КОМИТЕТ ПО ОБРАЗОВАНИЮ ПРАВИТЕЛЬСТВА САНКТ-ПЕТЕРБУРГА

**САНКТ- ПЕТЕРБУРГСКОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ   
ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ   
«КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»**

Специальность 09.02.03 «Программирование в компьютерных системах»

**УТВЕРЖДАЮ**

**Зам. директора по УМР**

**\_\_\_\_\_\_\_\_\_\_\_\_ Венедиктова О.Н.**

**«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г.**

**ДИПЛОМНЫЙ ПРОЕКТ**

**Пояснительная записка**

**Тема: Разработка мобильного приложения для предприятия общественного питания**

Группа 463

Рецензент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нормоконтроль \_\_\_\_\_\_\_\_ СОГЛАСОВАНО \_\_\_\_\_\_\_\_\_\_\_ О.Н.Венедиктова

Консультант по экономической части \_\_ СОГЛАСОВАНО \_\_\_\_\_ Т.В.Львова

Руководитель дипломного проекта \_\_\_\_\_СОГЛАСОВАНО\_\_\_\_ И.П.Смирнова

Выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.В. Денисенко

Санкт-Петербург, 2020

**Содержание**

[ВВЕДЕНИЕ 4](#_Toc42175118)

[1. ОБЩАЯ ЧАСТЬ 6](#_Toc42175119)

[1.1 Предметная область 6](#_Toc42175120)

[1.2 Цель разработки 7](#_Toc42175121)

[1.3 Обзор и анализ существующих программных решений 7](#_Toc42175122)

[1.4 Анализ средств программирования 10](#_Toc42175123)

[1.5 Анализ и выбор технологий разработки 11](#_Toc42175124)

[2. СПЕЦИАЛЬНАЯ ЧАСТЬ 14](#_Toc42175125)

[2.1 Техническое задание 14](#_Toc42175126)

[2.1.1 Постановка задачи 14](#_Toc42175127)

[2.1.2 Назначение программного продукта 15](#_Toc42175128)

[2.1.3 Основание для разработки 15](#_Toc42175129)

[2.1.4 Требования к программе 16](#_Toc42175130)

[2.1.4.1 Требования к функциональным характеристикам 16](#_Toc42175131)

[2.1.4.2 Требования к аппаратным и программным средствам 17](#_Toc42175132)

[2.1.5 Требования к надежности 18](#_Toc42175133)

[2.1.6 Защита сохранности данных 18](#_Toc42175134)

[2.2 Описание функционально-логической структуры программы 19](#_Toc42175135)

[2.2.1 Контекстная диаграмма 19](#_Toc42175136)

[2.2.2 Диаграмма декомпозиции первого уровня 21](#_Toc42175137)

[2.2.3 Диаграмма вариантов использования 22](#_Toc42175138)

[2.3 Логическая модель данных 22](#_Toc42175139)

[2.4 Пользовательский интерфейс 26](#_Toc42175140)

[2.5 Тестирование и отладка 28](#_Toc42175141)

[2.6 Инструкция пользователя 35](#_Toc42175142)

[2.6.1 Вход в приложение 35](#_Toc42175143)

[2.6.2 Главная страница приложения 38](#_Toc42175144)

[2.6.3 Активность с меню 40](#_Toc42175145)

[2.6.4 Активность с корзиной 53](#_Toc42175146)

[2.6.5 Выход 59](#_Toc42175147)

[3. ЭКОНОМИЧЕСКАЯ ЧАСТЬ 62](#_Toc42175148)

[3.1 Расчёт затрат на выполнение проекта 62](#_Toc42175149)

[3.2 Расчёт расходов на заработную плату 64](#_Toc42175150)

[3.3 Расчёт электроэнергии и амортизации оборудования 64](#_Toc42175151)

[3.4 Расчет затрат на материалы 64](#_Toc42175152)

[ЗАКЛЮЧЕНИЕ 66](#_Toc42175153)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 67](#_Toc42175154)

[ПРИЛОЖЕНИЕ 1. ИСХОДНЫЙ ТЕКСТ ПРОГРАММЫ 68](#_Toc42175155)

# **ВВЕДЕНИЕ**

В наше время трудно представить себе общество без мобильных устройств. Эти устройства все чаще и чаще используются в разных сферах деятельности. Мобильные устройства сейчас используются не только как средства связи, но и как навигатор, справочники, игровые приставки и так далее. Программ для мобильных устройств все больше и совершенней, а вычислительная мощь уже приближается к персональным компьютерам. Также стоит отметить, что смартфоны с каждым годом становятся более доступными для всех слоев населения, например, сегодня можно наблюдать, что представители пожилого возраста активно пользуются смартфонами вместо обычных кнопочных телефонов. Это означает, что мобильные устройства все больше и больше распространяются на рынке.

Мобильные устройства – это достаточно перспективное направление для разработки мобильных приложений. На мобильном сегменте программного рынка заметен рост количества программных приложений, а также вакансий для разработчиков и, следовательно, самих разработчиков. На мой взгляд, успех этой сфере приносит непосредственно доступность этих самых приложений для мобильных устройств. Под новые потребности людей появляется новые приложения. Из этого следует что, создавая такие новые приложения мобильным разработчикам удается автоматизировать все больше и больше действий, тем самым облегчая жизнь людям. Кроме того, развитие мобильного интернета и сетей Wi-Fi открывает больше возможностей для мобильной разработки.

Очень полезно разрабатывать мобильные приложения, которые помогали бы людям в жизни, тем самым делая мобильные устройства по-настоящему полезными. Мобильная разработка используется для таких сфер как: спорт, развлечения, питание, музыка и так далее. На мой взгляд, если уделять внимание приложениям, которые используются в повседневной жизни, то внедрение технологий на облегчение жизни повышает уровень жизни в целом.

Старые технологии имеют определенные недостатки, и внедрение новых технологий могло бы исправить все недостатки и сделать ситуацию во многих видах деятельности лучше. Несмотря на сложность улучшения технологий, можно отметить, что за новыми технологиями будущее.

Например, можно сказать, что относительно старой технологией является телефонная связь. Сейчас ее заменяет интернет, который имеет намного больше возможностей для обмена информацией. Одним из преимуществ является контроль потока информации. Человек, использующий интернет для обмена сообщениями, может контролировать, от кого он получает информацию, какого рода и когда. С использованием более старых технологий такое контролирование потока информации является сложно достижимым.

Использование сервисов, представляющие готовые инструменты для разработки мобильных приложений, таких как Google Firebase, выводит разработку на новый уровень. Вместо того, чтобы переписывать заново одни и те же популярные решения задач, не будучи при этом полностью уверенным в отказоустойчивости и безопасности итогового продукта, разрабатывать похожие друг на друга реализации серверов, разработчик может сконцентрировать внимание на удобстве своего приложения и спать спокойно, не думая о проблемах масштабирования его проекта или о наличии каких-либо дыр в безопасности.

Возвращаясь к теме о повседневных задачах, а также учитывая то, что необходимо сделать такое приложение, которым бы пользовались как можно больше людей, то можно выбрать сферу, без которой не может жить ни один человек, а именно, питание.

Учитывая нынешнюю ситуацию в мире, а также недоступность некоторых предприятий общественного питания от дома, то возникает необходимость вызова еды на дом. Но если слишком много людей начнут использовать мобильную связь для заказа еды, то может возникнуть телефонные очереди. Поэтому необходимо найти такое решение, которое могло бы разрешить такие проблемы.

Цель дипломного проекта – создать такое мобильно приложение для ОС Android, которое бы позволяло сделать заказ на дом, без необходимости делать звонок предприятию.

Данный диплом подтверждает освоение следующих компетенции профессиональных модулей:

ПК 1.1. Выполнять разработку спецификаций отдельных компонентов;

ПК 1.2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля;

ПК 1.3 Выполнять отладку программных модулей с использованием специализированных программных средств;

ПК 1.4 Выполнять тестирование программных модулей;

ПК 1.6. Разрабатывать компоненты проектной и технической документации с использованием графических языков спецификаций.

ПМ 02. Разработка и администрирование баз данных:

ПК 2.1. Разработка объектов базы данных;

ПК 2.2 Реализация базы данных в конкретной системе управления базами данных.

ПМ 03. Участие в интеграции программных модулей:

ПК 3.1. Анализировать проектную и техническую документацию на уровне взаимодействия компонент программного обеспечения;

ПК 3.2. Выполнять интеграцию модулей в программную систему;

ПК 3.3. Выполнять отладку программного продукта с использованием специализированных программных средств.

# **ОБЩАЯ ЧАСТЬ**

В наше время развитие информационных технологий и распространённость мобильных устройств позволяет облегчить работу предприятиям общественного питания, а также сделать доставку еды более удобной для пользователя. В связи с этим цель дипломного проекта – разработать приложение для мобильных устройств на ОС Android, которое бы позволило заказать еду по указанному адресу, а при необходимости, изменить размер порции или добавить что-нибудь дополнительно к блюдам.

# **1.1 Предметная область**

Данная программа содержит следующий функционал:

* аутентификация пользователя по номеру телефона;
* отображение меню, блюд, блюд в деталях;
* функция выбора размера блюд и добавок;
* поиск по блюдам;
* функция добавления блюд в корзину;
* функция указания адреса при оформлении заказа;
* получение уведомления по завершении заказа.

В целом программа предусмотрена для использования любыми людьми, которые хотят заказать еду на дом.

# **1.2 Цель разработки**

Цель разработки в данном дипломном проекте – создать полноценное мобильное приложение под ОС Android для предприятия общественного питания, которое обладает максимально простым и понятным интерфейсом, удобной навигацией и при этом сохраняя весь необходимый функционал для осуществления доставки еды на дом.

# **1.3 Обзор и анализ существующих программных решений**

На данный момент на рынке существует следующие приложения для предприятий общественного питания:

1. Delivery club.

Из плюсов этого приложения можно вынести следующие плюсы:

* большой охват сетей питания;
* большой функционал;
* различные акции;
* возможность более детально наблюдать за заказом.

Но из минусов этого приложения можно отнести следующие:

* приложение очень сложное в реализации;
* высокая цена затрат на разработку;
* поступает очень много оповещений от приложения;
* слабая техническая поддержка.

Скриншот данного приложения представлен на рисунке 1:

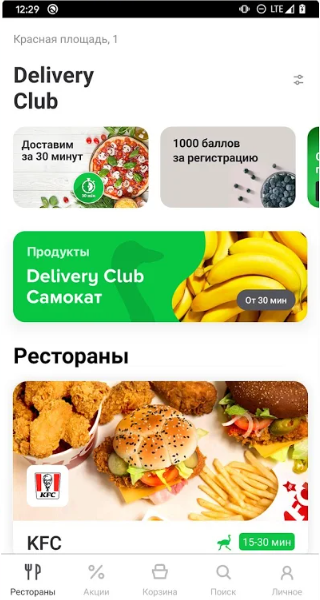


Рисунок 1 – Главная страница приложения Delivery Club

1. Dostaевский.

Из плюсов этого приложения можно вынести следующие плюсы:

* хорошо реализована функция заказа;
* удобное и наглядное приложение;
* работа приложения протекает быстро и стабильно.

Но из минусов этого приложения можно отнести следующие:

* частые ошибки в приложении;
* слабая поддержка приложения.

Скриншот данного приложения показан на рисунке 2:



Рисунок 2 – Dostаевcкий

1. ZakaZaka

Из плюсов этого приложения можно вынести следующие плюсы:

* хорошая тех. поддержка;
* возможность оплатить заказ с помощью баллов в приложении;

Но из минусов этого приложения можно отнести следующие:

* проблемы с нахождением местоположения пользователей;
* занимает много места на устройстве;
* проблема с возвращением денег за заказ.

Скриншот данного приложения показан на рисунке 3:



Рисунок 3 - ZakaZaka

# **1.4 Анализ средств программирования**

Интегрированная среда разработки – одна из самых важных частей удобной, быстрой и эффективной разработки приложений. Достаточно многие из сред разработки поддерживают разработку под ОС Android, но важно выбрать самую подходящую из них, чтобы процесс разработки доставлял как можно меньше трудностей.

В разработке мобильных приложений для мобильных устройств на базе ОС Android популярными являются следующие решения:

* Microsoft Visual Studio – имеет широкий инструментарий для разработки, но имеет недостаток: в этой среде запрещена разработка коммерческих проектов без приобретения лицензионной версии продукта, в то время как на рынке программного обеспечения для разработчиков присутствуют другие;
* Eclipse – среда разработки с бесплатной лицензией для многих языков программирования и платформ. До 2014 года поддерживалась Google как основная среда разработки под Android;
* Android Studio — интегрированная среда разработки производства Google, с помощью которой разработчикам становятся доступны инструменты для создания приложений на платформе Android OS. Android Studio можно установить на Windows, Mac и Linux Android Studio создавалась на базе IntelliJ IDEA. IDE можно загрузить и пользоваться бесплатно. В ней присутствуют макеты для создания UI, с чего обычно начинается работа над приложением. В Studio содержатся инструменты для разработки решений для смартфонов и планшетов, а также новые технологические решения. Среда Android Studio предназначена как для небольших команд разработчиков мобильных приложений (даже в количестве одного человека), или же крупных международных организаций с GIT или другими подобными системами управления версиями. Решения для ОС Android разрабатываются в Android Studio с использованием Java или C++. В основе рабочего процесса Android Studio заложен концепт непрерывной интеграции, позволяющий сразу же обнаруживать имеющиеся проблемы. Продолжительная проверка кода обеспечивает возможность эффективной обратной связи с разработчиками. Такая опция позволяет быстрее опубликовать версию мобильного приложения в Google Play App Store. Android Studio совместима с платформой Google App Engine для быстрой интеграции в облаке новых API и функций. В среде разработки вы найдете различные API, такие как Google Play, Android Pay и Health. Присутствует поддержка всех платформ ОС Android, начиная с Android 1.6. Есть варианты ОС Android, которые существенно отличаются от версии Google Android.

