Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студентки *Токаревої Вікторії Валеріївни*
(ПІБ)

академічної групи *122-18ск-2*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка мобільного додатка на платформі Android «Доставка їжі – Aspire»*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Приходченко С.Д.*			
розділів:				
спеціальний	*доц. Приходченко С.Д.*			
економічний	*проф. Вагонова О.Г.*			
Рецензент	*доц. Герасіна О.В.*			
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студентки *122-18ск-2* *Токаревої Вікторії Валріївни*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи: *Розробка мобільного додатка на платформі Android «Доставка їжі – Aspire»*

затверджена наказом ректора НТУ «ДП» від «07» червня 2021 р. № 317-с

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2021 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2021 р.***

Завдання видав _____ *доц. Приходченко С.Д.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняла до виконання _____ *Токарева В.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *11.06.2021 р.*

РЕФЕРАТ

Пояснювальна записка: 123 с., 22 рис., 3 дод., 19 джерел.

Об'єкт розробки: мобільний додаток на платформі Android «Доставка їжі – Aspire».

Мета кваліфікаційної роботи: Розробка мобільного додатка на платформі Android «Доставка їжі – Aspire», який дозволяє звичайному користувачеві замовити і відстежити блюда з обраних ресторанів.

У вступі виконується аналіз сучасного стану проблеми, уточнюється постановка завдання, мета кваліфікаційної роботи та галузь її застосування, обґрунтовується актуальність теми.

У першому розділі досліджується предметна галузь, визначається актуальність завдання та призначення розробки, розроблюється постановка завдання, зазначаються вимоги до її програмної реалізації.

У другому розділі обирається платформа для розробки, виконується проектування програми і її розробка, наводиться опис алгоритму і структури функціонування системи, визначаються вхідні і вихідні дані, наводяться характеристики складу параметрів технічних засобів, описується робота програми.

В економічному розділі визначається трудомісткість розробленого програмного продукта, виконується підрахунок вартості роботи по створенню програми додатка та розрахунок часу на його розробку.

Практичне значення полягає у створенні мобільного додатка на базі операційної системи Android, який реалізує замовлення та доставку їжі у різні локації обраного міста в Україні.

Актуальність даного програмного продукту у сучасному світі визначається великим попитом на послуги з доставки їжі.

Список ключових слів: ANDROID, МОБІЛЬНИЙ ДОДАТОК, KOTLIN, ДОСТАВКА, РЕСТОРАНИ, БАЗА ДАНИХ.

ABSTRACT

Explanatory note: 123 pp., 22 figs., 3 appendices, 19 sources.

Object of development: mobile application on the Android platform "Food Delivery - Aspire".

The purpose of the qualification work: Development of a mobile application on the Android platform "Food Delivery - Aspire", which allows the average user to order and track dishes from selected restaurants.

In the introduction it is analyzed the current state of the problem, clarified the problem, the purpose of the qualification work and the scope of its application, substantiated the relevance of the topic.

In the first section the subject branch is investigated, the urgency of the task and purpose of development is defined, statement of the task is developed, requirements to its program realization are noted.

In the second section the platform for development is chosen, the program design and its development are carried out, the description of algorithm and structure of functioning of system is given, input and output data are defined, characteristics of structure of parameters of technical means are given, work of the program is described.

The economic section determines the complexity of the developed software product, calculates the cost of work to create an application program and calculates the time for its development.

The practical significance lies in the creation of a mobile application based on the Android operating system, which orders and delivers food to various locations in the selected city in Ukraine.

The relevance of this software product in today world is determined by the high demand for food delivery services.

Keyword list: ANDROID, MOBILE APP, KOTLIN, DELIVERY, RESTAURANTS, DATABASE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

UI – користувальницький інтерфейс

UX – взаємодія користувача з інтерфейсом

СУБД – система управління базою даних

БД – база даних

API – програмний інтерфейс додатку

JSON – текстовий формат обміну даними, заснований на JavaScript

SQL – мова програмування структурованих запитів

ВСТУП

Темою кваліфікаційної роботи є розробка мобільного додатка на базі операційної системи Android «Доставка їжі - "Aspire"».

Метою кваліфікаційної роботи є вивчення та впровадження сучасних інструментів розробки в програмний продукт, який буде безпосередньо працювати з користувачем для полегшення повсякденного життя. В першу чергу була проведена аналітика UI / UX, так як подібні додатки і послуги почали набирати популярність в останні роки і потрібно розуміти зручність використання і сприйняття з боку клієнта. Далі була обрана нова мова програмування, яка на даний момент є одною з самих легких, зручних і досить безпечним в області програмування.

При розробці даного продукту можна ознайомитися з інструментами, які з'явилися на ринку досить нещодавно і зараз набирають популярність в їх використанні: так звані патерни, різні бібліотеки і плагіни, які і будуть описані і представлені в роботі.

Так як сам додаток працює безпосередньо з користувальницькими даними, то кожен користувач, реєструючись в системі, може не переживати за безпеку. Всі дані, занесені в базу даних, мають шифрування на рівні бази даних, що не дозволяє трапиться витоку.

Ресторанний бізнес, послуги з доставки - зараз як ніколи досить популярні і з кожним роком набирають лише обороти. Цей додаток дозволяє користувачеві замовити їжу з улюбленого ресторану і при цьому заощадити свій час, не вбиваючи в пошукову систему окремий ресторан. Все вже є в одній програмі - система відгуків, клієнтська підтримка і не одна доставка, яка співпрацює тільки з певними ресторанами, а в загальному і цілому, всі ресторани зі своєю особистою системою з доставки в одному місці, тобто в додатку.

Так що, завданням даної кваліфікаційної роботи є створення та розробка системи з доставки їжі з ресторанів "Aspire".

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Життя кипить навіть у невеликих містах. Все встигати - це справжній талант. Але не варто забувати і про нормальне харчування. Брати бутерброди з дому на роботу - не варіант. Це загрожує настанням проблем зі здоров'ям, чого ні в якому разі не можна допустити. Ось тут і приходять на поміч додатки та сервіси доставки їжі.

Закладам громадського харчування, які доставляють їжу на будинок (тощо), не обійтися без впровадження програми автоматизації. Вона необхідна для швидкої обробки замовлень, розподілу завдань на кухні, вибору маршруту для кур'єрів. Автоматизація доставки позитивно впливає на розвиток бізнесу в цілому. Впровадивши програму в свою організацію, можна розраховувати на повну оптимізацію бізнес-процесів: роботу кур'єрів, кухарів, керуючих, складальників, операторів.

Щоб домогтися успіху в бізнесі по доставці, важливо грамотно організувати робочі процеси. Клієнти повинні швидко отримувати замовлення з необхідною температурою страв. В даному сегменті важлива кожна хвилина. Для того, щоб процес роботи був злагодженим, слід впровадити систему автоматизації доставки їжі. Завдяки своїй багатофункціональності вона допомагає вирішує питань:

– прискорити виконання замовлень;

– контролювати замовлення;

– підключити IP-телефонію з автоматичним визначенням номера клієнта для швидкої обробки заявки;

– забезпечити виконання робочих процесів у відповідності зі стандартами;

– чітко планувати час операцій;

– інтегрувати заявки з різних сервісів: месенджерів, соціальних мереж, сайтів;

– створювати базу клієнтів;

– стежити за місцем розташування кур'єрів;

– змінювати вартість страв з урахуванням запитів клієнтів.

Послуга доставки їжі буде доречна і студенту, який не встиг поснідати вдома, і офісному працівнику, в якого перерва, тим, хто вирішив замовити додому щось до перегляду кіно. Тому вона є актуальною для всіх.

Це призвело до появи багатьох компаній, які доставляють їжу, також ресторани самі почали доставляти закази клієнтам і це значно спрощує життя багатьох людей. До основних переваг таких компаній відносяться:

– оперативна доставка, що дозволяє швидко розпланувати обід;

– великий вибір страв у меню, що дає можливість урізноманітнити свій раціон;

– невисока ціна - вона на порядок нижче, ніж в ресторанах;

– гідну якість продуктів, які використовуються під час приготування;

– відмінна можливість продегустувати страви, які складно приготувати в домашніх умовах;

– економія власних сил;

– простота оформлення замовлення. Це можна зробити по телефону або інтернету. Якщо в цьому є необхідність, то вдасться зробити попереднє замовлення.

І якщо раніше багато людей віддавали перевагу тому, щоб самим сходити до улюбленого кафе, посидіти та поспілкуватися, то у 2020 році ситуація у світі вимусила змінити звичний ритм життя. В умовах карантину ми побачили увесь потенціал таких служб. Коли жоден ресторан чи кафе не працювало і не було можливості сходити до улюбленого закладу, майже всі люди користувалися службами доставки аби замовити улюблену страву. Статистика свідчить, що популярність деяких служб збільшилася в кілька разів, а кількість заказів на 35-80%. Але не тільки в умовах карантину служби доставки користуються попитом, адже в них є декілька переваг.

Економія часу та легкість вибору- це найбільша перевага додатків доставки. Замість того, щоб бігати по різним закладам, шукати що вам до вподоби та губитися

у різноманітті усіх страв, можна просто встановити додаток і не виходячи з дому обрати страву, яка вам сподобається.

Якщо говорити про звичні нам доставки з ресторану, то в них є 1 великий мінус. Ресторани, як це не дивно, доставляють лише їжу зі свого закладу, а що робити, коли хочеш десерт з 1-го закладу, піцу з іншого, а суші для друзів з 3-го? Доводилося заходити на сайт кожного закладу і в кожному з них замовляти по 1-2 страві, це незручно, бо потім тобі треба чекати 3 кур'єра з різних закладів. Це закономірно призвело до створення додатку доставки, в якому будуть більшість ресторанів міста і можна одразу обрати усе, що ти хочеш.

До того ж, можна подивитися різні ресторани, почитати відгуки та вирішити де і що замовляти. Тобто, ви можете обрати на свій смак ресторан, кухню яка вам до вподоби, можна взагалі замовити декілька різних страв з різних ресторанів, суші, піца або вишуканий французький десерт і все це разом і в одному додатку, що значно легше, аніж замовляти це усе окремо. А потім вам привезуть це з кожного ресторану.

1.2. Призначення розробки та постановка задачі

Розробка мобільного додатку на платформі андроїд, через який можна здійснювати замовлення їжі з ресторанів.

Розроблений додаток буде використовуватися ресторанами для пропозиції своїх послуг з приготування і доставки їжі в будь-яку точку, зазначену «гостем» їхнього закладу.

Найголовнішою метою цього додатка є поєднання в одному місці (тобто в самому додатку) сотні ресторанів і вибору кухні / їжі / закладу по пошуковій системі або системі відгуків і пропозицій для того, щоб заощадити свій час і не шукати в інтернеті окремо кожен ресторан.

Тому метою є спроектувати та розробити клієнтський додаток на базі ОС Android для бази ресторанів міст України, який дозволяв би замовляти страви напряму з різних ресторанів, що значно економить час та зусилля.

1.3. Підстави для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується з наказом ректора.

Таким чином підставами для розробки (виконанням кваліфікаційної роботи) є:

- освітня програма спеціальності 122 “Комп’ютерні Науки”;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету “Дніпровська політехніка” №317-с від 07.06.2021р.;
- завдання на кваліфікаційну роботу на тему «Розробка мобільного додатку на платформі андроїд з доставки їжі - "Aspire"».

1.4. Постановка завдання

Метою завдання є спроектувати та розробити клієнтський додаток на базі ОС Android для бази ресторанів міст України з доставки їжі "Aspire".

Додаток містить в собі можливість:

- обрати ресторан, з якого клієнт хоче здійснити замовлення;
- переглянути меню ресторану;
- системи відгуків та оцінювання;
- відстежування на якому етапі знаходиться замовлення;
- особисті пропозиції з ресторанів;
- систему пошуку по кухням країни / стравам / ресторанам.

До додатку повинна бути підключена SQL база даних клієнтів, через сервер. Також буде застосовано API «Application Programming Interface» (інтерфейс програмування додатків) який так само як і база даних підключається окремо.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для коректного запуску додатку потрібно щоб на телефоні було встановлено операційну систему Android мінімум шостої версії, інакше "Aspire" не буде працювати.

Android, Андроїд - операційна система для смартфонів, планшетних комп'ютерів, електронних книг, цифрових програвачів, "розумних" наручних годинників, ігрових приставок, нетбуків, смартбуків та інших пристроїв.

ОС заснована на ядрі Linux і власної реалізації віртуальної машини Java від Google.

Сам додаток "Aspire" повинен встановлюватися з магазину Google Play Store (Play Market) та не потребує з боку користувачів ще маніпуляцій з установкою. Даний додаток призначений для безкоштовного користування з боку звичайних користувачів. [1]

Якщо ж користувач захоче встановити додаток з окремого джерела, то додаток буде упаковано в APK (формат архівних виконуваних файлів-додатків для Android і р інших операційних систем) файл, який треба буде завантажити та встановити самому.

1.5.2. Вимоги до інформаційної безпеки

Так як додаток вже запакований, якихось вимог до інформаційної безпеки дотримуватись не потрібно. Однак, в "Aspire" є найголовніший момент – це реєстрація користувачів (дані яких зберігаються в базі даних), саме вона проходить шифрування на рівні бази даних.

Дані шифруються перед записом на диск і дешифруються під час читання в пам'ять, що вирішує проблему захисту «неактивних» даних, але не забезпечує збереження інформації при передачі по каналах зв'язку або під час використання.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для більш надійного функціювання додатку потрібна на смартфоні (або планшеті) операційна система Android мінімум 6 версії. Вільна оперативна пам’ять та небагато (до 250 MB) місця зовнішньої пам’яті.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для реалізації мобільного додатку було обрано мову програмування Kotlin та середовище розробки Android studio.

Kotlin (Котлін) - статично типізована, об'єктно-орієнтована мова програмування, що працює поверх Java Virtual Machine і розробляється компанією JetBrains. Також компілюється в JavaScript і в виконуваний код ряду платформ через інфраструктуру LLVM.

Мова повністю сумісна з Java, що дозволяє java-розробникам поступово перейти до його використання; зокрема, в Android мова вбудовується за допомогою Gradle, що дозволяє для існуючого android-додатку впроваджувати нові функції на Kotlin без переписування програми цілком.

Синтаксис мови використовує елементи з JavaScript, Паскаля, TypeScript, Haxe, PL / SQL, F #, Go і Scala, C ++, Java, C #, Rust і D. [2]

Android Studio - універсальне середовище розробки, так як дозволяє оптимізувати роботу майбутніх додатків для роботи не тільки на смартфонах, але і на планшетах, портативних ПК, які працюють на основі даної операційної системи.

Особливості Android Studio: у програму вбудований емулятор, що дозволяє перевірити коректну роботу програми на пристроях з різними екранами, з різними співвідношеннями сторін. [3]

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення програми

Результатом даної кваліфікаційної роботи повинен бути програмний продукт, а саме - мобільний додаток створений за допомогою програми графічного редактору для розробки дизайну та середовища розробки програмної частини з вбудованим емулятором.

Основним призначенням додатку є наступне:

- створення власного акаунту в системі;
- безпечне зберігання даних користувачів;
- онлайн-замовлення їжі та відстежування доставки у реальному часі;
- редагування своїх даних та даних при замовленні чи залишених відгуків;
- зручний пошук за потрібними категоріями.

2.2. Опис застосованих математичних методів

Під час проектування та розробки мобільного додатку, використовувались прості арифметичні дії для: підрахунку загальної суми замовлення, де є сума всіх замовлених у «корзині покупок» страв, в тому числі і знижок; доставки кур’єра, якщо ресторан бере плату за доставку, вона також підсумовується з загальним підрахунком; та кінцевої оплати.

2.3. Опис використаних технологій та мов програмування

Мобільний додаток був розроблений з використанням наступних технологій:

- Kotlin;
- Node.js;
- SQL;

– Socket.io;
– Javalin RestAPI;
– Spark.

Kotlin - це нова мова програмування для платформи Java. Kotlin - лаконічна, безпечна і прагматична мова, сумісна з Java. Її можна використовувати практично скрізь, де застосовується Java: для розробки серверних додатків, додатків для Android і багато чого іншого. Kotlin прекрасно працює з усіма існуючими бібліотекам і фреймворками, написаними на Java, не уступаючи останньому в продуктивності. [4]

Kotlin дозволяє створювати об'єктно-орієнтований код, в якому використовуються класи, спадкування та поліморфізм, як і в мові Java. Але Kotlin також підтримує функціональне програмування, і завдяки цьому, є можливість користуватися кращими можливостями обох парадигм. [5]

Kotlin - статично типізована мова програмування. Тобто, тип кожного виразу в програмі відомий під час компіляції, і компілятор може перевірити, що методи і поля, до яких звертаються, дійсно існують в використовуваних об'єктах. [6]

Мова Kotlin пропонує виразний синтаксис, потужну та зрозумілу систему типів, чудову підтримку і безшовну сумісність з існуючим кодом на Java, багатий вибір бібліотек і фреймворків. Kotlin може компілюватися в байт-код Java, тому дану мову можна використовувати всюди, де використовується java, включаючи Android. А завдяки ефективному компілятору і маленькій стандартної бібліотеці Kotlin практично не привносить накладних витрат. [5]

На відміну від деяких мов, Kotlin надзвичайно компактний - в ньому можна виконувати складні операції в одному рядку коду. Kotlin надає скорочений запис для основних операцій, щоб не доводилося писати шаблонний код, що постійно повторюється, а також містить багату бібліотеку функцій, якими можна користуватися в своїх програмах. А чим менше коду доводиться переглядати, тим швидше читаються і пишуться програми, тим швидше можна зрозуміти їх логіку, і далі залишається більше часу для іншої роботи. [6]

Node.js - це серверна JavaScript-платформа, призначена для створення масштабованих розподілених мережевих додатків, що використовує подієво-орієнтовану архітектуру і неблокуючу асинхронну взаємодію. [7]

На рис. 2.1 зображено типовий веб-додаток Node, що використовує бібліотеку веб-програмування Express для обробки замовлень в магазині. Браузери видають запити на придбання продукту; додаток перевіряє поточний стан складських запасів, створює обліковий запис для користувача, відправляє квитанцію по електронній пошті і повертає HTTP-відповідь у форматі JSON (JavaScript Object Notation – текстовий формат обміну даними, заснований на JavaScript). Одночасно можуть відбуватися інші операції: квитанція надсилається електронною поштою, а база даних оновлюється інформацією від користувача і даними замовлення. По суті, перед нами прямолінійний імперативний код JavaScript, але виконавче середовище працює паралельно, тому що вона використовує введення / виведення, що не блокується.

Одна із сильних сторін Node і JavaScript взагалі - однопотокова модель програмування. Програмні потоки (threads) є стандартним джерелом помилок, і хоча деякі з мов програмування, що нещодавно з’явились, включаючи Go і Rust, намагаються надати безпечні інструменти паралельного програмування, Node працює з моделлю, що використовується в браузері. Браузерний код є послідовністю команд, які виконуються одна за однією; код не виконується паралельно. Для призначених для користувача інтерфейсів така модель не має сенсу: користувач не хоче чекати завершення повільних операцій (наприклад, звернень до даних по мережі або до файлів). Для вирішення цієї проблеми в браузерах використовуються події: коли користувач клацає на кнопці, ініціюється подія, і виконується функція, яка була визначена раніше, але ще не виконувалася. Тим самим запобігають деякі проблеми, що зустрічаються в багатопотоковому програмуванні, включаючи взаємні блокування (deadlocks) ресурсів і стану гонки (race conditions). [8]

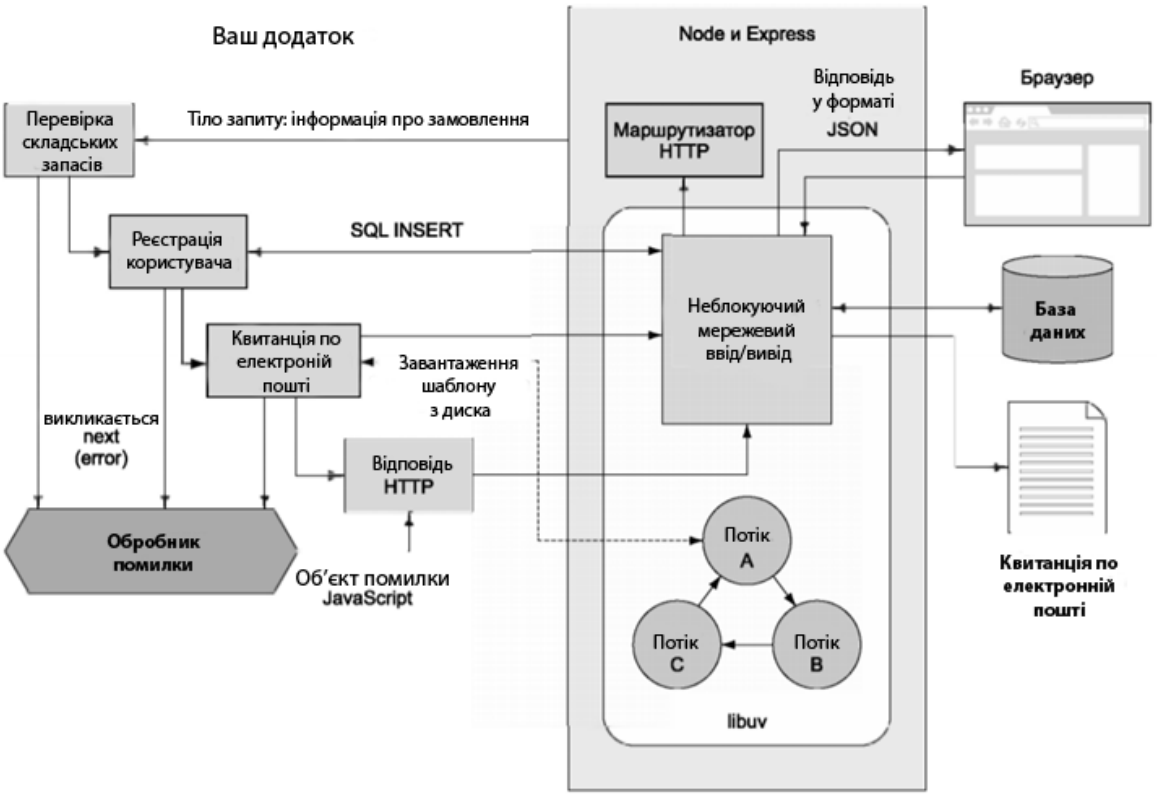

Рис. 2.1. Асинхронні і неблокуючі компоненти в додатках Node

Крім ефектної асинхронної моделі роботи, неблокуючих процесів, високої продуктивності, Node.js робить те, що вважалося принципово нездійсненним, - дає можливість розробнику створювати як server-side / backend-, так і frontend-додатки, користуючись єдиною технологією.

SQL є інструментом, призначеним для організації, управління, вибірки і обробки інформації, що міститься в базі даних. SQL є мовою програмування, яка застосовується для організації взаємодії користувача з базою даних. Насправді SQL працює тільки з базами даних певного типу, званими реляційними базами даних, які представляють собою основний спосіб організації даних в широкому діапазоні додатків.

На рис. 2.2 показана схема роботи SQL. Відповідно до цієї схеми, в обчислювальній системі є база даних, в якій зберігається важлива інформація. Якщо обчислювальна система відноситься до сфери бізнесу, то в базі даних можуть міститися відомості про матеріальні цінності, продукції що випускається, обсяги

продажів і зарплати. У базі даних на персональному комп'ютері може містити інформацію про виписані чеки, телефони і адреси або інформація, витягнута з більшої обчислювальної системи. Комп'ютерна програма, яка керує базою даних, називається системою управління базою даних, або СУБД.

Рис. 2.2. Застосування SQL для звернення до бази даних

Для того щоб отримати інформацію з БД, потрібно запросити її у СУБД за допомогою SQL. СУБД обробляє запит, знаходить необхідні дані і повертає їх. Процес запиту даних у БД і отримання результату називається запитом до бази даних (ось чому в назві мови є слово "запит" - мова структурованих запитів).

Однак ця назва не зовсім відповідає дійсності. По-перше, сьогодні SQL являє собою щось більше, ніж просто інструмент створення запитів, хоча саме для цього він і був спочатку розроблений. Незважаючи на те що вибірка даних як і раніше залишається однією з найбільш важливих функцій SQL, зараз ця мова використовується для реалізації всіх функціональних можливостей, які СУБД надає користувачеві. [9]

Нарешті, SQL - це слабо структурована мова, особливо в порівнянні з такими високо структуровані мовами, як C, Pascal або Java. Інструкції SQL нагадують звичайні пропозиції природної мови і містять "слова-пустушки", які не впливають на зміст інструкції, але полегшують її читання. У SQL майже немає нелогічностей, до того ж є ряд спеціальних правил, що запобігають створення інструкцій, які

виглядають як абсолютно правильні, але не мають сенсу. Незважаючи на не зовсім точну назва, SQL на сьогоднішній день є стандартом мови для роботи з реляційними базами даних. SQL - це досить потужна і в той же час відносно легка для вивчення мова. [10]

Також для роботи з мобільним додатком "Aspire" також використовувалось шифрування на рівні бази даних.

Шифрування всієї системи може бути невиправдано дорогим, коли потрібно захистити лише частину даних. Також не завжди можливо змінити код і розробити механізм шифрування на рівні додатку. У таких випадках важливо використовувати шифрування на рівні бази.

Одним із прикладів шифрування на рівні бази даних є шифрування на рівні стовпців, яке записує в базу даних вже зашифровані дані, а саму базу даних - без подальшого шифрування - в сховище. Особливістю шифрування на рівні стовпців є використання єдиного ключа при обробці даних одного стовпчика. Ключі можуть бути призначені користувачам і захищені паролем для запобігання автоматичної розшифровки, однак це ускладнює адміністрування БД. При використанні шифрування на рівні стовпців необхідно внесення змін до клієнтські програми. Крім цього зменшується продуктивність БД. [11]

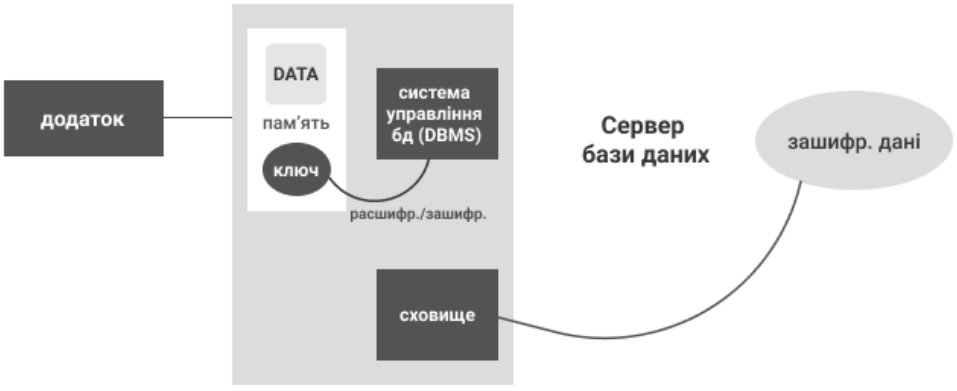

Рис. 2.3. Шифрування на рівні бази даних

Socket.IO – JavaScript-бібліотека для веб-додатків і обміну даними в реальному часі. Складається з двох частин: клієнтської, яка запускається в браузері і серверної для node.js. Обидва компоненти мають схожу API. Подібно node.js, Socket.IO подієво-орієнтована.

Як працює бібліотека: клієнт спробує встановити з'єднання WebSocket, якщо це можливо, і в іншому випадку повернеться до довгого опитування HTTP.

WebSocket – це протокол зв'язку, який забезпечує повнодуплексний канал з малою затримкою між сервером і браузером.

Функції, що надаються Socket.IO над звичайними WebSockets:

- надійність (відкат до довгого опитуванням HTTP в разі, якщо з'єднання WebSocket не може бути встановлено)
- автоматичне перепідключення;
- буферизація пакетів;
- мовлення всім клієнтам або підгрупі клієнтів;
- мультиплексування (те, що називається «простором імен»). [12]

Javalin – це дуже легкий веб-фреймворк для Java 8 (і більш пізніх версій) і Kotlin. Він підтримує сучасні функції, такі як HTTP / 2, WebSocket і асинхронні запити. Javalin заснований на сервлетах (інтерфейс Java, реалізація якого розширює функціональні можливості сервера), і його основні цілі - простота, відмінний досвід розробника і першокласна сумісність між Java і Kotlin.

Джавалін простий і об’єктивний, що робить його гарним вибором, якщо вам потрібно швидко почати роботу. Його шар абстракції тонкий, що дозволяє легко зрозуміти, що відбувається. Javalin також швидкий, обслуговуючи 1,1 мільйона запитів в секунду в тестах TechEmpower за жовтень 2018 року, що значно швидше, ніж у більшості більш важких і багатьох легких фреймворків.

Простота Javalin має свою ціну. Оскільки Javalin виконує тільки веб-додатки, розробникам необхідно вирішити настройку бази даних, впровадження залежностей, синтаксичний аналіз командного рядка і інші важливі аспекти програми. На веб-сайті Javalin є безліч посібників, які показують, як підійти до багатьох з цих завдань. [13]

Spark - вільна програма для миттєвого обміну повідомленнями в мережі Інтернет за протоколом Jabber. Має можливість створення закладок для кожного вікна чату, перевірки орфографії при наборі тексту. Відрізняється хорошою роботою з сервером Openfire. Зручна і проста реєстрація користувачів.

Дана програма розрахована на обмін короткими текстовими повідомленнями при корпоративному використанні в різних установах і на малих підприємствах, і має стандартний набір функцій, характерних для програм такого типу.

Основні можливості програми Spark. В першу чергу програма відноситься до розряду Jabber-клієнтів, має відкритий вихідний код (тобто оптимізувати його на свій розсуд можуть всі) і написано на Java. Саме тому Spark підтримує різні операційні системи. До безперечних достоїнств варто віднести можливість створення мультичатів і конференцій. У вікні повідомлень працювати зручно, оскільки для кожного співрозмовника виділяється закладка, а не нове вікно. Для кожного чату ви можете створювати власне вікно з додаванням різних співрозмовників. У вас також є можливість зробити знімок екрана (так звана «капча») та відправити персонально кожного співрозмовника, незалежно від того, що ви перебуваєте в режимі чату, де все бачать і власні і свої повідомлення та дії. За словами розробників цього програмного забезпечення, Spark має поліпшену систему безпеки.

