_margin="8dp"
    />

    <androidx.appcompat.widget.AppCompatTextView
        android:id="@+id/meal_price"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:textColor="@color/black"
        android:textSize="14sp"
        android:layout_margin="8dp"
        tools:text="123\n123"
        app:layout_constraintStart_toEndOf="@id/meal_image"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@id/meal_name"
    />

</androidx.constraintlayout.widget.ConstraintLayout>

```

A.1.16 Файл item_company_menu_item_section.xml

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <View
        android:layout_width="match_parent"
        android:layout_height="20dp"
        android:background="#C0C0C0"
    />

    <androidx.appcompat.widget.AppCompatTextView
        android:id="@+id/title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="@color/black"
        android:textStyle="bold"
        android:padding="8dp"
        tools:text="Salads"
    />

</LinearLayout>

```

A.1.17 Файл item_home_company.xml

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"

```

```

        android:orientation="horizontal">

        <androidx.appcompat.widget.AppCompatImageView
            android:id="@+id/company_image"
            android:layout_width="80dp"
            android:layout_height="80dp"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            android:layout_marginBottom="8dp" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="8dp"
            android:orientation="vertical">

            <androidx.appcompat.widget.AppCompatTextView
                android:id="@+id/company_name"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:ellipsize="end"
                android:maxLines="2"
                android:textSize="18sp"
                android:textStyle="bold"
                tools:text="Some company name" />

            <androidx.appcompat.widget.AppCompatTextView
                android:id="@+id/company_description"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="4dp"
                android:ellipsize="end"
                android:maxLines="3"
                android:textSize="14sp"
                tools:text="Some company description" />

        </LinearLayout>

</LinearLayout>

```

A.1.18 Файл item_order.xml

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:layout_marginBottom="8dp"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="12dp"
        android:layout_marginEnd="12dp"
        android:gravity="top|end"
        android:orientation="horizontal">

        <androidx.appcompat.widget.AppCompatTextView
            android:id="@+id/order_company"
            android:layout_width="0dp"

```

```

        android:layout_height="wrap_content"
        android:layout_width="1"
        android:textColor="@color/black"
        android:textSize="18sp"
        android:textStyle="bold"
        tools:text="Company name\nLong aw" />

<androidx.appcompat.widget.AppCompatTextView
    android:id="@+id/order_date"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:alpha="0.5"
    android:gravity="end"
    android:lines="1"
    android:textColor="@color/black"
    android:textSize="14sp"
    tools:text="12.12.12" />

</LinearLayout>

<androidx.appcompat.widget.AppCompatTextView
    android:id="@+id/order_meals"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="12dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="12dp"
    android:textColor="@color/black"
    android:textSize="16sp"
    tools:text="1 x Meat" />

<androidx.appcompat.widget.AppCompatTextView
    android:id="@+id/order_price"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="12dp"
    android:layout_marginTop="12dp"
    android:layout_marginEnd="12dp"
    android:lines="1"
    android:textColor="@color/black"
    android:textSize="16sp"
    android:textStyle="bold"
    tools:text="120 UAH" />

<View
    android:layout_width="match_parent"
    android:layout_height="1dp"
    android:layout_marginTop="12dp"
    android:background="@color/black" />

</LinearLayout>

```

A.1.19 Файл strings.xml

```

<resources>
    <string name="app_name">Coriliver</string>
    <string name="dialog_ok">OK</string>
    <string name="dialog_error">Помилка</string>
    <string name="dialog_cancel">Відміна</string>
    <string name="error_unknown">Невідома помилка. Спробуйте пізніше.</string>

```

```

<string name="error_validation_error">Не всі поля вірні.</string>
<string name="error_api_error">Помилка сервера. Спробуйте пізніше.</string>
<string name="error_try_again">Ще раз</string>
<string name="error_cancel">Відміна</string>
<string name="login_hint_email">E-mail</string>
<string name="login_hint_password">Пароль</string>
<string name="login_hint_confirm_password">Підтвердити пароль</string>
<string name="login_button_sign_in">Увійти</string>
<string name="login_button_sign_up">Зареєструватись</string>
<string name="validation_error_email">Не вірний формат</string>
<string name="validation_error_passwords_are_not_same">Не
відповідає</string>
<string name="validation_error_phone">Не правильний номер телефону</string>
<string name="validation_error_name">Обов'язково до заповнення</string>
<string name="register_back_to_sign_in_button">Повернутись</string>
<string name="login_scan_qr_button">Сканувати QR код</string>
<string name="register_hint_phone">Номер телефону</string>
<string name="register_hint_full_name">Повне ім'я</string>
<string name="resend_code">Відправити код ще раз</string>
<string name="resend_code_time">Відправити код через %s</string>
<string name="open_cart">Відкрити корзину: %d за %.2f грн.</string>
<string name="accept_order">Підтвердити</string>
<string name="orders_history">Історія замовлень</string>
<string name="my_orders">Мої замовлення</string>
<string name="we_have_sent_you_an_sms_with_verification_code">Вам було
надіслано SMS повідомлення з кодом.</string>
<string name="apply">Підтвердити</string>
<string name="validation_error_password">
    Усі символи латиніцею\n
    Повинна бути одна мала літера\n
    Повинна бути одна велика літера\n
    Повинне бути хоч одна цифра\n
    Повинне включати спеціальний символ (!@#$%^&*~)\n
    Має бути довше 8 символів\n
    Не повинне повторювати назву компанії
</string>
<string name="ready">Готово</string>
<string name="order_sent_message">Замовлення відправлено на
обробку.</string>
<string name="order_from">Замовлення з\u0020</string>
<string name="general_price">Загальна ціна:\u0020</string>
</resources>

```

A.2 Файли виконавчого коду програми

A.2.1 Файл CorliverApp.kt

```

class CorliverApp : Application() {

    private lateinit var cicerone: Cicerone<Router>

    val navigatorHolder
        get() = cicerone.getNavigatorHolder()

    val router
        get() = cicerone.router

    override fun onCreate() {
        super.onCreate()
        instance = this
    }
}

```

```

        cicerone = Cicerone.create()

        startKoin {
            modules(test)
            androidContext(this@CorliverApp)
        }
    }

    companion object {
        lateinit var instance: CorliverApp

        inline fun <reified T> inject() = instance.getKoin().inject<T>()
    }
}

```

A.2.2 Файл MainActivity.kt

```

class MainActivity : BaseActivity<MainViewModel>() {

    override val viewModel: MainViewModel by viewModel()

    private val navigatorHolder: NavigatorHolder by inject()

    private val navigator = object : AppNavigator(this, R.id.main_container) {
        override fun applyCommands(commands: Array<out Command>) {
            super.applyCommands(commands)
            hideKeyboard()
        }
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        viewModel.startFlow()
    }

    override fun onPause() {
        super.onPause()
        navigatorHolder.removeNavigator()
    }

    override fun onResumeFragments() {
        super.onResumeFragments()
        navigatorHolder.setNavigator(navigator)
    }
}

```

A.2.3 Файл MainViewModel.kt

```

class MainViewModel(
    private val repository: CompanyRepository
) : BaseViewModel() {

    fun startFlow() {

        viewModelScope.launch {

            safeWithContextAsync(this) {

```

```

        repository.clear()
    }.await()
    router.replaceScreen(HomeFragment.homeScreen())
}
}
}

```

A.2.4 Файл BaseActivity.kt

```

abstract class BaseActivity<T : BaseViewModel> : AppCompatActivity(),
    MessageHolder {

    abstract val viewModel: T

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        onBindLiveData()
    }

    @CallSuper
    open fun onBindLiveData() {
        observe(viewModel.errorLiveData) {
            showError(it)
        }
    }

    open fun showError(error: GeneralError) {
        showError(error, null)
        viewModel.clearError()
    }

    fun hideKeyboard() {
        val inputMethodManager =
            getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager

        // Check if no view has focus
        val currentFocusedView = currentFocus
        currentFocusedView?.let {
            inputMethodManager.hideSoftInputFromWindow(
                currentFocusedView.windowToken,
                InputMethodManager.HIDE_NOT_ALWAYS
            )
        }
    }
}

```

A.2.5 Файл BaseFragment.kt

```

abstract class BaseFragment<T: BaseViewModel>(
    @LayoutRes private val layoutId: Int
) : Fragment(), MessageHolder {

    abstract val viewModel: T

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        if (this is RouterProvider) viewModel.attachAdditionalRouter(router())
        else parentFragment?.let { if (it is RouterProvider)

```

```

viewModel.attachAdditionalRouter(it.router()) }
}

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

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

@CallSuper
open fun onBindLiveData() {
    observe(viewModel.progressLiveData, this::handleProgress)

    observe(viewModel.errorLiveData) {
        showError(it)
    }
}

open fun showError(error: GeneralError) {
    context?.showError(error, null)
    viewModel.clearError()
}

abstract fun handleProgress(isInProgress: Boolean)

fun closeKeyboard() = (activity as? BaseActivity<*>)?.hideKeyboard()
}

```

A.2.6 Файл BaseUtilsExt.kt

```

inline fun <T : Any> LifecycleOwner.observe(
    liveData: LiveData<T>,
    crossinline onChanged: (T) -> Unit
) {
    liveData.observe(this, { result: T? -> result?.let { onChanged.invoke(it) } })
}

```

A.2.7 Файл BaseViewModel.kt

```

abstract class BaseViewModel() : ViewModel() {

    private val generalRouter by CorliverApp.inject<Router>()
    protected val errorHandler: ErrorHandler by CorliverApp.inject()
    protected val sessionManager: SessionManager by CorliverApp.inject()
    private var additionalRouter: Router? = null

    val errorLiveData: MutableLiveData<GeneralError> = MutableLiveData()
    val progressLiveData: MutableLiveData<Boolean> = MutableLiveData()
    val messageLiveData: MutableLiveData<Message> = MutableLiveData()

    protected val router: Router
        get() = additionalRouter ?: generalRouter
}