# **1.5 Анализ и выбор технологий разработки**

После выбора среды разработки необходимо подумать, какой язык программирования может использоваться для разработки приложений для мобильных устройств. В выбранной среде разработки, а именно, Android Studio поддерживаются следующие языки программирования:

* C++, который, в свою очередь, не особо подходит для разработки мобильных приложений, так как на нем пишутся только некоторые части приложения, например библиотеки. Это говорится в официальной документации к Android Studio. Очень немногие разработчики используют его для разработки своих приложений.
* Kotlin - это относительно молодой язык от российской компании JetBrains. На конференции Google I/O 2017 команда разработчиков ОС Android сообщила, что Kotlin получил официальную поддержку для разработки Android-приложений. Как и Java, C++, Kotlin — это статически типизированный язык. Он поддерживает как объектно-ориентированное, так и процедурное программирование. Но в большинстве своем позиционируется как модификация для Java. Хорошо и то, что он может интерпретироваться в обе стороны и имеет более удобный синтаксис. В последнее время все больше и больше разработчиков используют этот язык как основной язык для разработки мобильных приложений.
* Java – самый популярный язык программирования для создания мобильных приложений. Этот язык не сложен в освоении, безопасен и имеет наибольшую поддержку сообщества при возникновении проблем с кодом. К тому же Android API очень похоже на API языка Java, и ОС Android поддерживает если не все доступные в J2SE SDK классы, то, по крайней мере, наиболее важные. Именно на этом языке программирует большая часть мобильных разработчиков.

Для разработки приложения был выбран язык программирования Java. Этот выбор обоснован, как и общей привлекательностью языка, так и личным опытом программирования. Во внимание стоит учесть еще и доступное количество руководств по разработке на Java за все время существования Android ОС.

Далее необходимо выбрать инструмент для удобной, а главное, безопасной аутентификации пользователей в системе, ведь обеспечение целостности системы, а также предоставление нужного доступа определённым пользователям, является одной из важнейших задач для построения хорошего приложения.

Для реализации безопасной, простой и быстрой аутентификации пользователей в системе было решено использовать набор инструментов от компании Google – Firebase. Этот набор удобно интегрирован с выбранной средой разработки Android Studio, что делает его использование простым и комфортным, а также предоставляет множество полезных средств для разработки современных и безопасных приложений. Кроме того, Firebase предоставляет подробную статистику созданных приложений, а также имеет огромные возможности для расширения приложения как с точки зрения масштаба, так и с точки зрения сложности программы.

Таким образом, в качестве инструмента аутентификации был выбран инструмент Firebase Authentication. Он имеет разные виды аутентификации на выбор: от связки email/пароль до входа через аккаунт Google Play. Также он содержит способ аутентификации через подтверждение входа в систему кодом, передающимся через SMS-сообщение на номер, который был введён в форме входа в приложение. Было решено, что данный способ аутентификации является самым удобным и безопасным вариантом для данного приложения исходя из лёгкости в использовании благодаря поддержке разработчиков компанией Google, которая предоставляет подробные руководства по использованию этих инструментов, а также из проверенной многими разработчиками надёжности этого инструмента.

Кроме всего прочего, набор Firebase содержит компонент Realtime Database. Это облачная система управления базами данных модели NoSQL, то есть в её основе лежит не таблица, как в реляционных СУБД, а файл формата JSON. Преимущество такой модели в гибкости: при грамотном построении структуры такой базы данных, чтение и изменение сведений в базе данных производятся очень просто. К тому же, хранение данных в ней не в виде таблицы, а в виде вложенных объектов и свойств, избавляет от необходимости придерживаться строгой структуры на этапе разработки, добавляет гибкости в реализации различных решений, а также даёт возможность хранить данные в любой удобной форме. Эта база данных позволяет только аутентифицированным пользователям взаимодействовать с базой данных, а данные разрабатываемого приложения будут защищены от несанкционированного доступа. Также она поддерживает создание правил доступа к данным, а пользователи не смогут совершать те действия, на которые у них нет прав. Но самая главная особенность данного инструмента описана в названии данного продукта: Realtime Database позволяет всем клиентам без усилий синхронизировать данные между собой в реальном времени, что делает разрабатываемое приложение удобным и отзывчивым, ведь пользователю не нужно заботиться о том, что данные в его приложении устарели – он получит все изменения через мгновения после того, как они были сделаны при условии, с его сетевым соединением всё в порядке. Все данные между базой данных и приложением передаются через программные запросы, а также «слушатели», роль которых – обновлять информацию, как только произошли какие-либо изменения в базе данных. Также Realtime Database автоматически кэширует данные, поэтому если произойдёт отсоединение от интернета, ничего страшного не случится.

В итоге, аутентификация в приложении происходит следующим образом: зарегистрированный пользователь вводит номер телефона в поле формы входа в программу, и если номер введён верно, то на этот номер отсылается сообщение с шестизначным цифровым кодом для подтверждения личности по номеру мобильного телефона, который пользователь должен ввести в изменившуюся форму входа. После этого, если код, введённый пользователем, оказался верным, в приложении появляется активный пользователь. Открывается главная активность, где пользователь может увидеть информацию о приложении, а также подсказку для продолжения использования приложением. После следования подсказке открывается меню, которое содержит различные фрагменты. После выбора определенного фрагмента, приложение делает запрос к базе данных для того, чтобы прогрузить меню и список блюд для отображения пользователю на фрагментах.

# **СПЕЦИАЛЬНАЯ ЧАСТЬ**

В данной части будут сформулированы различные требования к разрабатываемому программному продукту, его функциональным, аппаратным и программным характеристикам, описывается структура программы, используемые данные и её алгоритм, а также производится оценка результатов решения программой поставленных задач.

# **2.1 Техническое задание**

Целью проекта является разработка мобильного приложения для предприятий общественного питания, для людей, которые хотят заказать еду по адресу, которое позволит просмотреть меню, добавить желаемые блюда в корзину, указать адрес и получить уведомление о статусе заказа.

# **2.1.1 Постановка задачи**

Создать многофункциональное быстродействующее мобильное приложение, отвечающее следующим требованиям:

* формирование списков блюд;
* поиск по блюдам;
* возможность изменения количества, размера блюд;
* возможность добавить добавок к блюду;
* управление корзиной;
* формирование уведомлений, используя введённую информацию;
* возможность выхода из учетной записи.

Все данные синхронизированы с базой данных. Должны быть обработаны исключительные ситуации, чтобы избежать неожиданных ошибок во время исполнения, приводящих к «вылету» приложения. Цена должна рассчитываться с учетом выбранного количества, размера блюд, а также должен производится учет цены добавок. Пользователь должен иметь возможность удаления блюд из корзины.

# **2.1.2 Назначение программного продукта**

Программа предназначается для пользователей, которые желают заказать еду из азиатской кухни с доставкой по указанному адресу. Она должна решить задачу по облегчению работы предприятия общественного питания, а в связи с нынешней ситуацией в стране, то должна максимально полно предоставлять информацию о предприятии, тем самым исключая лишние посещения общественных мест.

Целью создания данного проекта является повышение доступности еды, разгрузка в потребности операторов, а также облегчение работы для предприятия общественного питания.

# **2.1.3 Основание для разработки**

В ходе изучения рынка с целью поиска аналогичных приложений, было найдено множество приложений, специализирующихся на доставке еды на дом. Но все они обладают достаточно большим функционалом и продвинутым интерфейсом. Исходя из этого, можно сказать, что и цена реализации этих приложений достаточно высокая. Поэтому, было принято создать мобильное приложения для предприятия общественного питания, которое обладает необходимым функционалом для данного рода приложений, но и очень просто в использовании путем максимально простой навигации и понятного интерфейса.

# **2.1.4 Требования к программе**

Приложение должно включать в себя простой и интуитивно-понятный интерфейс, должно быть достаточно надежным и удобным в использовании. Но самое главное, продукт должен выполнять поставленные перед ним задачи.

Кроме того, приложение не должно допускать утечки каких–либо персональных данных пользователей.

# **2.1.4.1 Требования к функциональным характеристикам**

Приложение должно включать в себя следующие сведения:

* данные о пользователях, зарегистрированных в системе. Хранимые данные пользователя должны быть представлены следующим образом:
* номер телефона;
* имя пользователя.

Также приложение должно обрабатывать все данные из базы данных и предоставлять их пользователям в удобном виде. Программа должна выполнять следующие функции:

* отображение данных из базы данных;
* поиск по данным из базы данных;
* иметь корзину;
* возможность управления корзиной;
* возможность указать адрес получения заказа;
* отправка уведомления о статусе заказа;
* возможность изменять размер блюда;
* возможность добавления добавок к блюду;
* возможность рассчитать цену с учетом размера и добавок;
* возможность выхода из аккаунта.

Интерфейс приложения должен отвечать следующим требованиям:

* быть интуитивно-понятным для людей, которые далеки от информационных технологий;
* не допускать некорректного поведения элементов приложения на экране мобильного устройства.

# **2.1.4.2 Требования к аппаратным и программным средствам**

Мобильное приложение должно иметь следующие минимальные требования к аппаратным и программным средствам:

* Android API 19(Android 4.4);
* подключение к сети Интернет (Wi-Fi или мобильная сеть);
* установленное приложение Google Play;
* установленное приложение Google Play Services.

Данные требования к аппаратным и программным средствам сформированы исходя из официальной статистики компании Google – поставщика ОС Android – в которой указано, что поддержка версии Android 19+ соответствует поддержке 98,1% мобильных устройств (рисунок 4). Таким образом, был достигнут максимальный потенциал функционала операционный системы и охвата аудитории приложения.

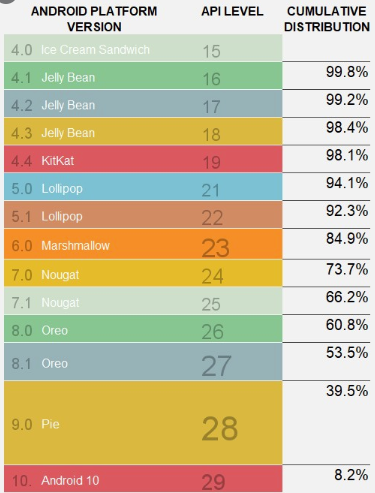


Рисунок 4 – Распределение версий ОС Android

# **2.1.5 Требования к надежности**

При продолжительном использовании приложения в нём не должно происходить сбоев, утечек памяти и процессорного времени. Приложение не должно неожиданно закрываться от действий или без действий пользователя. Все исключительные ситуации должны быть обработаны соответствующим образом, а само приложение должно быть тщательно протестировано. База данных должна быть защищена от несанкционированного доступа. Приложение также должно не допускать превышения привилегий пользователей.

Приложение должно иметь техническую поддержку. Это значит, что при обнаружении неисправности или нестандартного поведения программы, ситуация, в которой обнаружена неисправность, должна быть расследована, причина сбоя в работе программы должна быть обнаружена, а затем должно быть найдено решение, которое устранит все обнаруженные в ходе расследования неисправности, произведено внедрение изменений и повторное тестирование для подтверждения успешности произведённых действий. Необходимо поддерживать такой цикл в течение всего периода разработки. Рекомендуется, чтобы данные действия выполнялись и на стадии эксплуатации программы для того, чтобы минимизировать количество сбоев программы в случаях, не обнаруженных на стадии разработки программного продукта.

# **2.1.6 Защита сохранности данных**

Как уже упоминалось ранее, необходимо обеспечить надежную защиту данным пользователя. В используемой системе Firebase существует следующее требование:

В соответствии с требованиями Генерального регламента ЕС о защите персональных данных (GDPR), разработчик, который осуществляет масштабный сбор или обработку пользовательских данных, ведет сбор или обработку определенных категорий персональных данных либо является органом государственной власти, должен назначить сотрудника, ответственного за защиту данных, или представителя в ЕС.

Это значит, что при интенсивном внедрении программного продукта в сферу здравоохранения необходимо назначить ответственного за сохранность персональных данных.

Также при разработке приложения должны быть замечены и устранены любые сбои, которые могут привести к утечке персональных данных или несанкционированному доступу к ним.

В дополнение к вышеуказанным мерам является желательной минимизация количества чувствительной информации пользователей, хранящейся в СУБД Realtime Database и используемой в приложении, сохраняя при этом логику работы программы и принимая во внимания требования к функционалу разрабатываемого мобильного приложения.

# **2.2 Описание функционально-логической структуры программы**

Чтобы построить функционально структуру будущей программы, воспользуемся приёмами функционального моделирования. В данном проекте было решено использовать нотацию IDEF0. Это графическая нотация, предназначенная для формализации бизнес-процессов, а также их описания. В методологии IDEF0 существует два главных правила:

1. Наиболее важный модуль находится в левом верхнем углу диаграммы.
2. Правило стороны:

* стрелка входа приходит всегда в левую кромку активности;
* стрелка управления – в верхнюю кромку;
* стрелка механизма исходит из нижней кромки;
* стрелка выхода исходит из правой кромки.