Крім основної і самої головної функції для кожного месенджера - швидкого і зручного обміну текстовими повідомленнями між абонентами (прийом і відправлення повідомлень, створення контакт-листів і т.д.), програма Спарк готова надати своїм користувачам ще й цілий ряд інших дуже привабливих можливостей. Безумовно, однією з найважливіших з них, є повноцінна підтримка створення мультичатів і проведення групових конференцій, що останнім часом активно користуються різні приватні фірми, робітники підприємства, навчальні заклади і багато спільноти інтернет-користувачів. Додатково до цього, програма має вбудовану зручну записну книжку, інтегрований календар, багатофункціональний інструмент для створення списків завдань, може здійснювати перевірку орфографії для тексту, що набирається. Також, програма підтримує ексклюзивні смайли і Jingle,

працює з різними плагінами, що дають можливість розширити її функціонал, дозволяє приймати / передавати файли і володіє досить надійним захистом конфіденційності даних користувача. [14]

2.4. Опис структури системи та алгоритмів її функціонування

Організація вхідних та вихідних даних в більшості мала основу по роботі з патернами проектування.

Патерн проектування - це рішення певної проблеми при проектуванні архітектури програм.

На відміну від готових функцій або бібліотек, патерн можна просто взяти і скопіювати в програму. Патерн є не якийсь конкретний код, а загальна концепція вирішення тієї чи іншої проблеми, яку потрібно буде ще підлаштувати під потреби програми.

Патерни часто плутають з алгоритмами, адже обидва поняття описують типові рішення якихось відомих проблем. Але якщо алгоритм - це чіткий набір дій, то патерн - це високорівневий опис рішення, реалізація якого може відрізнятися в двох різних програмах.

Якщо привести аналогії, то алгоритм - це кулінарний рецепт з чіткими кроками, а патерн - інженерне креслення, на якому намальовано рішення, але не конкретні кроки його реалізації.

Описи патернів зазвичай дуже формальні і складаються з таких пунктів:

- проблема, яку вирішує патерн;
- мотивації до вирішення проблеми способом, який пропонує патерн;
- структури класів, складових рішення;
- прикладу на одній з мов програмування;
- особливостей реалізації в різних контекстах;
- зв'язків з іншими патернами.

Такий формалізм в описі дозволив створити великий каталог патернів, перевіривши кожен з них на спроможність. [15]

У кваліфікаційній роботі були використані: структурний патерн та патерни що породжують.

Структурний патерн називається Декоратор (Wrapper, Обгортка, Decorator).

Однак, в даному патерні є проблема. Наприклад, ви працюєте над бібліотекою сповіщень, яку можна підключати до різноманітних програм, щоб отримувати повідомлення про важливі події.

Основою бібліотеки є клас Notifier з методом send, який приймає на вхід рядок-повідомлення і висилає її всім адміністраторам по електронній пошті. Стороння програма повинна створити і налаштувати цей об'єкт, вказавши кому відправляти оповіщення, а потім використовувати його кожен раз, коли щось трапляється.

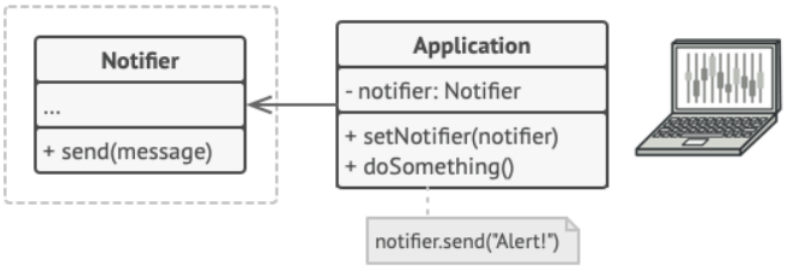

Рис. 2.4. Використання сторонньою програмою бібліотеку сповіщень

Далі користувачем захотілося би отримувати оповіщення різними способами: соціальні мережі, смс тощо. Можна спробувати реалізувати всі можливі комбінації підкласів сповіщень. Але після того як ви додали перший десяток класів, стало ясно, що такий підхід неймовірно роздуває код програми.

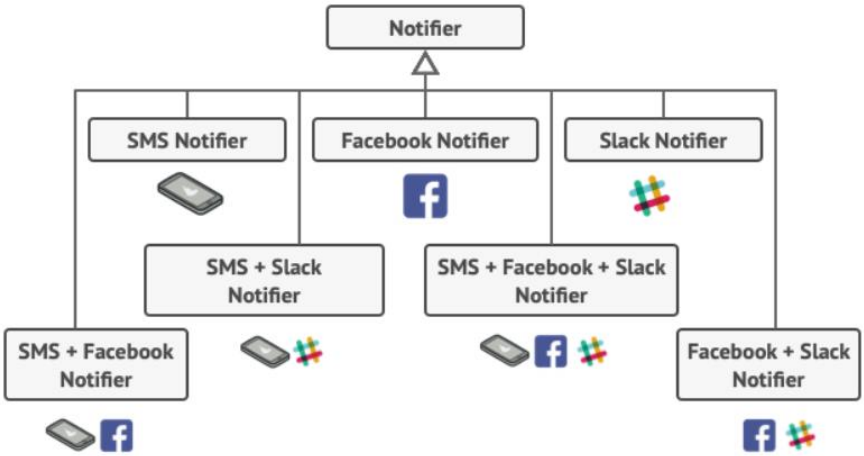

Рис. 2.5. Комбінаторний вибух підкласів при поєднанні типів сповіщень

Отже, потрібен якийсь інший спосіб комбінування поведінки об'єктів, який не призводить до вибуху кількості підкласів.

Тому є наступне рішення. Одним із способів обійти ці проблеми є заміна успадкування агрегацією або композицією. Це коли один об'єкт містить посилання на інший і делегує йому роботу, замість того щоб самому успадковувати його поведінку. Якраз на цьому принципі побудований патерн Декоратор.

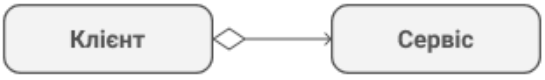

Рис. 2.6. Агрегація

Декоратор має альтернативну назву - обгортка. Воно більш точно описує суть патерну: ви ставите цільовий об'єкт в інший об'єкт-обгортку, який запускає базову поведінку об'єкта, а потім додає до результату щось своє.

Обидва об'єкти мають загальний інтерфейс, тому для користувача немає ніякої різниці, з яким об'єктом працювати - чистим або обгорнутим. Ви можете використовувати кілька різних обгорток одночасно - результат буде мати об'єднане поведінку всіх обгорток відразу.

У прикладі з оповіщенням ми залишимо в базовому класі просту відправку по електронній пошті, а розширені способи відправки зробимо декораторами.

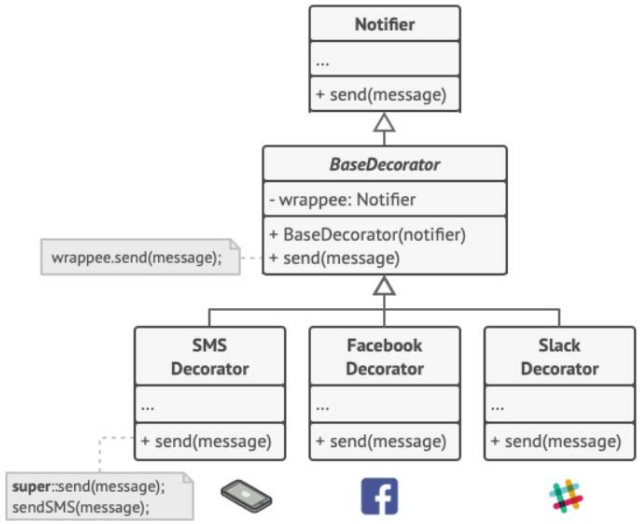

Рис. 2.7. Розширені способи оповіщення стають декораторами

Остання обгортка в списку і буде тим об'єктом, з яким клієнт буде працювати в інший час. Для решти клієнтського коду, по суті, нічого не зміниться, адже всі обгортки мають точно такий же інтерфейс, що і базовий клас сповіщень.

Таким же чином можна змінювати не тільки спосіб доставки повідомлень, а й форматування, список адресатів і так далі.

Структура декоратору наступна:

– компонент задає загальний інтерфейс обгорток і об'єктів, що обгортаються;

– конкретний компонент визначає клас об'єктів, що обгортаються. Він містить якусь базову поведінку, яку потім змінюють декоратори;

– базовий декоратор зберігає посилання на вкладений об'єкт-компонент. Їм може бути як конкретний компонент, так і один з конкретних декораторів. Базовий декоратор делегує всі свої операції вкладеному об'єкту. Додаткова поведінка буде жити в конкретних декораторах;

– конкретні декоратори - це різні варіації декораторів, які містять додаткову поведінку. Воно виконується до або після виклику аналогічного поведінки обгорненого об'єкта;

– клієнт може обертати прості компоненти і декоратори в інші декоратори, працюючи з усіма об'єктами через загальний інтерфейс компонентів.

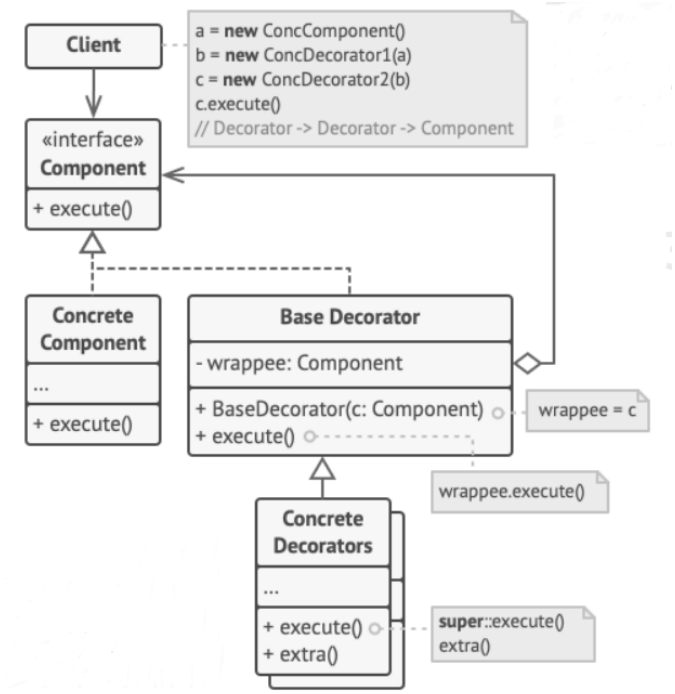

Рис. 2.8. Структура патерну Декоратор

Наступні патерни, що породжують.

Фабричний метод – це патерн проектування, який визначає загальний інтерфейс для створення об'єктів в суперкласі, дозволяючи підкласам змінювати тип створюваних об'єктів.

Його структура наступна:

– продукт визначає загальний інтерфейс об'єктів, який може зробити творець і його підкласи;

– конкретні продукти містять код різних продуктів. Продукти будуть відрізнятися реалізацією, але інтерфейс у них буде спільний.

– творець оголошує фабричний метод, який повинен повертати нові об'єкти продуктів. Важливо, щоб тип результату збігався із загальним інтерфейсом продуктів. Найчастіше фабричний метод оголошують абстрактним, щоб змусити всі підкласи реалізувати його по-своєму. Але він може повертати і якийсь стандартний продукт;

– конкретні творці по-своєму реалізують фабричний метод, виробляючи ті чи інші конкретні продукти.

– фабричний метод не зобов'язаний весь час створювати нові об'єкти. Його можна переписати так, щоб повертати існуючі об'єкти з якогось сховища або кеша.

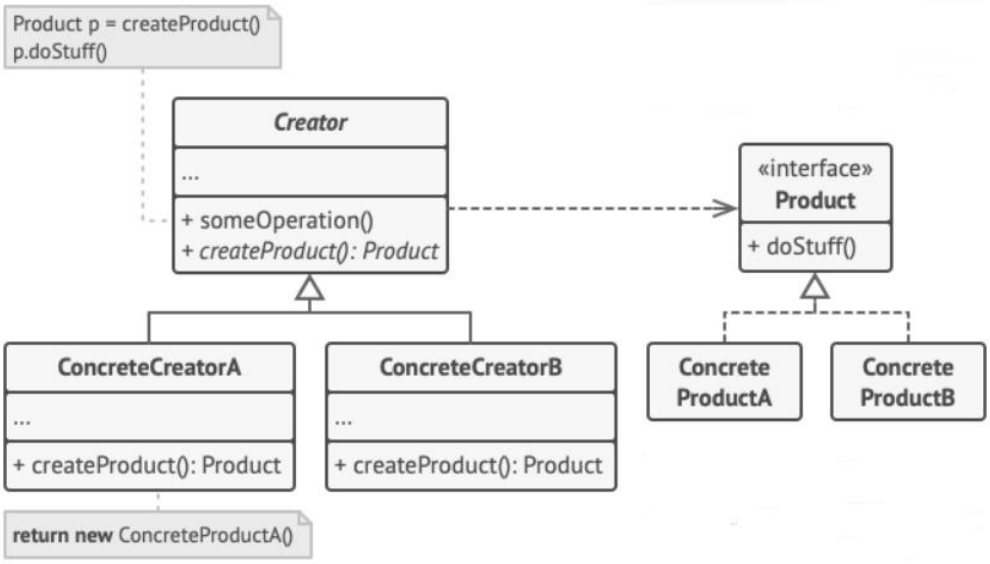

Рис. 2.9. Структура фабричного методу

Фабричний метод відокремлює код виробництва продуктів від решти коду, який ці продукти використовує.

Завдяки цьому, код виробництва можна розширювати, не чіпаючи основний. Так, щоб додати підтримку нового продукту, вам потрібно створити новий підклас і визначити в ньому фабричний метод, повертаючи звідти екземпляр нового продукту.

Будівельник - це патерн проектування, що породжує, який дозволяє створювати складні об'єкти покроково. Будівельник дає можливість використовувати один і той же код будівництва для отримання різних уявлень об'єктів.

Патерн пропонує розбити процес конструювання об'єкта на окремі кроки. Щоб створити об'єкт, вам потрібно по черзі викликати методи будівельника. Причому не потрібно запускати всі кроки, а тільки ті, що потрібні для виробництва об'єкта певної конфігурації.

Найчастіше один і той же крок будівництва може відрізнятися для різних варіацій вироблених об'єктів. Наприклад, дерев'яний будинок зажадає будівництва стін з дерева, а кам'яний - з каменю.

В цьому випадку ви можете створити кілька класів будівельників, які виконують одні і ті ж кроки по-різному. Використовуючи цих будівельників в одному і тому ж будівельному процесі, ви зможете отримувати на виході різні об'єкти.

Структура патерну Будівельник наступна:

- інтерфейс будівельника оголошує кроки конструювання продуктів, загальні для всіх видів будівельників;
- конкретні будівельники реалізують будівельні кроки, кожен по-своєму. Конкретні будівельники можуть виробляти різнорідні об'єкти, що не мають загального інтерфейсу;
- продукт - створюваний об'єкт. Продукти, зроблені різними будівельниками, не зобов'язані мати загальний інтерфейс;
- директор визначає порядок виклику будівельних кроків для виробництва тієї чи іншої конфігурації продуктів (випадок, коли можна виділити виклики методів будівельника в окремий клас, званий директором. У цьому випадку директор задаватиме порядок кроків будівництва, а будівельник - виконувати їх);
- зазвичай Клієнт подає в конструктор директора вже готовий об'єкт-будівельник, і в подальшому даний директор використовує тільки його. Але можливий і інший варіант, коли клієнт передає будівельника через параметр будівельного методу директора. В цьому випадку можна кожен раз застосовувати різних будівельників для виробництва різних уявлень об'єктів.

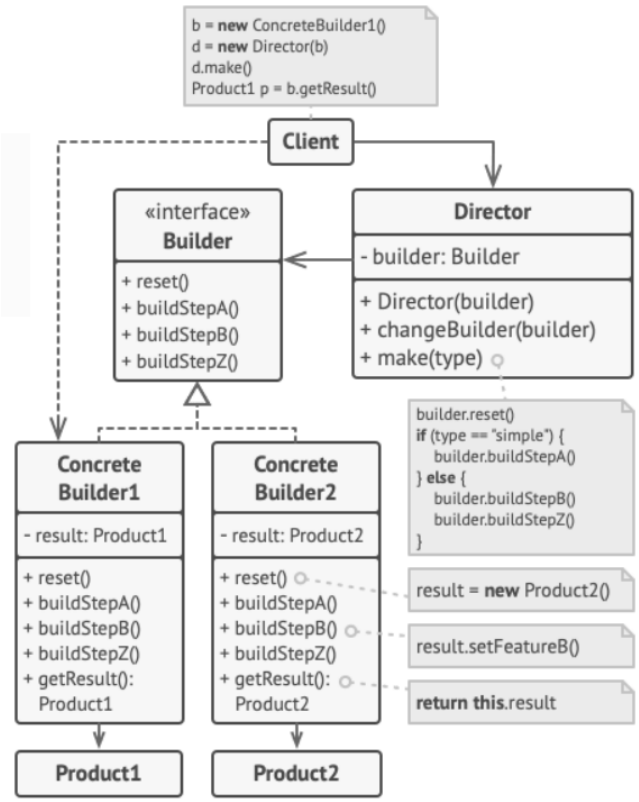

Рис. 2.10. Структура патерну Будівельник

Будівельник можна застосувати, якщо створення декількох уявлень об'єкта складається з однакових етапів, які відрізняються в деталях.

Інтерфейс будівельників визначить всі можливі етапи конструювання. Кожному представленню відповідатиме власний клас-будівельник. А порядок етапів будівництва буде задавати клас-директор.

Одинак - це патерн проектування, що породжує. Він гарантує, що у класу є тільки один екземпляр, і надає до нього глобальну точку доступу.

Всі реалізації одинаки зводяться до того, щоб приховати конструктор за замовчуванням і створити публічний статичний метод, який і буде контролювати життєвий цикл об'єкта-одинаки.

Якщо у вас є доступ до класу одинаки, значить, буде доступ і до цього статичного методу. З якої точки коду ви б його ні викликали, він завжди буде віддавати один і той же об'єкт.

Структура цього патерну наступна: одинак визначає статичний метод getInstance, який повертає єдиний екземпляр свого класу.

Конструктор одинаки повинен бути прихований від клієнтів. Виклик методу getInstance повинен стати єдиним способом отримати об'єкт цього класу.

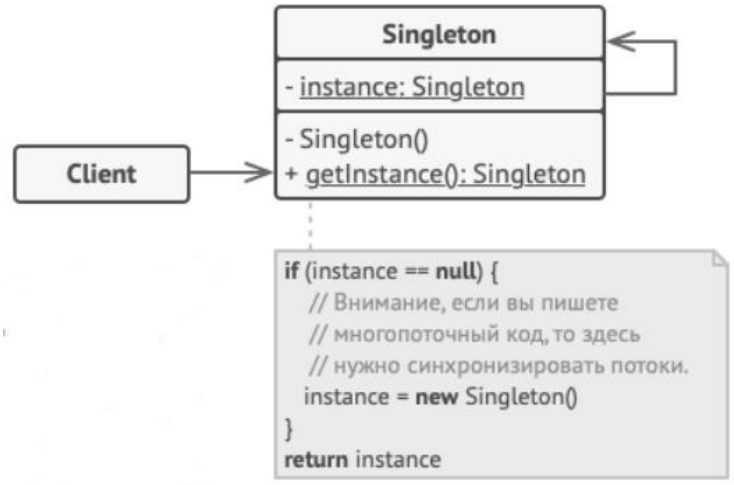

Рис. 2.11. Структура патерну Одинак

Одинак приховує від клієнтів все способи створення нового об'єкта, крім спеціального методу. Цей метод або створює об'єкт, або віддає існуючий об'єкт, якщо він вже був створений.

На відміну від глобальних змінних, Одинак гарантує, що жоден інший код не замінить створений екземпляр класу, тому ви завжди впевнені в наявності лише одного об'єкта-одинаки.

Проте, в будь-який момент ви можете розширити це обмеження і дозволити будь-яку кількість об'єктів-одинаків, помінявши код в одному місці (метод getInstance). [16]

2.5. Обґрунтування та організація вхідних та вихідних даних програми

В якості вхідних даних використовується файл JSON, який відповідає за інформацію клієнта. Та файл SQL, який відповідає за інформацію меню ресторану. Оформлення заказу теж зберігається в JSON файлі.

Кожний тип файлу має свій клас для введення та виведення даних у відповідному форматі. В даному додатку підтримуються файли з форматом JSON та SQL.

Для створення цієї моделі, введення та виведення даних ми використовуємо два патерни проектування, це фабричний та одиночний.

Single - це шаблон попереднього дизайну, який гарантує, що клас має лише один екземпляр і забезпечує глобальну точку доступу до нього.

Factory - це шаблонний дизайн, що породжує, який визначає загальний інтерфейс для створення об'єктів у суперкласі, що дозволяє підклас змінити тип створених об'єктів.

2.6. Опис роботи розробленої інформаційної системи

2.6.1. Використані технічні засоби

Для користувача є важливим мати систему, спроможну запускати та працювати з сучасними смартфонами, тож необхідно мати такі мінімальні параметри:

- ЦП [CPU]: Mediatek MT6739WW / 4 ядра / 1,5 ГГц.
- Відеоадаптер [GPU]: PowerVR GE8100 / Adreno 506.
- Оперативна пам’ять [RAM]: 2 ГБ;
- Постійна пам’ять [ROM]: 32 Гб;
- Операційна система: починаючи з Android 6 версії і вище.

2.6.2. Використані програмні засоби

Під час розробки даного застосунку були використані такі програмні засоби:

– Figma;

– Android Studio;

– Git, GitHub

Figma (Фігма) - це графічний онлайн-редактор для спільної роботи. У ньому можна створити прототип сайту, інтерфейс програми та обговорити правки з колегами в реальному часі.

У Figma можна «відмалювати» елементи інтерфейсу, створити інтерактивний прототип сайту і додатки, ілюстрації, векторну графіку. [17]

Android Studio - інтегроване середовище розробки виробництва Google, за допомогою якої розробникам стають доступні інструменти для створення додатків на платформі Android OS.

Під час створення додатків і утиліт для операційної системи Android, користувач програмного забезпечення може спостерігати за змінами в проекті, в режимі реального часу.

Android Studio - універсальне середовище розробки, так як дозволяє оптимізувати роботу майбутніх додатків для роботи не тільки на смартфонах, але і на планшетах, портативних ПК, які працюють на основі даної операційної системи.

У програму вбудований емулятор, що дозволяє перевірити коректну роботу програми на пристроях з різними екранами, з різними співвідношеннями сторін.

Відмітна особливість емулятора - перегляд приблизних показників продуктивності при запуску програми на найпопулярніших пристроях.

Середовище розробки для додатків Android Studio останньої версії стала по справжньому зручною навіть для початківців розробників. У програмі реалізовані всі сучасні засоби для упаковки коду, його маркування. [18]

Git - розподілена система контролю версій, яка дає можливість розробникам відстежувати зміни в файлах і працювати над одним проектом спільно з колегами.

GitHub - сервіс онлайн-хостингу репозиторіїв, що володіє всіма функціями розподіленого контролю версій і функціональністю управління вихідним кодом - все, що підтримує Git і навіть більше. [19]

2.6.3. Виклик та завантаження програми

Розроблений мобільний додаток завантажується та встановлюється через магазин мобільних додатків Google Play Store (або ще Play Market).

Якщо ж додаток Aspire завантажується стороннім шляхом, то безперечно спочатку треба завантажити APK-файл (формат архівних файлів) та шляхом його запуску встановити на телефон.

Після встановлення, в меню на головному екрані смартфона з’явиться іконка запуску мобільного додатку.

2.6.4. Опис інтерфейсу користувача

Робота починається з початкового екрану входу, після якого слідує другий слайдер, де можна перейти на сайт, який буде розроблений далі в ході проекту, але він буде націлений тільки на ресторатрів, які зможуть створити у системі цього додатку обліковий запис ресторану за допомогою окремої системи.

Тому для користувачів натискаємо на стрілочку Далі і попадаємо на екран реєстрації/ входу.

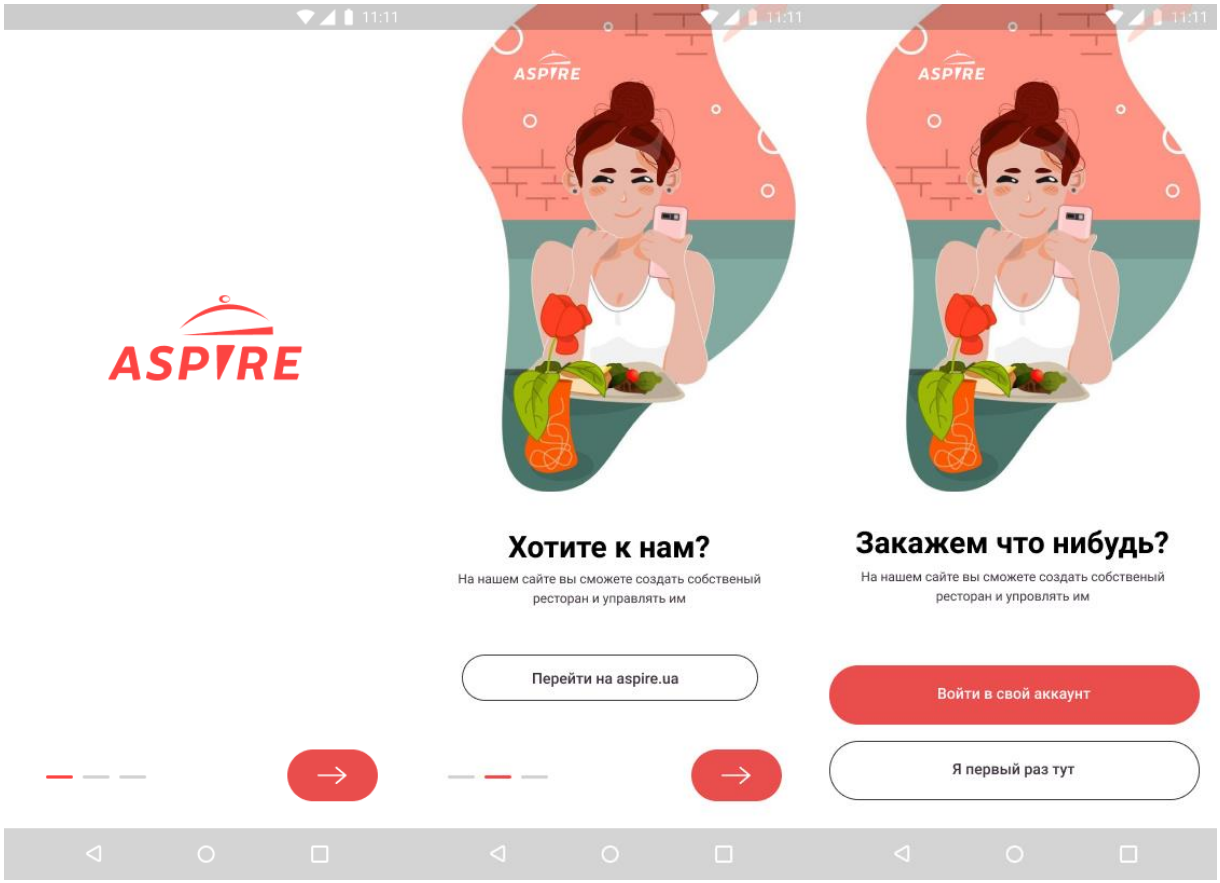

Рис. 2.12. Початковий екран додатку

Якщо немає акаунту в системі додатку, то треба здійснити реєстрацію. Якщо ж користувач зареєстрований, треба увійти. Надалі, вхід зберігається і постійно авторизуватись не треба.

Також можна здійснити авторизацію через соціальні мережі.

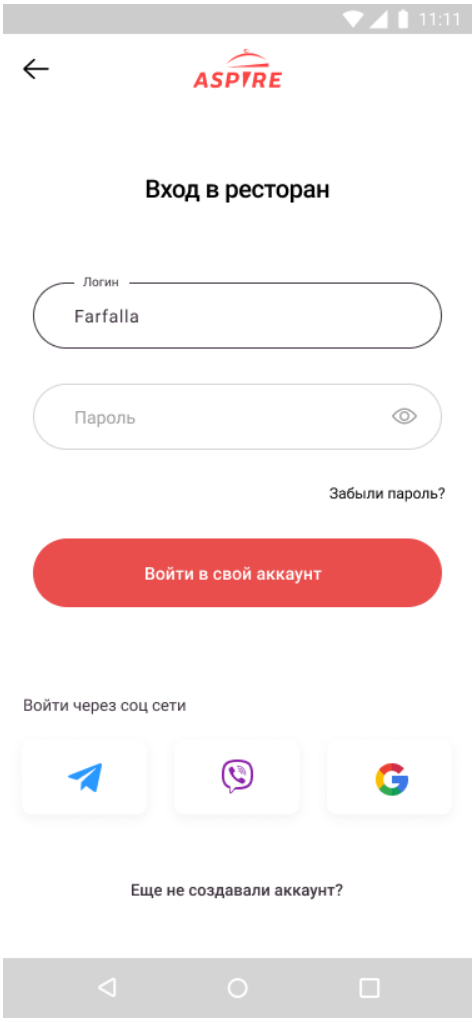

Рис. 2.13. Екран входу

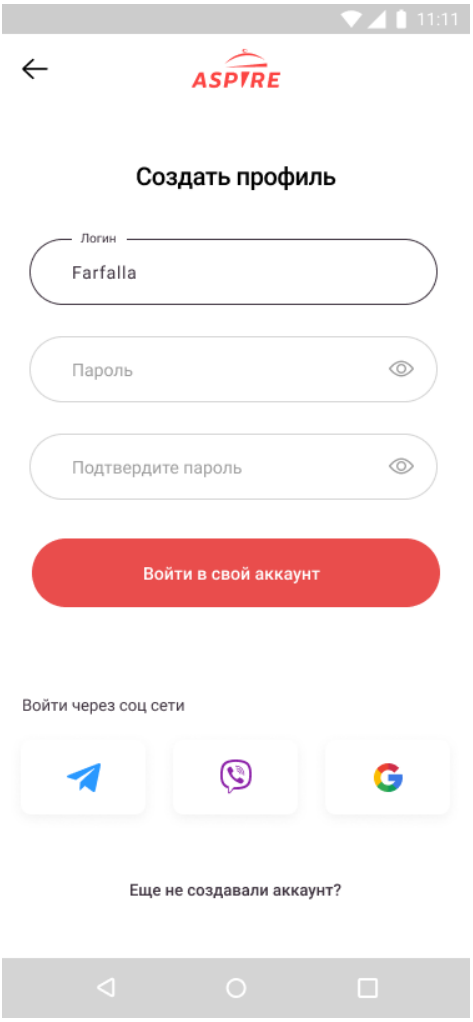

Рис. 2.14. Екран реєстрації

Після того як було реалізовано крок авторизації, користувач попадає на головну сторінку, де є:

– рекламний банер;

– фільтрація з: популярних ресторанів, нових ресторанів чи акційних пропозицій:

– перелік ресторанів з можливістю збереження їх в обране та перегляду рейтингів з годинами роботи;

– нижнє меню самого додатку (головна з переліком ресторанів, пошук, профіль користувача, вибране, замовлення).