```

```

fun attachAdditionalRouter(r: Router) {
    additionalRouter = r
}

@CallSuper
open fun exit() = router.exit()

protected suspend fun <T> safeWithContextAsync(
    coroutineScope: CoroutineScope,
    block: suspend () -> T
): Deferred<T?> {
    return coroutineScope.async {
        try {
            block.invoke()
        } catch (t: Throwable) {
            t.printStackTrace()
            handleError(t)
            null
        }
    }
}

/**
 * Ignore handling error
 */
protected suspend fun <T> safeIgnoringWithContext(block: suspend
CoroutineScope.() -> T): T? {
    return try {
        withContext(Dispatchers.IO, block)
    } catch (t: Throwable) {
        t.printStackTrace()
        null
    }
}

fun clearError() {
    errorLiveData.value = null
}

fun clearMessage() {
    messageLiveData.value = null
}

@CallSuper
open fun handleError(t: Throwable) {
    if (!logoutIfNeeded(t))
        errorLiveData.postValue(errorHandler.parseError(t))
}

/**
 * @return true if need to logout, false - otherwise
 */
private fun logoutIfNeeded(throwable: Throwable): Boolean {
    if (errorHandler.isUnauthorizedError(throwable)) {
        logout()
        return true
    }
    return false
}

fun logout() {
    viewModelScope.launch {

```

```

        safeWithContextAsync(this) { sessionManager.clearUserData()
    }.await()
    }
}

```

A.2.8 Файл FragmentViewBindingDelegate.kt

```

class FragmentViewBindingDelegate<T : ViewBinding>(
    val fragment: Fragment,
    val viewBindingFactory: (View) -> T
) : ReadOnlyProperty<Fragment, T> {
    private var binding: T? = null

    init {
        fragment.lifecycle.addObserver(object : DefaultLifecycleObserver {
            val viewLifecycleOwnerLiveDataObserver =
                Observer<LifecycleOwner?> {
                    val viewLifecycleOwner = it ?: return@Observer

                    viewLifecycleOwner.lifecycle.addObserver(object :
DefaultLifecycleObserver {
                        override fun onDestroy(owner: LifecycleOwner) {
                            binding = null
                        }
                    })
                }

            override fun onCreate(owner: LifecycleOwner) {

fragment.viewLifecycleOwnerLiveData.observeForever(viewLifecycleOwnerLiveDataObs
erver)
            }

            override fun onDestroy(owner: LifecycleOwner) {

fragment.viewLifecycleOwnerLiveData.removeObserver(viewLifecycleOwnerLiveDataObs
erver)
            }

        })
    }

    override fun getValue(thisRef: Fragment, property: KProperty<*>): T {
        val binding = binding
        if (binding != null) {
            return binding
        }

        val lifecycle = fragment.viewLifecycleOwner.lifecycle
        if (!lifecycle.currentState.isAtLeast(Lifecycle.State.INITIALIZED)) {
            throw IllegalStateException("Should not attempt to get bindings when
Fragment views are destroyed.")
        }

        return viewBindingFactory(thisRef.requireView()).also { this.binding =
it }
    }
}

fun <T : ViewBinding> Fragment.viewBinding(viewBindingFactory: (View) -> T) =
    FragmentViewBindingDelegate(this, viewBindingFactory)

```

A.2.9 Файл ProgressView.kt

```
class ProgressView
@JvmOverloads
constructor(
    context: Context,
    attrs: AttributeSet? = null,
    defStyleAttr: Int = 0,
    defStyleRes: Int = 0
) : FrameLayout(context, attrs, defStyleAttr, defStyleRes) {

    private val binding = ViewProgressBinding.inflate(
        LayoutInflater.from(context),
        this
    )

    init {
        isClickable = true
        isFocusable = true
        isFocusableInTouchMode = true
    }

    fun setIsVisible(isVisible: Boolean) {
        visibleOrGone(isVisible)
    }
}
```

A.2.10 Файл OrderDao.kt

```
@Dao
interface OrderDao {

    @Query("select * from `order` where companyId=:companyId")
    suspend fun getOrders(companyId: String): List<OrderData>

    @Query("select * from `order` where companyId=:companyId and mealId=:mealId")
    suspend fun getSingleOrder(companyId: String, mealId: String): OrderData?

    @Insert(onConflict = REPLACE)
    suspend fun insert(orderData: OrderData)

    @Delete
    suspend fun delete(orderData: OrderData)

    @Query("delete from `order`")
    suspend fun clear()
}
```

A.2.11 Файл OrderData.kt

```
@Entity(tableName = "order", primaryKeys = ["companyId", "mealId"])
data class OrderData(
    val companyId: String,
    val mealId: String,
    val count: Int,
)
```

```
        val price: Float
    )
}
```

A.2.12 Файл OrderingDataBase.kt

```
private const val Version = 1
private const val Name = "ordering_db"

@Database(entities = [OrderData::class], version = Version, exportSchema = true)
abstract class OrderingDataBase : RoomDatabase() {

    abstract fun dao(): OrderDao

}

class OrderingDatabaseProvider(context: Context) {

    private val db = Room.databaseBuilder(context, OrderingDataBase::class.java,
Name)
        .fallbackToDestructiveMigration()
        .build()

    fun getOrderDao(): OrderDao = db.dao()

}
```

A.2.13 Файл DeeplinkResolver.kt

```
interface DeeplinkResolver {

    suspend fun resolveData(params: DeeplinkParams) : DeeplinkResult?

}

data class DeeplinkParams(
    val params: Map<String, String>
)

@Parcelize
data class DeeplinkResult(
    val data: String?
) : Parcelable
```

A.2.14 Файл MockedDeeplinkResolver.kt

```
class MockedDeeplinkResolver : DeeplinkResolver {

    override suspend fun resolveData(params: DeeplinkParams): DeeplinkResult? {
        return null
    }

}
```

A.2.15 Файл KoinModule.kt

```
val module = module {

    single { CorliverApp.instance.router }
    single { CorliverApp.instance.navigatorHolder }
    single { ErrorHandler(get()) }
    single { Gson() }

}
```

```

single { SessionManager(get()) }
single { PreferencesManager(get()) }
single<DeeplinkResolver> { MockedDeeplinkResolver() }
single<AuthRepository> { MockedAuthRepository(get()) }
single<CompanyRepository> { MockedCompanyRepository(get()) }
single<UserRepository> { MockedUserRepository() }
single { OrderingDatabaseProvider(get()) }
single<CartRepository> { MockedCartRepository(get()) }
single<OrdersHistoryRepository> { MockedOrdersRepository() }

viewModel { MainViewModel(get()) }
viewModel { SplashViewModel(get(), get()) }
viewModel { LoginViewModel(get()) }
viewModel { RegisterViewModel(get()) }
viewModel { SmsViewModel(get()) }
viewModel { HomeViewModel(get(), get()) }

viewModel { CompanyDetailsViewModel(get()) }
viewModel { MealViewModel(get()) }
viewModel { CartViewModel(get()) }
viewModel { ProfileViewModel() }
viewModel { OrdersHistoryViewModel(get()) }
}

```

A.2.16 Файл ErrorHandler.kt

```

class ErrorHandler(
    private val gson: Gson
) {

    fun parseError(t: Throwable): GeneralError {
        return when {
            t.isApiError() -> (t as HttpException).proceedApiError()
            t.isNetworkError() -> t.proceedNetworkError()
            else -> t.toGeneralError()
        }
    }

    private fun Throwable.isNetworkError(): Boolean {
        return this is SocketException || this is UnknownHostException || this
is SocketTimeoutException
    }

    private fun Throwable.isApiError() = this is HttpException

    private fun Throwable.toGeneralError(): GeneralError {
        return GeneralError(this, message, R.string.error_unknown)
    }

    private fun HttpException.proceedApiError(): GeneralError {
        val error = response()?.errorBody()?.string()
        val errorBody = try {
            gson.fromJson(error ?: "", ErrorResponse::class.java)
        } catch (ex: Throwable) {
            ex.printStackTrace()
            null
        }
        return GeneralError(this, errorBody?.message, R.string.error_api_error)
    }

    private fun Throwable.proceedNetworkError(): GeneralError {

```

```

        return GeneralError(this, null, R.string.error_api_error)
    }

    fun isUnauthorizedError(t: Throwable): Boolean {
        return t is HttpException && t.code() ==
        HttpURLConnection.HTTP_UNAUTHORIZED
    }
}

data class ErrorResponse(
    val code: Int?,
    val status: Int?,
    val message: String?
)

```

A.2.17 Файл GeneralError.kt

```

open class GeneralError(
    override val cause: Throwable,
    override val message: String? = null,
    @StringRes
    val errorMessageId: Int
) : Throwable(message, cause)

```

A.2.18 Файл ViewExt.kt

```

fun View.gone() {
    if (visibility != View.GONE) {
        visibility = View.GONE
    }
}

fun View.visible() {
    if (visibility != View.VISIBLE) {
        visibility = View.VISIBLE
    }
}

fun View.invisible() {
    if (visibility != View.INVISIBLE) {
        visibility = View.INVISIBLE
    }
}

fun View.visibleOrGone(visible: Boolean) {
    visibility = if (visible) View.VISIBLE else View.GONE
}

fun View.visibleOrInvisible(visible: Boolean) {
    visibility = if (visible) View.VISIBLE else View.INVISIBLE
}

fun RecyclerView.ViewHolder.context(): Context = itemView.context

fun ViewGroup.inflater() = LayoutInflater.from(context)

```

A.2.19 Файл NumberUtils.kt

```

object NumberUtils {

```

```

fun getTotalMealsPrice(meals: List<CartMeal>): BigDecimal {
    val total = meals.map { it.meal.price.multiply(BigDecimal(it.count)) }
    val sum = total.sumOf { it }
    return sum
}
}

```

A.2.20 Файл SessionManager.kt

```

class SessionManager(
    private val preferencesManager: PreferencesManager
) {

    suspend fun saveUser(token: String): String? {
        clearUserData()
        preferencesManager.put(KeyUserToken, token, String::class.java)
        return userToken()
    }

    suspend fun userToken(): String? {
        return preferencesManager.read(KeyUserToken, "", String::class.java)
    }

    suspend fun isSessionActive() = !userToken().isNullOrBlank()

    suspend fun clearUserData() {
        preferencesManager.clear(KeyUserToken)
    }

    companion object {
        private const val KeyUserToken = "KeyUserToken"
    }
}