Для функционального описания данного программного продукта были составлены две диаграммы на основе нотации IDEF0: контекстная диаграмма и диаграмма декомпозиции первого уровня.

# **2.2.1 Контекстная диаграмма**

Данная диаграмма представляет из себя единственный блок процесса со стрелками, показывающими отношения объекта моделирования с окружающей средой. Эта диаграмма нужна для того, чтобы представить общий вид объекта: входные ресурсы, правила управления, механизм управления и выходные ресурсы. Приложение в данном случае представляется в виде «черного ящика», внутренние процессы которого скрыты, но при необходимости доступны при декомпозиции. Контекстная диаграмма для мобильного приложения, разрабатываемого в данном дипломном проекте представлена на рисунке 5.

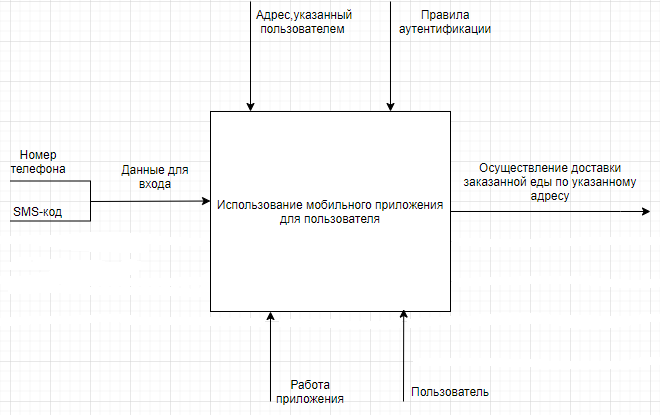
****

Рисунок 5 – Контекстная диаграмма приложения

Из данной диаграммы можно увидеть общий функциональный вид приложения: в качестве механизмов управления в приложении выступает «Работа приложения», что включает в себя любые действия, выполняемые самим приложением, включая работу базы данных, инструмент аутентификации и т.д., в качестве пользователя приложением выступает сам Пользователь. В качестве входных данных в приложении используются номер телефона и SMS-код, которые используются как данные, необходимые для аутентификации пользователя. Основными элементами управления и контроля являются правила аутентификации (разные условия, определяющие то, какие пользователи могут войти в систему приложения), а также адрес, указанный пользователем при оформлении заказа. Это свойство очень важно за счет того, что оно определяет, куда именно нужно доставить заказ, ведь без наличия адреса доставка еды по адресу становится проблематичной. Таким образом, адрес является важным организующим фактором в приложении. Наконец, выходным ресурсом приложения является осуществление доставки заказанной еды по указанному адресу, ведь это, по большому счёту, и является целью всех манипуляций данными внутри приложения.

# **2.2.2 Диаграмма декомпозиции первого уровня**

Эта диаграмма является более детальным взглядом на общий вид – контекстную диаграмму приложения. В этой диаграмме уже видны внутренние процессы, которые прежде были недоступны. Эта диаграмма отражает основную логику работы приложения, поэтому очень важно продумать все компоненты, из которых она состоит, а также связи между ними. При взгляде на эту диаграмму будет заметно, что она сложнее, чем первая, и состоит из многих компонентов, представляющих из себя функции, соединённых между собой различными видами связей, которые определяются правилом, как и в контекстной диаграмме, правилом стороны. По сути, каждая из функций, представленных на диаграмме декомпозиции, может быть представлена в виде контекстной диаграммы, как самостоятельный процесс. С другой стороны, каждая функция может быть разбита на подфункции. Диаграмма декомпозиции первого уровня нужна в данном случае, чтобы достаточно хорошо представить логику работы мобильного приложения, не погружаясь при этом слишком глубоко в детали. Она представлена на следующем рисунке 6.

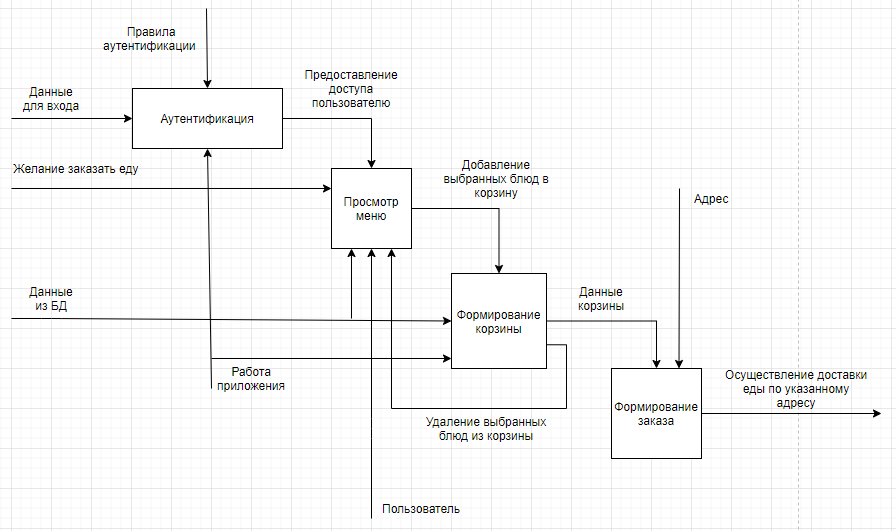
****

Рисунок 6 – Диаграмма декомпозиции 1-го уровня

Как видно из рисунка, эта диаграмма даёт намного более широкое представление о логике работы проектируемой программы. При этом, связи, которые были представлены на контекстной диаграмме, остались на месте, но теперь они имеют более важные роли, являясь теперь связующими звеньями совершенно разных компонентов, и находясь в разных частях диаграммы. Следует заметить, что вся эта диаграмма является лишь более детальным представлением процесса «Использование мобильного приложения для пользователя», которое до этого было единственным блоком процесса на всей диаграмме, представляясь теперь в виде логической цепочки.

# **2.2.3 Диаграмма вариантов использования**

Далее была построена диаграмма вариантов использования, в которой представляется совокупность функциональных требований к поведению программного продукта. На этих диаграммах представляются так называемые *актёры*, которые имеют некоторые отношения с *прецедентами*. Если говорить проще, то это моделирование использования разных возможностей приложения различными категориями пользователей. Главная вещь, которую даёт разработчику работа с данной диаграммной, это знание о том, какие возможности в приложении переиспользуются различными группами пользователей несколько раз, чтобы далее, на этапе разработки приложения, разработчик не усложнил программу, реализовав какой-либо функционал продукта больше одного раза. В данном случае будет использвована только одна категория – пользователь. Если представить все возможности, которые должны быть реализованы в текущем проекте, получится картина, представленная на рисунке 7.

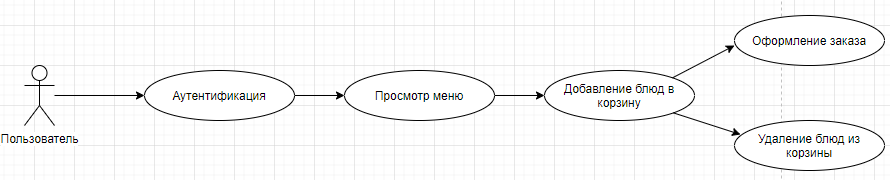


Рисунок 7 – Диаграмма вариантов использования

# **2.3 Логическая модель данных**

Так как на стадии выбора инструментов разработки было принято решение использовать СУБД Firebase Realtime Database, которая по своему классу является NoSQL базой данных объектно-ориентированной модели данных, для неё не представляется возможным представить логическую модель данных в виде диаграммы связей между таблицами базы данных. В связи с этим, в данном разделе будут приведены объяснения того, как хранятся, записываются и используются данные в выбранной базе данных.

Итак, данная СУБД хранит данные в виде файла формата JSON и предоставляет распределённый синхронизированный доступ к этим данным. Смеет возможность динамически изменять данные в базе данных, находясь при это. Этот вид хранения данных поддерживает следующие типы данных:

* строковый;
* целочисленный;
* числа с плавающей точкой;
* булевой;
* ассоциативный массив (хранящий записи в виде: строка – произвольный объект);
* список (хранящий произвольные объекты);
* произвольный объект.

Произвольный объект в данном контексте – это объект, содержащий любую комбинацию типов полей, а значит, разработчик может сам определять, какие данные ему нужны для хранения, без труда создать такой класс и оперировать экземплярами этого класса.

Было принято решение использовать следующие данные для хранения в базе данных. Для пользователей:

* имя;
* номер телефона;
* id пользователя.

Для категорий меню:

* список блюд;
* картинка с категорией блюд;
* название категории.

Для блюд:

* добавки;
* размер;
* описание;
* id блюда;
* изображение блюда;
* название блюда;
* цена блюда.

Для добавок:

* название добавки;
* цена.

Для размера:

* название размера;
* цена за размер.

Также для хранения товаров в корзине была сделана база данных «Корзина».

Инструментом разработки для создания использовалась SQLite. С помощью этой компактно встраиваемой СУБД удалось реализовать механизм добавления товаров в корзину с помощью запросов SQL. Также для упрощения создания этой базы данных использовалась библиотека Room от Google, которая представляет из себя высокоуровневый интерфейс для низкоуровневых привязок SQLite, встроенных в Android. Она выполняет большую часть своей работы во время компиляции, создавая API-интерфейс поверх встроенного SQLite API, исключая при это работу с Cursor или ContentResolver.

Итак, с помощью всех этих инструментов разработки была разработана база данных «Корзина».

Для хранения были использованы следующие данные:

* id блюда;
* название блюда;
* изображения блюда;
* цена блюда;
* количество;
* номер телефона пользователя;
* цена с учетом добавок и размера;
* добавки;
* размер;
* id пользователя.

Было решено, что в остальных случаях будет проще делать другие запросы, чем переделывать структуру базы данных. Поэтому для хранения записей на данный момент выбраны следующие модели данных (рисунки 8-13).

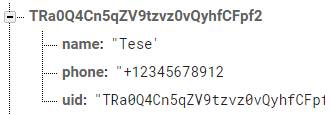


Рисунок 8 – Модель хранения пользователей в базе данных



Рисунок 9 – Модель хранения категорий меню в базе данных

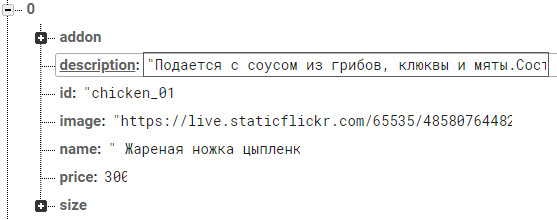


Рисунок 10 – Модель хранения блюд в базе данных

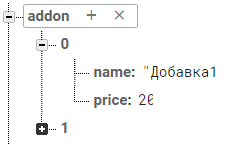


Рисунок 11 – Модель хранения добавок в базе данных

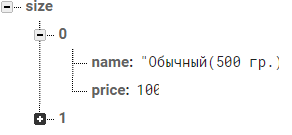


Рисунок 12 – Модель хранения размеров в базе данных

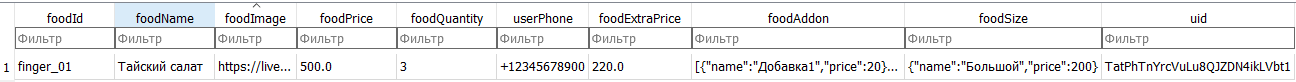


Рисунок 13 – Модель хранения данных в корзине

Все остальные второстепенные цели добиваются без дополнительных изменений в базе данных. Никаких связей, кроме иерархических, в этих объектах нет, все данные получаются из БД только с помощью правильных выражений. Доступ к данным пользователя происходит через номер мобильного телефона и id, так как это свойство у пользователей уникально внутри проекта, а также из-за использования номера телефона пользователя при аутентификации, после которой открывается домашняя страница, а на этот момент, у пользователя нет никаких данных, кроме введенного телефона и id.

# **2.4 Пользовательский интерфейс**

Пользовательский интерфейс – это средство, позволяющее пользователю человеку получать информацию от программы и взаимодействовать с ней. Чем удобнее и понятней пользовательский интерфейс программы, тем легче взаимодействовать с программой через него, поэтому при разработке приложения также необходимо уделять достаточно внимания на пользовательский интерфейс.

Пользовательский интерфейс программы на ОС Android представляет активности, которые сменяют друг друга. По своей сути активности это некое подобие окон. Для максимально простого и приятного опыта использования приложением необходимо обеспечить плавный переход между активностями, а сами экраны активностей должны быть разработаны с точки зрения удобства использования с применением инструментов Android Studio для создания и разметки активностей Android-приложений, а также основных правил дизайна. Кроме того, приложение должно иметь быстрый и синхронизированный поток данных.

Также, для упрощения использования приложения пользователям, были добавлены функции поиска блюд на экране списка блюд и полное очищение корзины на экране корзины.

Важно следить за тем, чтобы приложение занимало как можно меньше места на устройстве, ведь нехватка памяти это одна из самых частых проблем современных Android-пользователей. Поэтому, было принято решение достичь максимального функционала приложения путем использования не сильно требовательных инструментов, но достаточно эффективных для выполнения поставленной задачи.