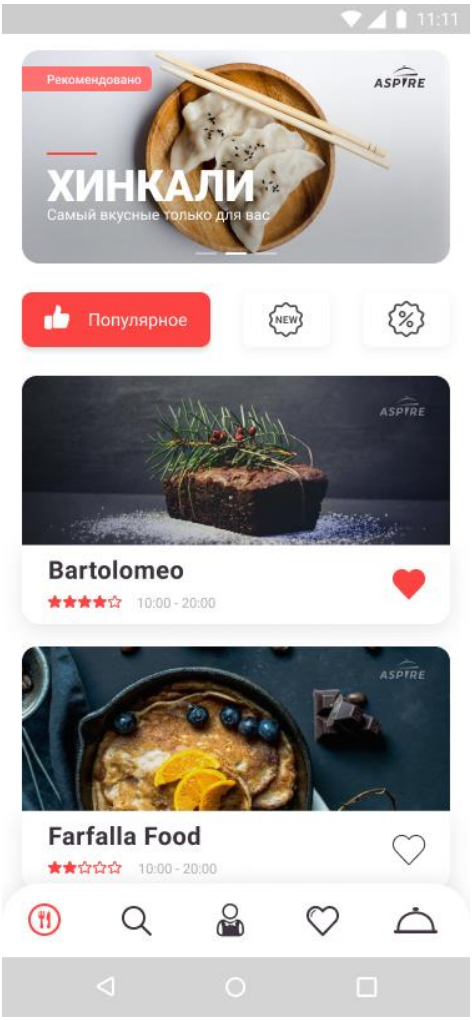

Рис. 2.15. Головна сторінка

Далі, як користувач обирає ресторан, який цікавить, йому стає доступне вікно, в якому Він може переглянути інформацію про даний ресторан. Та саме найголовніше – меню.

На цьому етапі також є вибірка, або ж фільтрація по меню, щоб можна було обрати з переліку «всього і відразу», холодних, гарячих блюд, або ж напоїв/десертів.

Блюдо дня виділяється рамкою для того, щоб привернути увагу.

Та відразу ж у «віконці» блюда, де є картинка, опис та ціна, можна за допомогою перемикачів «+» та «-» додати блюдо, чи певну його кількість у корзину для подальшого заказу.

Якщо ж натиснути на «Подробнее» в даному ресторані, можна отримати повну інформацію про: категорії кухні, на які орієнтовано заклад, опис, соціальні мережі.

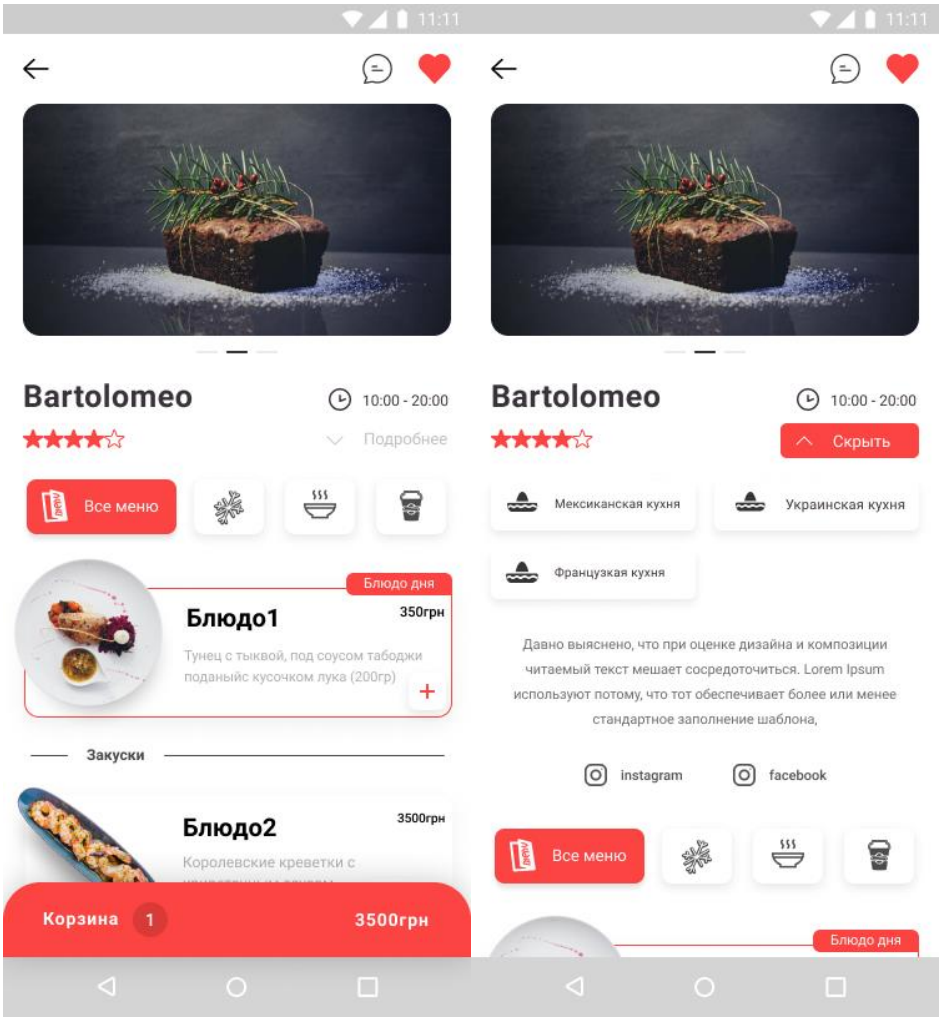

Рис. 2.16. Сторінка обраного ресторану

Якщо натиснути на блюдо, ми отримаємо про нього повну інформацію: фото самої страви, назву, ціну та вагу, зможемо обрати яку кількість блюд хочемо замовити, опис страви. Також додатково доступні інші блюда, які частіш за все обирають. Та кнопка, для того щоб додати в корзину для подальшого замовлення.

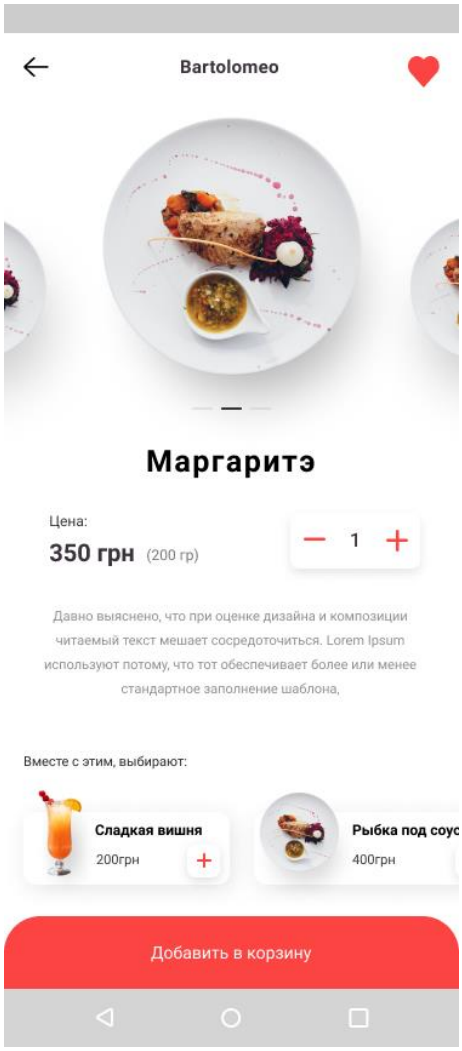

Рис. 2.17. Сторінка обраного блюда

Вікно пошуку. Тут доступний рядок для введення назви, потім категорія пошуку: ресторани, кухні або інгредієнти.

На даному прикладі продемонстрований пошук саме по ресторанам. По кухням чи інгредієнтам видача результатів буде в такому ж вигляді.

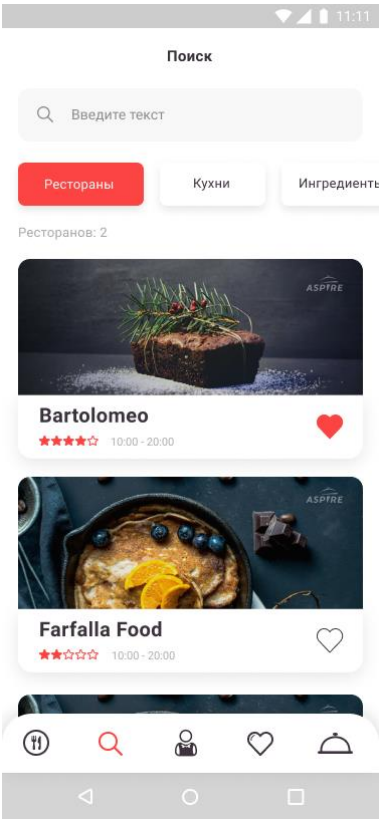

Рис. 2.18. Вікно пошуку (фільтр по ресторанам)

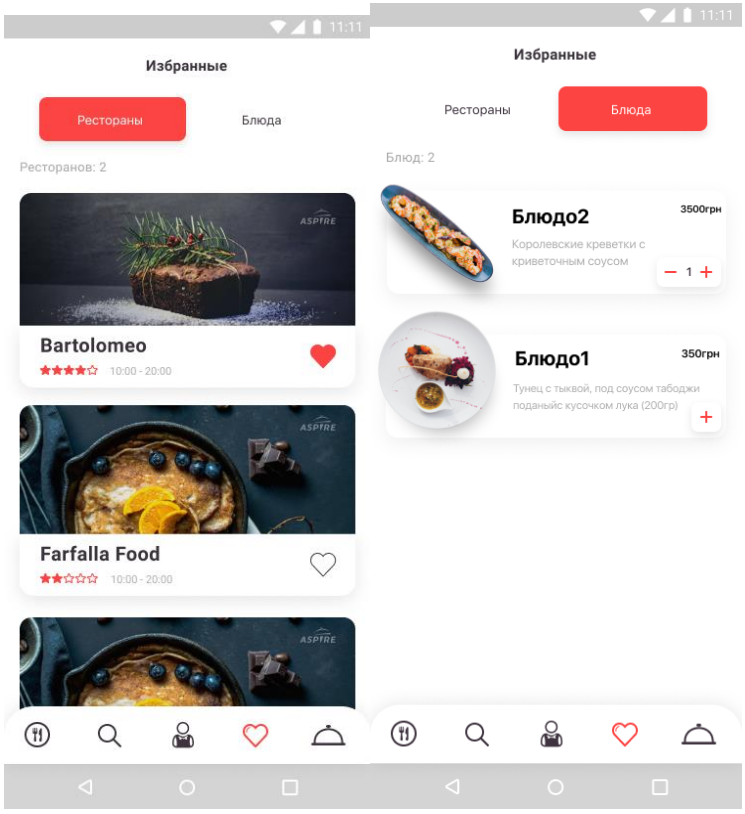

Рис. 2.19. Вікна обраного

Наступним вікном для огляду буде особистий профіль користувача, де зберігається необхідна інформація. Адреса доставки замовлення, промокоди до ресторанів на знижку чи акційної пропозиції, налаштування (наприклад, змінити пароль, пошту тощо), запросити друга у даний мобільний додаток (реферальні посилання), та підтримка служби, якщо додаток не працює за нормальними умовами.

Також в даному вікні можна налаштувати повідомлення чи вийти з профілю.

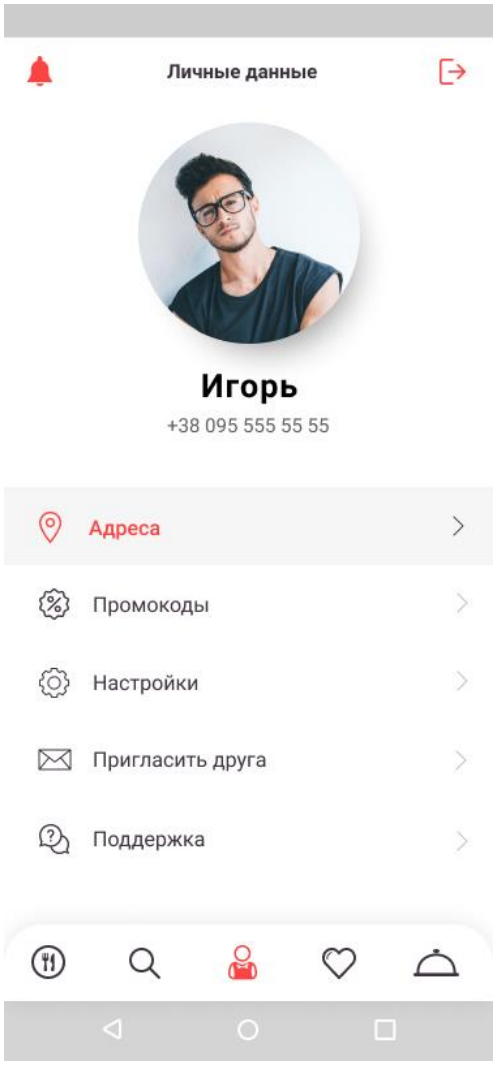

Рис. 2.20. Сторінка профілю

Наступною ланкою вікон є вікно, де можна переглянути замовлення: або, які очікуються, в доступі перегляд, або готування замовлення, або ж якщо блюдо вже «у дорозі».

Та архів з завершеними замовленнями.

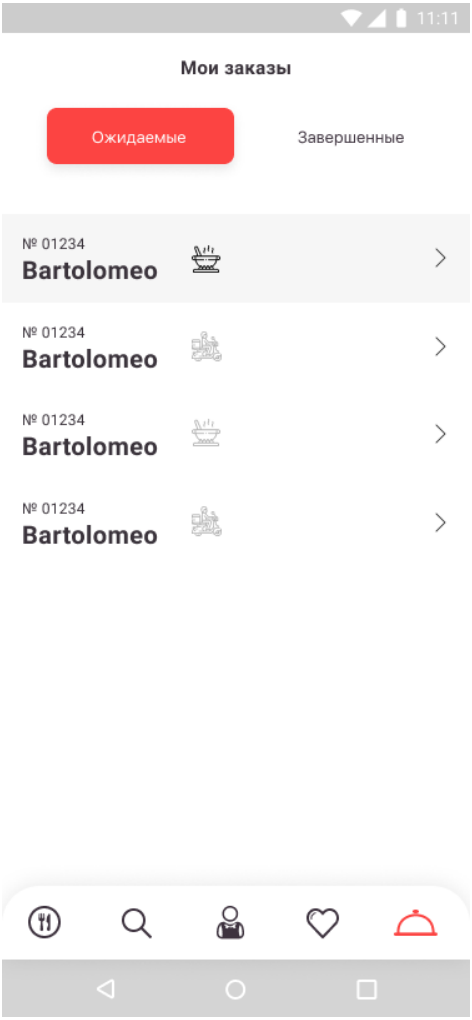

Рис. 2.21. Вікно замовлень

Вікно оформлення замовлення, де є перелік усіх налаштувань:

– обрані страви, з можливістю корегування кількості;

– кількість людей, на яку здійснюється замовлення для зручності розуміння сервірування;

– адреса доставки;

– в разі, якщо доставка потрібна якнайшвидше, або якщо на запланований та зручний для себе час, це також можна вказати;

– контактний номер для зв’язку зі споживачем;

– пункт, за яким можна вказати, який спосіб оплати буде здійснений;

– підсумкова вартість замовлення;

– кнопка для підтвердження та замовлення заказу.

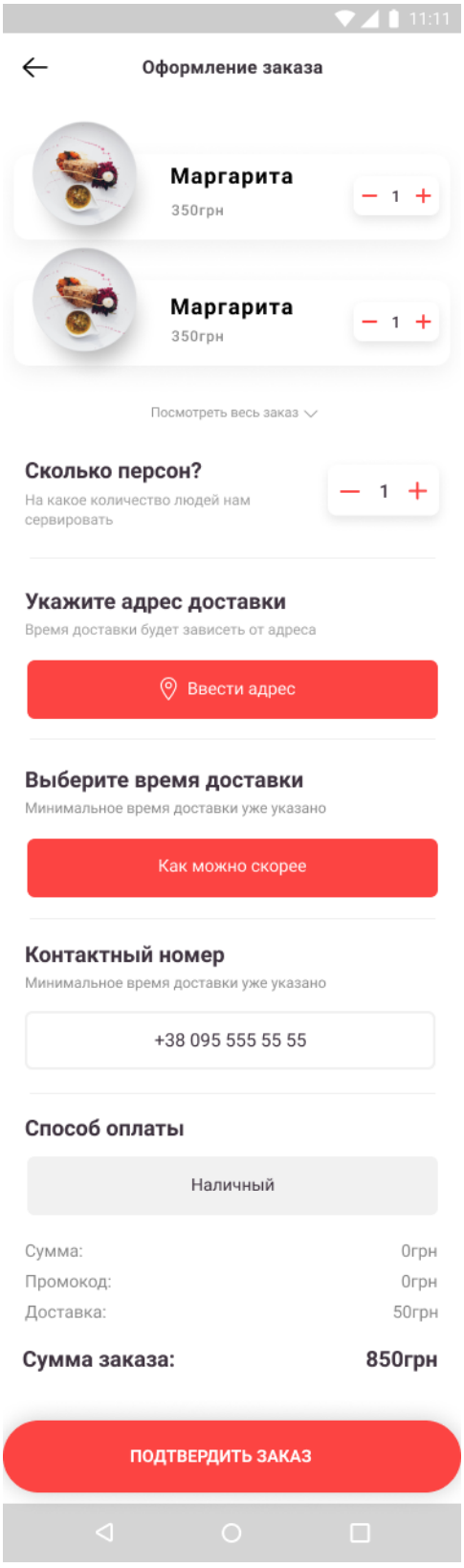

Рис. 2.22. Вікно оформлення замовлення

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

– передбачуване число операторів програми: 1400;

– коефіцієнт складності програми: 1,6;

– коефіцієнт корекції програми в ході її розробки: 0,07;

– годинна заробітна плата програміста: 90 грн/год;

– коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі: 1,2;

– коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності: 1,2;

– вартість машино-години ЕОМ: 13 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою (людино-годин):

$$t = t_o + t_{\text{и}} + t_{\text{п}} + t_{\text{отл}} + t_{\text{д}} \qquad (3.1)$$

де t_o- витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a- витрати праці на розробку блок-схеми алгоритму;

t_n-витрати праці на програмування по готовій блок-схемі;

$t_{отл}$-витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів (грн):

$$Q = q \cdot C \cdot (1 + p) \tag{3.2}$$

де q – передбачуване число операторів (1400);

C – коефіцієнт складності програми (1,6);

p – коефіцієнт корекції програми в ході її розробки (0,07).

Звідси умовне число операторів в програмі:

$$Q = 1{,}6 \cdot 1400 \cdot (1 + 0{,}07) = 2396{,}8$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста (людино-годин):

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k} \tag{3.3}$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 3 до 5 років він складає 1,2.

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (B = 1,2). З урахуванням коефіцієнта кваліфікації k = 1,2, отримуємо витрати праці на вивчення опису завдання:

$$t_u = \frac{2396{,}8 \cdot 1{,}2}{76 \cdot 1{,}2} = 31{,}53$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою (людино-годин):

$$t_a = \frac{Q}{(20..25) \cdot k} \tag{3.4}$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.4), людино-годин:

$$t_a = \frac{2396{,}8}{20 \cdot 1{,}2} = 99{,}86$$

Витрати на складання програми по готовій схемі (людино-годин):

$$t_{\text{п}} = \frac{Q}{(20..25) \cdot k} \tag{3.5}$$

$$t_{\text{п}} = \frac{2396{,}8}{25 \cdot 1{,}2} = 79{,}89$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання (людино-годин):

$$t_{\text{отл}} = \frac{Q}{(4..5) \cdot k} \tag{3.6}$$

$$t_{\text{отл}} = \frac{2396{,}8}{4 \cdot 1{,}2} = 499{,}33$$

– за умови комплексного налагодження завдання (людино-годин):

$$t_{\text{отл}}^{k} = 1{,}5 \cdot t_{\text{отл}} \tag{3.7}$$

$$t_{\text{отл}}^{k} = 1{,}5 \cdot 499{,}33 = 748{,}99$$

Витрати праці на підготовку документації визначаються за формулою (людино-годин):

$$t_{\text{д}} = t_{\text{др}} + t_{\text{до}} \tag{3.8}$$

де $t_{др}$-трудомісткість підготовки матеріалів і рукопису (людино-годин):

$$t_{\text{др}} = \frac{Q}{(15..20) \cdot k}$$

(3.9)

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації (людино-годин):

$$t_{\text{до}} = 0{,}75 \cdot t_{\text{др}} \tag{3.10}$$

Підставляючи відповідні значення, отримаємо (людино-годин):

$$t_{\text{др}} = \frac{2396{,}8}{18 \cdot 1{,}2} = 110{,}96$$

$$t_{\text{до}} = 0{,}75 \cdot 110{,}96 = 83{,}22$$

$$t_{\text{д}} = 110{,}96 + 83{,}22 = 194{,}18$$

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення (людино-годин):

$$t = 50 + 31{,}53 + 79{,}89 + 499{,}33 + 194{,}18 = 854{,}93$$

У результаті ми розрахували, що в загальній складності необхідно 854,93 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення додатку

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ (грн):

$$\text{К}_{\text{ПО}} = \text{З}_{\text{ЗП}} + \text{З}_{\text{МВ}} \tag{3.11}$$

Заробітна плата виконавців визначається за формулою (грн):

$$\text{З}_{\text{ЗП}} = t + \text{С}_{\text{ПР}} \tag{3.12}$$

де: t - загальна трудомісткість, людино-годин;

$C_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 65 грн / год, отримуємо (грн):

$$З_{ЗП} = 854{,}93 \cdot 90 = 76943{,}7$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою (грн):

$$З_{МВ} = t_{отл} \cdot C_{мч} \quad (3.13)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$C_{мч}$ - вартість машино-години ЕОМ, грн/год (13 грн/год).

Підставивши в формулу (3.3) відповідні значення, визначимо вартість необхідного для налагодження машинного часу (грн):

$$З_{МВ} = 499{,}33 \cdot 13 = 6491{,}29$$

Звідси витрати на створення програмного продукту (грн):

$$К_{ПО} = 76943{,}7 + 6491{,}29 = 83434{,}99$$

Очікуваний період створення ПЗ (міс):

$$T = \frac{t}{B_k \cdot F_p} \quad (3.14)$$

де B_k- число виконавців (дорівнює 1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

Звідси витрати на створення програмного продукту (міс):

$$T = \frac{854{,}93}{1 \cdot 176} \approx 4{,}8$$

Висновок: додаток розроблений з метою спростити процес створення циферблатів та їх редагування. Вартість даного додатку становить 83434,99 грн. та не потребує додаткових витрат при розробці проекту. Очікуваний час розробки приблизно 4,8 місяця. Цей термін пов'язаний зі значною кількістю операторів і включає в себе час для дослідження та розробки алгоритму розв'язання задачі, розробку дизайну, створення додатку та підготовку документації.

ВИСНОВКИ

В ході написання кваліфікаційної роботи були отримані цінні навички застосування сучасних інструментів розробки: відрисовка сучасного інтерфейсу (UI) і аналітика зручності при роботі і взаємодії користувача з інтерфейсом (UX). Були освоєні і впроваджені патерни проектування для зручної роботи і якісної розробки програмного коду. Були використані програми для підключення і підтримки клієнтів в форматі чату. За допомогою сучасних технологій емуляторів і нової мови Kotlin, сам додаток вийшов досить динамічним і яскравим в дії. Шифрування на рівні бази даних дозволяє зберігати в безпеці дані зареєстрованих користувачів.

Практичне завдання полягає в розробці мобільного додатку по доставці їжі з ресторанів і дозволяє користувачам заощаджувати свій час і розташовувати всі ресторани, систему відгуків і меню в одному місці.

Під час розробки даного дипломного проекту були виконані наступні завдання:

– аналітика та дослідження предметної області;
– ознайомлення з реалізацією нових програмних утиліт;
– відрисовка сучасного і яскравого дизайну;
– розробка програмного коду;
– написання рекомендацій та інструкції по використанню програми.

На даний момент, аналізуючи попит на подібні програми цей додаток єдиний, що не має свою доставку, а реалізує в одному місці (у системі самого програмного продукту) всі ресторани обраного міста України, тобто не потрібно в системі пошуку шукати окремий ресторан, чи встановлювати інші додатки по доставці, які просто співпрацюють з закладами громадського харчування. Це про те, що один додаток містить всі ресторани з окремими чи їхніми особистими способами доставки замовлення.

В «Економічному розділі» визначено трудомісткість розробки програмного забезпечення 854,93 людино-годин, підраховані витрати на створення програмного забезпечення 83434,99 грн. і гаданий період розробки приблизно 4,8 місяці.

СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Голощапов А. Google Android: програмування для мобільних пристроїв. 2010 р. 448 с.

2. Iyanu Adelekan. Kotlin Programming By Example. 2018 р. 732 с.

3. Head First. Программирование для Android. 2016 р. 704 с.

4. Josh Skeen, David Greenhalgh. Kotlin Programming: The Big Nerd Ranch Guide, First Edition. 2018 р. 549 с.

5. Head First. Kotlin. Девід и Дон Гриффітс. 2020 р. 464 с.

6. Жемеров Д., Ісакова С. Kotlin в дії. 2018 р. 402 с.

7. Сухов К. К. Node.js. Путівник по технології. 2015 р. 416 с.

8. Янг А., Мек Б., Кантелон М. Node.js в дії. 2-е вид. 201. 8 р. 432с.

9. Джеймс Грофф, Пол Вайнберг, Ендрю Опель, SQL Повне керівництво. Третє видання. 2015 р. 959 с.

10. Introduction to Transparent Data Encryption. Сайт з документацією Oracle Help Center. URL: https://docs.oracle.com/database/121/ASOAG/introduction-to-transparent-data-encryption.htm (переглянуто 05.06.2021)

11. Database Encryption in SQL Server 2008 Enterprise Edition, 09.04.2009. Офіційний сайт Microsoft. URL: https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008/ (переглянуто 11.06.2021)

12. Introduction: What Socket.IO is. Офіційний сайт Socket.IO. URL: https://socket.io/docs/v4/index.html (переглянуто 13.06.2021)

13. Javalin: A Simple, Modern Web Server Framework, by David Åse, March 15, 2019. Блог офіційного сайту Oracle. URL: https://blogs.oracle.com/javamagazine/javalin-a-simple-modern-web-server-framework (переглянуто 13.06.2021)

14. Spark – програма для спілкування секрети, 25.04.2020. Сайт Qipu. URL: https://qipu.ru/yota/spark-programma-dlya-obshcheniya-sekrety-ustanovka-i-nastroika-jabber-klientov.html (переглянуто 15.05.2021)

15. Александр Швець, Занурення в патерни проектування. 2018 р. 406 с.