```

A.2.21 Файл User.kt

```

data class User(
    val id: String
)

```

A.2.22 Файл MessageHolder.kt

```

typealias Message = String

interface MessageHolder {

    fun Context.showError(error: GeneralError, onAction: (() -> Unit)? = null) {
        if (error.message != null) showError(error.message ?: "", onAction)
        else showError(error.errorMessageId, onAction)
    }

    fun Context.showTryAgainError(
        error: GeneralError,
        onPositiveAction: (() -> Unit)?,
        onNegativeAction: (() -> Unit)?
    ) {
        val titleView = dialogTitleTextView(getString(R.string.dialog_error))
    }
}

```

```

val dialog = AlertDialog.Builder(this)
    .setCustomTitle(titleView)
    .setMessage(error.message ?: getString(error.errorMessageId))
    .setPositiveButton(getString(R.string.error_try_again)) { _, _ ->
        onPositiveAction?.invoke()
    }
    .setNegativeButton(getString(R.string.error_cancel)) { _, _ ->
        onNegativeAction?.invoke()
    }
    .setOnCancelListener {
        onNegativeAction?.invoke()
        it.dismiss()
    }
    .create()

dialog.show()

val messageText: TextView? = dialog.findViewById(R.id.message)
messageText?.gravity = Gravity.CENTER
}

fun Context.showError(@StringRes strResId: Int, onAction: (() -> Unit)?) {
    showError(getString(strResId), onAction)
}

fun Context.showError(error: String, onAction: (() -> Unit)?) {
    showDialog(getString(R.string.dialog_error), error, onAction)
}

fun Context.showDialog(title: String?, message: String?, onAction: (() ->
Unit)?) {
    showDialog(title, message, onAction, onAction, onAction)
}

fun Context.showDialog(
    title: String?,
    message: String?,
    @StringRes positiveButtonText: Int = R.string.dialog_ok,
    @StringRes negativeButtonText: Int? = null,
    isCancelable: Boolean,
    onPositiveAction: (() -> Unit)?,
    onNegativeAction: (() -> Unit)?,
    onCancel: (() -> Unit)?
) {
    val titleView = dialogTitleTextView(title)

    val dialogBuilder = AlertDialog.Builder(this)
        .setCustomTitle(titleView)
        .setMessage(message)
        .setPositiveButton(positiveButtonText) { _, _ ->
            onPositiveAction?.invoke()
        }
        .setCancelable(isCancelable)

    if (isCancelable) {
        dialogBuilder.setOnCancelListener {
            onCancel?.invoke()
            it.dismiss()
        }
    }

    negativeButtonText?.let {

```

```

        dialogBuilder.setNegativeButton(negativeButtonText) { _, _ ->
            onNegativeAction?.invoke()
        }
    }

    val dialog = dialogBuilder.create()
    dialog.show()

    val messageText: TextView? = dialog.findViewById(R.id.message)
    messageText?.gravity = Gravity.CENTER
}

fun Context.showDialog(
    title: String?,
    message: String?,
    onPositiveAction: (() -> Unit)?,
    onNegativeAction: (() -> Unit)?,
    onCancel: (() -> Unit)?
) {
    val titleView = dialogTitleTextView(title)

    val dialog = AlertDialog.Builder(this)
        .setCustomTitle(titleView)
        .setMessage(message)
        .setPositiveButton(R.string.dialog_ok) { _, _ ->
            onPositiveAction?.invoke()
        }
        .setOnCancelListener {
            onCancel?.invoke()
            it.dismiss()
        }
        .create()

    dialog.show()

    val messageText: TextView? = dialog.findViewById(R.id.message)
    messageText?.gravity = Gravity.CENTER
}

fun Context.showSingleChoiceDialog(
    title: String?,
    items: List<String>,
    defaultSelection: Int = -1,
    onOptionSelected: (Int) -> Unit
) {
    val titleView = dialogTitleTextView(title)
    AlertDialog.Builder(this)
        .setCustomTitle(titleView)
        .setSingleChoiceItems(items.toTypedArray(), defaultSelection) {
dialog, which ->
            onOptionSelected.invoke(which)
            dialog.dismiss()
        }
        .show()
}

fun Context.showListDialog(
    title: String?,
    items: List<String>,
    isCancelable: Boolean = true,
    onOptionSelected: (Int) -> Unit,
    @StringRes negativeButtonText: Int? = null,
    onNegativeAction: (() -> Unit)? = null

```

```

    ) {
        val titleView = dialogTitleTextView(title)
        AlertDialog.Builder(this)
            .setCustomTitle(titleView)
            .setItems(items.toTypedArray()) { dialog, which ->
                onOptionSelected.invoke(which)
                dialog.dismiss()
            }
            .setCancelable(isCancelable)
            .apply {
                if (negativeButtonText != null) {
                    setNegativeButton(negativeButtonText) { _, _ ->
                        onNegativeAction?.invoke()
                    }
                }
            }
            .show()
    }
}

private fun Context.dialogTitleTextView(title: String?): TextView? {
    if (title.isNullOrBlank()) return null
    return TextView(this).apply {
        text = title
        setPadding(10, 20, 10, 0)
        gravity = Gravity.CENTER
        setTextColor(Color.BLACK)
        textSize = 20f
    }
}
}

```

A.2.23 Файл PreferencesManager.kt

```

class PreferencesManager(context: Context) {

    private val preferences = context.getSharedPreferences(PREFS_NAME,
        MODE_PRIVATE)

    fun <T> put(key: String, value: T, type: Class<T>) {
        preferences.edit {
            when (type) {
                String::class.java -> {
                    putString(key, value as String)
                }
                Int::class.java -> {
                    putInt(key, value as Int)
                }
                Boolean::class.java -> {
                    putBoolean(key, value as Boolean)
                }
                Float::class.java -> {
                    putFloat(key, value as Float)
                }
                Long::class.java -> {
                    putLong(key, value as Long)
                }
            }
        }
    }

    @Suppress("UNCHECKED_CAST")
    fun <T> read(key: String, defValue: T?, type: Class<T>): T? {

```

```

val result = when (type) {
    String::class.java -> {
        preferences.getString(key, defValue as String)
    }
    Int::class.java -> {
        preferences.getInt(key, defValue as Int)
    }
    Boolean::class.java -> {
        preferences.getBoolean(key, defValue as Boolean)
    }
    Float::class.java -> {
        preferences.getFloat(key, defValue as Float)
    }
    Long::class.java -> {
        preferences.getLong(key, defValue as Long)
    }
    else -> null
}
if (result != null)
    return result as T
return defValue
}

fun clear(key: String) {
    preferences.edit().remove(key).apply()
}

fun contains(key: String): Boolean = preferences.contains(key)

companion object {
    private const val PREFS_NAME = "general_prefs"
}
}

```

A.2.24 Файл UserRepository.kt

```

interface UserRepository {

    fun user(forceRefresh: Boolean): UserModel

}

```

A.2.25 Файл UserModel.kt

```

@Parcelize
data class UserModel(
    val id: Int,
    val name: String,
    val phone: String,
    val email: String,
    val company: CompanyModel
): Parcelable

```

A.2.26 Файл SplashFragment.kt

```

class SplashFragment : BaseFragment<SplashViewModel>(R.layout.fragment_splash) {

    override val viewModel: SplashViewModel by viewModel()
    private val binding by viewBinding<FragmentSplashBinding>::bind
}

```

```

override fun handleProgress(isInProgress: Boolean) {

}

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    viewModel.checkSession()
}

companion object {

    fun SplashScreen() = FragmentScreen { SplashFragment() }

}
}

```

A.2.27 Файл SplashViewModel.kt

```

class SplashViewModel(
    private val userSessionManager: SessionManager,
    private val deeplinkResolver: DeeplinkResolver
) : BaseViewModel() {

    fun checkSession() {
        viewModelScope.launch {
            val isSessionActive = safeWithContextAsync(this) {
                userSessionManager.isSessionActive()
            }.await() ?: false

            val deeplink = safeWithContextAsync(this) {
                deeplinkResolver.resolveData(
                    DeeplinkParams(
                        emptyMap()
                    )
                )
            }.await()

            router.replaceScreen(
                when {
                    isSessionActive -> HomeFragment.homeScreen()
                    deeplink != null -> LoginScreen()
                    else -> LoginScreen()
                }
            )
        }
    }
}

```

A.2.28 Файл ProfileFragment.kt

```

class ProfileFragment :
    BaseFragment<ProfileViewModel>(R.layout.fragment_profile) {

    override val viewModel: ProfileViewModel by viewModel()
    private val binding by viewBinding<FragmentProfileBinding>::bind
    private val user by lazy {
        requireArguments().getParcelable<UserModel>(KeyUserModel)!!
    }
}

```

```

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

        back.setOnClickListener { viewModel.exit() }
        profileName.text = user.name
        profileEmail.text = user.email
        profilePhone.text = PhoneNumberUtils.formatNumber(user.phone,
Locale.getDefault().isO3Country)

        profileOrders.setOnClickListener {
            viewModel.openOrders(user)
        }
    }
}

override fun handleProgress(isInProgress: Boolean) {
    binding.progressBar.visibleOrGone(isInProgress)
}

companion object {

    private const val KeyUserModel = "user_model"

    fun profileScreen(user: UserModel) = FragmentScreen {
        ProfileFragment().apply {
            arguments = Bundle().apply {
                putParcelable(KeyUserModel, user)
            }
        }
    }
}
}

```

A.2.29 Файл ProfileViewModel.kt

```

class ProfileViewModel : BaseViewModel() {

    fun openOrders(user: UserModel) {
        router.navigateTo(OrdersHistoryFragment.ordersScreen())
    }
}

```

A.2.30 Файл HomeFragment.kt

```

class HomeFragment : BaseFragment<HomeViewModel>(R.layout.fragment_home) {

    override val viewModel: HomeViewModel by viewModel()
    private val binding by viewBinding(FragmentHomeBinding::bind)

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        viewModel.loadHomeData()
    }

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