В начале были разработаны разметки активностей приложения. Эти разметки представляют собой файл с XML разрешением, в котором описывается положение элементов интерфейса на экране с активностью. Программная часть активностей находится в файлах с разрешением .java.

Список активностей:

* activity\_main.xml, активность входа пользователя в систему, используя аутентификацию с помощью SMS-сообщения с шестизначным кодом;
* activity\_home.xml, активность с главной страницей приложения, содержащее информацию о приложении и подсказкой для дальнейшего продолжения использования приложением;
* layout\_register.xml, активность для регистрации пользователя в системе, которая содержит поля для ввода информации и кнопку регистрации;
* layout\_order.xml, активность для оформления заказа, которая содержит поле для ввода адреса, а также дополнительную информацию для пользователя, кнопки отмены и подтверждения.

Список фрагментов:

* fragment\_cart.xml, фрагмент корзины, который представляет из себя разметку для корзины, который содержит список с вложенными фрагментами блюд в корзине;
* layout\_cart\_item.xml, фрагмент корзины, в котором отображается блюдо в корзине;
* fragment\_food\_list.xml, фрагмент, содержащий список блюд из выбранной категории в меню;
* fragment\_food\_detail.xml, фрагмент, представляющий из себя подробное представление выбранного блюда;
* fragment\_menu.xml, фрагмент, содержащий список категорий меню;
* layout\_menu\_item.xml, разметка для элемента в категории меню;
* layout\_food\_item.xml, разметка для элементов в выбранной категории меню.

Для отображения данных в разных активностях используются различные инструменты и технологии. В активности для отображения категорий меню, блюд в выбранной категории в виде списка используются элемент RecylerView, который удобен тем, что позволяет отображать список элементов в виде, который определен разработчиком с возможностью прокрутки. При этом, если нажать на элемент в списке блюд, то откроется детальное представление блюда. Самая интересная особенность приложения, это меню с элементами. При нажатии на элемент в меню открывается активность с соответствующим элементом.

Демонстрация работы программы:

Главная функция программы – доставка выбранной еды по указанному адресу, и во время создания программы была приложена масса сил, чтобы сделать этот процесс максимально быстрым и простым для пользователя.

Базовая демонстрация на тестовых учетных записях:

1. Вход в аккаунт пользователя.
2. Переход на главную страницу -> Свайп вправо -> Выбор вкладки «Меню» в меню.
3. Выбор категории меню -> Выбор блюда -> Добавление в корзину.
4. Переход в корзину -> Нажатие на кнопку «Оформить заказ».
5. Ввод адреса -> Нажатие на кнопку «Оформить» -> Получение уведомления о статусе заказа.

# **2.5 Тестирование и отладка**

Отладка – это работа, направленная на обнаружение ошибок в определённом месте работы программы в режиме отладчика, то есть когда программа исполняется по шагам или приостанавливается при определённых условиях.

Тестирование – испытание программного продукта для проверки соответствия ожидаемого и реального её поведения на конечном наборе тестов.

Далее будут описаны шаги для теста и к ним будут приложены соответствующие снимки экрана:

1. В поле ввода телефона введите +12345678900 как показано на рисунке 14.

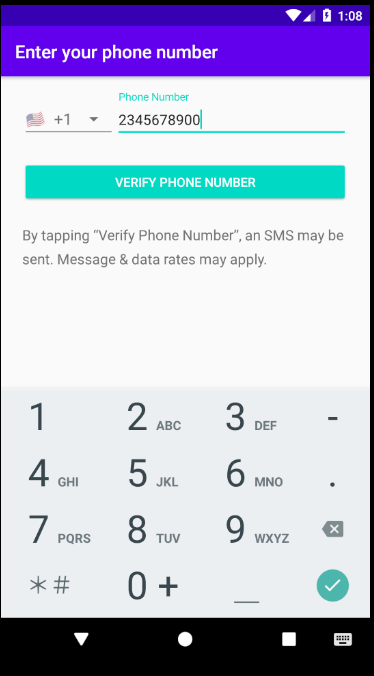


Рисунок 14 – Ввод номера телефона

1. Нажмите ОК.
2. Введите код 123456, как показано на рисунке 15.

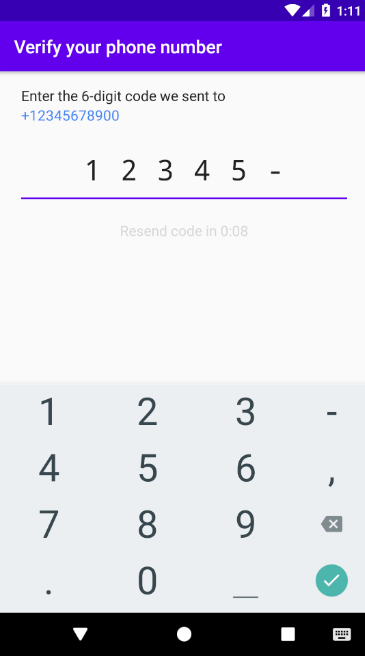


Рисунок 15 – Ввод кода

1. Сделайте свайп вправо, следуя подсказке на главной странице.
2. Нажмите на вкладку «Меню» как показано на рисунке 16.

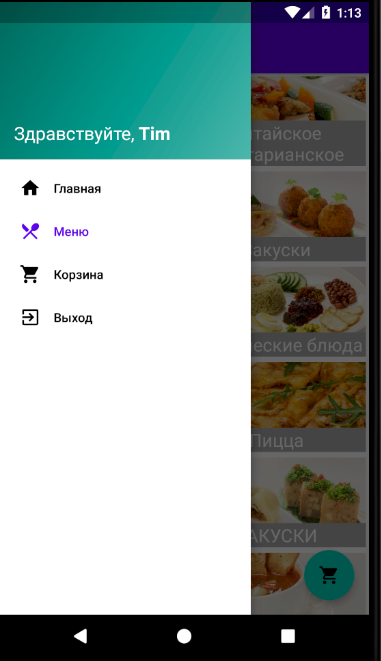


Рисунок 16 – Нажатие на вкладку «Меню»

1. Нажмите на категорию в меню.
2. Нажмите на значок корзины, чтобы добавить блюдо в меню.
3. Нажмите на значок корзины справа снизу экрана.
4. Нажмите на кнопку «оформить заказ».
5. Введите адрес как показано на рисунке 17.

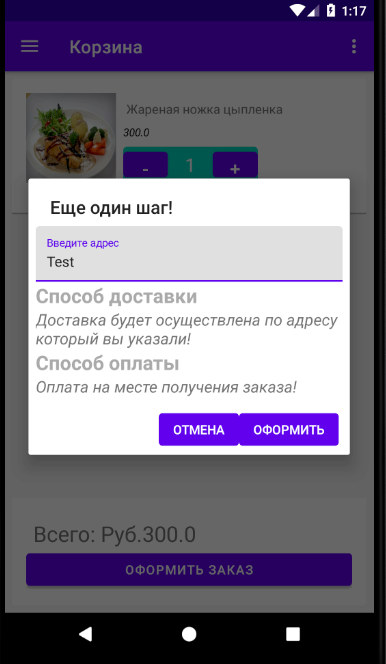


Рисунок 17 – Ввод адреса

1. Нажмите на кнопу «Оформить».
2. Откройте панель уведомлений как показано на рисунке 18.

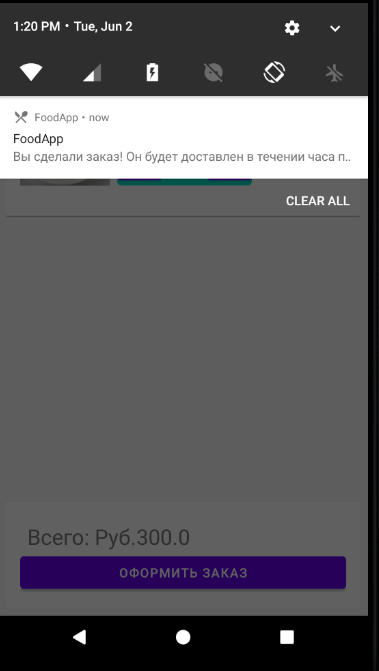


Рисунок 18 – Получение уведомления о статусе заказа

Если программный продукт проходит этот тест, значит основной функционал приложения работает исправно.

Также рекомендуется проделать данные шаги на реальных физических девайсах.

Также стоит отметить, что если приложение будет запущено при другой конфигурации, например, с более большим разрешением экрана, то элементы будут отображаться корректно, независимо от конфигурации экрана. Примеры работы представлены на рисунке 19-23.

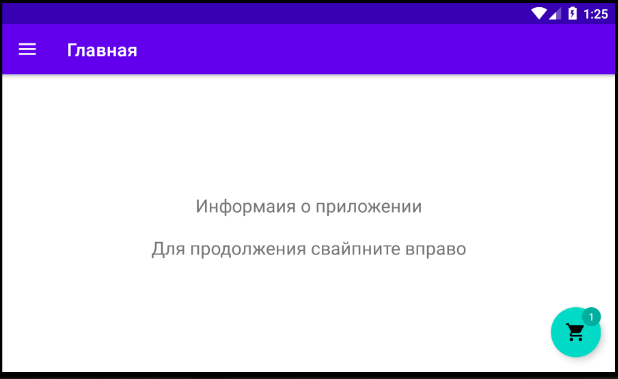


Рисунок 19 – Отображение главной страницы для другого разрешения экрана

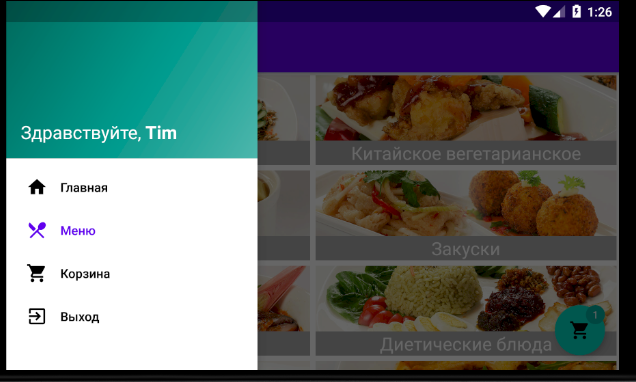


Рисунок 20 – Отображение Меню для другого разрешения экрана

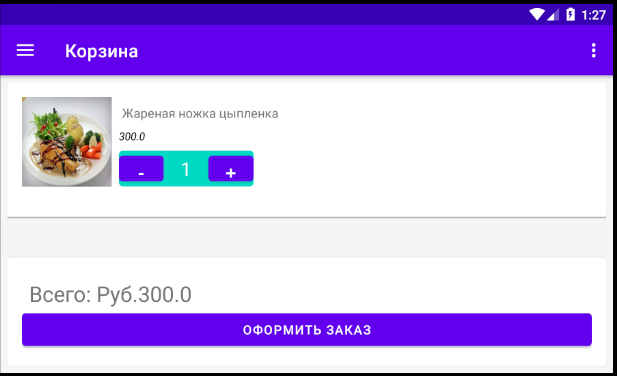


Рисунок 21 – Отображение корзины для другого разрешения экрана

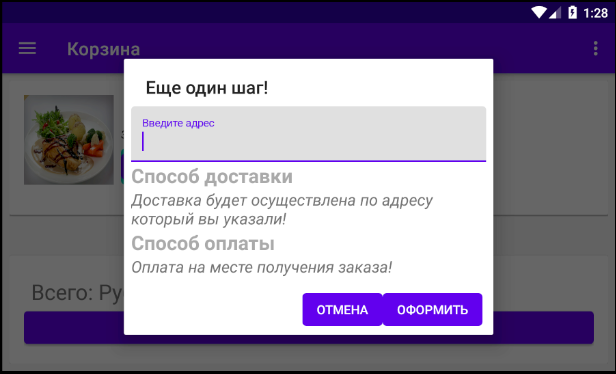


Рисунок 22 – Отображение окна оформления заказа для другого разрешения экрана

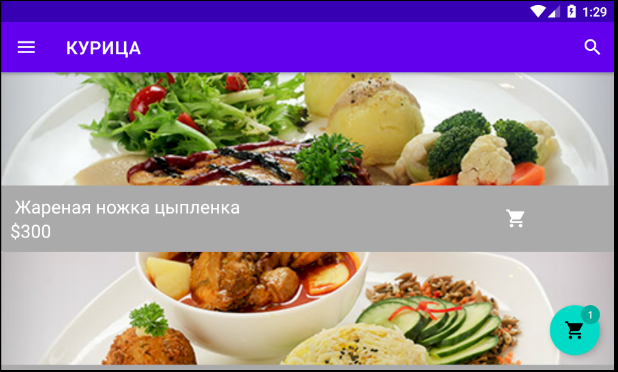


Рисунок 23 – Отображение списка блюд для другого разрешения экрана

# 

# **2.6 Инструкция пользователя**

# **2.6.1 Вход в приложение**

Для запуска программы нажмите на ярлык приложения FoodApp.

После запуска вас встречает форма входа в систему (рисунок 24). Для того, чтобы войти в систему, вы должны быть зарегистрированы в базе данных в качестве пользователя.