16. Head First. Патерни проектування. 2018 р. 657 с.

17. Гід по Figma для початківців веб-дизайнерів. Освітній журнал платформи для створення сайтів Tilda Education. URL: https://tilda.education/articles-figma (переглянуто 17.05.2021)

18. Android Studio: середовище розробки мобільних додатків. Сайт ArduinoPlus. URL: https://arduinoplus.ru/android-studio/ (переглянуто 17.05.2021)

19. Difference Between Git and GitHub, Eduo Shaun, 09.02.2017. Medium – платформа соціальної журналістики. URL: https://medium.com/@eduoshaun/difference-between-git-and-github-807f1a57d438

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

```
package com.example.myapplication

import androidx.test.platform.app.InstrumentationRegistry
import androidx.test.ext.junit.runners.AndroidJUnit4
import org.junit.Test
import org.junit.runner.RunWith
import org.junit.Assert.*

/** * Instrumented test, which will execute on an Android device. * * See [testing documentation](http://d.android.com/tools/testing). */
@RunWith(AndroidJUnit4::class)
class ExampleInstrumentedTest {
    @Test
    fun useAppContext() {
        // Context of the app under test.
        val appContext = InstrumentationRegistry.getInstrumentation().targetContext
        assertEquals("com.example.myapplication", appContext.packageName)
    }
}
package com.example.myapplication

import org.junit.Test
import org.junit.Assert.*

/** * Example local unit test, which will execute on the development machine (host). * * See [testing documentation](http://d.android.com/tools/testing). */
class ExampleUnitTest {
    @Test
    fun addition_isCorrect() {
        assertEquals(4, 2 + 2)
    }
}
package com.example.myapplication
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.Toast

class Regestration : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_regestration)
    }

    fun BackReg(view: View){
        val intent= Intent(this,Glavnaya::class.java)
        startActivity(intent)
        finish()
    }
    fun openMain(view: View){
        val randomIntent = Intent(this, Glavnaya1::class.java)
        startActivity(randomIntent)
    }
    fun myFun(view:View){
        Toast.makeText(this, "А я причем?))))", Toast.LENGTH_SHORT).show()
    }
```

```
    fun myFun1(view:View){
        Toast.makeText(this, "Телеграм не работет!", Toast.LENGTH_SHORT).show()
    }
    fun myFun2(view:View){
        Toast.makeText(this, "Вайбер тоже!)", Toast.LENGTH_SHORT).show()
    }
    fun myFun3(view:View){
        Toast.makeText(this, "Не ростраюйся)))", Toast.LENGTH_SHORT).show()
    }
}package com.example.myapplication
import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.view.View
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import androidx.fragment.app.Fragment
import androidx.fragment.app.FragmentManager
import androidx.fragment.app.FragmentPagerAdapter
import kotlinx.android.synthetic.main.activity_main.*


class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val pref = getSharedPreferences("mypref", Context.MODE_PRIVATE)

        if (pref.getBoolean("firststart", true)) {
            // update sharedpreference - another start wont be the first
            val editor = pref.edit()
            editor.putBoolean("firststart", false)
            editor.commit() // apply changes
            val adapter=MyAdapter(supportFragmentManager)
            adapter.addFragment(Slider1())
            adapter.addFragment(Slider0())
            viewPager.adapter=adapter
            // first start, show your dialog | first-run code goes here
        }
        else{
            startActivity(Intent(this,Glavnaya::class.java))
            finish()
        }

    }
    class MyAdapter(manager: FragmentManager):FragmentPagerAdapter(manager){
        private val FragmentList:MutableList<Fragment> = ArrayList()
        override fun getItem(position: Int): Fragment {
          return FragmentList[position]
        }

        override fun getCount(): Int {
            return FragmentList.size
        }
        fun addFragment(fragment: Fragment){
            FragmentList.add(fragment)

        }
    }
    fun onClick(view: View){
        val currentPage:Int=viewPager.currentItem+1
        val t1 = view.findViewById<TextView>(R.id.textView)
        val t2 = view.findViewById<TextView>(R.id.textView3)
        if(currentPage<2){
```

```
            viewPager.currentItem=currentPage
            if(currentPage==1){
                //t1.setBackgroundResource(R.drawable.white_slider)
                //t2.setBackgroundResource(R.drawable.slider)
            }
        }
        else{
            startActivity(Intent(this,Glavnaya::class.java))
            finish()
        }
    }
}package com.example.myapplication

import android.content.Intent
import android.graphics.Typeface
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.TypedValue
import android.view.View
import android.widget.*
import com.google.android.flexbox.FlexboxLayout

class Waiting : AppCompatActivity() {
    val elemnt: MutableList<FrameLayout> = ArrayList()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.waiting)
        for(i in 0..5){
            add_elements(R.drawable.dish1,"Блюдо"+i,"Кухни","false")
        }

        val flex = findViewById<FlexboxLayout>(R.id.flexwait)
        show_restourant(flex, elemnt.size-1)
        val text= findViewById<TextView>(R.id.textView31)
        text.setText("Заказ номер 1321")
        val waitimg=findViewById<ImageView>(R.id.waitimg)
        waitimg.setImageResource(R.drawable.mc)
        val text2= findViewById<TextView>(R.id.waitname)
        text2.setText("McDonalds")
        val text3= findViewById<TextView>(R.id.waittime)
        text3.setText("Ожидается в ...12")
    }


    fun int_to_dp(x:Int): Int {
        val value = TypedValue.applyDimension(
            TypedValue.COMPLEX_UNIT_DIP,
            x.toFloat(),
            resources.displayMetrics
        ).toInt()
        return value
    }
    fun create_blydo(img:Int,name:String, tag_name:String): FrameLayout {
        val mp = LinearLayout.LayoutParams.MATCH_PARENT
        val wc = LinearLayout.LayoutParams.WRAP_CONTENT

        val frame_layout = FrameLayout(baseContext)//Главный контейнер
        frame_layout.layoutParams = FrameLayout.LayoutParams(wc,wc)
        frame_layout.setPadding(0,int_to_dp(11),0,0)
        frame_layout.tag = tag_name

        val frame_layout1 = FrameLayout(baseContext)//Контейнер с картинкой
        frame_layout1.layoutParams = FrameLayout.LayoutParams(wc,wc)
```

```
        val image_view  = ImageView(baseContext)//Картинка
        var params3 = FrameLayout.LayoutParams(int_to_dp(136), int_to_dp(136))
        params3.setMargins(int_to_dp(-5),0,0,0)
        image_view.setImageResource(img)
        image_view.layoutParams = params3


        val frame_layout2 = FrameLayout(baseContext)//Блюдо
        var params1 = FrameLayout.LayoutParams(int_to_dp(343), int_to_dp(104))
        params1.setMargins(int_to_dp(10),int_to_dp(43),0,int_to_dp(10))
        frame_layout2.layoutParams = params1
        frame_layout2.setBackgroundResource(R.drawable.layout3)

        var text_view3 = TextView(baseContext)//Название
        var params5 = FrameLayout.LayoutParams(wc,wc)
        params5.setMargins(int_to_dp(120),int_to_dp(11),0,0)
        text_view3.layoutParams = params5
        text_view3.text = name
        text_view3.textSize = 18f
        text_view3.setTypeface(Typeface.SANS_SERIF, Typeface.BOLD)
        text_view3.setTextColor(getResources().getColor(R.color.textDark))
        val text_view2 = TextView(baseContext)//Цена
        val params4 = FrameLayout.LayoutParams(wc,wc)
        params4.setMargins(int_to_dp(121),int_to_dp(50),0,0)
        text_view2.layoutParams = params4
        text_view2.text = "350грн"
        text_view2.textSize = 12f
        text_view2.setTypeface(Typeface.SANS_SERIF, Typeface.NORMAL)
        text_view2.setTextColor(getResources().getColor(R.color.textSilver))
        frame_layout2.addView(text_view3)
        frame_layout2.addView(text_view2)
        frame_layout1.addView(frame_layout2)
        frame_layout1.addView(image_view)


        frame_layout.addView(frame_layout1)
        return frame_layout
    }
    fun add_elements(img:Int,name:String, tag_name:String, like:String) {
        var f1: FrameLayout
        f1 = create_blydo(img, name, tag_name)
        elemnt.add(f1)
    }
    fun show_restourant(flex: FlexboxLayout, length:Int){
        if(length<=elemnt.size)
            for(i in 0..length){
                flex.addView(elemnt[i])
            }
    }


    fun back(view: View){
        val intent= Intent(this,Order::class.java)
        startActivity(intent)
        finish()
    }
}package com.example.myapplication

import android.content.Intent
import android.os.Bundle
import android.text.InputType
import android.view.View
import android.widget.EditText
```

```
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity

class Sign_In : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.sign__in)
        val password = findViewById(R.id.Password) as EditText
        password.inputType = InputType.TYPE_CLASS_TEXT or
InputType.TYPE_TEXT_VARIATION_PASSWORD;
    }
    fun BackSign(view: View){
        val intent= Intent(this,Glavnaya::class.java)
        startActivity(intent)
        finish()
    }
    fun openRegestration(view: View) {
        val Intent = Intent(this, Regestration::class.java)
        startActivity(Intent)
    }

    fun openMain(view: View) {
        val randomIntent = Intent(this, Glavnaya1::class.java)
        val login = findViewById(R.id.Login) as EditText
        val password = findViewById(R.id.Password) as EditText
        if(login.text.toString() == "pikimell")
            if(password.text.toString() == "admin")
                startActivity(randomIntent)
            else{
                Toast.makeText(this, "Неверный пароль!", Toast.LENGTH_SHORT).show()
            }
        else{
            Toast.makeText(this, "Не верный логин!", Toast.LENGTH_SHORT).show()
        }

    }

    fun myFun(view:View){
        Toast.makeText(this, "А я причем?))))", Toast.LENGTH_SHORT).show()
    }
    fun myFun1(view:View){
        Toast.makeText(this, "Телеграм не работет!", Toast.LENGTH_SHORT).show()
    }
    fun myFun2(view:View){
        Toast.makeText(this, "Вайбер тоже!)", Toast.LENGTH_SHORT).show()
    }
    fun myFun3(view:View){
        Toast.makeText(this, "Не ростраюйся)))", Toast.LENGTH_SHORT).show()
    }

    fun showPassword(view:View){
        val password = findViewById(R.id.Password) as EditText
        if(password.tag == "hide"){
            password.inputType = InputType.TYPE_CLASS_TEXT or
InputType.TYPE_TEXT_VARIATION_PASSWORD;
            password.tag = "show"
        }else{
            password.inputType = InputType.TYPE_TEXT_VARIATION_VISIBLE_PASSWORD;
            password.tag = "hide"
        }

    }
}package com.example.myapplication
```

```
import android.content.Intent
import android.graphics.Color
import android.graphics.Typeface
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.TypedValue
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.view.animation.Animation
import android.view.animation.AnimationUtils
import android.widget.*
import androidx.fragment.app.Fragment
import com.google.android.flexbox.FlexboxLayout
import kotlinx.android.synthetic.main.activity_saved_menu.*

class Saved_menu : Fragment(R.layout.activity_saved_menu) {
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {

        val view: View = inflater.inflate(R.layout.activity_saved_menu, container, false)
        val buttons = arrayOf(
            view.findViewById(R.id.Restaurantsaved) as Button,
            view.findViewById(R.id.Bluda) as Button
        )
        val elemnt: MutableList<FrameLayout> = ArrayList()
        val animations: Array<Animation?> = arrayOf(
            AnimationUtils.loadAnimation(requireContext(), R.anim.alpha2)
        )


        fun int_to_dp(x:Int): Int {
            val value = TypedValue.applyDimension(
                TypedValue.COMPLEX_UNIT_DIP,
                x.toFloat(),
                resources.displayMetrics
            ).toInt()
            return value
        }

        fun create_frame(img:Int,name:String, tag_name:String, like:String): FrameLayout {
            var params1 = FrameLayout.LayoutParams(int_to_dp(343), int_to_dp(190))
            val mp = LinearLayout.LayoutParams.MATCH_PARENT
            val wc = LinearLayout.LayoutParams.WRAP_CONTENT



            val f = FrameLayout(this.requireContext())//Главный контейнер
            f.layoutParams = FrameLayout.LayoutParams(wc,wc)
            f.tag = tag_name

            val fl = FrameLayout(this.requireContext())//Главный контейнер
            params1.setMargins(0,int_to_dp(16),0,int_to_dp(0))
            fl.layoutParams = params1
            fl.setBackgroundResource(R.drawable.shadow)
            fl.setPadding(10,0,10,20)


            val iv = ImageView(this.requireContext())//Картинка ресторана
            iv.setImageResource(img)
            iv.layoutParams = LinearLayout.LayoutParams(mp,wc)
```

```
iv.scaleType = ImageView.ScaleType.CENTER_CROP
iv.setOnClickListener {
    activity?.let {
        val intent = Intent(it, Restaurant_menu::class.java)
        it.startActivity(intent)
    }
}


val fl2= FrameLayout(this.requireContext())//нижний контейнер
val params2 = LinearLayout.LayoutParams(0,0)
params2.width = mp
params2.height = int_to_dp(60)
params2.setMargins(0,int_to_dp(130),0,0)
fl2.layoutParams = params2
fl2.setBackgroundColor(Color.WHITE)
fl2.setBackgroundResource(R.drawable.bottom_block)


val tv = TextView(this.requireContext())//Название ресторана
val params3 = LinearLayout.LayoutParams(0,0,1000f)
tv.text = name
params3.width = wc
params3.height = wc
params3.setMargins(int_to_dp(21),int_to_dp(8),0,0)
tv.layoutParams = params3
tv.textSize = 20f
tv.setTypeface(Typeface.SANS_SERIF, Typeface.BOLD)
tv.setTextColor(getResources().getColor(R.color.textDark))
tv.setPadding(21, 9, 0, 0)


val img= ImageView(this.requireContext())//Звезды ресторана
val params4 = LinearLayout.LayoutParams(int_to_dp(58),int_to_dp(9))
params4.setMargins(int_to_dp(24),int_to_dp(40),0,0)
img.setImageResource(R.drawable.stars)
img.layoutParams= params4


val heart=ImageButton(this.requireContext())//лайк ресторана
val params5 = LinearLayout.LayoutParams(wc,wc)
params5.setMargins(int_to_dp(290),int_to_dp(7),0,0)
heart.setImageResource(R.drawable.ic_heart_black)
heart.layoutParams= params5
heart.setBackgroundColor(Color.TRANSPARENT)
heart.tag = like
if (heart.tag == "false") {
    heart.setImageResource(R.drawable.ic_heart_black)
    heart.tag = "true"
} else {
    heart.setImageResource(R.drawable.ic_heart_like_red)
    heart.tag = "false"
}
heart.setOnClickListener {
    if (heart.tag == "false") {
        heart.setImageResource(R.drawable.ic_heart_black)
        heart.tag = "true"
    } else {
        heart.setImageResource(R.drawable.ic_heart_like_red)
        heart.tag = "false"
    }
}
```

```
    val fl3 = FrameLayout(this.requireContext())//контейнер для текста с временем
    val params6 = LinearLayout.LayoutParams(wc,wc)
    params6.setMargins(int_to_dp(40),int_to_dp(25),0,0)
    fl3.layoutParams= params6
    fl3.setPadding(100,25,0,0)


    val txt = TextView(this.requireContext()) //Время работы ресторана
    val params7 = LinearLayout.LayoutParams(wc,wc)
    txt.text = "08:00 - 22:00"
    txt.layoutParams = params7
    txt.textSize = 11f
    txt.setTypeface(Typeface.SANS_SERIF, Typeface.NORMAL)
    txt.setTextColor(getResources().getColor(R.color.textSilver))
    txt.setPadding(21, 9, 0, 0)


    fl3.addView(txt)
    fl2.addView(tv)
    fl2.addView(img)
    fl2.addView(fl3)
    fl2.addView(heart)
    fl.addView(iv)
    fl.addView(fl2)
    f.addView(fl)
    return f
}

fun add_elements(img:Int,name:String, tag_name:String, like:String) {
    val f1 = create_frame(img, name, tag_name, like)
    elemnt.add(f1)
}

fun show_restourant(flex: FlexboxLayout, filter_tag:String){
    flex.removeAllViewsInLayout()
    for(i in 0..elemnt.size-1){
        if(elemnt[i].tag == filter_tag)
            flex.addView(elemnt[i])
        else if (filter_tag == "")
            flex.addView(elemnt[i])
    }
}

fun reset_button(my_iter:Int){
    for(i in 0..buttons.size-1){
        buttons[i].setBackgroundResource(R.drawable.shadow)
        buttons[i].setTextColor(getResources().getColor(R.color.textDark))
    }
    buttons[my_iter].setBackgroundResource(R.drawable.button_shadow)
    buttons[my_iter].setTextColor(getResources().getColor(R.color.colorWhite))
    buttons[my_iter].startAnimation(animations[0])
}


val txt=view.findViewById<TextView>(R.id.myText)
txt.setText("Всего"+elemnt.size)


buttons[0].setOnClickListener {
    var my_iter = 0
    val idx = 0
    reset_button(idx)
    show_restourant(view.findViewById(R.id.flexsaved) , "Рестораны")
```

```
            val txt=view.findViewById<TextView>(R.id.myText)
            for(i in 0..elemnt.size-1) {
                if (elemnt[i].tag == "Рестораны")
                    my_iter += 1
            }
            txt.setText("Ресторанов "+my_iter)
        }
        buttons[1].setOnClickListener {
            var my_iter = 0
            val idx = 1
            reset_button(idx)
            show_restourant(view.findViewById(R.id.flexsaved) , "Блюда")
            val txt=view.findViewById<TextView>(R.id.myText)
            for(i in 0..elemnt.size-1) {
                if (elemnt[i].tag == "Блюда")
                    my_iter += 1
            }
            txt.setText("Блюд "+my_iter)
        }

        fun but(view: View) {

        }


        for(i in 0..5){
            add_elements(R.drawable.ic_dark_tort,"Bartolomeo","Рестораны","false")
            add_elements(R.drawable.kfc_logo,"Борщ","Блюда","false")
            add_elements(R.drawable.chel,"Сосиска","Рестораны","false")
        }
        show_restourant(view.findViewById(R.id.flexsaved),"")


        return view

    }
}package com.example.myapplication

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment

class Slider1 :Fragment(R.layout.slider) {
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return super.onCreateView(inflater, container, savedInstanceState)
    }
}package com.example.myapplication

import android.content.Intent
import android.graphics.Color
import android.graphics.Typeface
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.TypedValue
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.view.animation.Animation
```

```
import android.view.animation.AnimationUtils
import android.widget.*
import androidx.fragment.app.Fragment
import com.google.android.flexbox.FlexboxLayout

class Basket_menu : Fragment(R.layout.activity_basket_menu) {
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {

        val view: View = inflater.inflate(R.layout.activity_basket_menu, container, false)

        val buttons = arrayOf(
            view.findViewById(R.id.Restaurant_bask) as Button,
            view.findViewById(R.id.Bluda_bask) as Button
        )
        val elemnt: MutableList<FrameLayout> = ArrayList()
        val animations: Array<Animation?> = arrayOf(
            AnimationUtils.loadAnimation(requireContext(), R.anim.alpha2)
        )


        fun int_to_dp(x:Int): Int {
            val value = TypedValue.applyDimension(
                TypedValue.COMPLEX_UNIT_DIP,
                x.toFloat(),
                resources.displayMetrics
            ).toInt()
            return value
        }

        fun create_frame(number:String,name:String, tag_name:String): FrameLayout {
            val mp = LinearLayout.LayoutParams.MATCH_PARENT
            val wc = LinearLayout.LayoutParams.WRAP_CONTENT
            var params1 = FrameLayout.LayoutParams(mp, int_to_dp(71))



            val f = FrameLayout(this.requireContext())//Главный контейнер
            f.layoutParams = params1
            f.tag = tag_name


            val tv = TextView(this.requireContext())//Номер заказа
            val params2 = LinearLayout.LayoutParams(0,0,1000f)
            tv.text = number
            params2.width = wc
            params2.height = int_to_dp(13)
            params2.setMargins(int_to_dp(26),int_to_dp(12),0,0)
            tv.layoutParams = params2
            tv.textSize = 12f
            tv.setTypeface(Typeface.SANS_SERIF, Typeface.BOLD)
            tv.setTextColor(getResources().getColor(R.color.textDark))


            val tv2 = TextView(this.requireContext())//Название
            val params3 = LinearLayout.LayoutParams(0,0,1000f)
            tv2.text = name
            params3.width = wc
            params3.height = int_to_dp(23)
            params3.setMargins(int_to_dp(26),int_to_dp(25),0,0)
            tv2.layoutParams = params3
```

```
    tv2.textSize = 20f
    tv2.setTypeface(Typeface.SANS_SERIF, Typeface.BOLD)
    tv2.setTextColor(getResources().getColor(R.color.textDark))


    val img= ImageView(this.requireContext())//Тарелка
    val params4 = LinearLayout.LayoutParams(int_to_dp(30),int_to_dp(30))
    params4.setMargins(int_to_dp(172),int_to_dp(15),0,0)
    img.setImageResource(R.drawable.ic_restaurants_black)
    img.layoutParams= params4
    val arrow= ImageView(this.requireContext())//Стрелка
    val params5 = LinearLayout.LayoutParams(int_to_dp(34),int_to_dp(20))
    params5.setMargins(int_to_dp(320),int_to_dp(22),0,0)
    arrow.setImageResource(R.drawable.ic_arrow_forward_black)
    arrow.layoutParams= params5



    f.addView(tv)
    f.addView(tv2)
    f.addView(img)
    f.addView(arrow)
    return f
}

fun add_elements(number:String,name:String, tag_name:String) {
    val f1 = create_frame(number, name, tag_name)
    elemnt.add(f1)
}

fun show_restourant(flex: LinearLayout, filter_tag:String){
    flex.removeAllViewsInLayout()
    for(i in 0..elemnt.size-1){
        if(elemnt[i].tag == filter_tag)
            flex.addView(elemnt[i])
        else if (filter_tag == "")
            flex.addView(elemnt[i])
    }
}

fun reset_button(my_iter:Int){
    for(i in 0..buttons.size-1){
        buttons[i].setBackgroundResource(R.drawable.shadow)
        buttons[i].setTextColor(getResources().getColor(R.color.textDark))
    }
    buttons[my_iter].setBackgroundResource(R.drawable.button_shadow)
    buttons[my_iter].setTextColor(getResources().getColor(R.color.colorWhite))
    buttons[my_iter].startAnimation(animations[0])
}




buttons[0].setOnClickListener {
    val my_iter = 0
    reset_button(my_iter)
    show_restourant(view.findViewById(R.id.basket_lin) , "Ожидаемые")
}
buttons[1].setOnClickListener{
    val my_iter = 1
    reset_button(my_iter)
    show_restourant(view.findViewById(R.id.basket_lin) , "Завершённые")
}
```

```
        fun but(view: View) {

        }


        for(i in 0..5){
            add_elements("1234","Bartolomeo","Ожидаемые")
            add_elements("3214","MC","Завершённые")
            add_elements("2314","Chelentano","Ожидаемые")
        }
        show_restourant(view.findViewById(R.id.basket_lin),"")

        return view

    }
}package com.example.myapplication

import android.content.Intent
import android.os.Bundle
import android.view.View
import android.widget.ImageButton
import android.widget.ImageView
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import com.denzcoskun.imageslider.ImageSlider
import com.denzcoskun.imageslider.models.SlideModel


class Dish: AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.dish)
        val imageSlider = findViewById<ImageSlider>(R.id.viewPager2)

        val slideModels: MutableList<SlideModel> = ArrayList()
        slideModels.add(
            SlideModel("фцвф")
        )
        slideModels.add(
            SlideModel("фцв")
        )
        slideModels.add(
            SlideModel("цфв")
        )
        imageSlider.setImageList(slideModels, true)

    }
    fun return_Rest(view: View){
        val intent= Intent(this,Restaurant_menu::class.java)
        startActivity(intent)
        finish()
    }
    fun likeClick(view: View){
        val b = view as ImageButton

        if(b.tag == "false"){
            b.setImageResource(R.drawable.ic_heart_black)
            b.tag = "true"
        }else{
            b.setImageResource(R.drawable.ic_heart_like_red)
            b.tag = "false"
        }
    }
```

```
    fun plus(view: View){
        val tt = findViewById(R.id.iterator) as TextView;
        tt.text = (tt.text.toString().toInt() + 1).toString()
    }

    fun minus(view: View){
        val tt = findViewById(R.id.iterator) as TextView;

        if(tt.text.toString().toInt()>=1)
            tt.text = (tt.text.toString().toInt() - 1).toString()
    }
}package com.example.myapplication

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment

class Slider0 :Fragment(R.layout.slider_0) {
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return super.onCreateView(inflater, container, savedInstanceState)
    }
}package com.example.myapplication

import android.content.Intent
import android.graphics.Color
import android.graphics.Typeface
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.TypedValue
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.view.animation.Animation
import android.view.animation.AnimationUtils
import android.widget.*
import androidx.core.view.marginLeft
import androidx.fragment.app.Fragment
import com.denzcoskun.imageslider.ImageSlider
import com.denzcoskun.imageslider.models.SlideModel
import com.google.android.flexbox.FlexboxLayout
import kotlinx.android.synthetic.main.activity_find_menu.*

class Find_menu : Fragment(R.layout.activity_find_menu) {
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {

        val view: View = inflater.inflate(R.layout.activity_find_menu, container, false)
        val buttons:Array<Button> = arrayOf(
            view.findViewById(R.id.Restaurant) as Button,
            view.findViewById(R.id.Kuhni) as Button,
            view.findViewById(R.id.Ingridients) as Button
        )
        val elemnt: MutableList<FrameLayout> = ArrayList()
        val animations: Array<Animation?> = arrayOf(
            AnimationUtils.loadAnimation(requireContext(), R.anim.alpha),
```

```
    AnimationUtils.loadAnimation(requireContext(), R.anim.alpha2),
    AnimationUtils.loadAnimation(requireContext(), R.anim.combo2)
)


fun int_to_dp(x:Int): Int {
    val value = TypedValue.applyDimension(
        TypedValue.COMPLEX_UNIT_DIP,
        x.toFloat(),
        resources.displayMetrics
    ).toInt()
    return value
}
fun create_frame(img:Int,name:String, tag_name:String, like:String): FrameLayout {
    var params1 = FrameLayout.LayoutParams(int_to_dp(343), int_to_dp(190))
    val mp = LinearLayout.LayoutParams.MATCH_PARENT
    val wc = LinearLayout.LayoutParams.WRAP_CONTENT



    val f = FrameLayout(this.requireContext())//Главный контейнер
    f.layoutParams = FrameLayout.LayoutParams(wc,wc)
    f.tag = tag_name

    val fl = FrameLayout(this.requireContext())//Главный контейнер
    params1.setMargins(0,int_to_dp(16),0,int_to_dp(0))
    fl.layoutParams = params1
    fl.setBackgroundResource(R.drawable.shadow)
    fl.setPadding(10,0,10,20)


    val iv = ImageView(this.requireContext())//Картинка ресторана
    iv.setImageResource(img)
    iv.layoutParams = LinearLayout.LayoutParams(mp,wc)
    iv.scaleType = ImageView.ScaleType.CENTER_CROP
    iv.setOnClickListener {
        activity?.let {
            val intent = Intent(it, Restaurant_menu::class.java)
            it.startActivity(intent)
        }
    }


    val fl2= FrameLayout(this.requireContext())//нижний контейнер
    val params2 = LinearLayout.LayoutParams(0,0)
    params2.width = mp
    params2.height = int_to_dp(60)
    params2.setMargins(0,int_to_dp(130),0,0)
    fl2.layoutParams = params2
    fl2.setBackgroundColor(Color.WHITE)
    fl2.setBackgroundResource(R.drawable.bottom_block)


    val tv = TextView(this.requireContext())//Название ресторана
    val params3 = LinearLayout.LayoutParams(0,0,1000f)
    tv.text = name
    params3.width = wc
    params3.height = wc
    params3.setMargins(int_to_dp(21),int_to_dp(8),0,0)
    tv.layoutParams = params3
    tv.textSize = 20f
    tv.setTypeface(Typeface.SANS_SERIF, Typeface.BOLD)
    tv.setTextColor(getResources().getColor(R.color.textDark))
    tv.setPadding(21, 9, 0, 0)
```

```
    val img= ImageView(this.requireContext())//3везды ресторана
    val params4 = LinearLayout.LayoutParams(int_to_dp(58),int_to_dp(9))
    params4.setMargins(int_to_dp(24),int_to_dp(40),0,0)
    img.setImageResource(R.drawable.stars)
    img.layoutParams= params4


    val heart=ImageButton(this.requireContext())//лайк ресторана
    val params5 = LinearLayout.LayoutParams(wc,wc)
    params5.setMargins(int_to_dp(290),int_to_dp(7),0,0)
    heart.setImageResource(R.drawable.ic_heart_black)
    heart.layoutParams= params5
    heart.setBackgroundColor(Color.TRANSPARENT)
    heart.tag = like
    if (heart.tag == "false") {
        heart.setImageResource(R.drawable.ic_heart_black)
        heart.tag = "true"
    } else {
        heart.setImageResource(R.drawable.ic_heart_like_red)
        heart.tag = "false"
    }
    heart.setOnClickListener {
        if (heart.tag == "false") {
            heart.setImageResource(R.drawable.ic_heart_black)
            heart.tag = "true"
        } else {
            heart.setImageResource(R.drawable.ic_heart_like_red)
            heart.tag = "false"
        }
    }


    val fl3 = FrameLayout(this.requireContext())//контейнер для текста с временем
    val params6 = LinearLayout.LayoutParams(wc,wc)
    params6.setMargins(int_to_dp(40),int_to_dp(25),0,0)
    fl3.layoutParams= params6
    fl3.setPadding(100,25,0,0)


    val txt = TextView(this.requireContext()) //Время работы ресторана
    val params7 = LinearLayout.LayoutParams(wc,wc)
    txt.text = "08:00 - 22:00"
    txt.layoutParams = params7
    txt.textSize = 11f
    txt.setTypeface(Typeface.SANS_SERIF, Typeface.NORMAL)
    txt.setTextColor(getResources().getColor(R.color.textSilver))