```

```

        super.onViewCreated(view, savedInstanceState)
        binding.homeCompanies.addItemDecoration(
            DividerItemDecoration(
                requireContext(),
                LinearLayoutManager.VERTICAL
            )
        )
    }

    override fun onBindLiveData() {
        super.onBindLiveData()
        observe(viewModel.homeLiveData, this::handleHomeData)
    }

    private fun handleHomeData(data: HomeScreenData) {
        binding.homeCompanyTitle.text = data.user.company.name
        binding.homeProfileImage.setOnClickListener {
            viewModel.navigateToUser(data.user)
        }
        handleCompanies(data.companies)
    }

    private fun handleCompanies(companies: List<CompanyModel>) {
        binding.homeCompanies.layoutManager =
            LinearLayoutManager(requireContext())
        binding.homeCompanies.adapter = HomeCompaniesAdapter(object :
            HomeCompaniesAdapter.OnHomeItemClickListener {
                override fun onClick(company: CompanyModel) {
                    handleCompanySelected(company)
                }
            })
        binding.homeCompanies.apply {
            setItems(companies)
        }
    }

    private fun handleCompanySelected(company: CompanyModel) {
        viewModel.navigateToCompanyInfo(company)
    }

    override fun handleProgress(isInProgress: Boolean) {
        binding.progressBar.visibleOrGone(isInProgress)
    }

    companion object {
        fun homeScreen() = FragmentScreen { HomeFragment() }
    }
}

```

A.2.31 Файл HomeScreenData.kt

```

data class HomeScreenData(
    val user: UserModel,
    val companies: List<CompanyModel>
)

```

A.2.32 Файл HomeViewModel.kt

```

class HomeViewModel(
    private val companyRepository: CompanyRepository,

```

```

        private val userRepository: UserRepository
    ) : BaseViewModel() {

        val homeLiveData = MutableLiveData<HomeScreenData>()

        fun loadHomeData() {
            viewModelScope.launch {
                progressLiveData.postValue(true)
                val result = safeWithContextAsync(this) {
                    val companies = companyRepository.getListOfSubscribedCompanies()
                    val user = userRepository.user(false)
                    HomeScreenData(user, companies)
                }.await()
                homeLiveData.postValue(result)
                progressLiveData.postValue(false)
            }
        }

        fun navigateToCompanyInfo(company: CompanyModel) {
            router.navigateTo(CompanyDetailsFragment.screen(company.id))
        }

        fun navigateToUser(user: UserModel) {
            router.navigateTo(ProfileFragment.profileScreen(user))
        }
    }
}

```

A.2.33 Файл HomeCompaniesAdapter.kt

```

class HomeCompaniesAdapter(
    private val onHomeClickListener: OnHomeItemClickListener
) : RecyclerView.Adapter<HomeCompaniesViewHolder>() {

    private val items = mutableListOf<CompanyModel>()

    fun setItems(new: List<CompanyModel>) {
        items.clear()
        items.addAll(new)
        notifyDataSetChanged()
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
        HomeCompaniesViewHolder {
        return HomeCompaniesViewHolder.newInstance(parent)
    }

    override fun onBindViewHolder(holder: HomeCompaniesViewHolder, position:
        Int) {
        holder.bind(items[position], onHomeClickListener)
    }

    override fun getItemCount(): Int = items.size

    interface OnHomeItemClickListener {

        fun onClick(company: CompanyModel)

    }
}

class HomeCompaniesViewHolder(

```

```

        private val binding: ItemHomeCompanyBinding
    ) : RecyclerView.ViewHolder(binding.root) {

        fun bind(
            company: CompanyModel,
            onHomeClickListener: HomeCompaniesAdapter.OnHomeItemClickListener
        ) {

            binding.root.setOnClickListener { onHomeClickListener.onClick(company) }
            binding.apply {
                companyDescription.text = company.description
                companyName.text = company.name
                Glide.with(context())
                    .load(company.image)
                    .centerCrop()
                    .placeholder(R.drawable.outline_image)
                    .into(companyImage)
            }
        }

        companion object {

            fun newInstance(parent: ViewGroup): HomeCompaniesViewHolder {
                return HomeCompaniesViewHolder(
                    ItemHomeCompanyBinding.inflate(
                        LayoutInflater.from(parent.context),
                        parent,
                        false
                    )
                )
            }
        }
    }
}

```

A.2.34 Файл OrdersHistoryFragment.kt

```

class OrdersHistoryFragment :
    BaseFragment<OrdersHistoryViewModel>(R.layout.fragment_orders) {

    override val viewModel: OrdersHistoryViewModel by viewModel()
    private val binding by viewBinding<FragmentOrdersBinding>::bind

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        viewModel.load()
    }

    override fun onBindLiveData() {
        super.onBindLiveData()
        observe(viewModel.ordersLiveData, this::handleOrdersData)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        binding.back.setOnClickListener { viewModel.exit() }
    }

    override fun handleProgress(isInProgress: Boolean) {
        //TODO("Not yet implemented")
    }

    private fun handleOrdersData(orders: List<OrderHistoryModel>) {

```

```

        with(binding) {
            orderHistoryList.layoutManager =
                LinearLayoutManager(requireContext())
            orderHistoryList.adapter = OrdersAdapter(orders, object :
                OnOrderSelectedListener {
                    override fun onOrderSelected(order: OrderHistoryModel) {
                        //todo
                    }
                })
        }
    }
}

companion object {
    fun ordersScreen() = FragmentScreen {
        OrdersHistoryFragment()
    }
}
}

```

A.2.35 Файл OrdersHistoryViewModel.kt

```

class OrdersHistoryViewModel(
    private val repository: OrdersHistoryRepository,
) : BaseViewModel() {

    val ordersLiveData = MutableLiveData<List<OrderHistoryModel>>()

    fun load() {
        viewModelScope.launch {
            safeWithContextAsync(this) {
                repository.getOrders()
            }.await()?.also(ordersLiveData::postValue)
        }
    }
}

```

A.2.36 Файл CompanyDetailsFragment.kt

```

private const val KeyCompanyId = "company_id"

class CompanyDetailsFragment :
    BaseFragment<CompanyDetailsViewModel>(R.layout.fragment_company_details) {

    override val viewModel: CompanyDetailsViewModel by viewModel()
    private val binding by viewBinding<FragmentCompanyDetailsBinding>::bind

    private val companyId by lazy { requireArguments().getString(KeyCompanyId)
        ?: "" }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        viewModel.load(companyId)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        binding.back.setOnClickListener { viewModel.exit() }
        binding.openCart.setOnClickListener { viewModel.openCart(companyId) }
    }
}

```

```

override fun onResume() {
    super.onResume()
    viewModel.checkOrder(companyId)
}

override fun onBindLiveData() {
    super.onBindLiveData()
    observe(viewModel.companyLiveData, this::handleCompany)
    observe(viewModel.orderLiveData, this::handleOrder)
}

private fun handleOrder(order: List<OrderData>) {
    if (order.isEmpty()) {
        binding.openCartLayout.gone()
    } else {
        binding.openCartLayout.visible()

        var count = 0
        var totalPrice = 0f
        order.forEach {
            count += it.count
            totalPrice += it.count * it.price
        }
        binding.openCart.text = getString(R.string.open_cart, count,
totalPrice)
    }
}

private fun handleCompany(detailModel: CompanyDetailModel) {
    binding.apply {

        Glide.with(requireContext())
            .load(detailModel.company.image)
            .centerCrop()
            .into(companyImage)

        companyName.text = detailModel.company.name
        companyDescription.text = detailModel.company.description

        companyMenu.isNestedScrollingEnabled = false
        companyMenu.layoutManager = LinearLayoutManager(context,
LinearLayoutManager.VERTICAL, false)
        companyMenu.adapter = CompanyMealsAdapter(detailModel.meals()) {
            viewModel.openMealDetails(it, companyId)
        }
    }
}

override fun handleProgress(isInProgress: Boolean) {
    binding.progressBar.visibleOrGone(isInProgress)
}

companion object {

    fun screen(id: String) = FragmentScreen {
        CompanyDetailsFragment().apply {
            arguments = Bundle().apply {
                putString(KeyCompanyId, id)
            }
        }
    }
}

```

```

    }
}

```

A.2.37 Файл CompanyDetailsViewModel.kt

```

class CompanyDetailsViewModel(
    private val companyRepository: CompanyRepository
) : BaseViewModel() {

    val companyLiveData = MutableLiveData<CompanyDetailModel>()
    val orderLiveData = MutableLiveData<List<OrderData>>()

    fun load(id: String) {
        viewModelScope.launch {
            progressLiveData.postValue(true)
            safeWithContextAsync(this) {
                companyRepository.getCompanyInfo(id)
            }.await()?.let {
                companyLiveData.postValue(it)
            }
            progressLiveData.postValue(false)
        }
    }

    fun openMealDetails(mealModel: MealModel, companyId: String) {
        router.navigateTo(MealFragment.mealScreen(mealModel, companyId))
    }

    override fun exit() {
        //todo check if user has active menu

        super.exit()
    }

    fun checkOrder(companyId: String) {
        viewModelScope.launch {

            safeWithContextAsync(this) {
                companyRepository.getOrder(companyId)
            }.await()?.let {
                orderLiveData.postValue(it)
            }
        }
    }

    fun openCart(companyId: String) {
        router.navigateTo(cartScreen(companyId))
    }
}

```

A.2.38 Файл MealFragment.kt

```

private const val KeyMealModel = "meal_model"
private const val KeyCompanyId = "company_id"

class MealFragment : BaseFragment<MealViewModel>(R.layout.fragment_meal) {

    override val viewModel: MealViewModel by viewModel()
}