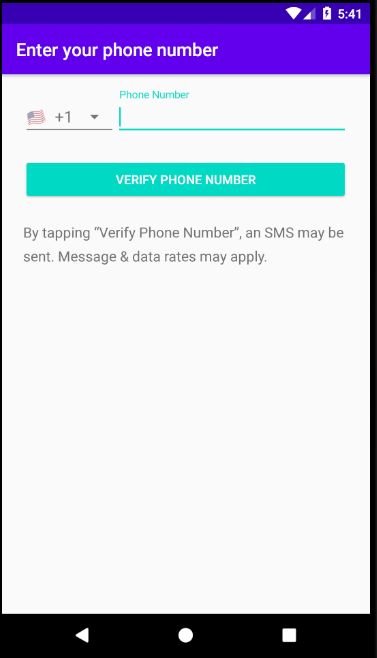


Рисунок 24 – Форма входа

Введите ваш номер телефона в поле ввода телефона, а затем нажмите на кнопку отправки сообщения. Если введённый номер является валидным, то на него будет отправлено сообщение с защитным кодом. При это происходит смена вида форма, то есть вместо поля ввода телефона будет отображаться поле ввода кода (рисунок 25).

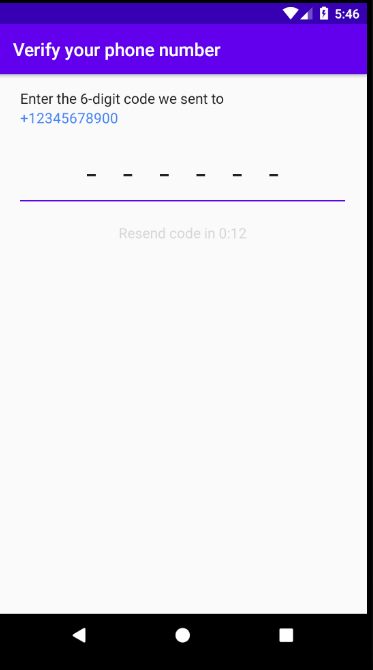


Рисунок 25 – Форма входа после отправленного сообщения

Дождитесь получения SMS-сообщения, отправленного на введённый номер. После этого введите полученный код в поле ввода кода и нажмите на кнопку входа. Если введённый номер корректен и пользователь зарегистрирован в базе данных, то пользователю отображается активность с главной страницей приложения.

Если же вы не зарегистрированы в базе данных, то после ввода кода выводится окно регистрации (рисунок 26), которое содержит поля для заполнения имени и телефона.

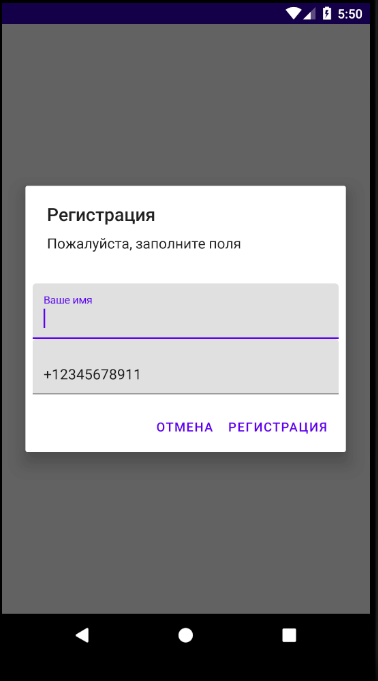


Рисунок 26 – Окно регистрации

После прохождения регистрации пользователю отображается активность с главной страницей приложения. В дальнейшем текущему пользователю достаточно ввести код, после чего сразу откроется активность с главной страницей приложения.

# **2.6.2 Главная страница приложения**

После ввода данных в форму входа, пользователю отображается активность, содержащая главную страницу приложения. Её внешний вид представлен на рисунке 27. Эта активность содержит информацию о приложении и подсказку.

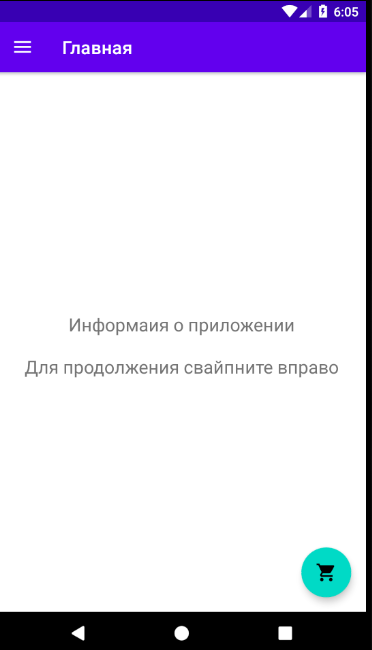


Рисунок 27 – Главная страница приложения

После свайпа вправо или нажатие на кнопку справа сверху открывается меню (рисунок 28), которое содержит следующие вкладки:

* главная;
* меню;
* корзина;
* выход.

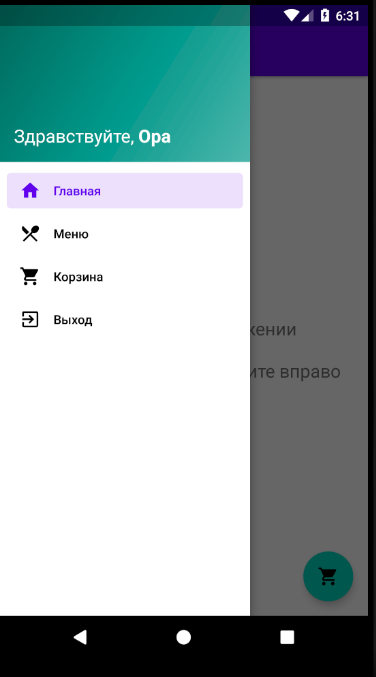


Рисунок 28 - Меню

# **2.6.3 Активность с меню**

На данной активности располагаются категории блюд в меню (рисунок 29).



Рисунок 29 – Активность с меню

При нажатии на категорию происходит открытие новой активности, содержащую список блюд в данной категории (рисунок 30), которые отсортированы в цене от наименьшей стоимости к наибольшей. Также на этой активности есть кнопка поиска по блюдам для более удобной навигации.



Рисунок 30 – Активность со списком блюд в выбранной категории

Нажав на кнопку поиска, вы можете ввести в поле нужное вам блюдо, если нужное вам блюдо есть в базе данных, то вы увидите результат вашего поиска (рисунок 31).

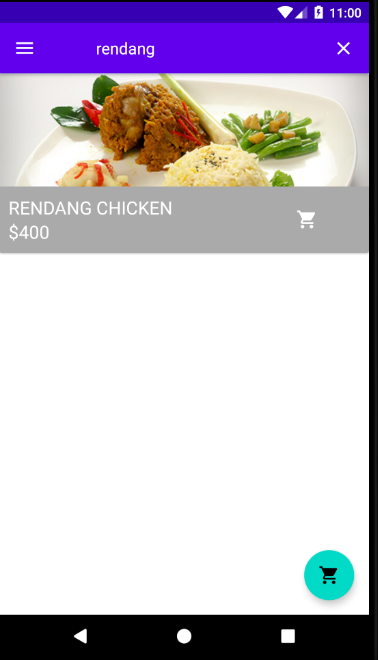


Рисунок 31 – Результат поиска

В противном же случае, если блюда нет в базе данных, то в результатах поиска ничего отображаться не будет (рисунок 32).

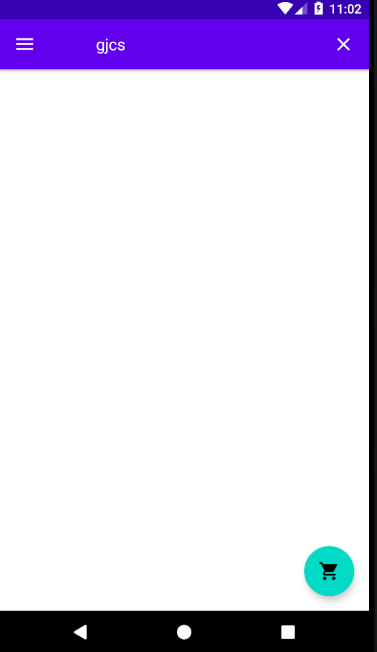


Рисунок 32 – Результат поиска

Далее, после нажатия на блюдо, в категории открывается подробная информация о нем (рисунок 33).

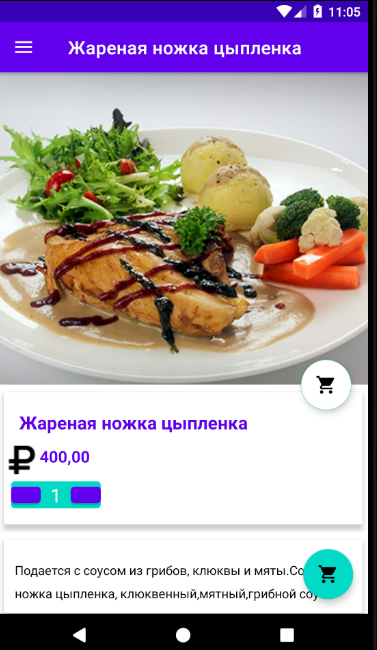


Рисунок 34 – Подробная информация о блюде

Тут отображается цена блюда с учетом количества, размера, добавок и описание блюда. Также на этой активности присутствует кнопки выбора количества, размера и добавок (рисунок 35).

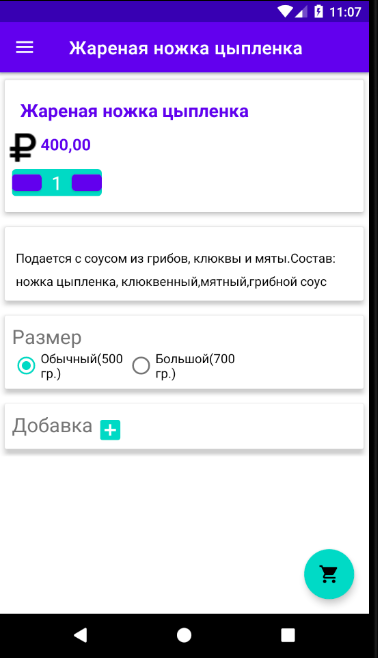


Рисунок 35 – Кнопки выбора размера, количества и добавок

При нажатии на «+» открывается окно с доступными добавками (рисунок 36).

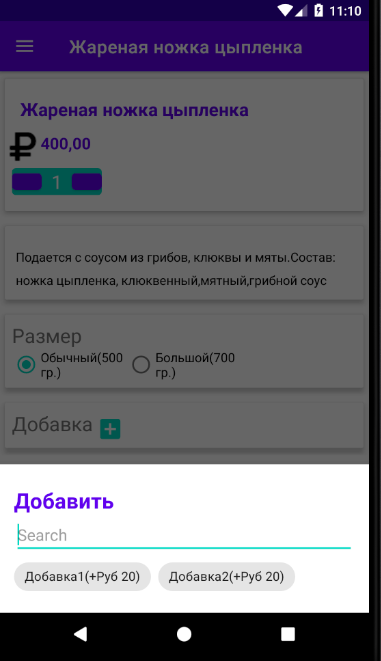


Рисунок 36 – Окно с добавками

Также на этом окне есть поиск по добавкам (рисунок 37).

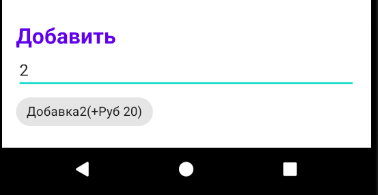


Рисунок 37 – Поиск по добавкам

После нажатия на понравившиеся добавки, они будут отображаться на активности (рисунок 38), при этом цена блюда изменится.

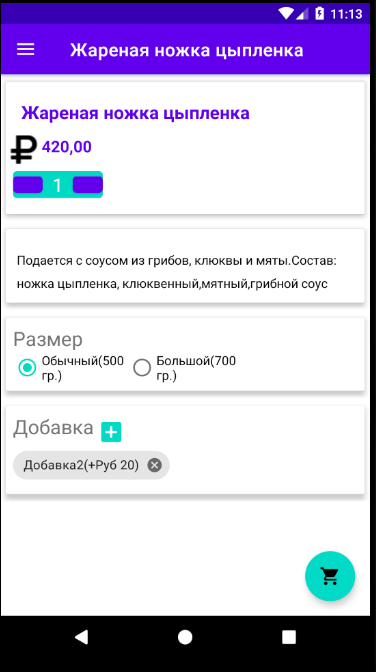


Рисунок 38 – Отображение добавок на активности

Также цена изменяется, если выбрать другой размер блюда (рисунок 39).

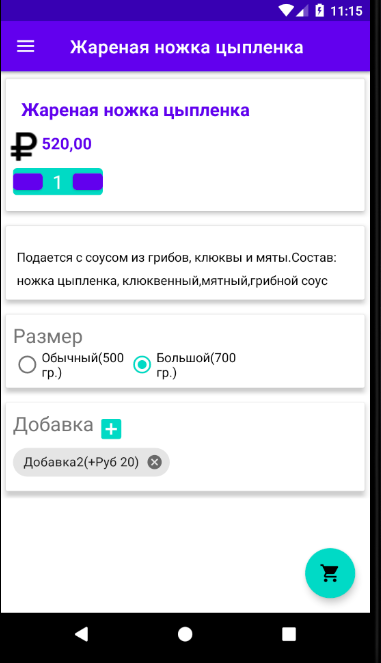


Рисунок 39 – Изменение цены

При этом, если убрать добавку, то цена также изменится (рисунок 40).