    txt.setPadding(21, 9, 0, 0)


    fl3.addView(txt)
    fl2.addView(tv)
    fl2.addView(img)
    fl2.addView(fl3)
    fl2.addView(heart)
    fl.addView(iv)
    fl.addView(fl2)
    f.addView(fl)
    return f
}
fun create_blydo(img:Int,name:String, tag_name:String):FrameLayout{
    val mp = LinearLayout.LayoutParams.MATCH_PARENT
    val wc = LinearLayout.LayoutParams.WRAP_CONTENT
```

```
    val frame_layout = FrameLayout(this.requireContext())//Главный контейнер
    frame_layout.layoutParams = FrameLayout.LayoutParams(wc,wc)
    frame_layout.setPadding(0,int_to_dp(11),0,0)
    frame_layout.tag = tag_name

    val frame_layout1 = FrameLayout(this.requireContext())
    frame_layout1.layoutParams = FrameLayout.LayoutParams(wc,wc)


    val image_view  = ImageView(this.requireContext())
    var params3 = FrameLayout.LayoutParams(int_to_dp(136), int_to_dp(136))
    params3.setMargins(int_to_dp(-5),0,0,0)
    image_view.setImageResource(R.drawable.dish1)
    image_view.layoutParams = params3


    val frame_layout2 = FrameLayout(this.requireContext())
    var params1 = FrameLayout.LayoutParams(int_to_dp(343), int_to_dp(104))
    params1.setMargins(int_to_dp(17),int_to_dp(43),0,0)
    frame_layout2.layoutParams = params1
    frame_layout2.setBackgroundResource(R.drawable.layout)

    var text_view1 = TextView(this.requireContext())
    var params2 = FrameLayout.LayoutParams(wc,wc)
    params2.setMargins(int_to_dp(130),int_to_dp(13),0,0)
    text_view1.layoutParams = params2
    text_view1.text = name
    text_view1.textSize = 20f
    text_view1.setTypeface(Typeface.SANS_SERIF, Typeface.BOLD)
    text_view1.setTextColor(getResources().getColor(R.color.slider))

    var text_view2 = TextView(this.requireContext())
    var params4 = FrameLayout.LayoutParams(wc,wc)
    params4.setMargins(int_to_dp(301),int_to_dp(13),0,0)
    text_view2.layoutParams = params4
    text_view2.text = "350грн"
    text_view2.textSize = 11f
    text_view2.setTypeface(Typeface.SANS_SERIF, Typeface.NORMAL)
    text_view2.setTextColor(getResources().getColor(R.color.slider))

    var text_view3 = TextView(this.requireContext())
    var params5 = FrameLayout.LayoutParams(wc,wc)
    params5.setMargins(int_to_dp(128),int_to_dp(46),0,0)
    text_view3.layoutParams = params5
    text_view3.text = "Тунец с тыквой, под соусом табоджи поданый с кусочком лука (200гр)"
    text_view3.textSize = 11f
    text_view3.setTypeface(Typeface.SANS_SERIF, Typeface.NORMAL)
    text_view3.setTextColor(getResources().getColor(R.color.textSilver))

    frame_layout2.addView(text_view1)
    frame_layout2.addView(text_view2)
    frame_layout2.addView(text_view3)

    frame_layout1.addView(frame_layout2)
    frame_layout1.addView(image_view)


    frame_layout.addView(frame_layout1)
    return frame_layout
}
fun add_elements(img:Int,name:String, tag_name:String, like:String) {
    var f1:FrameLayout
    if(tag_name == "Рестораны")
```

```
            f1 = create_frame(img, name, tag_name, "false")
        else
            f1 = create_blydo(img, name, tag_name)
        elemnt.add(f1)
    }
    fun show_restourant(flex: FlexboxLayout, filter_tag:String){
        flex.removeAllViewsInLayout()
        for(i in 0..elemnt.size-1){
            if(elemnt[i].tag == filter_tag)
                flex.addView(elemnt[i])
            else if (filter_tag == "")
                flex.addView(elemnt[i])
        }
    }
    fun reset_button(my_iter:Int){
        for(i in 0..buttons.size-1){
            buttons[i].setBackgroundResource(R.drawable.shadow)
            buttons[i].setTextColor(getResources().getColor(R.color.textDark))
        }
        buttons[my_iter].setBackgroundResource(R.drawable.button_shadow)
        buttons[my_iter].setTextColor(getResources().getColor(R.color.colorWhite))
        buttons[my_iter].startAnimation(animations[my_iter])
    }




    buttons[0].setOnClickListener {
        val my_iter = 0
        reset_button(my_iter)
        show_restourant(view.findViewById(R.id.flex) , "Рестораны")
    }
    buttons[1].setOnClickListener {
        val my_iter2 = 1
        reset_button(my_iter2)
        show_restourant(view.findViewById(R.id.flex) , "Кухни")
    }
    buttons[2].setOnClickListener {
        val my_iter3 = 2
        reset_button(my_iter3)
        show_restourant(view.findViewById(R.id.flex) , "Ингридиенты")
    }




    for(i in 0..5){
        add_elements(R.drawable.ic_dark_tort,"Bartolomeo","Рестораны","false")
        add_elements(R.drawable.kfc_logo,"Украинская","Кухни","false")
        add_elements(R.drawable.chel,"Сосиска","Ингридиенты","false")
    }
    show_restourant(view.findViewById(R.id.flex),"Рестораны")

    return view

  }
}package com.example.myapplication

import android.app.ActionBar
import android.content.Intent
import android.graphics.Color
import android.graphics.Typeface
import android.os.Bundle
import android.util.TypedValue
```

```
import android.view.View
import android.view.animation.Animation
import android.view.animation.AnimationUtils
import android.widget.*
import androidx.appcompat.app.AppCompatActivity
import androidx.constraintlayout.widget.ConstraintLayout
import com.denzcoskun.imageslider.ImageSlider
import com.denzcoskun.imageslider.models.SlideModel
import com.google.android.flexbox.FlexboxLayout
import kotlinx.android.synthetic.main.restoraunt.*


class Restaurant_menu : AppCompatActivity() {
    val elemnt: MutableList<FrameLayout> = ArrayList()
    val element2: MutableList<FrameLayout> = ArrayList()
    val element3: MutableList<FrameLayout> = ArrayList()
   /*  val buttons = arrayOf(
        findViewById<RelativeLayout>(R.id.relativeLay1R),
        findViewById(R.id.relativeLay2R),
        findViewById(R.id.relativeLay3R),
        findViewById(R.id.relativeLay4R)
    )
    val images= arrayOf<ImageView>(
        this.findViewById(R.id.relativeImg1),
        this.findViewById(R.id.relativeImg2),
        this.findViewById(R.id.relativeImg3),
        this.findViewById(R.id.relativeImg4)
    )
    val icons = arrayOf(
        R.drawable.ic_menu_red,
        R.drawable.ic_cold_red,
        R.drawable.ic_hot_red,
        R.drawable.ic_drink_red,
        R.drawable.ic_menu_black,
        R.drawable.ic_cold_black,
        R.drawable.ic_hot_black,
        R.drawable.ic_drink_black
    )
    val texts = arrayOf<TextView>(
        this.findViewById(R.id.relativeBut1),
        this.findViewById(R.id.relativeBut2),
        this.findViewById(R.id.relativeBut3),
        this.findViewById(R.id.relativeBut4)
    )
    val animations: Array<Animation?> = arrayOf(
        AnimationUtils.loadAnimation(baseContext, R.anim.alpha),
        AnimationUtils.loadAnimation(baseContext, R.anim.alpha2),
        AnimationUtils.loadAnimation(baseContext, R.anim.combo2),
        AnimationUtils.loadAnimation(baseContext, R.anim.combo)
    )*/
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.restoraunt)
        val container = findViewById(R.id.PlusOther) as ConstraintLayout
        val imageSlider = findViewById<ImageSlider>(R.id.viewPager3)

        val slideModels: MutableList<SlideModel> = ArrayList()
        slideModels.add(
            SlideModel("https://-tbn0.gstatic.com/images?q=tbn:ANd9GcTeY-x0-ozntxe-
jwblHmlddIc1PqUrdDn6Nw&usqp=CAU")
        )
        slideModels.add(
            SlideModel("https://reston.ua/uploads/img/zavedeniya/original/54212b9.jpg")
        )
```

```
    slideModels.add(
        SlideModel("https://www.dniprohotel.ua/images/m/dnipro-m.jpg")
    )
    imageSlider.setImageList(slideModels, true)
    for(i in 0..5){
        add_elements(R.drawable.dish1,"Dish1","Закуски","false")
        add_elements(R.drawable.dish2,"Dish2","Напитки","false")
        add_elements(R.drawable.dish1,"Dish3","Горячее","false")
        add_elements(R.drawable.dish2,"Dish4","Закуски","false")
    }
    show_restourant(flexrestor,elemnt.size-1)
}

fun int_to_dp(x:Int): Int {
    val value = TypedValue.applyDimension(
        TypedValue.COMPLEX_UNIT_DIP,
        x.toFloat(),
        resources.displayMetrics
    ).toInt()
    return value
}
fun create_blydo(img:Int,name:String, tag_name:String): FrameLayout {
    val mp = LinearLayout.LayoutParams.MATCH_PARENT
    val wc = LinearLayout.LayoutParams.WRAP_CONTENT

    val frame_layout = FrameLayout(baseContext)//Главный контейнер
    frame_layout.layoutParams = FrameLayout.LayoutParams(wc,wc)
    frame_layout.setPadding(0,int_to_dp(11),0,0)
    frame_layout.tag = tag_name

    val frame_layout1 = FrameLayout(baseContext)//Контейнер с картинкой
    frame_layout1.layoutParams = FrameLayout.LayoutParams(wc,wc)
    frame_layout1.setOnClickListener {
        startActivity(Intent(this,Dish::class.java))
    }

    val image_view  = ImageView(baseContext)//Картинка
    var params3 = FrameLayout.LayoutParams(int_to_dp(136), int_to_dp(136))
    params3.setMargins(int_to_dp(-5),0,0,0)
    image_view.setImageResource(img)
    image_view.layoutParams = params3


    val frame_layout2 = FrameLayout(baseContext)//Блюдо
    var params1 = FrameLayout.LayoutParams(int_to_dp(343), int_to_dp(104))
    params1.setMargins(int_to_dp(10),int_to_dp(33),0,int_to_dp(10))
    frame_layout2.layoutParams = params1
    frame_layout2.setBackgroundResource(R.drawable.layout3)

    var text_view3 = TextView(baseContext)//Название
    var params5 = FrameLayout.LayoutParams(wc,wc)
    params5.setMargins(int_to_dp(120),int_to_dp(11),0,0)
    text_view3.layoutParams = params5
    text_view3.text = name
    text_view3.textSize = 18f
    text_view3.setTypeface(Typeface.SANS_SERIF, Typeface.BOLD)
    text_view3.setTextColor(getResources().getColor(R.color.textDark))
    val text_view2 = TextView(baseContext)//Цена
    val params4 = FrameLayout.LayoutParams(wc,wc)
    params4.setMargins(int_to_dp(121),int_to_dp(50),0,0)
    text_view2.layoutParams = params4
    text_view2.text = "Тунец с тыквой, под соусом табоджи поданый с кусочком лука (200гр)"
    text_view2.textSize = 12f
    text_view2.setTypeface(Typeface.SANS_SERIF, Typeface.NORMAL)
```

```
    text_view2.setTextColor(getResources().getColor(R.color.textSilver))
    val frame_layout3 = FrameLayout(baseContext)//Кнопки
    val params2=FrameLayout.LayoutParams(int_to_dp(75),int_to_dp(42))
    params2.setMargins(int_to_dp(259),int_to_dp(30),int_to_dp(13),int_to_dp(20))
    frame_layout3.layoutParams=params2
    frame_layout3.setBackgroundResource(R.drawable.shadow)
    var i=1
    var text_view1 = TextView(baseContext)//Колво
    var params8 = FrameLayout.LayoutParams(wc,int_to_dp(27))
    params8.setMargins(int_to_dp(25),int_to_dp(10),int_to_dp(8),0)
    text_view1.layoutParams = params8
    text_view1.text = ""+i
    text_view1.textSize = 12f
    text_view1.setTypeface(Typeface.SANS_SERIF, Typeface.BOLD)
    text_view1.setTextColor(getResources().getColor(R.color.textDark))
    val imgb1 = ImageButton(baseContext)//Минус
    val params6=FrameLayout.LayoutParams(int_to_dp(11),int_to_dp(10))
    params6.setMargins(int_to_dp(5),int_to_dp(13),int_to_dp(50),int_to_dp(15))

    imgb1.layoutParams=params6
    imgb1.setImageResource(R.drawable.minus)
    imgb1.setOnClickListener {
       if(i>0) {
          i--
          text_view1.text = "" + i
       }
       else text_view1.text=" "

    }
    val imgb2 = ImageButton(baseContext)//Плюс
    val params7=FrameLayout.LayoutParams(int_to_dp(12),int_to_dp(33))
    params7.setMargins(int_to_dp(45),int_to_dp(5),int_to_dp(8),int_to_dp(10))
    imgb2.layoutParams=params7
    imgb2.setImageResource(R.drawable.plus)
    imgb2.setOnClickListener {
       i++
       text_view1.text=""+i
    }
    frame_layout3.addView(imgb1)
    frame_layout3.addView(text_view1)
    frame_layout3.addView(imgb2)
    frame_layout2.addView(frame_layout3)
    frame_layout2.addView(text_view3)
    frame_layout2.addView(text_view2)
    frame_layout1.addView(frame_layout2)
    frame_layout1.addView(image_view)


    frame_layout.addView(frame_layout1)
    return frame_layout

 }
 fun add_elements(img:Int,name:String, tag_name:String, like:String) {
    var f1: FrameLayout
    f1 = create_blydo(img, name, tag_name)
    elemnt.add(f1)
 }
 fun show_restourant(flex: FlexboxLayout, length:Int){
    if(length<=elemnt.size)
       for(i in 0..length){
          flex.addView(elemnt[i])
       }
 }
/*fun reset_buttons(myIter:Int){
```

```
        val anim: Animation? = AnimationUtils.loadAnimation(this, R.anim.alpha);
        val animtext: Animation? = AnimationUtils.loadAnimation(this, R.anim.combo);
        val size = 3
        val sizeInPX1 =
TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP,64f,resources.displayMetrics).toInt()
        val sizeInPX2 =
TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP,150f,resources.displayMetrics).toInt()

        for(i in 0..size){
            if(i != myIter){
                buttons[i].setBackgroundResource(R.drawable.shadow)
                images[i].setImageResource(icons[i+4])
                texts[i].text = ""
                buttons[i].layoutParams.width=sizeInPX1
            }
        }

        val textViewanim = findViewById<TextView>(R.id.relativeBut1)
        buttons[myIter].setBackgroundResource(R.drawable.button_shadow)
        images[myIter].setImageResource(icons[myIter])
        buttons[myIter].startAnimation(anim)
        textViewanim.startAnimation(animtext)
        buttons[myIter].layoutParams.width=sizeInPX2
    }*/
    fun plus(view: View){
        val tt = findViewById(R.id.iterator)as TextView;
        tt.text = (tt.text.toString().toInt() + 1).toString()
    }
    fun minus(view: View){
        val tt = findViewById(R.id.iterator)as TextView;
        if(tt.text.toString().toInt()>=1)
            tt.text = (tt.text.toString().toInt() - 1).toString()
    }
    fun openDish(view: View){
        val intent= Intent(this,Dish::class.java)
        startActivity(intent)
    }
    fun Ordered(view: View){
        val intent= Intent(this,Order::class.java)
        startActivity(intent)
    }
    fun openRestaurantMenu(view: View){
        val intent= Intent(this,Glavnaya1::class.java)
        startActivity(intent)
        finish()
    }
    fun Podrobnee(view: View){
        val but = findViewById(R.id.butText) as TextView
        val ico = findViewById(R.id.strelka) as ImageView
        val container = findViewById(R.id.PlusOther) as ConstraintLayout
        val kuhni = findViewById(R.id.podrobneeLay) as ConstraintLayout

        if(but.text == "Подробнее") {
            container.layoutParams.height = ConstraintLayout.LayoutParams.WRAP_CONTENT
            but.text = "Скрыть"
            ico.setImageResource(R.drawable.ic_arrow_up)
            kuhni.layoutParams.height = ConstraintLayout.LayoutParams.WRAP_CONTENT
        }
        else {
            container.layoutParams.height = int_to_dp(50);
            but.text = "Подробнее"
            ico.setImageResource(R.drawable.ic_arrow_down)
        }
    }
```

```
/*
   fun all_menu(view: View){
       val myIter = 0
       reset_button(myIter)
       texts[myIter].text = "Все меню"
    }
    fun cold_platters(view: View){
       val myIter = 1
       reset_button(myIter)
       texts[myIter].text = "Закуски"
    }
    fun soups(view: View){
       val myIter = 2
       reset_button(myIter)
       texts[myIter].text = "Горячее"
    }
    fun beverages(view: View){
       val myIter = 3
       reset_button(myIter)
       texts[myIter].text = "Напитки"
    }*/
    fun likeClick(view: View){
       val b = view as ImageButton
       if(b.tag == "false"){
          b.setImageResource(R.drawable.ic_heart_black)
          b.tag = "true"
       }else{
          b.setImageResource(R.drawable.ic_heart_like_red)
          b.tag = "false"
       }
    }

}package com.example.myapplication


import android.content.Intent
import android.graphics.Bitmap
import android.graphics.Color
import android.view.Gravity
import android.graphics.Typeface
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import android.util.TypedValue
import android.view.Gravity.apply
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.view.animation.Animation
import android.view.animation.AnimationUtils
import android.widget.*
import androidx.annotation.WorkerThread
import androidx.core.view.GravityCompat.apply
import androidx.fragment.app.Fragment
import com.denzcoskun.imageslider.ImageSlider
import com.denzcoskun.imageslider.models.SlideModel
import com.google.android.flexbox.FlexboxLayout
import com.google.android.gms.vision.Frame
import kotlinx.android.synthetic.main.activity_pinned_buttons.*
import okhttp3.*
import org.json.JSONArray
import java.io.IOException
import java.io.InputStreamReader
import java.net.HttpURLConnection
```

```
import java.net.URL
import java.time.format.TextStyle

private const val ENDPOINT ="http://10.0.2.2:3000/api"
private const val RESTORAUNT="/restaurant"
private const val NAME="name"
class PinnedButtons : Fragment(R.layout.activity_pinned_buttons) {
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {


        val view: View = inflater.inflate(R.layout.activity_pinned_buttons, container, false)
        val elemnt: MutableList<FrameLayout> = ArrayList()
        val relativeLay1 = view.findViewById(R.id.relativeLay1) as RelativeLayout
        val relativeLay2 = view.findViewById(R.id.relativeLay2) as RelativeLayout
        val relativeLay3 = view.findViewById(R.id.relativeLay3) as RelativeLayout
        val buttons: Array<RelativeLayout> = arrayOf(
            view.findViewById(R.id.relativeLay1) as RelativeLayout,
            view.findViewById(R.id.relativeLay2) as RelativeLayout,
            view.findViewById(R.id.relativeLay3) as RelativeLayout
        )
        val images: Array<ImageView> = arrayOf(
            view.findViewById(R.id.relativeImg1) as ImageView,
            view.findViewById(R.id.relativeImg2) as ImageView,
            view.findViewById(R.id.relativeImg3) as ImageView
        )
        var icons = arrayOf(
            R.drawable.ic_hand_like_white,
            R.drawable.ic_new_white,
            R.drawable.ic_sale_white,
            R.drawable.ic_hand_like_black,
            R.drawable.ic_new_black,
            R.drawable.ic_sale_black
        )
        val texts: Array<TextView> = arrayOf(
            view.findViewById(R.id.relativeBut1) as TextView,
            view.findViewById(R.id.relativeBut2) as TextView,
            view.findViewById(R.id.relativeBut3) as TextView
        )
        val animations: Array<Animation?> = arrayOf(
            AnimationUtils.loadAnimation(requireContext(), R.anim.alpha),
            AnimationUtils.loadAnimation(requireContext(), R.anim.alpha2),
            AnimationUtils.loadAnimation(requireContext(), R.anim.combo2),
            AnimationUtils.loadAnimation(requireContext(), R.anim.combo)
        )



        fun int_to_dp(x:Int): Int {
            val value = TypedValue.applyDimension(
                TypedValue.COMPLEX_UNIT_DIP,
                x.toFloat(),
                resources.displayMetrics
            ).toInt()
            return value
        }

        fun create_frame(img:Int,name:String, tag_name:String, like:String): FrameLayout {
            var params1 = FrameLayout.LayoutParams(int_to_dp(343), int_to_dp(190))
            val mp = LinearLayout.LayoutParams.MATCH_PARENT
            val wc = LinearLayout.LayoutParams.WRAP_CONTENT
```

```
val f = FrameLayout(this.requireContext())//Главный контейнер
f.layoutParams = FrameLayout.LayoutParams(wc,wc)
f.tag = tag_name

val fl = FrameLayout(this.requireContext())//Главный контейнер
params1.setMargins(0,int_to_dp(16),0,int_to_dp(0))
fl.layoutParams = params1
fl.setBackgroundResource(R.drawable.shadow)
fl.setPadding(10,0,10,20)


val iv = ImageView(this.requireContext())//Картинка ресторана
iv.setImageResource(img)
iv.layoutParams = LinearLayout.LayoutParams(mp,wc)
iv.scaleType = ImageView.ScaleType.CENTER_CROP
iv.setOnClickListener {
   activity?.let {
      val intent = Intent(it, Restaurant_menu::class.java)
      it.startActivity(intent)
   }
}


val fl2=FrameLayout(this.requireContext())//нижний контейнер
val params2 = LinearLayout.LayoutParams(0,0)
params2.width = mp
params2.height = int_to_dp(60)
params2.setMargins(0,int_to_dp(130),0,0)
fl2.layoutParams = params2
fl2.setBackgroundColor(Color.WHITE)
fl2.setBackgroundResource(R.drawable.bottom_block)


val tv = TextView(this.requireContext())//Название ресторана
val params3 = LinearLayout.LayoutParams(0,0,1000f)
tv.text = name
params3.width = wc
params3.height = wc
params3.setMargins(int_to_dp(21),int_to_dp(8),0,0)
tv.layoutParams = params3
tv.textSize = 20f
tv.setTypeface(Typeface.SANS_SERIF,Typeface.BOLD)
tv.setTextColor(getResources().getColor(R.color.textDark))
tv.setPadding(21, 9, 0, 0)


val img=ImageView(this.requireContext())//Звезды ресторана
val params4 = LinearLayout.LayoutParams(int_to_dp(58),int_to_dp(9))
params4.setMargins(int_to_dp(24),int_to_dp(40),0,0)
img.setImageResource(R.drawable.stars)
img.layoutParams= params4


val heart=ImageButton(this.requireContext())//лайк ресторана
val params5 = LinearLayout.LayoutParams(wc,wc)
params5.setMargins(int_to_dp(290),int_to_dp(7),0,0)
heart.setImageResource(R.drawable.ic_heart_black)
heart.layoutParams= params5
heart.setBackgroundColor(Color.TRANSPARENT)
heart.tag = like
if (heart.tag == "false") {
   heart.setImageResource(R.drawable.ic_heart_black)
   heart.tag = "true"
} else {
```

```
            heart.setImageResource(R.drawable.ic_heart_like_red)
            heart.tag = "false"
        }
        heart.setOnClickListener {
            if (heart.tag == "false") {
                heart.setImageResource(R.drawable.ic_heart_black)
                heart.tag = "true"
            } else {
                heart.setImageResource(R.drawable.ic_heart_like_red)
                heart.tag = "false"
            }
        }


        val fl3 = FrameLayout(this.requireContext())//контейнер для текста с временем
        val params6 = LinearLayout.LayoutParams(wc,wc)
        params6.setMargins(int_to_dp(24),int_to_dp(25),0,0)
        fl3.layoutParams= params6
        fl3.setPadding(100,25,0,0)


        val txt =TextView(this.requireContext()) //Время работы ресторана
        val params7 = LinearLayout.LayoutParams(wc,wc)
        txt.text = "08:00 - 22:00"
        txt.layoutParams = params7
        txt.textSize = 11f
        txt.setTypeface(Typeface.SANS_SERIF,Typeface.NORMAL)
        txt.setTextColor(getResources().getColor(R.color.textSilver))
        txt.setPadding(21, 9, 0, 0)


        fl3.addView(txt)
        fl2.addView(tv)
        fl2.addView(img)
        fl2.addView(fl3)
        fl2.addView(heart)
        fl.addView(iv)
        fl.addView(fl2)
        f.addView(fl)
        return f
    }

    fun add_elements(img:Int,name:String, tag_name:String, like:String) {
        val f1 = create_frame(img, name, tag_name, like)
        elemnt.add(f1)
    }

    fun show_restourant(flex:FlexboxLayout, filter_tag:String){
        flex.removeAllViewsInLayout()
        for(i in 0..elemnt.size-1){
            if(elemnt[i].tag == filter_tag)
                flex.addView(elemnt[i])
            else if (filter_tag == "")
                flex.addView(elemnt[i])
        }
    }

    fun reset_button(myIter:Int){
        for (i in 0..2) {
            if (i != myIter) {
                buttons[i].setBackgroundResource(R.drawable.shadow)
                images[i].setImageResource(icons[i + 3])
                texts[i].text = ""
                buttons[i].layoutParams.width = int_to_dp(84)
```

```
            }
        }
        buttons[myIter].setBackgroundResource(R.drawable.button_shadow)
        images[myIter].setImageResource(icons[myIter])
        buttons[myIter].startAnimation(animations[myIter])
        texts[myIter].startAnimation(animations[3])
        buttons[myIter].layoutParams.width = int_to_dp(150)
    }




    relativeLay1.setOnClickListener {
        val myIter = 0 as Int;
        texts[myIter].text = "Популярное"
        reset_button(myIter)
        texts[myIter].text = "Популярное"
        buttons[myIter].startAnimation(animations[myIter])
        texts[myIter].startAnimation(animations[3])
        buttons[myIter].layoutParams.width = int_to_dp(170)
        Thread{
            getrest()
        }.start()
        show_restourant(view.findViewById(R.id.flex),"")
        //Toast.makeText(this, "Its toast!", Toast.LENGTH_SHORT).show()
    }


    relativeLay2.setOnClickListener {
        val myIter = 1 as Int;
        texts[myIter].text = "Новинки"
        reset_button(myIter)
        for (i in 0..2) {
            if (i != myIter) {
                buttons[i].setBackgroundResource(R.drawable.shadow)
                images[i].setImageResource(icons[i + 3])
                texts[i].text = ""
                buttons[i].layoutParams.width = int_to_dp(84)
            }
        }

        val textViewanim = view.findViewById<TextView>(R.id.relativeBut2)
        buttons[myIter].setBackgroundResource(R.drawable.button_shadow)
        images[myIter].setImageResource(icons[myIter])
        texts[myIter].text = "Новинки"
        buttons[myIter].startAnimation(animations[myIter])
        texts[myIter].startAnimation(animations[3])
        buttons[myIter].layoutParams.width = int_to_dp(170)
        show_restourant(view.findViewById(R.id.flex),"Новинки")
        //Toast.makeText(view.context,                    "${(R.drawable.ic_dark_tort)::class.simpleName}",
Toast.LENGTH_SHORT).show()
    }



    relativeLay3.setOnClickListener {
        val myIter = 2 as Int;
        texts[myIter].text = "Акция"
        reset_button(myIter)

        for (i in 0..2) {
            if (i != myIter) {
                buttons[i].setBackgroundResource(R.drawable.shadow)
```

```
                images[i].setImageResource(icons[i + 3])
                texts[i].text = ""
                buttons[i].layoutParams.width = int_to_dp(84)
            }
        }

        val textViewanim = view.findViewById<TextView>(R.id.relativeBut3)
        buttons[myIter].setBackgroundResource(R.drawable.button_shadow)
        images[myIter].setImageResource(icons[myIter])
        texts[myIter].text = "Акция"
        buttons[myIter].startAnimation(animations[myIter])
        texts[myIter].startAnimation(animations[3])
        buttons[myIter].layoutParams.width = int_to_dp(170)
        show_restourant(view.findViewById(R.id.flex),"Акция")
        //Toast.makeText(view.context,                    "${(R.drawable.ic_dark_tort)::class.qualifiedName}",
Toast.LENGTH_SHORT).show()
    }




    //ПЕРВОЕ ЗАПОЛНЕНИЕ//
    for(i in 0..5){
        add_elements(R.drawable.ic_dark_tort,"Bartolomeo","Популярное","false")
        add_elements(R.drawable.kfc_logo,"KFC","Акция","false")
        add_elements(R.drawable.chel,"Chelentano","Новинки","false")
    }
    show_restourant(view.findViewById(R.id.flex),"")
    val imageSlider = view.findViewById<ImageSlider>(R.id.imageView8)

    val slideModels: MutableList<SlideModel> = ArrayList()
    slideModels.add(
        SlideModel("error")
    )
    slideModels.add(
        SlideModel("https://smachno.ua/wp-content/uploads/old_uploads/img/pg/72/11/1.jpg")
    )
    slideModels.add(
        SlideModel("https://i.obozrevatel.com/food/recipemain/2019/1/9/126.webp?size=600x400")
    )
    imageSlider.setImageList(slideModels, true)

   //  fetchJson()
    return view
}

@WorkerThread
private fun getrest() {
    val httpURLConnection= URL(ENDPOINT+RESTORAUNT).openConnection() as HttpURLConnection
    httpURLConnection.apply{
        connectTimeout=10000
        requestMethod="GET"
        doInput=true
    }
    if(httpURLConnection.responseCode!=HttpURLConnection.HTTP_OK){
        return
    }
    val streamReader=InputStreamReader(httpURLConnection.inputStream)
    var text:String=""
    streamReader.use {
        text=it.readText()
    }
    val restaurants= mutableListOf<String>()
```

```
    val json=JSONArray(text)
    for(i in 0 until json.length()){
        val jsonrest=json.getJSONObject(i)
        val name=jsonrest.getString(NAME)
        restaurants.add(name)
    }
    httpURLConnection.disconnect()
    Handler(Looper.getMainLooper()).post {
        for(i in 0..10) {
            add_elements(
                R.drawable.dish1,
                restaurants.reduce { acc, s -> "$acc\n$s" },
                "Популярное",
                "false"
            )
        }
        textViewaddjson.text=restaurants.reduce { acc, s -> "$acc\n$s" }
    }
}
val elemnt: MutableList<FrameLayout> = ArrayList()

fun int_to_dp(x:Int): Int {
    val value = TypedValue.applyDimension(
        TypedValue.COMPLEX_UNIT_DIP,
        x.toFloat(),
        resources.displayMetrics
    ).toInt()
    return value
}
fun create_frame(img:Int,name:String, tag_name:String, like:String): FrameLayout {
    var params1 = FrameLayout.LayoutParams(int_to_dp(343), int_to_dp(190))
    val mp = LinearLayout.LayoutParams.MATCH_PARENT
    val wc = LinearLayout.LayoutParams.WRAP_CONTENT
    val f = FrameLayout(this.requireContext())//Главный контейнер
    f.layoutParams = FrameLayout.LayoutParams(wc,wc)
    f.tag = tag_name

    val fl = FrameLayout(this.requireContext())//Главный контейнер
    params1.setMargins(0,int_to_dp(16),0,int_to_dp(0))
    fl.layoutParams = params1
    fl.setBackgroundResource(R.drawable.shadow)
    fl.setPadding(10,0,10,20)


    val iv = ImageView(this.requireContext())//Картинка ресторана
    iv.setImageResource(img)
    iv.layoutParams = LinearLayout.LayoutParams(mp,wc)
    iv.scaleType = ImageView.ScaleType.CENTER_CROP
    iv.setOnClickListener {
        activity?.let {
            val intent = Intent(it, Restaurant_menu::class.java)
            it.startActivity(intent)
        }
    }


    val fl2=FrameLayout(this.requireContext())//нижний контейнер
    val params2 = LinearLayout.LayoutParams(0,0)
    params2.width = mp
    params2.height = int_to_dp(60)
    params2.setMargins(0,int_to_dp(130),0,0)
    fl2.layoutParams = params2
    fl2.setBackgroundColor(Color.WHITE)