```

```

        private val binding by viewBinding<FragmentMealBinding>::bind
        private val meal by lazy {
            requireArguments().getParcelable<MealModel>(KeyMealModel)!! }
        private val companyId by lazy { requireArguments().getString(KeyCompanyId)!! }
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        viewModel.load(companyId, meal)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        binding.apply {
            menuAdd.setOnClickListener {
                viewModel.increase(companyId, meal,
                menuCounter.text.toString().toIntOrNull() ?: 0)
            }
            menuRemove.setOnClickListener {
                viewModel.decrease(companyId, meal,
                menuCounter.text.toString().toIntOrNull() ?: 0)
            }

            back.setOnClickListener { viewModel.exit() }
        }
    }

    override fun onBindLiveData() {
        super.onBindLiveData()
        observe(viewModel.dataLiveData, this::handleMealData)
    }

    override fun handleProgress(isInProgress: Boolean) {
        binding.progressBar.visibleOrGone(isInProgress)
    }

    private fun handleMealData(data: LocalMealData) {
        binding.apply {

            val meal = data.meal
            Glide.with(root)
                .load(meal.image)
                .centerCrop()
                .into(mealImage)

            mealName.text = meal.name
            mealDescription.text = meal.desc
            menuCounter.text = data.amount.toString()
            mealPrice.text = String.format("%.2f", meal.price)
            totalPrice.text = getString(R.string.general_price) +
            String.format("%.2f", data.totalPrice.toDouble()) + " ррн."

        }
    }

    companion object {

        fun mealScreen(mealModel: MealModel, companyId: String) = FragmentScreen
    }
}

MealFragment().apply {
    arguments = Bundle().apply {
        putParcelable(KeyMealModel, mealModel)
        putString(KeyCompanyId, companyId)
    }
}

```

```

        }
    }
}

}

```

A.2.39 Файл MealViewModel.kt

```

class MealViewModel(
    private val repository: CompanyRepository
) : BaseViewModel() {

    val dataLiveData = MutableLiveData<LocalMealData>()

    fun load(companyId: String, mealModel: MealModel) {

        viewModelScope.launch {

            progressLiveData.postValue(true)

            safeWithContextAsync(this) {
                repository.getLocalMeal(companyId, mealModel)
            }.await()?.let {
                dataLiveData.postValue(it)
            } ?: router.exit()

            progressLiveData.postValue(false)

        }

    }

    fun increase(companyId: String, meal: MealModel, currentAmount: Int) {
        viewModelScope.launch {

            progressLiveData.postValue(true)

            safeWithContextAsync(this) {
                repository.updateMeal(companyId, LocalMealData(meal,
currentAmount + 1, BigDecimal.ZERO))
            }.await()
            load(companyId, meal)
            progressLiveData.postValue(false)

        }

    }

    fun decrease(companyId: String, meal: MealModel, currentAmount: Int) {
        if (currentAmount == 0) return

        viewModelScope.launch {

            progressLiveData.postValue(true)

            safeWithContextAsync(this) {
                repository.updateMeal(companyId, LocalMealData(meal,
currentAmount - 1, BigDecimal.ZERO))
            }.await()
            load(companyId, meal)

        }

    }

}

```

```

        progressLiveData.postValue(false)
    }
}
}

```

A.2.40 Файл CartFragment.kt

```

private const val KeyCompanyId = "companyId"

class CartFragment : BaseFragment<CartViewModel>(R.layout.fragment_cart) {

    override val viewModel: CartViewModel by viewModel()
    private val binding by viewBinding<FragmentCartBinding>::bind
    private val companyId: String by lazy {
        requireArguments().getString(KeyCompanyId)!!
    }

    private val cartListener = object : CartListener {
        override fun onDelete(cart: CartMeal) {
            viewModel.delete(companyId, cart)
        }

        override fun onClick(cart: CartMeal) {
            viewModel.open(companyId, cart)
        }
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        viewModel.loadCart(companyId)
        binding.back.setOnClickListener { viewModel.exit() }
    }

    override fun onBindLiveData() {
        super.onBindLiveData()
        observe(viewModel.cartLiveData, this::handleCart)
        observe(viewModel.orderApprovedLiveData, this::handleOrderApproved)
    }

    private fun handleOrderApproved(isApproved: Boolean) {
        if (isApproved) {
            AlertDialog.Builder(requireContext())
                .setTitle(getString(R.string.ready))
                .setMessage(getString(R.string.order_sended_message))
                .setPositiveButton(R.string.dialog_ok) { d, v ->
                    d.dismiss()
                    viewModel.endOrderFlow()
                }
                .setCancelable(false)
                .show()
        }
    }

    private fun handleCart(cart: Cart) {
        with(binding) {
            cartTopTitle.text = getString(R.string.order_from) +
            cart.company.name
            cartOrderList.adapter = CartAdapter(cart.meals, cartListener)
            cartOrderList.layoutManager = LinearLayoutManager(context)
            cartOrderTotalSum.text = getString(R.string.general_price) +

```

```

getTotalMealsPrice(cart.meals) + " грн."
        cartConfirm.setOnClickListener { viewModel.applyCart(cart) }
    }
}

override fun handleProgress(isInProgress: Boolean) {
    binding.progressBar.visibleOrGone(isInProgress)
}

companion object {

    fun cartScreen(companyId: String) = FragmentScreen {
        CartFragment().apply {
            arguments = Bundle().apply {
                putString(KeyCompanyId, companyId)
            }
        }
    }

}
}

```

A.2.41 Файл CartViewModel.kt

```

class CartViewModel(
    private val cartRepository: CartRepository
) : BaseViewModel() {

    val cartLiveData = MutableLiveData<Cart>()
    val orderApprovedLiveData = MutableLiveData<Boolean>()

    fun loadCart(companyId: String) {
        viewModelScope.launch {
            progressLiveData.postValue(true)

            val cartInfo = safeWithContextAsync(this) {
                cartRepository.loadCart(companyId)
            }.await()
            cartLiveData.postValue(cartInfo)
            progressLiveData.postValue(false)
        }
    }

    fun clear() {
        viewModelScope.launch {

            progressLiveData.postValue(true)
            safeWithContextAsync(this) {
                cartRepository.clear()
            }.await()
            progressLiveData.postValue(false)

            router.newRootScreen(homeScreen())
        }
    }

    fun applyCart(cart: Cart) {
        viewModelScope.launch {
            progressLiveData.postValue(true)

```

```

        safeWithContextAsync(this) {
            cartRepository.applyOrder(cart)
        }.await()?.let {
            orderApprovedLiveData.postValue(it)
        }

        progressLiveData.postValue(false)
    }
}

fun open(companyId: String, cart: CartMeal) {
    router.navigateTo(mealScreen(cart.meal, companyId))
}

fun delete(companyId: String, cart: CartMeal) {
    viewModelScope.launch {
        progressLiveData.postValue(true)

        safeWithContextAsync(this) {
            cartRepository.remove(companyId, cart.meal)
        }.await()?.let {
            loadCart(companyId)
        }

        progressLiveData.postValue(false)
    }
}

fun endOrderFlow() {
    router.newRootChain(homeScreen())
}
}

```

A.2.42 Файл LoginFragment.kt

```

class LoginFragment
private constructor() : BaseFragment<LoginViewModel>(R.layout.fragment_login) {

    private val binding by viewBinding<FragmentLoginBinding>::bind

    override val viewModel: LoginViewModel by viewModel()

    override fun handleProgress(isInProgress: Boolean) {
        binding.progressBar.visibleOrGone(isInProgress)
    }

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

        binding.loginSignInButton.setOnClickListener {
            viewModel.signIn(
                binding.loginEmailInput.text.toString(),
                binding.loginPasswordInput.text.toString()
            )
        }

        binding.loginSignUpButton.setOnClickListener {
            viewModel.signUp(binding.loginEmailInput.text?.toString() ?: "")
        }
    }
}

```

```

    }

    override fun onBindLiveData() {
        super.onBindLiveData()
        observe(viewModel.stateLiveData, this::handleState)
    }

    private fun handleState(state: LoginViewModel.LoginState) {
        handleProgress(state.isInProgress)

        binding.loginEmail.setError(state.emailValidationSucceed,
R.string.validation_error_email)
        binding.loginPassword.setError(
            state.passwordValidationSucceed,
            R.string.validation_error_password
        )
    }

    private fun TextInputLayout.setError(isOk: Boolean, @StringRes error: Int) {
        setError(if (isOk) null else getString(error))
    }

    companion object {

        fun LoginScreen() = FragmentScreen{ newInstance() }

        fun newInstance(): LoginFragment {
            return LoginFragment().apply {
                arguments = Bundle()
            }
        }

    }
}

```

A.2.43 Файл LoginViewModel.kt

```

class LoginViewModel(
    private val authRepository: AuthRepository
) : BaseViewModel() {

    private val _stateLiveData = MutableLiveData<LoginState>()
    val stateLiveData: LiveData<LoginState>
        get() = _stateLiveData

    fun signIn(email: String, password: String) {
        viewModelScope.launch {
            _stateLiveData.postValue(
                LoginState(
                    isInProgress = true,
                    emailValidationSucceed = true,
                    passwordValidationSucceed = true
                )
            )

            val isEmailValid =
PatternsCompat.EMAIL_ADDRESS.matcher(email).matches()
            val isPasswordValid = passwordMatcher().matcher(password).matches()

            if (!isEmailValid || !isPasswordValid) {
                _stateLiveData.postValue(LoginState(
                    isInProgress = false,

```

```

        emailValidationSucceed = isEmailValid,
        passwordValidationSucceed = isPasswordValid
    ))
    return@launch
}

val loginResult = safeWithContextAsync(this) {
    authRepository.login(email, password)
}.await()

_stateLiveData.postValue(LoginState(
    isInProgress = false,
    emailValidationSucceed = true,
    passwordValidationSucceed = true
))

if (loginResult?.isSucceed == true) {
    router.replaceScreen(homeScreen())
}
}

fun signUp(email: String) {
    router.navigateTo(RegisterScreen(RegisterDataContainer(email)))
}

data class LoginState(
    val isInProgress: Boolean,
    val emailValidationSucceed: Boolean,
    val passwordValidationSucceed: Boolean
)

private fun passwordMatcher() = Pattern.compile(
    "^(?=.*[0-9])(?=.*[a-z])(?=.*[A-
Z])(?=.*[!@#\\$%^&*~])(?=\\S+$).{8,256}$",
)
}

```

A.2.44 Файл RegisterFragment.kt

```

class RegisterFragment
private constructor() :
BaseFragment<RegisterViewModel>(R.layout.fragment_register) {

    private val params by lazy {
        arguments?.getParcelable<RegisterDataContainer>(
            BundleKeyRegisterParams
        )!!
    }
    private val binding by viewBinding(FragmentRegisterBinding::bind)

    override val viewModel: RegisterViewModel by viewModel()

    override fun handleProgress(isInProgress: Boolean) {
        binding.progressView.visibleOrGone(isInProgress)
    }

    private val scanQrCode = registerForActivityResult(ScanQRCode(),
::handleResult)
    private var secretKey: String? = null
}