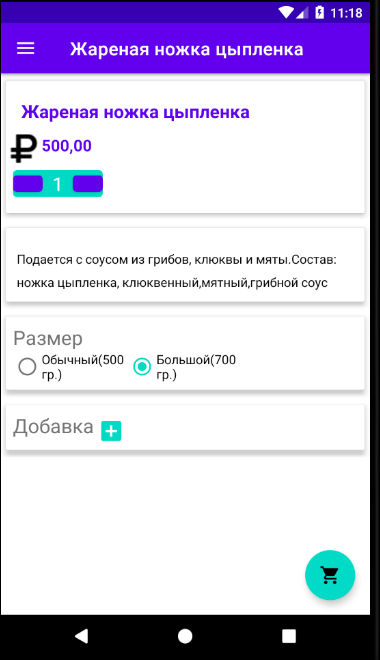


Рисунок 40 – Изменение цены при убранной добавке

Наконец, если вы определились с выбором товара, то нужно нажать на кнопку корзины на белом фоне. Тогда блюдо добавится в корзину с учетом всех изменений. Вы увидите сообщение о статусе операции, и на кнопке корзины справа снизу появится счетчик, который показывает количество блюд в корзине (рисунок 41).

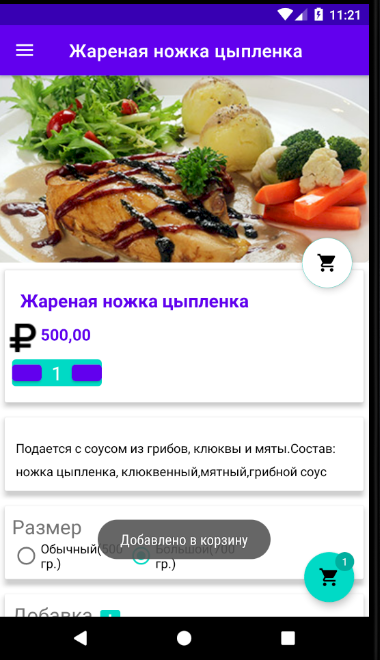


Рисунок 41 – Добавление блюда в корзину

Также, если вы закажите тоже самое блюдо, но, например, другого размера и с добавками то корзина обновится (рисунок 42).

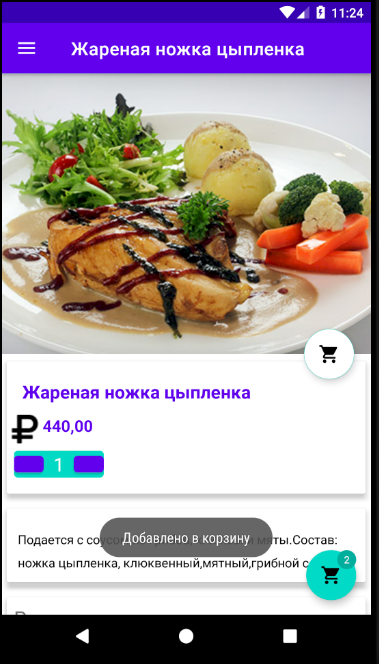


Рисунок 42 – Обновление корзины

При добавлении в корзину, также учитывается количество блюд. Для примера добавим 2 блюда без всяких добавок и обычного размера (рисунок 43).

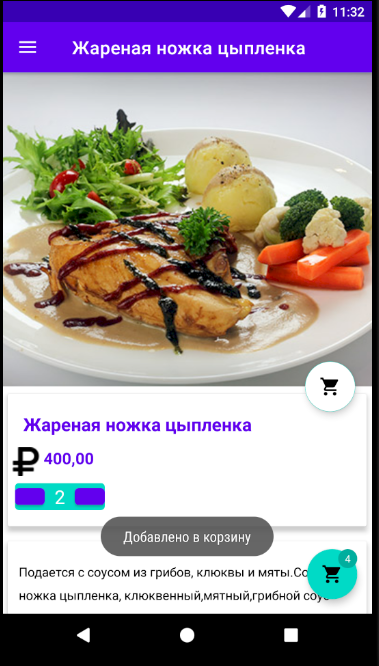


Рисунок 43– Добавление в корзину с учетом количества

# **2.6.4 Активность с корзиной**

На данной активности отображаются блюда, которые выбрал пользователь. При этом, если пользователь добавил одно и тоже блюдо, но с разыми добавками и размерами, то в корзине они отображаются как две вариации одного и того же блюда (рисунок 44).

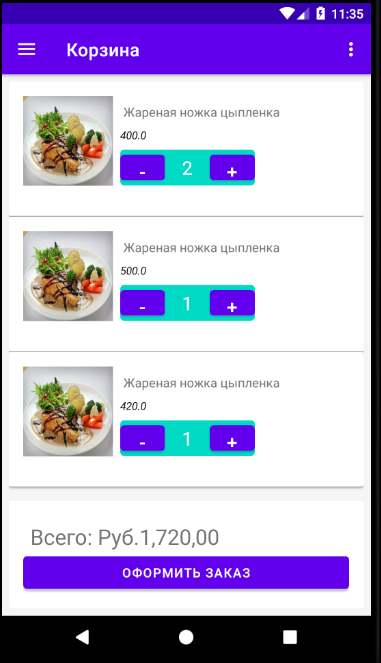


Рисунок 44 – Корзина

Цена этих всех блюд также учитывается и отображается над кнопкой «Оформить заказ». Также из корзины можно указать количество блюд, цена также будет учитываться (рисунок 45).

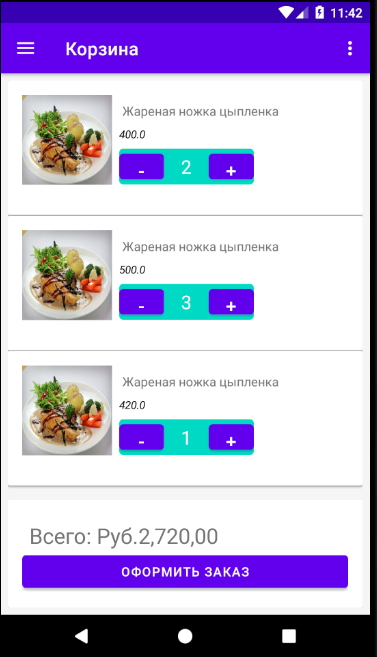


Рисунок 45 – Учет цены при изменении количества

Если же пользователь хочет удалить блюдо из корзины, то для этого ему нужно свайпнуть влево выбранное блюдо и нажать на кнопку «Удалить» (рисунок 46).

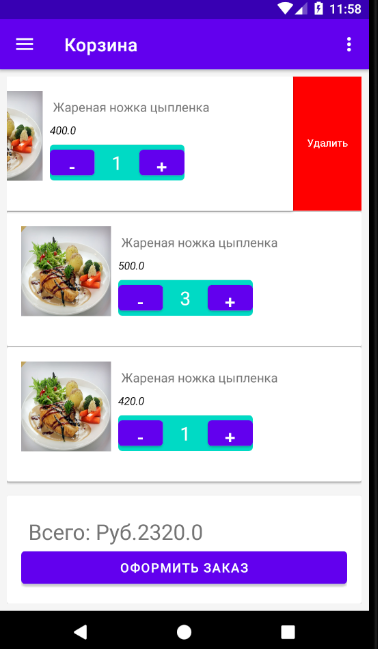


Рисунок 46 – Удаление из корзины

После нажатия на кнопку блюдо удалится из корзины и покажется соответствующее сообщение (рисунок 47).

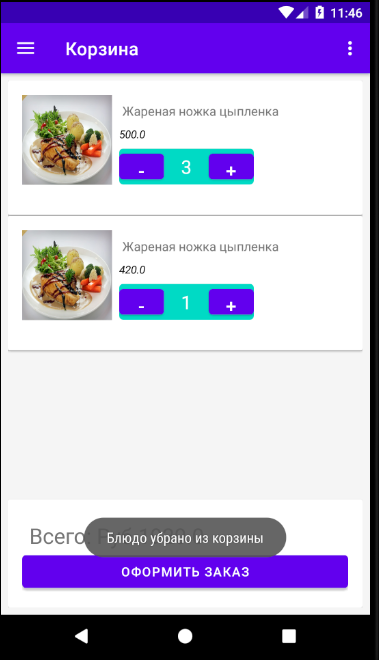


Рисунок 47 – Успешное удаление из корзины

Также пользователь может очистить всю корзину в пару нажатий. Для этого требуется нажать на кнопку справа сверху, а затем выбрать «Очистить все» как показано на рисунке 48.

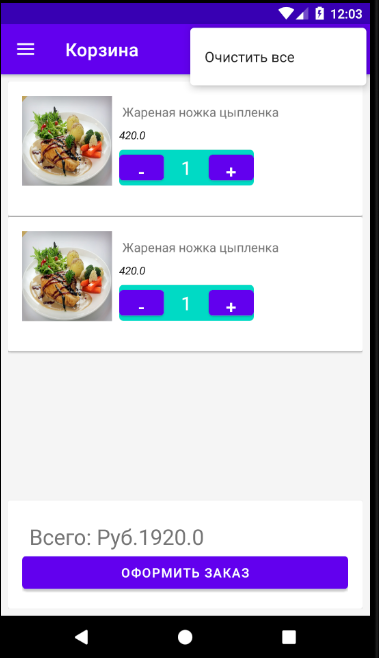


Рисунок 48 – Кнопка «Очистить все»

После этого все блюда, которые были в корзине, удалятся и появится надпись о том, что корзина пуста (рисунок 49).



Рисунок 49 – Пустая корзина

# **2.6.5 Выход**

Выход из текущего аккаунта осуществляется посредством открытия меню и нажатия кнопки выход (рисунок 50).

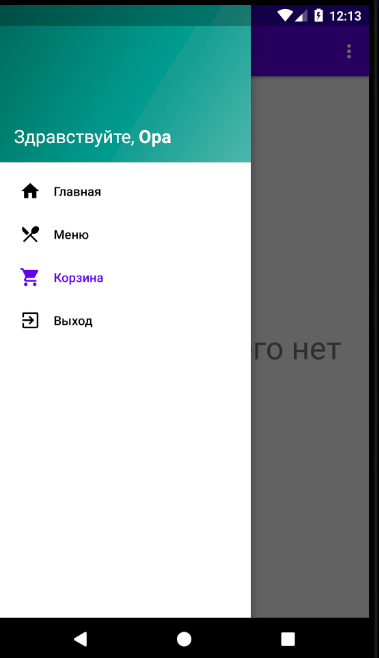


Рисунок 50 – Выход

После чего появится окно, в котором требуется подтвердить выход (рисунок 51).

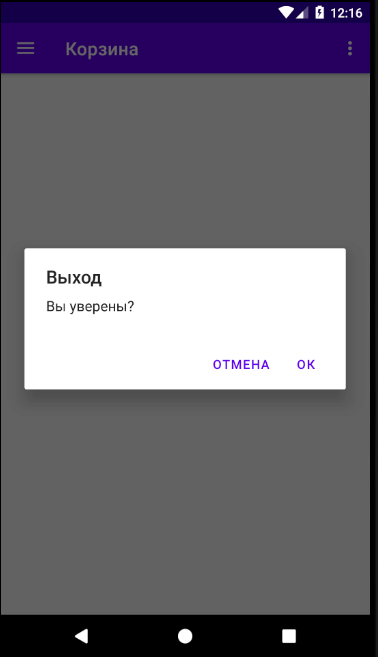


Рисунок 51 – Подтверждение выхода

После нажатия кнопки «ОК» приложение закроется, а при последующем его запуске откроется форма входа в приложение, как это было описано в пункте «2.6.1 Вход в приложение» (страница 36).

# **ЭКОНОМИЧЕСКАЯ ЧАСТЬ**

**РАСЧЕТ ПРЕДПОЛОГАЕМОЙ ПРИБЫЛИ ОТ РЕАЛИЗАЦИИ СОЗДАННОГО ПРОГРАММНОГО ПРОДУКТА**

Данный проект предназначен для предприятий общественного питания преимущественно азиатской кухни, для упрощения его работы, за счет реализованных функций в приложении. Таким образом, человеку, который желает покушать, необязательно идти куда-то в ресторан, а достаточно скачать приложение и заказать еду по адресу. Работа программы начинается с окна входа в систему, где пользователь проходит аутентификацию через SMS-сообщение. Если пользователь не зарегистрирован, то ему предлагается пройти простую регистрацию. Затем, открывается главная страница приложения, на которой находится информация о приложении, а также подсказка для пользователя. Далее, пользователь может осуществлять навигацию по меню, в котором можно открыть меню, корзину или просто выйти с аккаунта. Меню содержит категории блюд, а те, в свою очередь, содержат списки блюд в выбранной пользователем категории. Если пользователь желает как-то изменить блюдо, например, изменить размер или добавить добавок, то он может нажать на блюдо в списке и посмотреть подробное описание блюда с размером и добавками, после чего добавить его в корзину. В корзине пользователь может удалять блюда или вообще целиком очистить корзину. Если пользователь определился с выбором, то он оформляет заказ и получает уведомление о статусе заказа.

# **3.1 Расчёт затрат на выполнение проекта**

Разработка любого программного обеспечения, как и любой другой труд требует определённых затрат, а, следовательно, при продаже разработанного ПО необходимо добиться окупаемости этого товара, а это возможно тогда, когда доходы от реализации продукта покроют расходы, которые были затрачены на его производство.