    fl2.setBackgroundResource(R.drawable.bottom_block)
```

```
val tv = TextView(this.requireContext())//Название ресторана
val params3 = LinearLayout.LayoutParams(0,0,1000f)
tv.text = name
params3.width = wc
params3.height = wc
params3.setMargins(int_to_dp(21),int_to_dp(8),0,0)
tv.layoutParams = params3
tv.textSize = 20f
tv.setTypeface(Typeface.SANS_SERIF,Typeface.BOLD)
tv.setTextColor(getResources().getColor(R.color.textDark))
tv.setPadding(21, 9, 0, 0)


val img=ImageView(this.requireContext())//Звезды ресторана
val params4 = LinearLayout.LayoutParams(int_to_dp(58),int_to_dp(9))
params4.setMargins(int_to_dp(24),int_to_dp(40),0,0)
img.setImageResource(R.drawable.stars)
img.layoutParams= params4


val heart=ImageButton(this.requireContext())//лайк ресторана
val params5 = LinearLayout.LayoutParams(wc,wc)
params5.setMargins(int_to_dp(290),int_to_dp(7),0,0)
heart.setImageResource(R.drawable.ic_heart_black)
heart.layoutParams= params5
heart.setBackgroundColor(Color.TRANSPARENT)
heart.tag = like
if (heart.tag == "false") {
   heart.setImageResource(R.drawable.ic_heart_black)
   heart.tag = "true"
} else {
   heart.setImageResource(R.drawable.ic_heart_like_red)
   heart.tag = "false"
}
heart.setOnClickListener {
   if (heart.tag == "false") {
      heart.setImageResource(R.drawable.ic_heart_black)
      heart.tag = "true"
   } else {
      heart.setImageResource(R.drawable.ic_heart_like_red)
      heart.tag = "false"
   }
}


val fl3 = FrameLayout(this.requireContext())//контейнер для текста с временем
val params6 = LinearLayout.LayoutParams(wc,wc)
params6.setMargins(int_to_dp(24),int_to_dp(25),0,0)
fl3.layoutParams= params6
fl3.setPadding(100,25,0,0)


val txt =TextView(this.requireContext()) //Время работы ресторана
val params7 = LinearLayout.LayoutParams(wc,wc)
txt.text = "08:00 - 22:00"
txt.layoutParams = params7
txt.textSize = 11f
txt.setTypeface(Typeface.SANS_SERIF,Typeface.NORMAL)
txt.setTextColor(getResources().getColor(R.color.textSilver))
txt.setPadding(21, 9, 0, 0)
```

```
        fl3.addView(txt)
        fl2.addView(tv)
        fl2.addView(img)
        fl2.addView(fl3)
        fl2.addView(heart)
        fl.addView(iv)
        fl.addView(fl2)
        f.addView(fl)
        return f
    }

    fun add_elements(img:Int,name:String, tag_name:String, like:String) {
        val f1 = create_frame(img, name, tag_name, like)
        elemnt.add(f1)
    }

    fun show_restourant(flex:FlexboxLayout, filter_tag:String){
        flex.removeAllViewsInLayout()
        for(i in 0..elemnt.size-1){
            if(elemnt[i].tag == filter_tag)
                flex.addView(elemnt[i])
            else if (filter_tag == "")
                flex.addView(elemnt[i])
        }
    }
    /*  fun fetchJson(){
  val url ="http://localhost:3000/api/profile/0"
         val request = Request.Builder().url(url).build()
         val client =OkHttpClient()
         client.newCall(request).enqueue(object: Callback{
             override fun onResponse(call: Call, response: Response) {
                 val body = response?.body?.string()
                 println(body)
             }
             override fun onFailure(call: Call, e: IOException) {
               println("Fail")
             }
         })
     }*/
}package com.example.myapplication

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.*
import androidx.fragment.app.Fragment

class Person_menu : Fragment(R.layout.activity_person_menu) {
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {

        val view: View = inflater.inflate(R.layout.activity_person_menu, container, false)

        val notification_sound = view.findViewById(R.id.notification_sound) as ImageView

        val rel1 = view.findViewById(R.id.rel1) as RelativeLayout
        val rel2 = view.findViewById(R.id.rel2) as RelativeLayout
        val rel3 = view.findViewById(R.id.rel3) as RelativeLayout
```

```
val rel4 = view.findViewById(R.id.rel4) as RelativeLayout
val rel5 = view.findViewById(R.id.rel5) as RelativeLayout

notification_sound.setOnClickListener {
   Toast.makeText(requireContext(), "Уведомления выключены!", Toast.LENGTH_SHORT).show()
   val b = it as ImageView
   if (b.tag == "false") {
      b.setImageResource(R.drawable.bell_1)
      Toast.makeText(requireContext(), "Уведомления включены!", Toast.LENGTH_SHORT).show()
      b.tag = "true"
   } else {
      b.setImageResource(R.drawable.bell_1)
      Toast.makeText(requireContext(), "Уведомления выключены!", Toast.LENGTH_SHORT).show()
      b.tag = "false"
   }
}

fun outLogin(view: View) {
   activity?.let {
      val intent = Intent(it, Sign_In::class.java)
      it.startActivity(intent)
   }
}


fun smena(it:View){

}


rel1.setOnClickListener {

   val imageView: ImageView = view.findViewById(R.id.imageView18)
   imageView.setImageResource(R.drawable.gps)
   val imageView2: ImageView = view.findViewById(R.id.imageView20)
   imageView2.setImageResource(R.drawable.ic_sale)
   val imageView3: ImageView = view.findViewById(R.id.imageView22)
   imageView3.setImageResource(R.drawable.settings)
   val imageView4: ImageView = view.findViewById(R.id.imageView24)
   imageView4.setImageResource(R.drawable.email_b)
   val imageView5: ImageView = view.findViewById(R.id.imageView26)
   imageView5.setImageResource(R.drawable.faq)

}

rel2.setOnClickListener {
   val imageView: ImageView = view.findViewById(R.id.imageView18)
   imageView.setImageResource(R.drawable.gps_b)
   val imageView2: ImageView = view.findViewById(R.id.imageView20)
   imageView2.setImageResource(R.drawable.sales_r)
   val imageView3: ImageView = view.findViewById(R.id.imageView22)
   imageView3.setImageResource(R.drawable.settings)
   val imageView4: ImageView = view.findViewById(R.id.imageView24)
   imageView4.setImageResource(R.drawable.email_b)
   val imageView5: ImageView = view.findViewById(R.id.imageView26)
   imageView5.setImageResource(R.drawable.faq)

}

rel3.setOnClickListener {
   val imageView: ImageView = view.findViewById(R.id.imageView18)
   imageView.setImageResource(R.drawable.gps_b)
   val imageView2: ImageView = view.findViewById(R.id.imageView20)
   imageView2.setImageResource(R.drawable.ic_sale)
```

```
            val imageView3: ImageView = view.findViewById(R.id.imageView22)
            imageView3.setImageResource(R.drawable.settings_r)
            val imageView4: ImageView = view.findViewById(R.id.imageView24)
            imageView4.setImageResource(R.drawable.email_b)
            val imageView5: ImageView = view.findViewById(R.id.imageView26)
            imageView5.setImageResource(R.drawable.faq)
        }

        rel4.setOnClickListener {
            val imageView: ImageView = view.findViewById(R.id.imageView18)
            imageView.setImageResource(R.drawable.gps_b)
            val imageView2: ImageView = view.findViewById(R.id.imageView20)
            imageView2.setImageResource(R.drawable.ic_sale)
            val imageView3: ImageView = view.findViewById(R.id.imageView22)
            imageView3.setImageResource(R.drawable.settings)
            val imageView4: ImageView = view.findViewById(R.id.imageView24)
            imageView4.setImageResource(R.drawable.email_r)
            val imageView5: ImageView = view.findViewById(R.id.imageView26)
            imageView5.setImageResource(R.drawable.faq)
        }

        rel5.setOnClickListener {
            val imageView: ImageView = view.findViewById(R.id.imageView18)
            imageView.setImageResource(R.drawable.gps_b)
            val imageView2: ImageView = view.findViewById(R.id.imageView20)
            imageView2.setImageResource(R.drawable.ic_sale)
            val imageView3: ImageView = view.findViewById(R.id.imageView22)
            imageView3.setImageResource(R.drawable.settings)
            val imageView4: ImageView = view.findViewById(R.id.imageView24)
            imageView4.setImageResource(R.drawable.email_b)
            val imageView5: ImageView = view.findViewById(R.id.imageView26)
            imageView5.setImageResource(R.drawable.faq_r)

        }

        fun but(view: View) {

        }

        return view

    }
}package com.example.myapplication

import android.animation.ObjectAnimator
import android.content.Intent
import android.os.Bundle
import android.util.TypedValue
import android.view.View
import android.view.animation.Animation
import android.view.animation.AnimationUtils
import android.widget.ImageButton
import android.widget.ImageView
import android.widget.RelativeLayout
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import androidx.fragment.app.Fragment
import kotlinx.android.synthetic.main.activity_glavnaya1.*
import kotlinx.android.synthetic.main.activity_pinned_buttons.*


class Glavnaya1 : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
```

```
        setContentView(R.layout.activity_glavnaya1)
        val basketmenu=Basket_menu()
        val findmenu=Find_menu()
        val personmenu=Person_menu()
        val savedmenu=Saved_menu()
        val pinnedbuttons=PinnedButtons()
        setFragment(pinnedbuttons)
        navigation.setOnNavigationItemSelectedListener {
            when(it.itemId){
                R.id.menu->setFragment(pinnedbuttons)
                R.id.orders->setFragment(basketmenu)
                R.id.find->setFragment(findmenu)
                R.id.person->setFragment(personmenu)
                R.id.saved->setFragment(savedmenu)
            }
            true
        }
    }
    private fun setFragment(fragment: Fragment)=supportFragmentManager.beginTransaction().apply {
        replace(R.id.flFragment,fragment)
        commit()
    }

}package com.example.myapplication

import android.content.Context
import android.os.Parcel
import android.os.Parcelable
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.viewpager.widget.PagerAdapter

class MyAdapter(var layouts:IntArray, var context: Context ): PagerAdapter() {
    private lateinit var inflater: LayoutInflater
    override fun isViewFromObject(view: View, `object`: Any): Boolean {
      return view==`object`
    }

    override fun getCount(): Int {
   return layouts.size
    }

    override fun instantiateItem(container: ViewGroup, position: Int): Any {
        inflater=context.getSystemService(Context.LAYOUT_INFLATER_SERVICE) as LayoutInflater
        val v= inflater.inflate(layouts[position],container,false)
        container.addView(v)
        return  v
    }

    override fun destroyItem(container: ViewGroup, position: Int, `object`: Any) {
        val v=`object` as View
        container.removeView(v)
    }
}package com.example.myapplication

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View

class Glavnaya : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
```

```
        setContentView(R.layout.activity_glavnaya)
    }

    fun onClick2(view: View){
        val intent= Intent(this,Sign_In::class.java)
        startActivity(intent)
    }

    fun secondClick(view: View){
        val intent= Intent(this,Regestration::class.java)
        startActivity(intent)
    }
}package com.example.myapplication

import android.content.Intent
import android.graphics.Typeface
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.TypedValue
import android.view.View
import android.widget.*
import com.google.android.flexbox.FlexboxLayout

class Order : AppCompatActivity() {
    val elemnt: MutableList<FrameLayout> = ArrayList()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.order)
        for(i in 0..5){
            add_elements(R.drawable.dish1,"Блюдо"+i,"Кухни","false")
        }

        val flex = findViewById<FlexboxLayout>(R.id.flexorder)
        show_restourant(flex, elemnt.size-1)
    }

    fun int_to_dp(x:Int): Int {
        val value = TypedValue.applyDimension(
            TypedValue.COMPLEX_UNIT_DIP,
            x.toFloat(),
            resources.displayMetrics
        ).toInt()
        return value
    }
    fun create_blydo(img:Int,name:String, tag_name:String): FrameLayout {
        val mp = LinearLayout.LayoutParams.MATCH_PARENT
        val wc = LinearLayout.LayoutParams.WRAP_CONTENT

        val frame_layout = FrameLayout(baseContext)//Главный контейнер
        frame_layout.layoutParams = FrameLayout.LayoutParams(wc,wc)
        frame_layout.setPadding(0,int_to_dp(11),0,0)
        frame_layout.tag = tag_name

        val frame_layout1 = FrameLayout(baseContext)//Контейнер с картинкой
        frame_layout1.layoutParams = FrameLayout.LayoutParams(wc,wc)


        val image_view  = ImageView(baseContext)//Картинка
        var params3 = FrameLayout.LayoutParams(int_to_dp(136), int_to_dp(136))
        params3.setMargins(int_to_dp(-5),0,0,0)
        image_view.setImageResource(img)
        image_view.layoutParams = params3
```

```
val frame_layout2 = FrameLayout(baseContext)//Блюдо
var params1 = FrameLayout.LayoutParams(int_to_dp(343), int_to_dp(104))
params1.setMargins(int_to_dp(10),int_to_dp(43),0,int_to_dp(10))
frame_layout2.layoutParams = params1
frame_layout2.setBackgroundResource(R.drawable.layout3)

var text_view3 = TextView(baseContext)//Название
var params5 = FrameLayout.LayoutParams(wc,wc)
params5.setMargins(int_to_dp(120),int_to_dp(11),0,0)
text_view3.layoutParams = params5
text_view3.text = name
text_view3.textSize = 18f
text_view3.setTypeface(Typeface.SANS_SERIF, Typeface.BOLD)
text_view3.setTextColor(getResources().getColor(R.color.textDark))
val text_view2 = TextView(baseContext)//Цена
val params4 = FrameLayout.LayoutParams(wc,wc)
params4.setMargins(int_to_dp(121),int_to_dp(50),0,0)
text_view2.layoutParams = params4
text_view2.text = "350грн"
text_view2.textSize = 12f
text_view2.setTypeface(Typeface.SANS_SERIF, Typeface.NORMAL)
text_view2.setTextColor(getResources().getColor(R.color.textSilver))
val frame_layout3 = FrameLayout(baseContext)//Кнопки
val params2=FrameLayout.LayoutParams(int_to_dp(75),int_to_dp(42))
params2.setMargins(int_to_dp(259),int_to_dp(30),int_to_dp(13),int_to_dp(20))
frame_layout3.layoutParams=params2
frame_layout3.setBackgroundResource(R.drawable.shadow)
var i=1
var text_view1 = TextView(baseContext)//Колво
var params8 = FrameLayout.LayoutParams(wc,int_to_dp(27))
params8.setMargins(int_to_dp(25),int_to_dp(10),int_to_dp(8),0)
text_view1.layoutParams = params8
text_view1.text = ""+i
text_view1.textSize = 12f
text_view1.setTypeface(Typeface.SANS_SERIF, Typeface.BOLD)
text_view1.setTextColor(getResources().getColor(R.color.textDark))
val imgb1 = ImageButton(baseContext)//Минус
val params6=FrameLayout.LayoutParams(int_to_dp(11),int_to_dp(10))
params6.setMargins(int_to_dp(5),int_to_dp(13),int_to_dp(50),int_to_dp(15))

imgb1.layoutParams=params6
imgb1.setImageResource(R.drawable.minus)
imgb1.setOnClickListener {
   if(i>0) {
      i--
      text_view1.text = "" + i
   }
   else text_view1.text=" "

}
val imgb2 = ImageButton(baseContext)//Плюс
val params7=FrameLayout.LayoutParams(int_to_dp(12),int_to_dp(33))
params7.setMargins(int_to_dp(45),int_to_dp(5),int_to_dp(8),int_to_dp(10))
imgb2.layoutParams=params7
imgb2.setImageResource(R.drawable.plus)
imgb2.setOnClickListener {
   i++
   text_view1.text=""+i
}
frame_layout3.addView(imgb1)
frame_layout3.addView(text_view1)
frame_layout3.addView(imgb2)
frame_layout2.addView(frame_layout3)
frame_layout2.addView(text_view3)
```

```
        frame_layout2.addView(text_view2)
        frame_layout1.addView(frame_layout2)
        frame_layout1.addView(image_view)


        frame_layout.addView(frame_layout1)
        return frame_layout
    }
    fun add_elements(img:Int,name:String, tag_name:String, like:String) {
        var f1: FrameLayout
        f1 = create_blydo(img, name, tag_name)
        elemnt.add(f1)
    }
    fun show_restourant(flex: FlexboxLayout, length:Int){
        if(length<=elemnt.size)
            for(i in 0..length){
                flex.addView(elemnt[i])
            }
    }



    fun back(view: View){
        val intent= Intent(this,Restaurant_menu::class.java)
        startActivity(intent)
        finish()
    }
    fun confirm(view: View){
        val intent= Intent(this,Waiting::class.java)
        startActivity(intent)
    }

    fun showMore(view: View){
        val More = view.findViewById(R.id.More) as RelativeLayout
        val flex = view.findViewById(R.id.flexorder) as FlexboxLayout
        var length = 2
        if(More.tag == "true"){
            length = elemnt.size-1
            More.tag = "false"
        }else{
            More.tag = "tag"
        }
        show_restourant(flex, length)
    }
}package com.example.myapplication

import androidx.test.platform.app.InstrumentationRegistry
import androidx.test.ext.junit.runners.AndroidJUnit4

import org.junit.Test
import org.junit.runner.RunWith

import org.junit.Assert.*

/**
 * Instrumented test, which will execute on an Android device.
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
@RunWith(AndroidJUnit4::class)
class ExampleInstrumentedTest {
    @Test
    fun useAppContext() {
```

```
        // Context of the app under test.
        val appContext = InstrumentationRegistry.getInstrumentation().targetContext
        assertEquals("com.example.myapplication", appContext.packageName)
    }
}package com.example.myapplication

import org.junit.Test

import org.junit.Assert.*

/**
 * Example local unit test, which will execute on the development machine (host).
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
class ExampleUnitTest {
    @Test
    fun addition_isCorrect() {
        assertEquals(4, 2 + 2)
    }
}package com.example.myapplication

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.Toast

class Regestration : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_regestration)
    }

    fun BackReg(view: View){
        val intent= Intent(this,Glavnaya::class.java)
        startActivity(intent)
        finish()
    }
    fun openMain(view: View){
        val randomIntent = Intent(this, Glavnaya1::class.java)
        startActivity(randomIntent)
    }
    fun myFun(view:View){
        Toast.makeText(this, "А я причем?))))", Toast.LENGTH_SHORT).show()
    }
    fun myFun1(view:View){
        Toast.makeText(this, "Телеграм не работет!", Toast.LENGTH_SHORT).show()
    }
    fun myFun2(view:View){
        Toast.makeText(this, "Вайбер тоже!)", Toast.LENGTH_SHORT).show()
    }
    fun myFun3(view:View){
        Toast.makeText(this, "Не ростраюйся)))", Toast.LENGTH_SHORT).show()
    }
}package com.example.myapplication
import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.view.View
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import androidx.fragment.app.Fragment
import androidx.fragment.app.FragmentManager
```

```
import androidx.fragment.app.FragmentPagerAdapter
import kotlinx.android.synthetic.main.activity_main.*


class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val pref = getSharedPreferences("mypref", Context.MODE_PRIVATE)

        if (pref.getBoolean("firststart", true)) {
            // update sharedpreference - another start wont be the first
            val editor = pref.edit()
            editor.putBoolean("firststart", false)
            editor.commit() // apply changes
            val adapter=MyAdapter(supportFragmentManager)
            adapter.addFragment(Slider1())
            adapter.addFragment(Slider0())
            viewPager.adapter=adapter
            // first start, show your dialog | first-run code goes here
        }
        else{
            startActivity(Intent(this,Glavnaya::class.java))
            finish()
        }

    }
    class MyAdapter(manager: FragmentManager):FragmentPagerAdapter(manager){
        private val FragmentList:MutableList<Fragment> = ArrayList()
        override fun getItem(position: Int): Fragment {
            return FragmentList[position]
        }

        override fun getCount(): Int {
            return FragmentList.size
        }
        fun addFragment(fragment: Fragment){
            FragmentList.add(fragment)

        }
    }
    fun onClick(view: View){
        val currentPage:Int=viewPager.currentItem+1
        val t1 = view.findViewById<TextView>(R.id.textView)
        val t2 = view.findViewById<TextView>(R.id.textView3)
        if(currentPage<2){
            viewPager.currentItem=currentPage
            if(currentPage==1){
                //t1.setBackgroundResource(R.drawable.white_slider)
                //t2.setBackgroundResource(R.drawable.slider)
            }
        }
        else{
            startActivity(Intent(this,Glavnaya::class.java))
            finish()
        }
    }
}package com.example.myapplication

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
```

```
class Waiting : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.waiting)
    }
    fun back(view: View){
        val intent= Intent(this,Order::class.java)
        startActivity(intent)
        finish()
    }
}package com.example.myapplication

import android.content.Intent
import android.os.Bundle
import android.text.InputType
import android.view.View
import android.widget.EditText
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity

class Sign_In : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.sign__in)
        val password = findViewById(R.id.Password) as EditText
        password.inputType = InputType.TYPE_CLASS_TEXT or
InputType.TYPE_TEXT_VARIATION_PASSWORD;
    }
    fun BackSign(view: View){
        val intent= Intent(this,Glavnaya::class.java)
        startActivity(intent)
        finish()
    }
    fun openRegestration(view: View) {
        val Intent = Intent(this, Regestration::class.java)
        startActivity(Intent)
    }

    fun openMain(view: View) {
        val randomIntent = Intent(this, Glavnaya1::class.java)
        val login = findViewById(R.id.Login) as EditText
        val password = findViewById(R.id.Password) as EditText
        if(login.text.toString() == "pikimell")
            if(password.text.toString() == "admin")
                startActivity(randomIntent)
            else{
                Toast.makeText(this, "Неверный пароль!", Toast.LENGTH_SHORT).show()
            }
        else{
            Toast.makeText(this, "Не верный логин!", Toast.LENGTH_SHORT).show()
        }

    }

    fun myFun(view:View){
        Toast.makeText(this, "А я причем?))))", Toast.LENGTH_SHORT).show()
    }
    fun myFun1(view:View){
        Toast.makeText(this, "Телеграм не работет!", Toast.LENGTH_SHORT).show()
    }
    fun myFun2(view:View){
        Toast.makeText(this, "Вайбер тоже!)", Toast.LENGTH_SHORT).show()
    }
    fun myFun3(view:View){
```

```
            Toast.makeText(this, "Не ростраюйся)))", Toast.LENGTH_SHORT).show()
        }

        fun showPassword(view:View){
            val password = findViewById(R.id.Password) as EditText
            if(password.tag == "hide"){
                password.inputType = InputType.TYPE_CLASS_TEXT or
InputType.TYPE_TEXT_VARIATION_PASSWORD;
                password.tag = "show"
            }else{
                password.inputType = InputType.TYPE_TEXT_VARIATION_VISIBLE_PASSWORD;
                password.tag = "hide"
            }

        }
    }package com.example.myapplication

    import android.content.Intent
    import android.graphics.Color
    import android.graphics.Typeface
    import androidx.appcompat.app.AppCompatActivity
    import android.os.Bundle
    import android.util.TypedValue
    import android.view.LayoutInflater
    import android.view.View
    import android.view.ViewGroup
    import android.view.animation.Animation
    import android.view.animation.AnimationUtils
    import android.widget.*
    import androidx.fragment.app.Fragment
    import com.google.android.flexbox.FlexboxLayout
    import kotlinx.android.synthetic.main.activity_saved_menu.*

    class Saved_menu : Fragment(R.layout.activity_saved_menu) {
        override fun onCreateView(
            inflater: LayoutInflater,
            container: ViewGroup?,
            savedInstanceState: Bundle?
        ): View? {

            val view: View = inflater.inflate(R.layout.activity_saved_menu, container, false)
            val buttons = arrayOf(
                view.findViewById(R.id.Restaurantsaved) as Button,
                view.findViewById(R.id.Bluda) as Button
            )
            val elemnt: MutableList<FrameLayout> = ArrayList()
            val animations: Array<Animation?> = arrayOf(
                AnimationUtils.loadAnimation(requireContext(), R.anim.alpha2)
            )


            fun int_to_dp(x:Int): Int {
                val value = TypedValue.applyDimension(
                    TypedValue.COMPLEX_UNIT_DIP,
                    x.toFloat(),
                    resources.displayMetrics
                ).toInt()
                return value
            }

            fun create_frame(img:Int,name:String, tag_name:String, like:String): FrameLayout {
                var params1 = FrameLayout.LayoutParams(int_to_dp(343), int_to_dp(190))
                val mp = LinearLayout.LayoutParams.MATCH_PARENT
                val wc = LinearLayout.LayoutParams.WRAP_CONTENT
```

```
val f = FrameLayout(this.requireContext())//Главный контейнер
f.layoutParams = FrameLayout.LayoutParams(wc,wc)
f.tag = tag_name

val fl = FrameLayout(this.requireContext())//Главный контейнер
params1.setMargins(0,int_to_dp(16),0,int_to_dp(0))
fl.layoutParams = params1
fl.setBackgroundResource(R.drawable.shadow)
fl.setPadding(10,0,10,20)


val iv = ImageView(this.requireContext())//Картинка ресторана
iv.setImageResource(img)
iv.layoutParams = LinearLayout.LayoutParams(mp,wc)
iv.scaleType = ImageView.ScaleType.CENTER_CROP
iv.setOnClickListener {
   activity?.let {
      val intent = Intent(it, Restaurant_menu::class.java)
      it.startActivity(intent)
   }
}


val fl2= FrameLayout(this.requireContext())//нижний контейнер
val params2 = LinearLayout.LayoutParams(0,0)
params2.width = mp
params2.height = int_to_dp(60)
params2.setMargins(0,int_to_dp(130),0,0)
fl2.layoutParams = params2
fl2.setBackgroundColor(Color.WHITE)
fl2.setBackgroundResource(R.drawable.bottom_block)


val tv = TextView(this.requireContext())//Название ресторана
val params3 = LinearLayout.LayoutParams(0,0,1000f)
tv.text = name
params3.width = wc
params3.height = wc
params3.setMargins(int_to_dp(21),int_to_dp(8),0,0)
tv.layoutParams = params3
tv.textSize = 20f
tv.setTypeface(Typeface.SANS_SERIF, Typeface.BOLD)
tv.setTextColor(getResources().getColor(R.color.textDark))
tv.setPadding(21, 9, 0, 0)


val img= ImageView(this.requireContext())//3везды ресторана
val params4 = LinearLayout.LayoutParams(int_to_dp(58),int_to_dp(9))
params4.setMargins(int_to_dp(24),int_to_dp(40),0,0)
img.setImageResource(R.drawable.stars)
img.layoutParams= params4


val heart=ImageButton(this.requireContext())//лайк ресторана
val params5 = LinearLayout.LayoutParams(wc,wc)
params5.setMargins(int_to_dp(290),int_to_dp(7),0,0)
heart.setImageResource(R.drawable.ic_heart_black)
heart.layoutParams= params5
heart.setBackgroundColor(Color.TRANSPARENT)
heart.tag = like
if (heart.tag == "false") {
```

```
            heart.setImageResource(R.drawable.ic_heart_black)
            heart.tag = "true"
        } else {
            heart.setImageResource(R.drawable.ic_heart_like_red)
            heart.tag = "false"
        }
        heart.setOnClickListener {
            if (heart.tag == "false") {
                heart.setImageResource(R.drawable.ic_heart_black)
                heart.tag = "true"
            } else {
                heart.setImageResource(R.drawable.ic_heart_like_red)
                heart.tag = "false"
            }
        }


        val fl3 = FrameLayout(this.requireContext())//контейнер для текста с временем
        val params6 = LinearLayout.LayoutParams(wc,wc)
        params6.setMargins(int_to_dp(40),int_to_dp(25),0,0)
        fl3.layoutParams= params6
        fl3.setPadding(100,25,0,0)


        val txt = TextView(this.requireContext()) //Время работы ресторана
        val params7 = LinearLayout.LayoutParams(wc,wc)
        txt.text = "08:00 - 22:00"
        txt.layoutParams = params7
        txt.textSize = 11f
        txt.setTypeface(Typeface.SANS_SERIF, Typeface.NORMAL)
        txt.setTextColor(getResources().getColor(R.color.textSilver))
        txt.setPadding(21, 9, 0, 0)


        fl3.addView(txt)
        fl2.addView(tv)
        fl2.addView(img)
        fl2.addView(fl3)
        fl2.addView(heart)
        fl.addView(iv)
        fl.addView(fl2)
        f.addView(fl)
        return f
    }

    fun add_elements(img:Int,name:String, tag_name:String, like:String) {
        val f1 = create_frame(img, name, tag_name, like)
        elemnt.add(f1)
    }

    fun show_restourant(flex: FlexboxLayout, filter_tag:String){
        flex.removeAllViewsInLayout()
        for(i in 0..elemnt.size-1){
            if(elemnt[i].tag == filter_tag)
                flex.addView(elemnt[i])
            else if (filter_tag == "")
                flex.addView(elemnt[i])
        }
    }

    fun reset_button(my_iter:Int){
        for(i in 0..buttons.size-1){
            buttons[i].setBackgroundResource(R.drawable.shadow)
            buttons[i].setTextColor(getResources().getColor(R.color.textDark))
```

```
            }
            buttons[my_iter].setBackgroundResource(R.drawable.button_shadow)
            buttons[my_iter].setTextColor(getResources().getColor(R.color.colorWhite))
            buttons[my_iter].startAnimation(animations[0])
        }


        val txt=view.findViewById<TextView>(R.id.myText)
        txt.setText("Всего"+elemnt.size)


        buttons[0].setOnClickListener {
            var my_iter = 0
            val idx = 0
            reset_button(idx)
            show_restourant(view.findViewById(R.id.flexsaved) , "Рестораны")

            val txt=view.findViewById<TextView>(R.id.myText)
            for(i in 0..elemnt.size-1) {
                if (elemnt[i].tag == "Рестораны")
                    my_iter += 1
            }
            txt.setText("Ресторанов "+my_iter)
        }
        buttons[1].setOnClickListener {
            var my_iter = 0
            val idx = 1
            reset_button(idx)