```

```

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

    checkSecretKeyState()
    binding.apply {
        registerEmail.editText?.setText(params.email)

        registerSignInButton.setOnClickListener {
            viewModel.backToSignIn()
        }

        registerScanQrButton.setOnClickListener {
            if (secretKey.isNullOrBlank()) {
                scanQrCode.launch(null)
            } else {
                signUp()
            }
        }
    }
}

private fun handleResult(result: QRResult) {
    when (result) {
        // is QRResult.QRSuccess -> {
        //     secretKey = result.content.rawValue
        //     Log.d("handleResult", secretKey.toString())
        //     signUp()
        // }
        QRResult.QRUserCanceled -> Unit
        QRResult.QRMissingPermission -> Unit
        is QRResult.QRError -> Unit
    }
    checkSecretKeyState()
}

private fun signUp() {
    binding.apply {
        viewModel.signUp(
            SignUpParams(
                registerEmailInput.text.toString(),
                registerPasswordInput.text.toString(),
                registerConfirmPasswordInput.text.toString(),
                registerNameInput.text.toString(),
                registerPhoneInput.text.toString(),
            ),
            secretKey ?: return scanQrCode.launch(null)
        )
    }
}

private fun Editable?.toString() = this?.toString() ?: ""

override fun onBindLiveData() {
    super.onBindLiveData()
    observe(viewModel.stateLiveData, this::handleState)
}

private fun handleState(state: RegisterViewModel.RegisterState) {
    handleProgress(state.isInProgress)

    binding.apply {
        registerEmail.setError(state.isEmailValid,

```

```

R.string.validation_error_email)
        registerPassword.setError(state.isPasswordValid,
R.string.validation_error_password)
        registerConfirmPassword.setError(
            state.arePasswordsTheSame,
            R.string.validation_error_passwords_are_not_same
        )
        registerPhone.setError(state.isPhoneValid,
R.string.validation_error_phone)
        registerName.setError(state.isNameValid,
R.string.validation_error_name)
    }

    checkSecretKeyState()
}

private fun checkSecretKeyState() {
    binding.apply {
        if (secretKey.isNullOrBlank()) {
            registerScanQrButton.setText(R.string.login_scan_qr_button)
        } else {
            registerScanQrButton.setText(R.string.login_button_sign_up)
        }
    }
}

private fun TextInputLayout.setError(isOk: Boolean, @StringRes error: Int) {
    setError(if (isOk) null else getString(error))
}

companion object {

    private const val BundleKeyRegisterParams = "RegisterParams"

    fun newInstance(dataContainer: RegisterDataContainer): RegisterFragment
{
    return RegisterFragment().apply {
        arguments = Bundle().apply {
            putParcelable(BundleKeyRegisterParams, dataContainer)
        }
    }
}
}
}

```

A.2.45 Файл RegisterViewModel.kt

```

class RegisterViewModel(
    private val authRepository: AuthRepository
) : BaseViewModel() {

    private val _stateLiveData = MutableLiveData<RegisterState>()
    val stateLiveData: LiveData<RegisterState>
        get() = _stateLiveData

    fun signUp(params: SignUpParams, secret: String) {
        viewModelScope.launch {
            _stateLiveData.postValue(
                RegisterState(
                    isInProgress = true
                )
            )
        }
    }
}

```

```

        )
    )

    val validationResult = RegisterValidator.validate(params)

    if (!validationResult.isValid) {
        _stateLiveData.postValue(RegisterState(
            isInProgress = false,
            isEmailValid = validationResult.isEmailValid,
            isPasswordValid = validationResult.isPasswordValid,
            arePasswordsTheSame = validationResult.arePasswordsTheSame,
            isPhoneValid = validationResult.isPhoneValid,
            isNameValid = validationResult.isNameValid
        ))
        return@launch
    }

    val result = safeWithContextAsync(this) {
        authRepository.signUp(
            params.email,
            params.password,
            params.phone,
            params.fullName,
            secret
        )
    }.await()

    if (result?.isSucceed == true) {
        router.newRootChain(HomeFragment.homeScreen())
        _stateLiveData.postValue(RegisterState(isInProgress = false))
    } else {
        _stateLiveData.postValue(RegisterState(isInProgress = false,
isSecretValid = false))
    }
}

}

fun backToSignIn() {
    router.exit()
}

data class RegisterState(
    val isInProgress: Boolean,
    val isEmailValid: Boolean = true,
    val isPasswordValid: Boolean = true,
    val arePasswordsTheSame: Boolean = true,
    val isPhoneValid: Boolean = true,
    val isNameValid: Boolean = true,
    val isSecretValid: Boolean = true
)

}

data class SignUpParams(
    val email: String,
    val password: String,
    val confirmPassword: String,
    val fullName: String,
    val phone: String
)

```

A.2.46 Файл SmsFragment.kt

```

class SmsFragment : BaseFragment<SmsViewModel>(R.layout.fragment_sms) {

    override val viewModel: SmsViewModel by viewModel()
    private val binding by viewBinding<FragmentSmsBinding>::bind
    private var intentFilter: IntentFilter? = null
    private var smsReceiver: SmsReceiver? = null

    private val timeFormatter = SimpleDateFormat("mm:ss", Locale.getDefault())

    private val timer = object : CountDownTimer(10L * 1000L, 500) {

        override fun onTick(millisUntilFinished: Long) {
            val time = (millisUntilFinished / 1000).toInt()
            if (time != 0) {
                binding.smsButtonResend.isEnabled = false
            }
            val formatted = timeFormatter.format(Date(millisUntilFinished))
            binding.smsButtonResend.text = getString(R.string.resend_code_time,
formatted)
        }

        override fun onFinish() {
            binding.smsButtonResend.text = getString(R.string.resend_code)
            binding.smsButtonResend.isEnabled = true
        }

    }

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

        binding.smsButtonResend.setOnClickListener {
            viewModel.validateUser()
            startTimer()
        }

    }

    private fun startSmsState() {
        viewModel.validateUser()
        setupSmsListing()
        startTimer()
    }

    private fun startTimer() {
        binding.smsButtonResend.isEnabled = false
        timer.start()
    }

    private fun stopTimer() {
        timer.cancel()
    }

    private fun setupSmsListing() {
        intentFilter = IntentFilter(SmsRetriever.SMS_RETRIEVED_ACTION)
        if (smsReceiver == null) {
            smsReceiver = SmsReceiver()
            smsReceiver?.setOTPListener(object : SmsReceiver.OTPReceiveListener
{
            override fun onOTPReceived(otp: String?) {

```

```

        if (!otp.isNullOrBlank()) {
            binding.txtPinEntry.setText(otp)
        }
    }

    override fun onError() {
        startSmsState()
    }
})
} else {
    requireActivity().unregisterReceiver(smsReceiver)
}
requireActivity().registerReceiver(smsReceiver, intentFilter)

val client = SmsRetriever.getClient(requireContext())

val task: Task<Void> = client.startSmsRetriever()
task.addOnSuccessListener {

}
task.addOnFailureListener {
    setupSmsListing()
}

binding.txtPinEntry.setOnPinEnteredListener {
    viewModel.validateCode(it.toString())
}

}

override fun onResume() {
    super.onResume()
    smsReceiver?.let {
        requireActivity().registerReceiver(smsReceiver, intentFilter)
    }
}

override fun onPause() {
    super.onPause()
    smsReceiver?.let {
        requireActivity().unregisterReceiver(smsReceiver)
    }
}

override fun onDestroy() {
    super.onDestroy()
    smsReceiver = null
}

override fun onDestroyView() {
    super.onDestroyView()
    stopTimer()
}

override fun handleProgress(isInProgress: Boolean) {

}

}

class SmsReceiver : BroadcastReceiver() {

    private var otpListener: OTPReceiveListener? = null

```

```

fun setOTPListener(otpListener: OTPReceiveListener?) {
    this.otpListener = otpListener
}

override fun onReceive(context: Context?, intent: Intent?) {
    if (intent?.action == SmsRetriever.SMS_RETRIEVED_ACTION) {
        val extras = intent.extras
        val status = extras?.get(SmsRetriever.EXTRA_STATUS) as Status?
        when (status?.statusCode) {
            CommonStatusCodes.SUCCESS -> {
                val sms = extras?.get(SmsRetriever.EXTRA_SMS_MESSAGE) as
String?

                sms?.let {
                    val p = Pattern.compile("\\d+")
                    val m = p.matcher(it)
                    if (m.find()) {
                        otpListener?.onOTPReceived(m.group())
                    }
                }
            }
            else -> {
                otpListener?.onError()
            }
        }
    }
}

interface OTPReceiveListener {
    fun onOTPReceived(otp: String?)
    fun onError()
}

```

A.2.47 Файл SmsViewModel.kt

```

class SmsViewModel(
    private val authRepository: AuthRepository
) : BaseViewModel() {

    fun validateUser() {
        viewModelScope.launch {
            progressLiveData.postValue(true)
            safeWithContextAsync(this) {
                authRepository.sendValidationCode()
            }.await()
            progressLiveData.postValue(false)
        }
    }

    fun validateCode(code: String) {
        viewModelScope.launch {
            progressLiveData.postValue(true)
            val isValid = safeWithContextAsync(this) {
                authRepository.validateSmsCode(code)
            }.await()?.isValid ?: false
            if (isValid) {
                //TODO()
            } else {
                //TODO()
            }
            progressLiveData.postValue(false)
        }
    }
}

```

}
 }
 }

ДОДАТОК Б
Слайди презентації

Національний університет «Запорізька політехніка»

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСОБІВ РОЗРОБЛЕННЯ АГРЕГАТОРІВ НА ПРИКЛАДІ
ЗАСТОСУНКУ З ЛОГІСТИКИ КОРПОРАТИВНОГО ХАРЧУВАННЯ
**RESEARCH AND SOFTWARE IMPLEMENTATION OF THE DESIGNING AGGREGATOR
TOOLS ON THE CASE OF THE CORPORATE CATERING LOGISTICS APPLICATION**

Виконав:
студент групи КНТ-220М

Шаврін М.В.