Чтобы оценить затраты на изготовление программного обеспечения, необходимо определить *трудоёмкость* данного процесса. Трудоёмкость разработки программного продукта состоит из этапов и видов работ, которые необходимо выполнить, чтобы изготовить продукт.

Имеется перечень таких работ и этапов, упорядоченный в соответствии со смысловым содержанием каждого вида работ и взаимосвязями между видами работ.

1. Форма расположения работ по этапам:
   * разработка технического задания (ТЗ): получение ТЗ.
2. Подготовительный этап:
   * сбор информации;
   * выбор объектного построения программы;
   * разработка общей методики создания продукта.
3. Основной этап:
   * разработка основного алгоритма;
   * создание интерфейса;
   * отладка.
4. Завершающий этап:
   * подготовка технической документации;
   * сдача продукта.

Трудоёмкость выполнения работ оценивается в человеко-часах и носит вероятностный характер. Расчёт трудоёмкости представлен в таблице 1.

Таблица 1 – Расчёт трудоемкости по проекту

|  |  |
| --- | --- |
| Виды работ | Трудоёмкость, чел./час |
| 1. Получение ТЗ | 2 |
| 1. Сбор информации и ознакомление с предметной областью | 28 |
| 1. Выбор объектного построения программы | 10 |
| 1. Разработка общей методики создания продукта | 15 |
| 1. Разработка основного алгоритма | 110 |
| 1. Создание интерфейса | 36 |
| 1. Отладка | 12 |
| 1. Подготовка технической документации | 10 |
| 1. Сдача продукта | 2 |
| Итого | 225 |

Суммарная трудоёмкость составляет 225 чел./час.

# **3.2 Расчёт расходов на заработную плату**

На основе данных о трудоёмкости и средней заработной плате по отрасли рассчитываем основную заработную плату. Средняя зарплата Android-разработчика без опыта работы в Санкт-Петербурге на начало 2020 года составила ~ 40000 руб. в месяц (21 рабочий день, 8 часовой рабочий день) или 238,95 руб./час.

Таким образом, расходы на заработную плату по нашему проекту составляют:

ЗП = 238,95 руб./час \* 225 чел./час = 53763,75 руб.

Отчисления на ЗП (Пенсионный Фонд, Фонд социального страхования, Фонд обязательного медицинского страхования) составляют 30%. В денежном выражении это составит: 53763,75 руб. \* 0,3 = 16128,9 руб.

# **3.3 Расчёт электроэнергии и амортизации оборудования**

Так как персональный компьютер стоимостью 45000 руб. Используется в течении 6 лет. Следовательно, амортизация не учитывается.

ПК в среднем потребляет 70 Вт./час,

70 \* 225 = 15750 Вт = 15,75 КВт.

Стоимость 1 Квт./ч по городской зоне СПБ составляет 4,65 руб.

Итого: 15,75 \* 4,65 = 73,23 руб.

# **3.4 Расчет затрат на материалы**

Расчет затрат на материал приведен в таблице 2.

Таблица 2 – Расчет затрат на материалы

|  |  |  |  |
| --- | --- | --- | --- |
| Расходные материалы | Количество | Цена за единицу, руб. | Сумма, руб. |
| Бумага для принтера, уп. | 2 | 235 | 235 |
| Картридж с краской, шт. | 1 | 1880 | 1880 |
| Интернет, мес. | 1 | 550 | 550 |
| Папка, шт. | 1 | 187 | 187 |
| Набор файлов, уп. | 2 | 89 | 178 |
| Итого |  |  | 3030 |

# **3.5 Расчет сметы затрат**

Смета затрат приведена в таблице 3.

Таблица 3 – Смета всех затрат

|  |  |
| --- | --- |
| Наименование статей затрат | Сумма, руб. |
| Расходные материалы | 3030 |
| Основная заработная плата | 53763,75 |
| Расходы на электроэнергию | 73,23 |
| Отчисления на ЗП | 16128,9 |
| Итого | 72935,88 |

Таким образом, суммарная стоимость затрат на создание данного программного продукта составляет 72935,88 рублей. Следовательно, эту программу ниже этой цены продать нельзя, так как в этом случае программа не будет являться экономически эффективной.

Созданный программный продукт актуален на фоне эпидемии, да и целом актуален для населения.

Главными достоинствами программного продукта является:

* максимально простой, понятный и удобный в навигации интерфейс, позволяющий максимально быстро сделать заказ;
* программа имеет очень широкий круг поддерживаемых устройств;
* большой потенциал применения.

При установке рыночной цены важно учитывать, что аналоги подобного рода приложений существуют. При этом и цена реализации этих приложений куда выше. Следовательно, можно установить цену немного ниже средней, но с гарантией того, что приложение окупится. Средняя цена на небольшое мобильное приложение достигает 200000 руб.

При цене в 100000 руб., реализация данного продукта полностью покроет все затраты, а прибыль от реализации программы составит 27064,12 руб.

# **ЗАКЛЮЧЕНИЕ**

В данном дипломном проекте было разработано мобильное приложение для упрощения и оптимизации работы предприятия общественного питания. В ходе выполнения работы были освоены основные приёмы разработки приложений для ОС Android, изучены и применены такие инструменты, как Firebase Authentication, позволяющий интегрировать в приложение разные типы аутентификации, и Firebase Realtime Database, используемый для соединения и работы с облачной объектно-ориентированной базой данных в режиме реального времени.

В приложении реализован поиск по блюдам. А также была создана отдельная база данных для корзины, а также функции, позволяющие добавлять блюда и удалять их из корзины.

Данный программный продукт имеет аналоги на рынке. Но разработанная программа обладает максимально простым интерфейсом и очень легка в использовании, при это сохраняя необходимый функционал для приложений подобного вида.

Что немаловажно, проект имеет зоны роста для расширения и добавления нового функционала. В частности, можно добавить теги для блюд и осуществлять сортировку блюд по ним, также можно включить поддержку удаленной оплаты заказа.

# **ЛИТЕРАТУРА**

1. Брюс Эккель. Философия Java. – 4-е изд. – СПб: Питер, 2017. – 1168 с.
2. П. Дейтел, Х. Дейтел, А. Уолд. Android для разработчиков. – 3-е изд. – СПб: Питер, 2016. – 512 с.
3. Брайан Харди, Билл Филлипс, Крис Стюарт, Кристин Марсикано. Программирование под Android. – 2-е изд. – СПб: Питер, 2016. – 567 с.
4. Дон Гриффитс, Дэвид Гриффитс. Head First. Программирование для Android. – 2-е изд. – СПб: Питер, 2018. – 592 с.
5. Герберт Шилдт. Java 8. Руководство для начинающих. – 6-е изд. – СПб: Вильямс, 2015. – 720 с.
6. Кей Хорстманн, Гари Корнелл. Java. Библиотека профессионала. Том 1. – 10-е изд. – СПб: Вильямс. – 2016. – 1008 с.

# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. ГОСТ Р 7.0.97-2016. Национальный стандарт Российской Федерации. Система стандартов по информации, библиотечному и издательскому делу. Организационно-распорядительная документация. Требования к оформлению документов" (ред. от 14.05.2018).
2. ГОСТ 19.202-78 Спецификация. Требования к содержанию и оформлению;
3. ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению.
4. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом.
5. https://developers.google.com – справочный ресурс для Android-разработчиков.
6. https://habr.com – русскоязычное IT сообщество.
7. https://stackoverflow.com – заграничный ресурс обмена опытом среди разработчиков.
8. https://github.com – проблемы и неисправности различных инструментов.

# **ПРИЛОЖЕНИЕ А. Исходный текст программы**

**AndroidManifest.xml**

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.foodapp">

<uses-permission android:name="android.permission.INTERNET" />

<application

android:allowBackup="true"

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:roundIcon="@mipmap/ic\_launcher\_round"

android:supportsRtl="true"

android:theme="@style/AppTheme">

<activity

android:name=".HomeActivity"

android:label="@string/title\_activity\_home"

android:theme="@style/AppTheme.NoActionBar"></activity>

<activity android:name=".MainActivity">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

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

</intent-filter>

</activity>

</application>

</manifest>

**MainActivity.java**

package com.example.foodapp;

import androidx.annotation.NonNull;

import androidx.annotation.Nullable;

import androidx.appcompat.app.AppCompatActivity;

import android.app.AlertDialog;

import android.content.DialogInterface;

import android.content.Intent;

import android.os.Bundle;

import android.text.TextUtils;

import android.view.LayoutInflater;

import android.view.View;

import android.widget.EditText;

import android.widget.Toast;

import com.firebase.ui.auth.AuthUI;

import com.firebase.ui.auth.IdpResponse;

import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

import java.util.Arrays;

import java.util.List;

import Common.Common;

import Models.UserModel;

import dmax.dialog.SpotsDialog;

import io.reactivex.disposables.CompositeDisposable;

public class MainActivity extends AppCompatActivity {

private static int APP\_REQUEST\_CODE = 4201;

private FirebaseAuth firebaseAuth;

private FirebaseAuth.AuthStateListener listener;

private AlertDialog dialog;

private CompositeDisposable disposable = new CompositeDisposable();

private DatabaseReference userRef;

private List<AuthUI.IdpConfig> providers;

@Override

protected void onStart() {

super.onStart();

firebaseAuth.addAuthStateListener(listener);

}

@Override

protected void onStop() {

if(listener != null){

firebaseAuth.removeAuthStateListener(listener);

disposable.clear();

}

super.onStop();

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

init();

}

private void init() {

providers = Arrays.asList(new AuthUI.IdpConfig.PhoneBuilder().build());

userRef = FirebaseDatabase.getInstance().getReference(Common.USER\_REF);

firebaseAuth = FirebaseAuth.getInstance();

dialog = new SpotsDialog.Builder().setCancelable(false).setContext(this).build();

listener = new FirebaseAuth.AuthStateListener() {

@Override

public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {

FirebaseUser user = firebaseAuth.getCurrentUser();

if(user != null){

CheckUserFormFirebase(user);

}else{

phoneLogin();

}

}

};

}

private void CheckUserFormFirebase(FirebaseUser user) {

dialog.show();

userRef.child(user.getUid()).addListenerForSingleValueEvent(new ValueEventListener() {

@Override

public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

if(dataSnapshot.exists()){

//Toast.makeText(MainActivity.this, "You already registered!", Toast.LENGTH\_SHORT).show();

UserModel userModel = dataSnapshot.getValue(UserModel.class);

goToHomeActivity(userModel);

}else{

showRegisterDialog(user);

}

dialog.dismiss();

}

@Override

public void onCancelled(@NonNull DatabaseError databaseError) {

dialog.dismiss();

Toast.makeText(MainActivity.this, ""+databaseError.getMessage(), Toast.LENGTH\_SHORT).show();

}

});

}

private void phoneLogin(){

startActivityForResult(AuthUI.getInstance()

.createSignInIntentBuilder()

.setAvailableProviders(providers).build(),

APP\_REQUEST\_CODE);

}

private void showRegisterDialog(FirebaseUser user){

androidx.appcompat.app.AlertDialog.Builder builder = new androidx.appcompat.app.AlertDialog.Builder(this);

builder.setTitle("Регистрация");

builder.setMessage("Пожалуйста, заполните поля");

View itemView = LayoutInflater.from(this).inflate(R.layout.layout\_register,null);

EditText edt\_name = (EditText)itemView.findViewById(R.id.edt\_name);

EditText edt\_phone = (EditText)itemView.findViewById(R.id.edt\_phone);

edt\_phone.setText(user.getPhoneNumber());

builder.setView(itemView);

builder.setNegativeButton("Отмена", (dialog, which) -> dialog.dismiss());

builder.setPositiveButton("Регистрация", (dialog, which) -> {

if(TextUtils.isEmpty(edt\_name.getText().toString())){

Toast.makeText(MainActivity.this, "Пожалуйста, введите имя", Toast.LENGTH\_SHORT).show();

return;

}

UserModel userModel = new UserModel();

userModel.setUid(user.getUid());

userModel.setName(edt\_name.getText().toString());

userModel.setPhone(edt\_phone.getText().toString());

userRef.child(user.getUid()).setValue(userModel).addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

if(task.isSuccessful()){

dialog.dismiss();

Toast.makeText(MainActivity.this, "Регистрация завершена!", Toast.LENGTH\_SHORT).show();

goToHomeActivity(userModel);

}

}

});

});

builder.setView(itemView);

androidx.appcompat.app.AlertDialog dialog = builder.create();

dialog.show();

}

private void goToHomeActivity(UserModel userModel) {

Common.currentUser = userModel;

startActivity(new Intent(MainActivity.this, HomeActivity.class));

finish();

}

@Override

protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if(requestCode == APP\_REQUEST\_CODE){

IdpResponse response = IdpResponse.fromResultIntent(data);

if(resultCode == RESULT\_OK){

FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();

}else{

Toast.makeText(this, "Ошибка входа", Toast.LENGTH\_SHORT).show();