            show_restourant(view.findViewById(R.id.flexsaved) , "Блюда")
            val txt=view.findViewById<TextView>(R.id.myText)
            for(i in 0..elemnt.size-1) {
                if (elemnt[i].tag == "Блюда")
                    my_iter += 1
            }
            txt.setText("Блюд "+my_iter)
        }

        fun but(view: View) {

        }


        for(i in 0..5){
            add_elements(R.drawable.ic_dark_tort,"Bartolomeo","Рестораны","false")
            add_elements(R.drawable.kfc_logo,"Борщ","Блюда","false")
            add_elements(R.drawable.chel,"Сосиска","Рестораны","false")
        }
        show_restourant(view.findViewById(R.id.flexsaved),"")


        return view

    }
}package com.example.myapplication

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment

class Slider1 :Fragment(R.layout.slider) {
    override fun onCreateView(
        inflater: LayoutInflater,
```

```
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return super.onCreateView(inflater, container, savedInstanceState)
    }
}package com.example.myapplication

import android.content.Intent
import android.graphics.Color
import android.graphics.Typeface
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.TypedValue
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.view.animation.Animation
import android.view.animation.AnimationUtils
import android.widget.*
import androidx.fragment.app.Fragment
import com.google.android.flexbox.FlexboxLayout

class Basket_menu : Fragment(R.layout.activity_basket_menu) {
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {

        val view: View = inflater.inflate(R.layout.activity_basket_menu, container, false)

        val buttons = arrayOf(
            view.findViewById(R.id.Restaurant_bask) as Button,
            view.findViewById(R.id.Bluda_bask) as Button
        )
        val elemnt: MutableList<FrameLayout> = ArrayList()
        val animations: Array<Animation?> = arrayOf(
            AnimationUtils.loadAnimation(requireContext(), R.anim.alpha2)
        )


        fun int_to_dp(x:Int): Int {
            val value = TypedValue.applyDimension(
                TypedValue.COMPLEX_UNIT_DIP,
                x.toFloat(),
                resources.displayMetrics
            ).toInt()
            return value
        }

        fun create_frame(number:String,name:String, tag_name:String): FrameLayout {
            val mp = LinearLayout.LayoutParams.MATCH_PARENT
            val wc = LinearLayout.LayoutParams.WRAP_CONTENT
            var params1 = FrameLayout.LayoutParams(mp, int_to_dp(71))



            val f = FrameLayout(this.requireContext())//Главный контейнер
            f.layoutParams = params1
            f.tag = tag_name


            val tv = TextView(this.requireContext())//Номер заказа
            val params2 = LinearLayout.LayoutParams(0,0,1000f)
```

```
    tv.text = number
    params2.width = wc
    params2.height = int_to_dp(13)
    params2.setMargins(int_to_dp(26),int_to_dp(12),0,0)
    tv.layoutParams = params2
    tv.textSize = 12f
    tv.setTypeface(Typeface.SANS_SERIF, Typeface.BOLD)
    tv.setTextColor(getResources().getColor(R.color.textDark))


    val tv2 = TextView(this.requireContext())//Название
    val params3 = LinearLayout.LayoutParams(0,0,1000f)
    tv2.text = name
    params3.width = wc
    params3.height = int_to_dp(23)
    params3.setMargins(int_to_dp(26),int_to_dp(25),0,0)
    tv2.layoutParams = params3
    tv2.textSize = 20f
    tv2.setTypeface(Typeface.SANS_SERIF, Typeface.BOLD)
    tv2.setTextColor(getResources().getColor(R.color.textDark))


    val img= ImageView(this.requireContext())//Тарелка
    val params4 = LinearLayout.LayoutParams(int_to_dp(30),int_to_dp(30))
    params4.setMargins(int_to_dp(172),int_to_dp(15),0,0)
    img.setImageResource(R.drawable.ic_restaurants_black)
    img.layoutParams= params4
    val arrow= ImageView(this.requireContext())//Стрелка
    val params5 = LinearLayout.LayoutParams(int_to_dp(34),int_to_dp(20))
    params5.setMargins(int_to_dp(320),int_to_dp(22),0,0)
    arrow.setImageResource(R.drawable.ic_arrow_forward_black)
    arrow.layoutParams= params5



    f.addView(tv)
    f.addView(tv2)
    f.addView(img)
    f.addView(arrow)
    return f
}

fun add_elements(number:String,name:String, tag_name:String) {
    val f1 = create_frame(number, name, tag_name)
    elemnt.add(f1)
}

fun show_restourant(flex: LinearLayout, filter_tag:String){
    flex.removeAllViewsInLayout()
    for(i in 0..elemnt.size-1){
        if(elemnt[i].tag == filter_tag)
            flex.addView(elemnt[i])
        else if (filter_tag == "")
            flex.addView(elemnt[i])
    }
}

fun reset_button(my_iter:Int){
    for(i in 0..buttons.size-1){
        buttons[i].setBackgroundResource(R.drawable.shadow)
        buttons[i].setTextColor(getResources().getColor(R.color.textDark))
    }
    buttons[my_iter].setBackgroundResource(R.drawable.button_shadow)
    buttons[my_iter].setTextColor(getResources().getColor(R.color.colorWhite))
```

```
            buttons[my_iter].startAnimation(animations[0])
        }



        buttons[0].setOnClickListener {
            val my_iter = 0
            reset_button(my_iter)
            show_restourant(view.findViewById(R.id.basket_lin) , "Ожидаемые")
        }
        buttons[1].setOnClickListener{
            val my_iter = 1
            reset_button(my_iter)
            show_restourant(view.findViewById(R.id.basket_lin) , "Завершённые")
        }

        fun but(view: View) {

        }


        for(i in 0..5){
            add_elements("1234","Bartolomeo","Ожидаемые")
            add_elements("3214","MC","Завершённые")
            add_elements("2314","Chelentano","Ожидаемые")
        }
        show_restourant(view.findViewById(R.id.basket_lin),"")

        return view

    }
}package com.example.myapplication

import android.content.Intent
import android.os.Bundle
import android.view.View
import android.widget.ImageButton
import android.widget.ImageView
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import com.denzcoskun.imageslider.ImageSlider
import com.denzcoskun.imageslider.models.SlideModel


class Dish: AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.dish)
        val imageSlider = findViewById<ImageSlider>(R.id.viewPager2)

        val slideModels: MutableList<SlideModel> = ArrayList()
        slideModels.add(
            SlideModel("фцвф")
        )
        slideModels.add(
            SlideModel("фцв")
        )
        slideModels.add(
            SlideModel("цфв")
        )
        imageSlider.setImageList(slideModels, true)

    }
```

```
    fun return_Rest(view: View){
        val intent= Intent(this,Restaurant_menu::class.java)
        startActivity(intent)
        finish()
    }
    fun likeClick(view: View){
        val b = view as ImageButton

        if(b.tag == "false"){
            b.setImageResource(R.drawable.ic_heart_black)
            b.tag = "true"
        }else{
            b.setImageResource(R.drawable.ic_heart_like_red)
            b.tag = "false"
        }
    }

    fun plus(view: View){
        val tt = findViewById(R.id.iterator) as TextView;
        tt.text = (tt.text.toString().toInt() + 1).toString()
    }

    fun minus(view: View){
        val tt = findViewById(R.id.iterator) as TextView;

        if(tt.text.toString().toInt()>=1)
            tt.text = (tt.text.toString().toInt() - 1).toString()
    }
}package com.example.myapplication

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment

class Slider0 :Fragment(R.layout.slider_0) {
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return super.onCreateView(inflater, container, savedInstanceState)
    }
}package com.example.myapplication

import android.content.Intent
import android.graphics.Color
import android.graphics.Typeface
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.TypedValue
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.view.animation.Animation
import android.view.animation.AnimationUtils
import android.widget.*
import androidx.core.view.marginLeft
import androidx.fragment.app.Fragment
import com.denzcoskun.imageslider.ImageSlider
import com.denzcoskun.imageslider.models.SlideModel
import com.google.android.flexbox.FlexboxLayout
import kotlinx.android.synthetic.main.activity_find_menu.*
```

```
class Find_menu : Fragment(R.layout.activity_find_menu) {
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {

        val view: View = inflater.inflate(R.layout.activity_find_menu, container, false)
        val buttons:Array<Button> = arrayOf(
            view.findViewById(R.id.Restaurant) as Button,
            view.findViewById(R.id.Kuhni) as Button,
            view.findViewById(R.id.Ingridients) as Button
        )
        val elemnt: MutableList<FrameLayout> = ArrayList()
        val animations: Array<Animation?> = arrayOf(
            AnimationUtils.loadAnimation(requireContext(), R.anim.alpha),
            AnimationUtils.loadAnimation(requireContext(), R.anim.alpha2),
            AnimationUtils.loadAnimation(requireContext(), R.anim.combo2)
        )


        fun int_to_dp(x:Int): Int {
            val value = TypedValue.applyDimension(
                TypedValue.COMPLEX_UNIT_DIP,
                x.toFloat(),
                resources.displayMetrics
            ).toInt()
            return value
        }
        fun create_frame(img:Int,name:String, tag_name:String, like:String): FrameLayout {
            var params1 = FrameLayout.LayoutParams(int_to_dp(343), int_to_dp(190))
            val mp = LinearLayout.LayoutParams.MATCH_PARENT
            val wc = LinearLayout.LayoutParams.WRAP_CONTENT



            val f = FrameLayout(this.requireContext())//Главный контейнер
            f.layoutParams = FrameLayout.LayoutParams(wc,wc)
            f.tag = tag_name

            val fl = FrameLayout(this.requireContext())//Главный контейнер
            params1.setMargins(0,int_to_dp(16),0,int_to_dp(0))
            fl.layoutParams = params1
            fl.setBackgroundResource(R.drawable.shadow)
            fl.setPadding(10,0,10,20)


            val iv = ImageView(this.requireContext())//Картинка ресторана
            iv.setImageResource(img)
            iv.layoutParams = LinearLayout.LayoutParams(mp,wc)
            iv.scaleType = ImageView.ScaleType.CENTER_CROP
            iv.setOnClickListener {
                activity?.let {
                    val intent = Intent(it, Restaurant_menu::class.java)
                    it.startActivity(intent)
                }
            }


            val fl2= FrameLayout(this.requireContext())//нижний контейнер
            val params2 = LinearLayout.LayoutParams(0,0)
            params2.width = mp
            params2.height = int_to_dp(60)
```

```
params2.setMargins(0,int_to_dp(130),0,0)
fl2.layoutParams = params2
fl2.setBackgroundColor(Color.WHITE)
fl2.setBackgroundResource(R.drawable.bottom_block)


val tv = TextView(this.requireContext())//Название ресторана
val params3 = LinearLayout.LayoutParams(0,0,1000f)
tv.text = name
params3.width = wc
params3.height = wc
params3.setMargins(int_to_dp(21),int_to_dp(8),0,0)
tv.layoutParams = params3
tv.textSize = 20f
tv.setTypeface(Typeface.SANS_SERIF, Typeface.BOLD)
tv.setTextColor(getResources().getColor(R.color.textDark))
tv.setPadding(21, 9, 0, 0)


val img= ImageView(this.requireContext())//3звезды ресторана
val params4 = LinearLayout.LayoutParams(int_to_dp(58),int_to_dp(9))
params4.setMargins(int_to_dp(24),int_to_dp(40),0,0)
img.setImageResource(R.drawable.stars)
img.layoutParams= params4


val heart=ImageButton(this.requireContext())//лайк ресторана
val params5 = LinearLayout.LayoutParams(wc,wc)
params5.setMargins(int_to_dp(290),int_to_dp(7),0,0)
heart.setImageResource(R.drawable.ic_heart_black)
heart.layoutParams= params5
heart.setBackgroundColor(Color.TRANSPARENT)
heart.tag = like
if (heart.tag == "false") {
   heart.setImageResource(R.drawable.ic_heart_black)
   heart.tag = "true"
} else {
   heart.setImageResource(R.drawable.ic_heart_like_red)
   heart.tag = "false"
}
heart.setOnClickListener {
   if (heart.tag == "false") {
      heart.setImageResource(R.drawable.ic_heart_black)
      heart.tag = "true"
   } else {
      heart.setImageResource(R.drawable.ic_heart_like_red)
      heart.tag = "false"
   }
}


val fl3 = FrameLayout(this.requireContext())//контейнер для текста с временем
val params6 = LinearLayout.LayoutParams(wc,wc)
params6.setMargins(int_to_dp(40),int_to_dp(25),0,0)
fl3.layoutParams= params6
fl3.setPadding(100,25,0,0)


val txt = TextView(this.requireContext()) //Время работы ресторана
val params7 = LinearLayout.LayoutParams(wc,wc)
txt.text = "08:00 - 22:00"
txt.layoutParams = params7
txt.textSize = 11f
txt.setTypeface(Typeface.SANS_SERIF, Typeface.NORMAL)
```

```
    txt.setTextColor(getResources().getColor(R.color.textSilver))
    txt.setPadding(21, 9, 0, 0)


    fl3.addView(txt)
    fl2.addView(tv)
    fl2.addView(img)
    fl2.addView(fl3)
    fl2.addView(heart)
    fl.addView(iv)
    fl.addView(fl2)
    f.addView(fl)
    return f
}
fun create_blydo(img:Int,name:String, tag_name:String):FrameLayout{
    val mp = LinearLayout.LayoutParams.MATCH_PARENT
    val wc = LinearLayout.LayoutParams.WRAP_CONTENT

    val frame_layout = FrameLayout(this.requireContext())//Главный контейнер
    frame_layout.layoutParams = FrameLayout.LayoutParams(wc,wc)
    frame_layout.setPadding(0,int_to_dp(11),0,0)
    frame_layout.tag = tag_name

    val frame_layout1 = FrameLayout(this.requireContext())
    frame_layout1.layoutParams = FrameLayout.LayoutParams(wc,wc)


    val image_view  = ImageView(this.requireContext())
    var params3 = FrameLayout.LayoutParams(int_to_dp(136), int_to_dp(136))
    params3.setMargins(int_to_dp(-5),0,0,0)
    image_view.setImageResource(R.drawable.dish1)
    image_view.layoutParams = params3


    val frame_layout2 = FrameLayout(this.requireContext())
    var params1 = FrameLayout.LayoutParams(int_to_dp(343), int_to_dp(104))
    params1.setMargins(int_to_dp(17),int_to_dp(43),0,0)
    frame_layout2.layoutParams = params1
    frame_layout2.setBackgroundResource(R.drawable.layout)

    var text_view1 = TextView(this.requireContext())
    var params2 = FrameLayout.LayoutParams(wc,wc)
    params2.setMargins(int_to_dp(130),int_to_dp(13),0,0)
    text_view1.layoutParams = params2
    text_view1.text = name
    text_view1.textSize = 20f
    text_view1.setTypeface(Typeface.SANS_SERIF, Typeface.BOLD)
    text_view1.setTextColor(getResources().getColor(R.color.slider))

    var text_view2 = TextView(this.requireContext())
    var params4 = FrameLayout.LayoutParams(wc,wc)
    params4.setMargins(int_to_dp(301),int_to_dp(13),0,0)
    text_view2.layoutParams = params4
    text_view2.text = "350грн"
    text_view2.textSize = 11f
    text_view2.setTypeface(Typeface.SANS_SERIF, Typeface.NORMAL)
    text_view2.setTextColor(getResources().getColor(R.color.slider))

    var text_view3 = TextView(this.requireContext())
    var params5 = FrameLayout.LayoutParams(wc,wc)
    params5.setMargins(int_to_dp(128),int_to_dp(46),0,0)
    text_view3.layoutParams = params5
    text_view3.text = "Тунец с тыквой, под соусом табоджи поданый с кусочком лука (200гр)"
    text_view3.textSize = 11f
```

```
    text_view3.setTypeface(Typeface.SANS_SERIF, Typeface.NORMAL)
    text_view3.setTextColor(getResources().getColor(R.color.textSilver))

    frame_layout2.addView(text_view1)
    frame_layout2.addView(text_view2)
    frame_layout2.addView(text_view3)

    frame_layout1.addView(frame_layout2)
    frame_layout1.addView(image_view)


    frame_layout.addView(frame_layout1)
    return frame_layout
}
fun add_elements(img:Int,name:String, tag_name:String, like:String) {
    var f1:FrameLayout
    if(tag_name == "Рестораны")
        f1 = create_frame(img, name, tag_name, "false")
    else
        f1 = create_blydo(img, name, tag_name)
    elemnt.add(f1)
}
fun show_restourant(flex: FlexboxLayout, filter_tag:String){
    flex.removeAllViewsInLayout()
    for(i in 0..elemnt.size-1){
        if(elemnt[i].tag == filter_tag)
            flex.addView(elemnt[i])
        else if (filter_tag == "")
            flex.addView(elemnt[i])
    }
}
fun reset_button(my_iter:Int){
    for(i in 0..2){
        buttons[i].setBackgroundResource(R.drawable.shadow)
        buttons[i].setTextColor(getResources().getColor(R.color.textDark))
    }
    buttons[my_iter].setBackgroundResource(R.drawable.button_shadow)
    buttons[my_iter].setTextColor(getResources().getColor(R.color.colorWhite))
    buttons[my_iter].startAnimation(animations[my_iter])
}




buttons[0].setOnClickListener {
    val my_iter = 0
    reset_button(my_iter)
    show_restourant(view.findViewById(R.id.flex) , "Рестораны")
}
buttons[1].setOnClickListener {
    val my_iter = 1
    reset_button(my_iter)
    show_restourant(view.findViewById(R.id.flex) , "Кухни")
}
buttons[2].setOnClickListener {
    val my_iter = 2
    reset_button(my_iter)
    show_restourant(view.findViewById(R.id.flex) , "Ингридиенты")
}




for(i in 0..5){
```

```
                add_elements(R.drawable.ic_dark_tort,"Bartolomeo","Рестораны","false")
                add_elements(R.drawable.kfc_logo,"Украинская","Кухни","false")
                add_elements(R.drawable.chel,"Сосиска","Ингридиенты","false")
            }
            show_restourant(view.findViewById(R.id.flex),"Рестораны")

            return view

        }
    }package com.example.myapplication

    import android.app.ActionBar
    import android.content.Intent
    import android.graphics.Typeface
    import android.os.Bundle
    import android.util.TypedValue
    import android.view.View
    import android.view.animation.Animation
    import android.view.animation.AnimationUtils
    import android.widget.*
    import androidx.appcompat.app.AppCompatActivity
    import androidx.constraintlayout.widget.ConstraintLayout
    import com.denzcoskun.imageslider.ImageSlider
    import com.denzcoskun.imageslider.models.SlideModel
    import com.google.android.flexbox.FlexboxLayout
    import kotlinx.android.synthetic.main.restoraunt.*


    class Restaurant_menu : AppCompatActivity() {
        val elemnt: MutableList<FrameLayout> = ArrayList()
        val element2: MutableList<FrameLayout> = ArrayList()
        val element3: MutableList<FrameLayout> = ArrayList()
        override fun onCreate(savedInstanceState: Bundle?) {
            super.onCreate(savedInstanceState)
            setContentView(R.layout.restoraunt)

            val imageSlider = findViewById<ImageSlider>(R.id.viewPager3)

            val slideModels: MutableList<SlideModel> = ArrayList()
            slideModels.add(
                SlideModel("https://-tbn0.gstatic.com/images?q=tbn:ANd9GcTeY-x0-ozntxe-
jwblHmlddIc1PqUrdDn6Nw&usqp=CAU")
            )
            slideModels.add(
                SlideModel("https://reston.ua/uploads/img/zavedeniya/original/54212b9.jpg")
            )
            slideModels.add(
                SlideModel("https://www.dniprohotel.ua/images/m/dnipro-m.jpg")
            )
            imageSlider.setImageList(slideModels, true)
            for(i in 0..5){
                add_elements(R.drawable.dish1,"Dish1","Закуски","false")
                add_elements(R.drawable.dish2,"Dish2","Напитки","false")
                add_elements(R.drawable.dish1,"Dish3","Горячее","false")
                add_elements(R.drawable.dish2,"Dish4","Закуски","false")
            }
            show_restourant(flexrestor,elemnt.size-1)
        }

        fun int_to_dp(x:Int): Int {
            val value = TypedValue.applyDimension(
                TypedValue.COMPLEX_UNIT_DIP,
                x.toFloat(),
                resources.displayMetrics
```

```
    ).toInt()
    return value
}
fun create_blydo(img:Int,name:String, tag_name:String): FrameLayout {
    val mp = LinearLayout.LayoutParams.MATCH_PARENT
    val wc = LinearLayout.LayoutParams.WRAP_CONTENT

    val frame_layout = FrameLayout(baseContext)//Главный контейнер
    frame_layout.layoutParams = FrameLayout.LayoutParams(wc,wc)
    frame_layout.setPadding(0,int_to_dp(11),0,0)
    frame_layout.tag = tag_name

    val frame_layout1 = FrameLayout(baseContext)//Контейнер с картинкой
    frame_layout1.layoutParams = FrameLayout.LayoutParams(wc,wc)
    frame_layout1.setOnClickListener {
        startActivity(Intent(this,Dish::class.java))
    }

    val image_view  = ImageView(baseContext)//Картинка
    var params3 = FrameLayout.LayoutParams(int_to_dp(136), int_to_dp(136))
    params3.setMargins(int_to_dp(-5),0,0,0)
    image_view.setImageResource(img)
    image_view.layoutParams = params3


    val frame_layout2 = FrameLayout(baseContext)//Блюдо
    var params1 = FrameLayout.LayoutParams(int_to_dp(343), int_to_dp(104))
    params1.setMargins(int_to_dp(10),int_to_dp(33),0,int_to_dp(10))
    frame_layout2.layoutParams = params1
    frame_layout2.setBackgroundResource(R.drawable.layout3)

    var text_view3 = TextView(baseContext)//Название
    var params5 = FrameLayout.LayoutParams(wc,wc)
    params5.setMargins(int_to_dp(120),int_to_dp(11),0,0)
    text_view3.layoutParams = params5
    text_view3.text = name
    text_view3.textSize = 18f
    text_view3.setTypeface(Typeface.SANS_SERIF, Typeface.BOLD)
    text_view3.setTextColor(getResources().getColor(R.color.textDark))
    val text_view2 = TextView(baseContext)//Цена
    val params4 = FrameLayout.LayoutParams(wc,wc)
    params4.setMargins(int_to_dp(121),int_to_dp(50),0,0)
    text_view2.layoutParams = params4
    text_view2.text = "Тунец с тыквой, под соусом табоджи поданый с кусочком лука (200гр)"
    text_view2.textSize = 12f
    text_view2.setTypeface(Typeface.SANS_SERIF, Typeface.NORMAL)
    text_view2.setTextColor(getResources().getColor(R.color.textSilver))
    val frame_layout3 = FrameLayout(baseContext)//Кнопки
    val params2=FrameLayout.LayoutParams(int_to_dp(75),int_to_dp(42))
    params2.setMargins(int_to_dp(259),int_to_dp(30),int_to_dp(13),int_to_dp(20))
    frame_layout3.layoutParams=params2
    frame_layout3.setBackgroundResource(R.drawable.shadow)
    var i=1
    var text_view1 = TextView(baseContext)//Колво
    var params8 = FrameLayout.LayoutParams(wc,int_to_dp(27))
    params8.setMargins(int_to_dp(25),int_to_dp(10),int_to_dp(8),0)
    text_view1.layoutParams = params8
    text_view1.text = ""+i
    text_view1.textSize = 12f
    text_view1.setTypeface(Typeface.SANS_SERIF, Typeface.BOLD)
    text_view1.setTextColor(getResources().getColor(R.color.textDark))
    val imgb1 = ImageButton(baseContext)//Минус
    val params6=FrameLayout.LayoutParams(int_to_dp(11),int_to_dp(10))
    params6.setMargins(int_to_dp(5),int_to_dp(13),int_to_dp(50),int_to_dp(15))
```

```
    imgb1.layoutParams=params6
    imgb1.setImageResource(R.drawable.minus)
    imgb1.setOnClickListener {
        if(i>0) {
            i--
            text_view1.text = "" + i
        }
        else text_view1.text=" "

    }
    val imgb2 = ImageButton(baseContext)//Плюс
    val params7=FrameLayout.LayoutParams(int_to_dp(12),int_to_dp(33))
    params7.setMargins(int_to_dp(45),int_to_dp(5),int_to_dp(8),int_to_dp(10))
    imgb2.layoutParams=params7
    imgb2.setImageResource(R.drawable.plus)
    imgb2.setOnClickListener {
        i++
        text_view1.text=""+i
    }
    frame_layout3.addView(imgb1)
    frame_layout3.addView(text_view1)
    frame_layout3.addView(imgb2)
    frame_layout2.addView(frame_layout3)
    frame_layout2.addView(text_view3)
    frame_layout2.addView(text_view2)
    frame_layout1.addView(frame_layout2)
    frame_layout1.addView(image_view)


    frame_layout.addView(frame_layout1)
    return frame_layout

}
fun add_elements(img:Int,name:String, tag_name:String, like:String) {
    var f1: FrameLayout
    f1 = create_blydo(img, name, tag_name)
    elemnt.add(f1)
}
fun show_restourant(flex: FlexboxLayout, length:Int){
    if(length<=elemnt.size)
        for(i in 0..length){
            flex.addView(elemnt[i])
        }
}
fun plus(view: View){
    val tt = findViewById(R.id.iterator)as TextView;
    tt.text = (tt.text.toString().toInt() + 1).toString()
}

fun minus(view: View){
    val tt = findViewById(R.id.iterator)as TextView;
    if(tt.text.toString().toInt()>=1)
    tt.text = (tt.text.toString().toInt() - 1).toString()
}

fun openDish(view: View){
    val intent= Intent(this,Dish::class.java)
    startActivity(intent)
}
fun Ordered(view: View){
    val intent= Intent(this,Order::class.java)
    startActivity(intent)
}
```

```
    fun openRestaurantMenu(view: View){
        val intent= Intent(this,Glavnaya1::class.java)
        startActivity(intent)
        finish()
    }

    /*fun Podrobnee(view: View){
        val ico = findViewById(R.id.strelka) as ImageView
        val container = findViewById(R.id.PlusOther) as ConstraintLayout
        val kuhni = findViewById(R.id.podrobneeLay) as ConstraintLayout

        if(but.text == "Подробнее") {
            container.layoutParams.height = ConstraintLayout.LayoutParams.WRAP_CONTENT
            but.text = "Скрыть"
            ico.setImageResource(R.drawable.ic_arrow_up)
            kuhni.layoutParams.height = ConstraintLayout.LayoutParams.WRAP_CONTENT
        }
        else {
            container.layoutParams.height = int_to_dp(40);
            but.text = "Подробнее"
            ico.setImageResource(R.drawable.ic_arrow_down)
        }
    }*/


    fun all_menu(view: View){
        val anim: Animation? = AnimationUtils.loadAnimation(this, R.anim.alpha);
        val animtext: Animation? = AnimationUtils.loadAnimation(this, R.anim.combo);
        val myIter = 0
        val size = 4-1
        val buttons = arrayOf(findViewById(R.id.relativeLay1) as RelativeLayout, findViewById(R.id.relativeLay2) as
RelativeLayout, findViewById(R.id.relativeLay3) as RelativeLayout, findViewById(R.id.relativeLay4) as RelativeLayout)
        val images= arrayOf(findViewById(R.id.relativeImg1) as ImageView, findViewById(R.id.relativeImg2) as
ImageView, findViewById(R.id.relativeImg3) as ImageView, findViewById(R.id.relativeImg4) as ImageView)
        val icons =
arrayOf(R.drawable.ic_menu_red,R.drawable.ic_cold_red,R.drawable.ic_hot_red,R.drawable.ic_drink_red,
            R.drawable.ic_menu_black,R.drawable.ic_cold_black,R.drawable.ic_hot_black,R.drawable.ic_drink_black)
        val texts = arrayOf(findViewById(R.id.relativeBut1) as TextView, findViewById(R.id.relativeBut2) as TextView,
findViewById(R.id.relativeBut3) as TextView, findViewById(R.id.relativeBut4) as TextView)
        val sizeInPX1 =
TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP,64f,resources.displayMetrics).toInt()
        val sizeInPX2 =
TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP,150f,resources.displayMetrics).toInt()

        for(i in 0..size){
            if(i != myIter){
                buttons[i].setBackgroundResource(R.drawable.shadow)
                images[i].setImageResource(icons[i+4])
                texts[i].text = ""
                buttons[i].layoutParams.width=sizeInPX1
            }
        }

        val textViewanim = findViewById<TextView>(R.id.relativeBut1)
        buttons[myIter].setBackgroundResource(R.drawable.button_shadow)
        images[myIter].setImageResource(icons[myIter])
        texts[myIter].text = "Всё меню"
        buttons[myIter].startAnimation(anim)
        textViewanim.startAnimation(animtext)
        buttons[myIter].layoutParams.width=sizeInPX2
    }

    fun cold_platters(view: View){
```

```
        val anim: Animation? = AnimationUtils.loadAnimation(this, R.anim.alpha2);
        val animtext: Animation? = AnimationUtils.loadAnimation(this, R.anim.combo);
        val myIter = 1
        val size = 4-1
        val buttons = arrayOf(findViewById(R.id.relativeLay1) as RelativeLayout, findViewById(R.id.relativeLay2) as
RelativeLayout, findViewById(R.id.relativeLay3) as RelativeLayout, findViewById(R.id.relativeLay4) as RelativeLayout)
        val images= arrayOf(findViewById(R.id.relativeImg1) as ImageView, findViewById(R.id.relativeImg2) as
ImageView, findViewById(R.id.relativeImg3) as ImageView, findViewById(R.id.relativeImg4) as ImageView)
        val icons =
arrayOf(R.drawable.ic_menu_red,R.drawable.ic_cold_red,R.drawable.ic_hot_red,R.drawable.ic_drink_red,
            R.drawable.ic_menu_black,R.drawable.ic_cold_black,R.drawable.ic_hot_black,R.drawable.ic_drink_black)
        val texts = arrayOf(findViewById(R.id.relativeBut1) as TextView, findViewById(R.id.relativeBut2) as TextView,
findViewById(R.id.relativeBut3) as TextView, findViewById(R.id.relativeBut4) as TextView)
        val sizeInPX1 =
TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP,64f,resources.displayMetrics).toInt()
        val sizeInPX2 =
TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP,150f,resources.displayMetrics).toInt()

        for(i in 0..size){
          if(i != myIter){
            buttons[i].setBackgroundResource(R.drawable.shadow)
            images[i].setImageResource(icons[i+4])
            texts[i].text = ""
            buttons[i].layoutParams.width=sizeInPX1
          }
        }
        val textViewanim = findViewById<TextView>(R.id.relativeBut2)
        buttons[myIter].setBackgroundResource(R.drawable.button_shadow)
        images[myIter].setImageResource(icons[myIter])
        texts[myIter].text = "Закуски"
        buttons[myIter].startAnimation(anim)
        textViewanim.startAnimation(animtext)
        buttons[myIter].layoutParams.width=sizeInPX2
    }

    fun soups(view: View){
        val anim: Animation? = AnimationUtils.loadAnimation(this, R.anim.alpha2);
        val animtext: Animation? = AnimationUtils.loadAnimation(this, R.anim.combo);
        val myIter = 2
        val size = 4-1
        val buttons = arrayOf(findViewById(R.id.relativeLay1) as RelativeLayout, findViewById(R.id.relativeLay2) as
RelativeLayout, findViewById(R.id.relativeLay3) as RelativeLayout, findViewById(R.id.relativeLay4) as RelativeLayout)
        val images= arrayOf(findViewById(R.id.relativeImg1) as ImageView, findViewById(R.id.relativeImg2) as