Керівник:
професор, к.т.н.,

Табунщик Г.В.

Рисунок Б.1 – Слайд 1

2

МЕТА, ОБ'ЄКТ, ПРЕДМЕТ ДОСЛІДЖЕННЯ

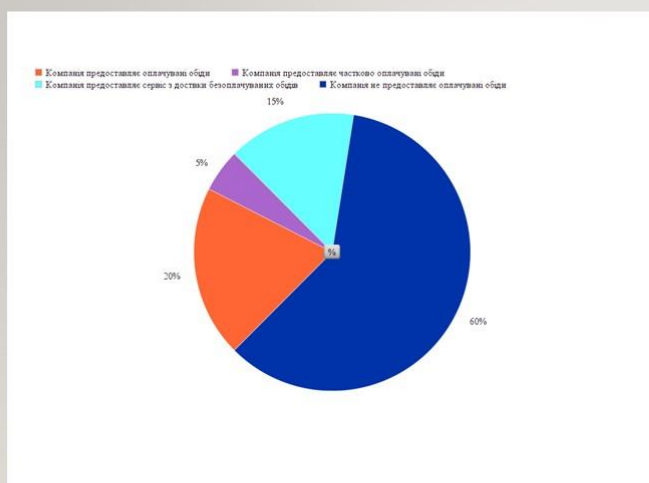
Об'єкт дослідження – процес забезпечення логістики корпоративного харчування.

Предмет дослідження – дослідження та реалізація засобів розроблення агрегаторів з логістики корпоративного харчування.

Мета роботи – покращення логістики корпоративного харчування за рахунок використання мобільного застосунка.

Рисунок Б.2 – Слайд 2

ПОСТАНОВКА ЗАДАЧІ



На сьогодні у сфері ІТ працює більше 250 тис. розробників. З них:

- 20% мають оплачувані обіди
- 5% мають частково оплачувані обіди
- 15% компанія представляє сервіс з доставки платних обідів
- 60% не мають оплачувані обіди

Рисунок Б.3 – Слайд 3

Огляд існуючих мобільних агрегаторів

Назва	Переваги	Недоліки
Grubhub	- мобільний та веб за стосунки; - корпоративна версія сервісу; - власний штат кур'єрів; - фільтрація ресторанів за типами кухні; - можливість залишати відгуки.	- відсутність механізму відслідковування статусу замовлення; - відсутність зворотнього зв'язку між клієнтами і закладами; - відсутність функціоналу, який би «радив» заклади найближчі до місця знаходження клієнта на основі його вподобань; - сервіс не працює з закладами України.
Notch	- великий вибір продуктів з місцевих ферм; - приємний та зрозумілий дизайн; - відстеження місцезнаходження замовлення; - засоби комунікації з кур'єрами та закладами.	- не працює в Україні; - відсутність функціоналу, який би «радив» заклади найближчі до місця знаходження клієнта на основі його вподобань; - розділення закладів за кухнею та типами їжі.
EatFirst	- мобільний та веб за стосунки; - корпоративна версія сервісу; - напрямок у сфері здорового харчування.	- відсутність механізму відслідковування статусу замовлення; - відсутність зворотнього зв'язку між клієнтами і закладами; - відсутність функціоналу, який би «радив» заклади найближчі до місця знаходження клієнта на основі його вподобань; - сервіс не працює з закладами України.

Рисунок Б.4 – Слайд 4

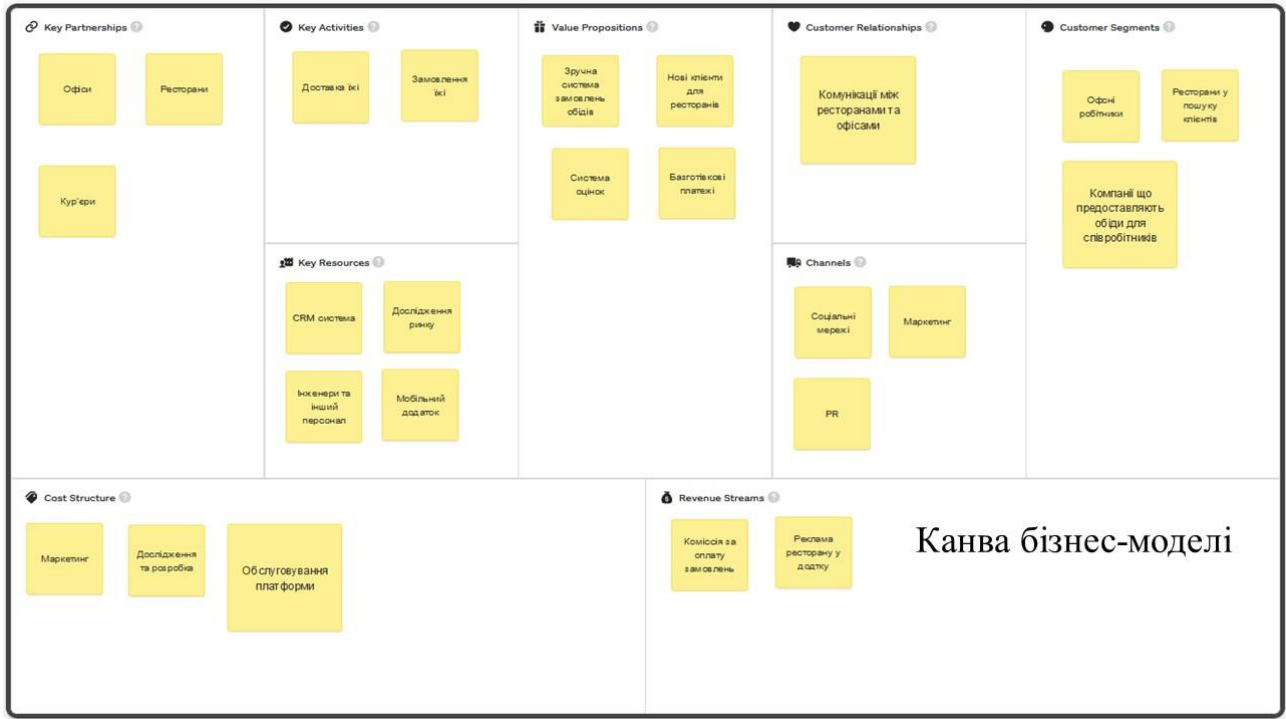


Рисунок Б.5 – Слайд 5

Вибір інструментів розробки

Функціонал	Java (8)	Kotlin
Лаконічність	-	+
Зворотна сумісність	-	+
Офіційна підтримка Google	+	+
Підтримка спільнотою	+/-	+
Продуктивність роботи	+/-	+

Обрано мови програмування Kotlin

Рисунок Б.6 – Слайд 6

Вибір архітектурного шаблону розробки

Характеристика	MVC	MVP	MVVM	MVI
Підтримка системою Android	-	+-	+	-
Робота з життєвим циклом Android	-	-	+	+-
Підтримка спільнотою розробників під Android	-	+	+	+-
Простота у реалізації	+-	+	+-	-
Простота у використанні у системі Android	-	+-	+-	+-

Рисунок Б.7 – Слайд 7

Блок-схема функціоналу програми

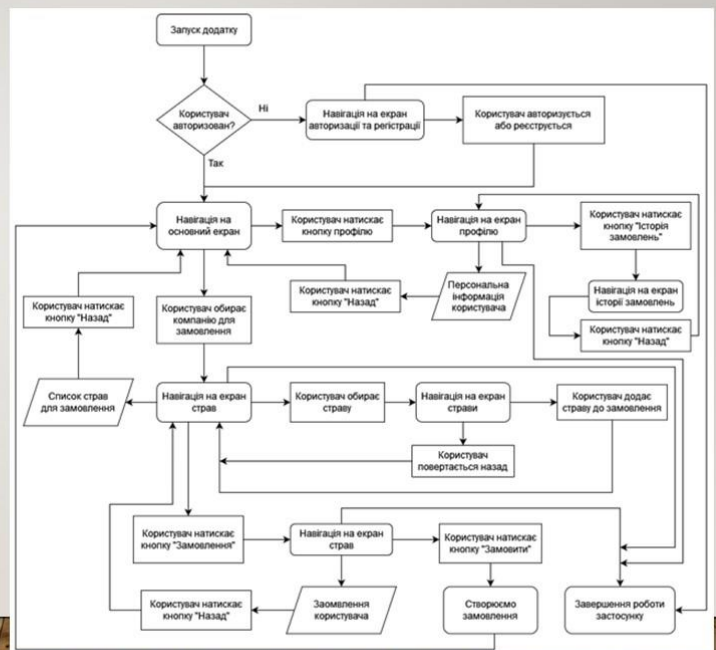


Рисунок Б.8 – Слайд 8

Спроектowana UML діаграма класів

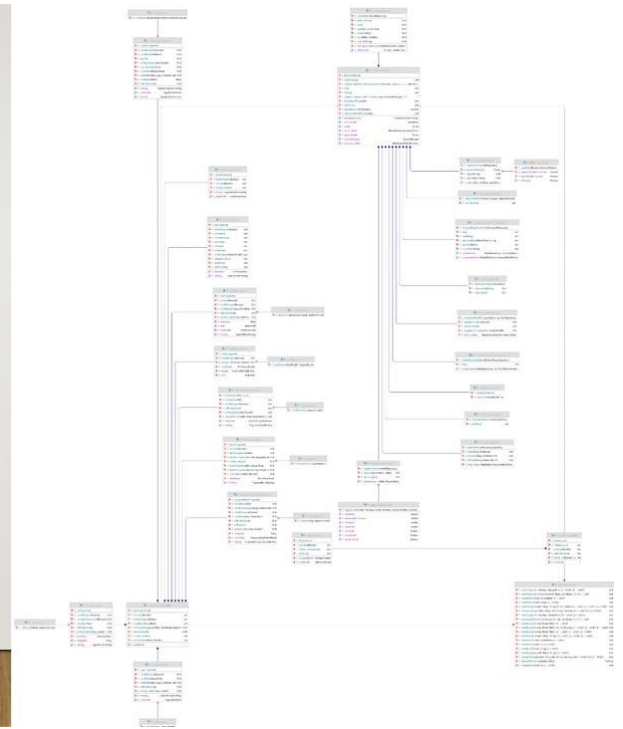


Рисунок Б.9 – Слайд 9

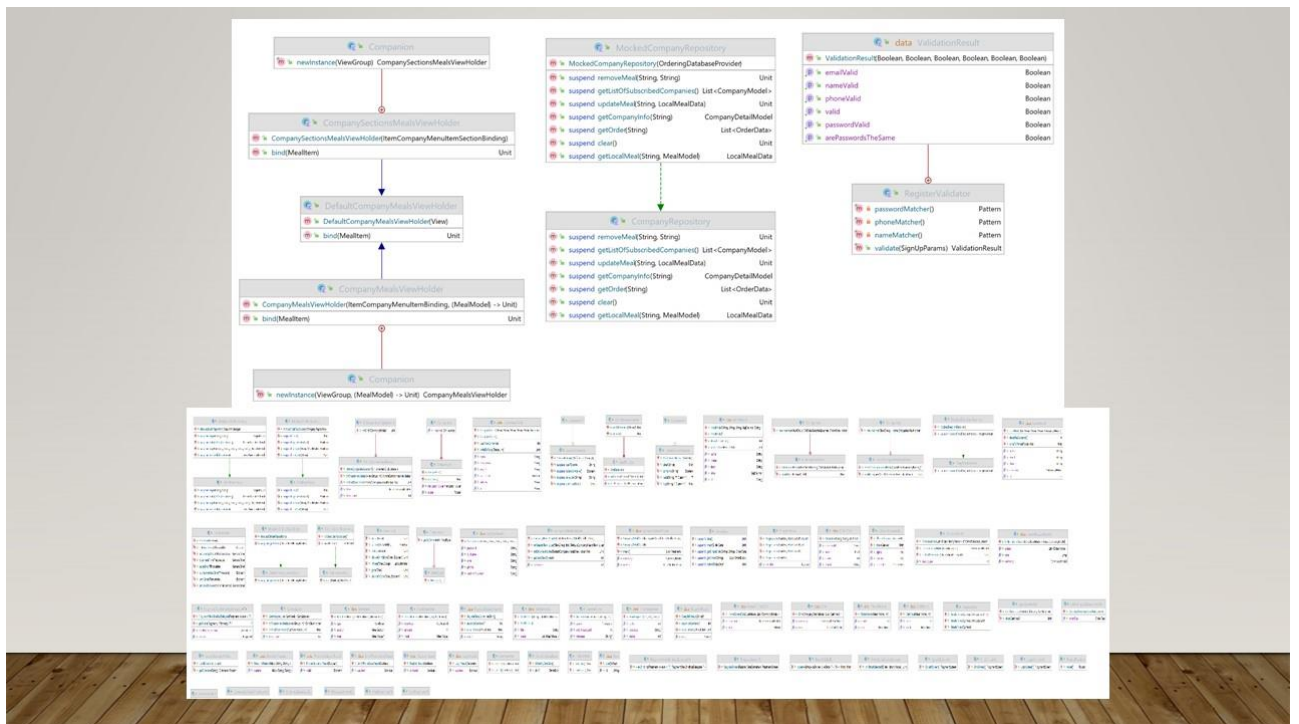


Рисунок Б.10 – Слайд 10

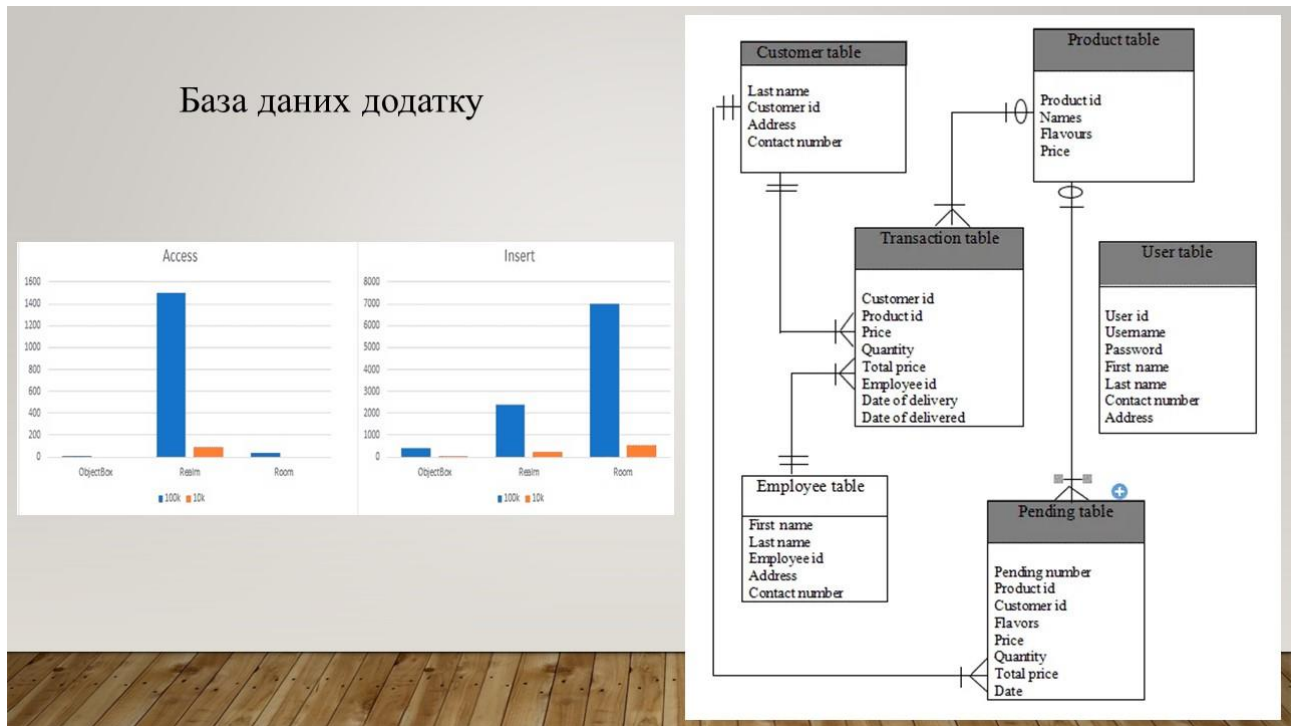


Рисунок Б.11 – Слайд 11

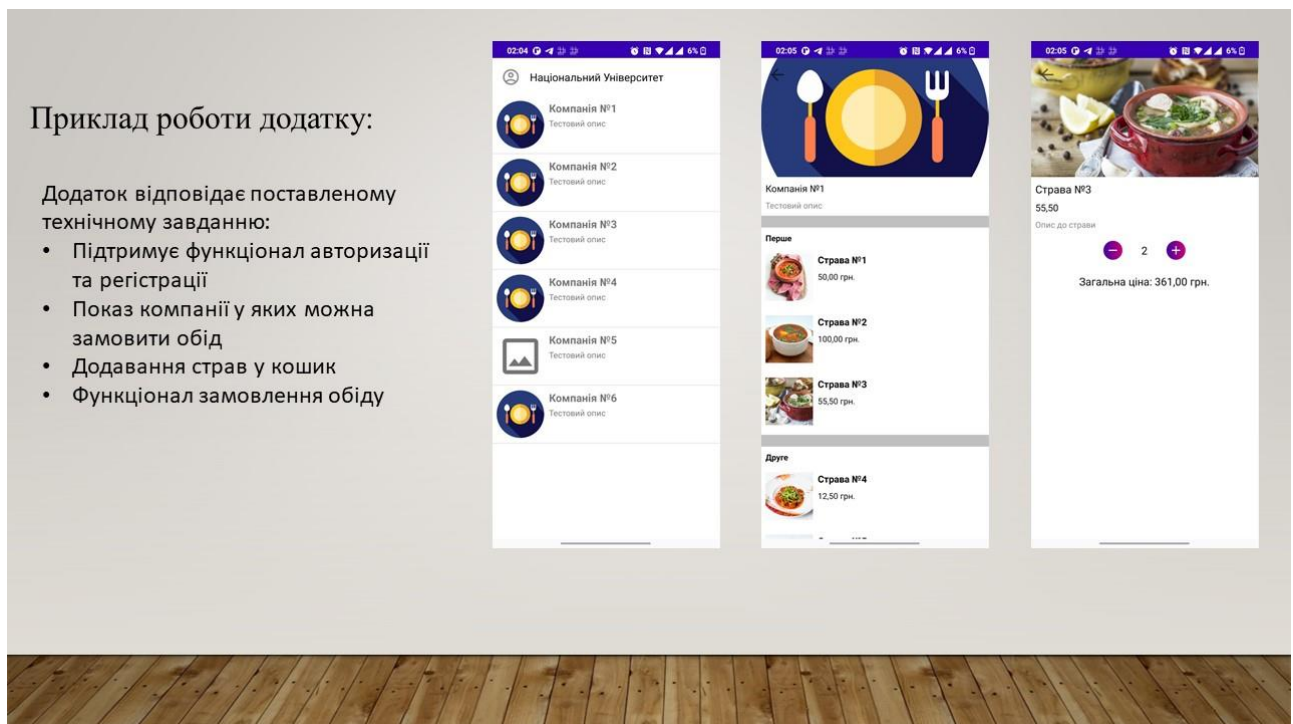


Рисунок Б.12 – Слайд 12

ВИСНОВОК

Під час роботи над виконання дипломної роботи магістра було проаналізовані різні системи з створення замовлень обідів, було сформовані функціональні вимоги для системи, котрі були реалізовані у застосунку:

- ознайомлення та аналіз предметної області
- аналіз існуючих інструментів при розробці та проектуванні систем. Обрана мова Kotlin та відповідні бібліотеки для реалізації компонентів системи. Як середовище розробки була обрана Android Studio
- розроблення системи з замовлення обідів у офіси, що відповідає сучасним архітектурним підходам

Рисунок Б.13 – Слайд 13