ImageView, findViewById(R.id.relativeImg3) as ImageView, findViewById(R.id.relativeImg4) as ImageView)
        val icons =
arrayOf(R.drawable.ic_menu_red,R.drawable.ic_cold_red,R.drawable.ic_hot_red,R.drawable.ic_drink_red,
            R.drawable.ic_menu_black,R.drawable.ic_cold_black,R.drawable.ic_hot_black,R.drawable.ic_drink_black)
        val texts = arrayOf(findViewById(R.id.relativeBut1) as TextView, findViewById(R.id.relativeBut2) as TextView,
findViewById(R.id.relativeBut3) as TextView, findViewById(R.id.relativeBut4) as TextView)
        val sizeInPX1 =
TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP,64f,resources.displayMetrics).toInt()
        val sizeInPX2 =
TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP,150f,resources.displayMetrics).toInt()

        for(i in 0..size){
          if(i != myIter){
            buttons[i].setBackgroundResource(R.drawable.shadow)
            images[i].setImageResource(icons[i+4])
            texts[i].text = ""
            buttons[i].layoutParams.width=sizeInPX1
          }
        }
        val textViewanim = findViewById<TextView>(R.id.relativeBut3)
        buttons[myIter].setBackgroundResource(R.drawable.button_shadow)
```

```
            images[myIter].setImageResource(icons[myIter])
            texts[myIter].text = "Горячее"
            buttons[myIter].startAnimation(anim)
            textViewanim.startAnimation(animtext)
            buttons[myIter].layoutParams.width=sizeInPX2
        }

        fun beverages(view: View){
            val anim: Animation? = AnimationUtils.loadAnimation(this, R.anim.combo2);
            val animtext: Animation? = AnimationUtils.loadAnimation(this, R.anim.combo);
            val myIter = 3
            val size = 4-1
            val buttons = arrayOf(findViewById(R.id.relativeLay1) as RelativeLayout, findViewById(R.id.relativeLay2) as
RelativeLayout, findViewById(R.id.relativeLay3) as RelativeLayout, findViewById(R.id.relativeLay4) as RelativeLayout)
            val images= arrayOf(findViewById(R.id.relativeImg1) as ImageView, findViewById(R.id.relativeImg2) as
ImageView, findViewById(R.id.relativeImg3) as ImageView, findViewById(R.id.relativeImg4) as ImageView)
            val icons =
arrayOf(R.drawable.ic_menu_red,R.drawable.ic_cold_red,R.drawable.ic_hot_red,R.drawable.ic_drink_red,
                R.drawable.ic_menu_black,R.drawable.ic_cold_black,R.drawable.ic_hot_black,R.drawable.ic_drink_black)
            val texts = arrayOf(findViewById(R.id.relativeBut1) as TextView, findViewById(R.id.relativeBut2) as TextView,
findViewById(R.id.relativeBut3) as TextView, findViewById(R.id.relativeBut4) as TextView)
            val sizeInPX1 =
TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP,64f,resources.displayMetrics).toInt()
            val sizeInPX2 =
TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP,150f,resources.displayMetrics).toInt()

            for(i in 0..size){
                if(i != myIter){
                    buttons[i].setBackgroundResource(R.drawable.shadow)
                    images[i].setImageResource(icons[i+4])
                    texts[i].text = ""
                    buttons[i].layoutParams.width=sizeInPX1
                }
            }
            val textViewanim = findViewById<TextView>(R.id.relativeBut4)
            buttons[myIter].setBackgroundResource(R.drawable.button_shadow)
            images[myIter].setImageResource(icons[myIter])
            texts[myIter].text = "Напитки"
            buttons[myIter].startAnimation(anim)
            textViewanim.startAnimation(animtext)
            buttons[myIter].layoutParams.width=sizeInPX2
        }


        fun likeClick(view: View){
            val b = view as ImageButton
            if(b.tag == "false"){
                b.setImageResource(R.drawable.ic_heart_black)
                b.tag = "true"
            }else{
                b.setImageResource(R.drawable.ic_heart_like_red)
                b.tag = "false"
            }
        }

    }package com.example.myapplication


    import android.content.Intent
    import android.graphics.Color
    import android.graphics.Typeface
    import android.os.Bundle
    import android.util.TypedValue
    import android.view.LayoutInflater
```

```
import android.view.View
import android.view.ViewGroup
import android.view.animation.Animation
import android.view.animation.AnimationUtils
import android.widget.*
import androidx.fragment.app.Fragment
import com.denzcoskun.imageslider.ImageSlider
import com.denzcoskun.imageslider.models.SlideModel
import com.google.android.flexbox.FlexboxLayout
import com.google.android.gms.vision.Frame
import java.time.format.TextStyle


class PinnedButtons : Fragment(R.layout.activity_pinned_buttons) {
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {


        val view: View = inflater.inflate(R.layout.activity_pinned_buttons, container, false)
        val elemnt: MutableList<FrameLayout> = ArrayList()
        val relativeLay1 = view.findViewById(R.id.relativeLay1) as RelativeLayout
        val relativeLay2 = view.findViewById(R.id.relativeLay2) as RelativeLayout
        val relativeLay3 = view.findViewById(R.id.relativeLay3) as RelativeLayout
        val buttons: Array<RelativeLayout> = arrayOf(
            view.findViewById(R.id.relativeLay1) as RelativeLayout,
            view.findViewById(R.id.relativeLay2) as RelativeLayout,
            view.findViewById(R.id.relativeLay3) as RelativeLayout
        )
        val images: Array<ImageView> = arrayOf(
            view.findViewById(R.id.relativeImg1) as ImageView,
            view.findViewById(R.id.relativeImg2) as ImageView,
            view.findViewById(R.id.relativeImg3) as ImageView
        )
        var icons = arrayOf(
            R.drawable.ic_hand_like_white,
            R.drawable.ic_new_white,
            R.drawable.ic_sale_white,
            R.drawable.ic_hand_like_black,
            R.drawable.ic_new_black,
            R.drawable.ic_sale_black
        )
        val texts: Array<TextView> = arrayOf(
            view.findViewById(R.id.relativeBut1) as TextView,
            view.findViewById(R.id.relativeBut2) as TextView,
            view.findViewById(R.id.relativeBut3) as TextView
        )
        val animations: Array<Animation?> = arrayOf(
            AnimationUtils.loadAnimation(requireContext(), R.anim.alpha),
            AnimationUtils.loadAnimation(requireContext(), R.anim.alpha2),
            AnimationUtils.loadAnimation(requireContext(), R.anim.combo2),
            AnimationUtils.loadAnimation(requireContext(), R.anim.combo)
        )



        fun int_to_dp(x:Int): Int {
            val value = TypedValue.applyDimension(
                TypedValue.COMPLEX_UNIT_DIP,
                x.toFloat(),
                resources.displayMetrics
            ).toInt()
```

```
    return value
}

fun create_frame(img:Int,name:String, tag_name:String, like:String): FrameLayout {
    var params1 = FrameLayout.LayoutParams(int_to_dp(343), int_to_dp(190))
    val mp = LinearLayout.LayoutParams.MATCH_PARENT
    val wc = LinearLayout.LayoutParams.WRAP_CONTENT



    val f = FrameLayout(this.requireContext())//Главный контейнер
    f.layoutParams = FrameLayout.LayoutParams(wc,wc)
    f.tag = tag_name

    val fl = FrameLayout(this.requireContext())//Главный контейнер
    params1.setMargins(0,int_to_dp(16),0,int_to_dp(0))
    fl.layoutParams = params1
    fl.setBackgroundResource(R.drawable.shadow)
    fl.setPadding(10,0,10,20)


    val iv = ImageView(this.requireContext())//Картинка ресторана
    iv.setImageResource(img)
    iv.layoutParams = LinearLayout.LayoutParams(mp,wc)
    iv.scaleType = ImageView.ScaleType.CENTER_CROP
    iv.setOnClickListener {
        activity?.let {
            val intent = Intent(it, Restaurant_menu::class.java)
            it.startActivity(intent)
        }
    }


    val fl2=FrameLayout(this.requireContext())//нижний контейнер
    val params2 = LinearLayout.LayoutParams(0,0)
    params2.width = mp
    params2.height = int_to_dp(60)
    params2.setMargins(0,int_to_dp(130),0,0)
    fl2.layoutParams = params2
    fl2.setBackgroundColor(Color.WHITE)
    fl2.setBackgroundResource(R.drawable.bottom_block)


    val tv = TextView(this.requireContext())//Название ресторана
    val params3 = LinearLayout.LayoutParams(0,0,1000f)
    tv.text = name
    params3.width = wc
    params3.height = wc
    params3.setMargins(int_to_dp(21),int_to_dp(8),0,0)
    tv.layoutParams = params3
    tv.textSize = 20f
    tv.setTypeface(Typeface.SANS_SERIF,Typeface.BOLD)
    tv.setTextColor(getResources().getColor(R.color.textDark))
    tv.setPadding(21, 9, 0, 0)


    val img=ImageView(this.requireContext())//Звезды ресторана
    val params4 = LinearLayout.LayoutParams(int_to_dp(58),int_to_dp(9))
    params4.setMargins(int_to_dp(24),int_to_dp(40),0,0)
    img.setImageResource(R.drawable.stars)
    img.layoutParams= params4


    val heart=ImageButton(this.requireContext())//лайк ресторана
```

```
    val params5 = LinearLayout.LayoutParams(wc,wc)
    params5.setMargins(int_to_dp(290),int_to_dp(7),0,0)
    heart.setImageResource(R.drawable.ic_heart_black)
    heart.layoutParams= params5
    heart.setBackgroundColor(Color.TRANSPARENT)
    heart.tag = like
    if (heart.tag == "false") {
        heart.setImageResource(R.drawable.ic_heart_black)
        heart.tag = "true"
    } else {
        heart.setImageResource(R.drawable.ic_heart_like_red)
        heart.tag = "false"
    }
    heart.setOnClickListener {
        if (heart.tag == "false") {
            heart.setImageResource(R.drawable.ic_heart_black)
            heart.tag = "true"
        } else {
            heart.setImageResource(R.drawable.ic_heart_like_red)
            heart.tag = "false"
        }
    }


    val fl3 = FrameLayout(this.requireContext())//контейнер для текста с временем
    val params6 = LinearLayout.LayoutParams(wc,wc)
    params6.setMargins(int_to_dp(40),int_to_dp(25),0,0)
    fl3.layoutParams= params6
    fl3.setPadding(100,25,0,0)


    val txt =TextView(this.requireContext()) //Время работы ресторана
    val params7 = LinearLayout.LayoutParams(wc,wc)
    txt.text = "08:00 - 22:00"
    txt.layoutParams = params7
    txt.textSize = 11f
    txt.setTypeface(Typeface.SANS_SERIF,Typeface.NORMAL)
    txt.setTextColor(getResources().getColor(R.color.textSilver))
    txt.setPadding(21, 9, 0, 0)


    fl3.addView(txt)
    fl2.addView(tv)
    fl2.addView(img)
    fl2.addView(fl3)
    fl2.addView(heart)
    fl.addView(iv)
    fl.addView(fl2)
    f.addView(fl)
    return f
}

fun add_elements(img:Int,name:String, tag_name:String, like:String) {
    val f1 = create_frame(img, name, tag_name, like)
    elemnt.add(f1)
}

fun show_restourant(flex:FlexboxLayout, filter_tag:String){
    flex.removeAllViewsInLayout()
    for(i in 0..elemnt.size-1){
        if(elemnt[i].tag == filter_tag)
            flex.addView(elemnt[i])
        else if (filter_tag == "")
            flex.addView(elemnt[i])
```

```
        }
    }

    fun reset_button(myIter:Int){
        for (i in 0..2) {
            if (i != myIter) {
                buttons[i].setBackgroundResource(R.drawable.shadow)
                images[i].setImageResource(icons[i + 3])
                texts[i].text = ""
                buttons[i].layoutParams.width = int_to_dp(84)
            }
        }
        buttons[myIter].setBackgroundResource(R.drawable.button_shadow)
        images[myIter].setImageResource(icons[myIter])
        buttons[myIter].startAnimation(animations[myIter])
        texts[myIter].startAnimation(animations[3])
        buttons[myIter].layoutParams.width = int_to_dp(150)
    }




    relativeLay1.setOnClickListener {
        val myIter = 0 as Int;
        texts[myIter].text = "Популярное"
        reset_button(myIter)
        texts[myIter].text = "Популярное"
        buttons[myIter].startAnimation(animations[myIter])
        texts[myIter].startAnimation(animations[3])
        buttons[myIter].layoutParams.width = int_to_dp(170)
        show_restourant(view.findViewById(R.id.flex),"")
        //Toast.makeText(this, "Its toast!", Toast.LENGTH_SHORT).show()
    }


    relativeLay2.setOnClickListener {
        val myIter = 1 as Int;
        texts[myIter].text = "Новинки"
        reset_button(myIter)
        for (i in 0..2) {
            if (i != myIter) {
                buttons[i].setBackgroundResource(R.drawable.shadow)
                images[i].setImageResource(icons[i + 3])
                texts[i].text = ""
                buttons[i].layoutParams.width = int_to_dp(84)
            }
        }

        val textViewanim = view.findViewById<TextView>(R.id.relativeBut2)
        buttons[myIter].setBackgroundResource(R.drawable.button_shadow)
        images[myIter].setImageResource(icons[myIter])
        texts[myIter].text = "Новинки"
        buttons[myIter].startAnimation(animations[myIter])
        texts[myIter].startAnimation(animations[3])
        buttons[myIter].layoutParams.width = int_to_dp(170)
        show_restourant(view.findViewById(R.id.flex),"Новинки")
        //Toast.makeText(view.context,                    "${(R.drawable.ic_dark_tort)::class.simpleName}",
Toast.LENGTH_SHORT).show()
    }



    relativeLay3.setOnClickListener {
```

```
                val myIter = 2 as Int;
                texts[myIter].text = "Акция"
                reset_button(myIter)

                for (i in 0..2) {
                    if (i != myIter) {
                        buttons[i].setBackgroundResource(R.drawable.shadow)
                        images[i].setImageResource(icons[i + 3])
                        texts[i].text = ""
                        buttons[i].layoutParams.width = int_to_dp(84)
                    }
                }

                val textViewanim = view.findViewById<TextView>(R.id.relativeBut3)
                buttons[myIter].setBackgroundResource(R.drawable.button_shadow)
                images[myIter].setImageResource(icons[myIter])
                texts[myIter].text = "Акция"
                buttons[myIter].startAnimation(animations[myIter])
                texts[myIter].startAnimation(animations[3])
                buttons[myIter].layoutParams.width = int_to_dp(170)
                show_restourant(view.findViewById(R.id.flex),"Акция")
                //Toast.makeText(view.context,                    "${(R.drawable.ic_dark_tort)::class.qualifiedName}",
Toast.LENGTH_SHORT).show()
            }




            //ПЕРВОЕ ЗАПОЛНЕНИЕ//
            for(i in 0..5){
                add_elements(R.drawable.ic_dark_tort,"Bartolomeo","Популярное","false")
                add_elements(R.drawable.kfc_logo,"KFC","Акция","false")
                add_elements(R.drawable.chel,"Chelentano","Новинки","false")
            }
            show_restourant(view.findViewById(R.id.flex),"")
            val imageSlider = view.findViewById<ImageSlider>(R.id.imageView8)

            val slideModels: MutableList<SlideModel> = ArrayList()
            slideModels.add(
                SlideModel("error")
            )
            slideModels.add(
                SlideModel("https://smachno.ua/wp-content/uploads/old_uploads/img/pg/72/11/1.jpg")
            )
            slideModels.add(
                SlideModel("https://i.obozrevatel.com/food/recipemain/2019/1/9/126.webp?size=600x400")
            )
            imageSlider.setImageList(slideModels, true)


            return view
        }
    }package com.example.myapplication

    import android.content.Intent
    import androidx.appcompat.app.AppCompatActivity
    import android.os.Bundle
    import android.view.LayoutInflater
    import android.view.View
    import android.view.ViewGroup
    import android.widget.*
    import androidx.fragment.app.Fragment
```

```
class Person_menu : Fragment(R.layout.activity_person_menu) {
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {

        val view: View = inflater.inflate(R.layout.activity_person_menu, container, false)

        val notification_sound = view.findViewById(R.id.notification_sound) as ImageView

        val rel1 = view.findViewById(R.id.rel1) as RelativeLayout
        val rel2 = view.findViewById(R.id.rel2) as RelativeLayout
        val rel3 = view.findViewById(R.id.rel3) as RelativeLayout
        val rel4 = view.findViewById(R.id.rel4) as RelativeLayout
        val rel5 = view.findViewById(R.id.rel5) as RelativeLayout

        notification_sound.setOnClickListener {
            Toast.makeText(requireContext(), "Уведомления выключены!", Toast.LENGTH_SHORT).show()
            val b = it as ImageView
            if (b.tag == "false") {
                b.setImageResource(R.drawable.bell_1)
                Toast.makeText(requireContext(), "Уведомления включены!", Toast.LENGTH_SHORT).show()
                b.tag = "true"
            } else {
                b.setImageResource(R.drawable.bell_1)
                Toast.makeText(requireContext(), "Уведомления выключены!", Toast.LENGTH_SHORT).show()
                b.tag = "false"
            }
        }

        fun outLogin(view: View) {
            activity?.let {
                val intent = Intent(it, Sign_In::class.java)
                it.startActivity(intent)
            }
        }


        fun smena(it:View){

        }


        rel1.setOnClickListener {

            val imageView: ImageView = view.findViewById(R.id.imageView18)
            imageView.setImageResource(R.drawable.gps)
            val imageView2: ImageView = view.findViewById(R.id.imageView20)
            imageView2.setImageResource(R.drawable.ic_sale)
            val imageView3: ImageView = view.findViewById(R.id.imageView22)
            imageView3.setImageResource(R.drawable.settings)
            val imageView4: ImageView = view.findViewById(R.id.imageView24)
            imageView4.setImageResource(R.drawable.email_b)
            val imageView5: ImageView = view.findViewById(R.id.imageView26)
            imageView5.setImageResource(R.drawable.faq)

        }

        rel2.setOnClickListener {
            val imageView: ImageView = view.findViewById(R.id.imageView18)
            imageView.setImageResource(R.drawable.gps_b)
            val imageView2: ImageView = view.findViewById(R.id.imageView20)
            imageView2.setImageResource(R.drawable.sales_r)
```

```
            val imageView3: ImageView = view.findViewById(R.id.imageView22)
            imageView3.setImageResource(R.drawable.settings)
            val imageView4: ImageView = view.findViewById(R.id.imageView24)
            imageView4.setImageResource(R.drawable.email_b)
            val imageView5: ImageView = view.findViewById(R.id.imageView26)
            imageView5.setImageResource(R.drawable.faq)

        }

        rel3.setOnClickListener {
            val imageView: ImageView = view.findViewById(R.id.imageView18)
            imageView.setImageResource(R.drawable.gps_b)
            val imageView2: ImageView = view.findViewById(R.id.imageView20)
            imageView2.setImageResource(R.drawable.ic_sale)
            val imageView3: ImageView = view.findViewById(R.id.imageView22)
            imageView3.setImageResource(R.drawable.settings_r)
            val imageView4: ImageView = view.findViewById(R.id.imageView24)
            imageView4.setImageResource(R.drawable.email_b)
            val imageView5: ImageView = view.findViewById(R.id.imageView26)
            imageView5.setImageResource(R.drawable.faq)
        }

        rel4.setOnClickListener {
            val imageView: ImageView = view.findViewById(R.id.imageView18)
            imageView.setImageResource(R.drawable.gps_b)
            val imageView2: ImageView = view.findViewById(R.id.imageView20)
            imageView2.setImageResource(R.drawable.ic_sale)
            val imageView3: ImageView = view.findViewById(R.id.imageView22)
            imageView3.setImageResource(R.drawable.settings)
            val imageView4: ImageView = view.findViewById(R.id.imageView24)
            imageView4.setImageResource(R.drawable.email_r)
            val imageView5: ImageView = view.findViewById(R.id.imageView26)
            imageView5.setImageResource(R.drawable.faq)
        }

        rel5.setOnClickListener {
            val imageView: ImageView = view.findViewById(R.id.imageView18)
            imageView.setImageResource(R.drawable.gps_b)
            val imageView2: ImageView = view.findViewById(R.id.imageView20)
            imageView2.setImageResource(R.drawable.ic_sale)
            val imageView3: ImageView = view.findViewById(R.id.imageView22)
            imageView3.setImageResource(R.drawable.settings)
            val imageView4: ImageView = view.findViewById(R.id.imageView24)
            imageView4.setImageResource(R.drawable.email_b)
            val imageView5: ImageView = view.findViewById(R.id.imageView26)
            imageView5.setImageResource(R.drawable.faq_r)

        }

        fun but(view: View) {

        }

        return view

    }
}package com.example.myapplication

import android.animation.ObjectAnimator
import android.content.Intent
import android.os.Bundle
import android.util.TypedValue
import android.view.View
import android.view.animation.Animation
```

```
import android.view.animation.AnimationUtils
import android.widget.ImageButton
import android.widget.ImageView
import android.widget.RelativeLayout
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import androidx.fragment.app.Fragment
import kotlinx.android.synthetic.main.activity_glavnaya1.*
import kotlinx.android.synthetic.main.activity_pinned_buttons.*


class Glavnaya1 : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_glavnaya1)
        val basketmenu=Basket_menu()
        val findmenu=Find_menu()
        val personmenu=Person_menu()
        val savedmenu=Saved_menu()
        val pinnedbuttons=PinnedButtons()
        setFragment(pinnedbuttons)
        navigation.setOnNavigationItemSelectedListener {
            when(it.itemId){
                R.id.menu->setFragment(pinnedbuttons)
                R.id.orders->setFragment(basketmenu)
                R.id.find->setFragment(findmenu)
                R.id.person->setFragment(personmenu)
                R.id.saved->setFragment(savedmenu)
            }
            true
        }
    }
    private fun setFragment(fragment: Fragment)=supportFragmentManager.beginTransaction().apply {
        replace(R.id.flFragment,fragment)
        commit()
    }

}package com.example.myapplication

import android.content.Context
import android.os.Parcel
import android.os.Parcelable
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.viewpager.widget.PagerAdapter

class MyAdapter(var layouts:IntArray, var context: Context ): PagerAdapter() {
    private lateinit var inflater: LayoutInflater
    override fun isViewFromObject(view: View, `object`: Any): Boolean {
      return view==`object`
    }

    override fun getCount(): Int {
   return layouts.size
    }

    override fun instantiateItem(container: ViewGroup, position: Int): Any {
        inflater=context.getSystemService(Context.LAYOUT_INFLATER_SERVICE) as LayoutInflater
        val v= inflater.inflate(layouts[position],container,false)
        container.addView(v)
        return  v
    }
```

```
    override fun destroyItem(container: ViewGroup, position: Int, `object`: Any) {
        val v=`object` as View
        container.removeView(v)
    }
}package com.example.myapplication

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View

class Glavnaya : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_glavnaya)
    }

    fun onClick2(view: View){
        val intent= Intent(this,Sign_In::class.java)
        startActivity(intent)
    }

    fun secondClick(view: View){
        val intent= Intent(this,Regestration::class.java)
        startActivity(intent)
    }
}package com.example.myapplication

import android.content.Intent
import android.graphics.Typeface
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.TypedValue
import android.view.View
import android.widget.*
import com.google.android.flexbox.FlexboxLayout

class Order : AppCompatActivity() {
    val elemnt: MutableList<FrameLayout> = ArrayList()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.order)
        for(i in 0..5){
            add_elements(R.drawable.dish1,"Блюдо"+i,"Кухни","false")
        }

        val flex = findViewById<FlexboxLayout>(R.id.flexorder)
        show_restourant(flex, elemnt.size-1)
    }

    fun int_to_dp(x:Int): Int {
        val value = TypedValue.applyDimension(
            TypedValue.COMPLEX_UNIT_DIP,
            x.toFloat(),
            resources.displayMetrics
        ).toInt()
        return value
    }
    fun create_blydo(img:Int,name:String, tag_name:String): FrameLayout {
        val mp = LinearLayout.LayoutParams.MATCH_PARENT
        val wc = LinearLayout.LayoutParams.WRAP_CONTENT

        val frame_layout = FrameLayout(baseContext)//Главный контейнер
        frame_layout.layoutParams = FrameLayout.LayoutParams(wc,wc)
```

```
frame_layout.setPadding(0,int_to_dp(11),0,0)
frame_layout.tag = tag_name

val frame_layout1 = FrameLayout(baseContext)//Контейнер с картинкой
frame_layout1.layoutParams = FrameLayout.LayoutParams(wc,wc)


val image_view  = ImageView(baseContext)//Картинка
var params3 = FrameLayout.LayoutParams(int_to_dp(136), int_to_dp(136))
params3.setMargins(int_to_dp(-5),0,0,0)
image_view.setImageResource(img)
image_view.layoutParams = params3


val frame_layout2 = FrameLayout(baseContext)//Блюдо
var params1 = FrameLayout.LayoutParams(int_to_dp(343), int_to_dp(104))
params1.setMargins(int_to_dp(10),int_to_dp(43),0,int_to_dp(10))
frame_layout2.layoutParams = params1
frame_layout2.setBackgroundResource(R.drawable.layout3)

var text_view3 = TextView(baseContext)//Название
var params5 = FrameLayout.LayoutParams(wc,wc)
params5.setMargins(int_to_dp(120),int_to_dp(11),0,0)
text_view3.layoutParams = params5
text_view3.text = name
text_view3.textSize = 18f
text_view3.setTypeface(Typeface.SANS_SERIF, Typeface.BOLD)
text_view3.setTextColor(getResources().getColor(R.color.textDark))
val text_view2 = TextView(baseContext)//Цена
val params4 = FrameLayout.LayoutParams(wc,wc)
params4.setMargins(int_to_dp(121),int_to_dp(50),0,0)
text_view2.layoutParams = params4
text_view2.text = "350грн"
text_view2.textSize = 12f
text_view2.setTypeface(Typeface.SANS_SERIF, Typeface.NORMAL)
text_view2.setTextColor(getResources().getColor(R.color.textSilver))
val frame_layout3 = FrameLayout(baseContext)//Кнопки
val params2=FrameLayout.LayoutParams(int_to_dp(75),int_to_dp(42))
params2.setMargins(int_to_dp(259),int_to_dp(30),int_to_dp(13),int_to_dp(20))
frame_layout3.layoutParams=params2
frame_layout3.setBackgroundResource(R.drawable.shadow)
var i=1
var text_view1 = TextView(baseContext)//Колво
var params8 = FrameLayout.LayoutParams(wc,int_to_dp(27))
params8.setMargins(int_to_dp(25),int_to_dp(10),int_to_dp(8),0)
text_view1.layoutParams = params8
text_view1.text = ""+i
text_view1.textSize = 12f
text_view1.setTypeface(Typeface.SANS_SERIF, Typeface.BOLD)
text_view1.setTextColor(getResources().getColor(R.color.textDark))
val imgb1 = ImageButton(baseContext)//Минус
val params6=FrameLayout.LayoutParams(int_to_dp(11),int_to_dp(10))
params6.setMargins(int_to_dp(5),int_to_dp(13),int_to_dp(50),int_to_dp(15))

imgb1.layoutParams=params6
imgb1.setImageResource(R.drawable.minus)
imgb1.setOnClickListener {
   if(i>0) {
      i--
      text_view1.text = "" + i
   }
   else text_view1.text=" "

}
```

```
        val imgb2 = ImageButton(baseContext)//Плюс
        val params7=FrameLayout.LayoutParams(int_to_dp(12),int_to_dp(33))
        params7.setMargins(int_to_dp(45),int_to_dp(5),int_to_dp(8),int_to_dp(10))
        imgb2.layoutParams=params7
        imgb2.setImageResource(R.drawable.plus)
        imgb2.setOnClickListener {
            i++
            text_view1.text=""+i
        }
        frame_layout3.addView(imgb1)
        frame_layout3.addView(text_view1)
        frame_layout3.addView(imgb2)
        frame_layout2.addView(frame_layout3)
        frame_layout2.addView(text_view3)
        frame_layout2.addView(text_view2)
        frame_layout1.addView(frame_layout2)
        frame_layout1.addView(image_view)


        frame_layout.addView(frame_layout1)
        return frame_layout
    }
    fun add_elements(img:Int,name:String, tag_name:String, like:String) {
        var f1: FrameLayout
        f1 = create_blydo(img, name, tag_name)
        elemnt.add(f1)
    }
    fun show_restourant(flex: FlexboxLayout, length:Int){
        if(length<=elemnt.size)
            for(i in 0..length){
                flex.addView(elemnt[i])
            }
    }


    fun back(view: View){
        val intent= Intent(this,Restaurant_menu::class.java)
        startActivity(intent)
        finish()
    }
    fun confirm(view: View){
        val intent= Intent(this,Waiting::class.java)
        startActivity(intent)
    }
    fun showMore(view: View){
        val More = view.findViewById(R.id.More) as RelativeLayout
        val flex = view.findViewById(R.id.flexorder) as FlexboxLayout
        var length = 2
        if(More.tag == "true"){
            length = elemnt.size-1
            More.tag = "false"
        }else{
            More.tag = "tag"
        }
        show_restourant(flex, length)
    }
}
```

ДОДАТОК Б

Відгук керівника економічного розділу

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота Токарева В.В.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота Токарева В.В.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Токарева В.В.rar	Архів. Містить коди програми і скомпільовану програму
Презентація	
Токарева В.В.ppt	Презентація кваліфікаційної роботи

ВІДГУК

на кваліфікаційну роботу бакалавра
на тему:
"Розробка мобільного додатка на платформі Android «Доставка їжі – Aspire»"
студентки групи 122-18ск-2 Токаревої Вікторії Валеріївни

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра
на тему:
"Розробка мобільного додатка на платформі Android «Доставка їжі – Aspire»"
студентки групи 122-18ск-2 Токаревої Вікторії Валеріївни