}

}

}

}

**HomeActivity.java**

package com.example.foodapp;

import android.content.DialogInterface;

import android.content.Intent;

import android.os.Bundle;

import android.view.MenuItem;

import android.view.View;

import android.view.Menu;

import android.widget.TextView;

import android.widget.Toast;

import com.andremion.counterfab.CounterFab;

import com.google.android.material.floatingactionbutton.FloatingActionButton;

import com.google.android.material.snackbar.Snackbar;

import com.google.android.material.navigation.NavigationView;

import com.google.firebase.auth.FirebaseAuth;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AlertDialog;

import androidx.navigation.NavController;

import androidx.navigation.Navigation;

import androidx.navigation.ui.AppBarConfiguration;

import androidx.navigation.ui.NavigationUI;

import androidx.drawerlayout.widget.DrawerLayout;

import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.widget.Toolbar;

import Common.Common;

import Database.CartDataBase;

import Database.CartDataSource;

import Database.LocalCartDataSource;

import EventBus.CounterCartEvent;

import EventBus.CategoryClick;

import EventBus.HideFABCart;

import EventBus.FoodItemClick;

import butterknife.BindView;

import butterknife.ButterKnife;

import io.reactivex.SingleObserver;

import io.reactivex.android.schedulers.AndroidSchedulers;

import io.reactivex.disposables.Disposable;

import io.reactivex.schedulers.Schedulers;

import org.greenrobot.eventbus.EventBus;

import org.greenrobot.eventbus.Subscribe;

import org.greenrobot.eventbus.ThreadMode;

public class HomeActivity extends AppCompatActivity implements NavigationView.OnNavigationItemSelectedListener {

private AppBarConfiguration mAppBarConfiguration;

private DrawerLayout drawer;

private NavController navController;

private CartDataSource cartDataSource;

@BindView(R.id.fab)

CounterFab fab;

@Override

protected void onResume() {

super.onResume();

countCartItem();

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_home);

ButterKnife.bind(this);

cartDataSource = new LocalCartDataSource(CartDataBase.getInstance(this).cartDAO());

Toolbar toolbar = findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

FloatingActionButton fab = findViewById(R.id.fab);

fab.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

navController.navigate(R.id.nav\_cart);

}

});

drawer = findViewById(R.id.drawer\_layout);

NavigationView navigationView = findViewById(R.id.nav\_view);

// Passing each menu ID as a set of Ids because each

// menu should be considered as top level destinations.

mAppBarConfiguration = new AppBarConfiguration.Builder(

R.id.nav\_home, R.id.nav\_menu, R.id.nav\_food\_list,R.id.nav\_food\_detail,R.id.nav\_cart)

.setDrawerLayout(drawer)

.build();

navController = Navigation.findNavController(this, R.id.nav\_host\_fragment);

NavigationUI.setupActionBarWithNavController(this, navController, mAppBarConfiguration);

NavigationUI.setupWithNavController(navigationView, navController);

navigationView.setNavigationItemSelectedListener(this);

navigationView.bringToFront();

View headerView = navigationView.getHeaderView(0);

TextView txt\_username = (TextView) headerView.findViewById(R.id.txt\_username);

Common.setSpanString("Здравствуйте, ",Common.currentUser.getName(),txt\_username);

countCartItem();

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.home, menu);

return true;

}

@Override

public boolean onSupportNavigateUp() {

NavController navController = Navigation.findNavController(this, R.id.nav\_host\_fragment);

return NavigationUI.navigateUp(navController, mAppBarConfiguration)

|| super.onSupportNavigateUp();

}

@Override

public boolean onNavigationItemSelected(@NonNull MenuItem item) {

item.setChecked(true);

drawer.closeDrawers();

switch (item.getItemId()){

case R.id.nav\_home:

navController.navigate(R.id.nav\_home);

break;

case R.id.nav\_menu:

navController.navigate(R.id.nav\_menu);

break;

case R.id.nav\_cart:

navController.navigate(R.id.nav\_cart);

break;

case R.id.nav\_sign\_out:

signOut();

break;

}

return true;

}

private void signOut() {

AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setTitle("Выход").setMessage("Вы уверены?").setNegativeButton("Отмена", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

dialog.dismiss();

}

}).setPositiveButton("Ок", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

Common.selectedFood = null;

Common.categorySelected = null;

Common.currentUser = null;

FirebaseAuth.getInstance().signOut();

Intent intent = new Intent(HomeActivity.this, MainActivity.class);

intent.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK | Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);

finish();

}

});

AlertDialog dialog = builder.create();

dialog.show();

}

//EventBus

@Override

protected void onStart() {

super.onStart();

EventBus.getDefault().register(this);

}

@Override

protected void onStop() {

EventBus.getDefault().unregister(this);

super.onStop();

}

@Subscribe(sticky = true, threadMode = ThreadMode.MAIN)

public void onCategorySelected(CategoryClick event){

if(event.isSuccess()){

//Toast.makeText(this, "Click to "+event.getCategoryModel().getName(), Toast.LENGTH\_SHORT).show();

navController.navigate(R.id.nav\_food\_list);

}

}

@Subscribe(sticky = true, threadMode = ThreadMode.MAIN)

public void onFoodItemClick(FoodItemClick event){

if(event.isSuccess()){

navController.navigate(R.id.nav\_food\_detail);

}

}

@Subscribe(sticky = true, threadMode = ThreadMode.MAIN)

public void onCartCounter(CounterCartEvent event){

if(event.isSuccess()){

countCartItem();

}

}

@Subscribe(sticky = true, threadMode = ThreadMode.MAIN)

public void onHideFABEvent(HideFABCart event){

if(event.isHidden()){

fab.hide();

}else{

fab.show();

}

}

private void countCartItem() {

cartDataSource.countItemInCart(Common.currentUser.getUid())

.subscribeOn(Schedulers.io())

.observeOn(AndroidSchedulers.mainThread())

.subscribe(new SingleObserver<Integer>() {

@Override

public void onSubscribe(Disposable d) {

}

@Override

public void onSuccess(Integer integer) {

fab.setCount(integer);

}

@Override

public void onError(Throwable e) {

if(!e.getMessage().contains("Query returned empty")){

Toast.makeText(HomeActivity.this, "[COUNT CART]"+e.getMessage(), Toast.LENGTH\_SHORT).show();

}else{

fab.setCount(0);

}

}

});

}

}

**FoodListAdapter.java**

package Adapters;

import android.content.Context;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageView;

import android.widget.TextView;

import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;

import com.example.foodapp.R;

import org.greenrobot.eventbus.EventBus;

import java.util.List;

import Database.CartDataBase;

import Database.CartDataSource;

import Database.CartItem;

import Database.LocalCartDataSource;

import EventBus.FoodItemClick;

import EventBus.CounterCartEvent;

import Callbacks.RecyclerClickListener;

import Common.Common;

import Models.FoodModel;

import butterknife.BindView;

import butterknife.ButterKnife;

import butterknife.Unbinder;

import io.reactivex.SingleObserver;

import io.reactivex.android.schedulers.AndroidSchedulers;

import io.reactivex.disposables.CompositeDisposable;

import io.reactivex.disposables.Disposable;

import io.reactivex.schedulers.Schedulers;

public class FoodListAdapter extends RecyclerView.Adapter<FoodListAdapter.ViewHolder> {

private Context context;

private List<FoodModel> foodModelList;

private CompositeDisposable compositeDisposable;

private CartDataSource cartDataSource;

public FoodListAdapter(Context context, List<FoodModel> foodModelList) {

this.context = context;

this.foodModelList = foodModelList;

this.compositeDisposable = new CompositeDisposable();

this.cartDataSource = new LocalCartDataSource(CartDataBase.getInstance(context).cartDAO());

}

@NonNull

@Override

public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

return new ViewHolder(LayoutInflater.from(context)

.inflate(R.layout.layout\_food\_item,parent,false));

}

@Override

public void onBindViewHolder(@NonNull ViewHolder holder, int position) {

Glide.with(context).load(foodModelList.get(position).getImage()).into(holder.food\_item\_image);

holder.food\_item\_price.setText(new StringBuilder("$")

.append(foodModelList.get(position).getPrice()));

holder.food\_item\_name.setText(new StringBuilder("")

.append(foodModelList.get(position).getName()));

//Event

holder.setRecyclerClickListener(new RecyclerClickListener() {

@Override

public void onItemClickListener(View view, int position) {

Common.selectedFood = foodModelList.get(position);

EventBus.getDefault().postSticky(new FoodItemClick(true,foodModelList.get(position)));

}

});

holder.img\_cart.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

CartItem cartItem = new CartItem();

cartItem.setUid(Common.currentUser.getUid());

cartItem.setUserPhone(Common.currentUser.getPhone());

cartItem.setFoodId(foodModelList.get(position).getId());

cartItem.setFoodName(foodModelList.get(position).getName());

cartItem.setFoodImage(foodModelList.get(position).getImage());

cartItem.setFoodPrice(Double.valueOf(String.valueOf(foodModelList.get(position).getPrice())));

cartItem.setFoodQuantity(1);

cartItem.setFoodExtraPrice(0.0);

cartItem.setFoodAddon("Default");

cartItem.setFoodSize("Default");

cartDataSource.getItemWithAllOptionsInCart(Common.currentUser.getUid(),

cartItem.getFoodId(),

cartItem.getFoodSize(),

cartItem.getFoodAddon())

.subscribeOn(Schedulers.io())

.observeOn(AndroidSchedulers.mainThread())

.subscribe(new SingleObserver<CartItem>() {

@Override

public void onSubscribe(Disposable d) {

}

@Override

public void onSuccess(CartItem cartItemFromDB) {

if(cartItemFromDB.equals(cartItem)){

//Just update

cartItemFromDB.setFoodExtraPrice(cartItem.getFoodExtraPrice());

cartItemFromDB.setFoodAddon(cartItem.getFoodAddon());

cartItemFromDB.setFoodSize(cartItem.getFoodSize());

cartItemFromDB.setFoodQuantity(cartItemFromDB.getFoodQuantity()+cartItem.getFoodQuantity());

cartDataSource.updateCartItems(cartItemFromDB)

.subscribeOn(Schedulers.io())

.observeOn(AndroidSchedulers.mainThread())

.subscribe(new SingleObserver<Integer>() {

@Override

public void onSubscribe(Disposable d) {

}

@Override

public void onSuccess(Integer integer) {

Toast.makeText(context, "Корзина обновлена!", Toast.LENGTH\_SHORT).show();

EventBus.getDefault().postSticky(new CounterCartEvent(true));

}

@Override

public void onError(Throwable e) {

Toast.makeText(context, "[UPDATE CART]"+e.getMessage(), Toast.LENGTH\_SHORT).show();

}

});

}else{

//New item

compositeDisposable.add(cartDataSource.insertOrReplaceAll(cartItem)

.subscribeOn(Schedulers.io())

.observeOn(AndroidSchedulers.mainThread())

.subscribe(()->{

Toast.makeText(context, "Добавлено в корзину", Toast.LENGTH\_SHORT).show();

//Update Counter

EventBus.getDefault().postSticky(new CounterCartEvent(true));

}, throwable -> {

Toast.makeText(context, "[CART ERROR]"+throwable.getMessage(), Toast.LENGTH\_SHORT).show();

}));

}

}

@Override

public void onError(Throwable e) {

if(e.getMessage().contains("empty")){

//Default

compositeDisposable.add(cartDataSource.insertOrReplaceAll(cartItem)

.subscribeOn(Schedulers.io())

.observeOn(AndroidSchedulers.mainThread())

.subscribe(()->{

Toast.makeText(context, "Добавлено в корзину", Toast.LENGTH\_SHORT).show();

//Update Counter

EventBus.getDefault().postSticky(new CounterCartEvent(true));

}, throwable -> {

Toast.makeText(context, "[CART ERROR]"+throwable.getMessage(), Toast.LENGTH\_SHORT).show();

}));

}else {

Toast.makeText(context, "[GET CART]" + e.getMessage(), Toast.LENGTH\_SHORT).show();

}

}

});

}

});

}

@Override

public int getItemCount() {

return foodModelList.size();

}

public class ViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {

private Unbinder unbinder;

@BindView(R.id.food\_item\_name)

TextView food\_item\_name;

@BindView(R.id.food\_item\_price)

TextView food\_item\_price;

@BindView(R.id.food\_item\_image)

ImageView food\_item\_image;

@BindView(R.id.img\_cart)

ImageView img\_cart;

RecyclerClickListener recyclerClickListener;

public void setRecyclerClickListener(RecyclerClickListener recyclerClickListener) {

this.recyclerClickListener = recyclerClickListener;

}

public ViewHolder(@NonNull View itemView) {

super(itemView);

unbinder = ButterKnife.bind(this,itemView);

itemView.setOnClickListener(this);

}

@Override

public void onClick(View v) {

recyclerClickListener.onItemClickListener(v,getAdapterPosition());

}

}

}

**CartAdapter.java**

package Adapters;

import android.content.Context;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageView;

import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;

import com.cepheuen.elegantnumberbutton.view.ElegantNumberButton;

import com.example.foodapp.R;

import org.greenrobot.eventbus.EventBus;

import java.util.List;

import EventBus.UpdateItemInCart;

import Database.CartItem;

import butterknife.BindView;

import butterknife.ButterKnife;

import butterknife.Unbinder;

public class CartAdapter extends RecyclerView.Adapter<CartAdapter.ViewHolder> {

Context context;

List<CartItem> cartItemList;

public CartAdapter(Context context, List<CartItem> cartItemList) {

this.context = context;

this.cartItemList = cartItemList;

}

@NonNull

@Override

public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

return new ViewHolder(LayoutInflater.from(context).inflate(R.layout.layout\_cart\_item,parent,false));

}

@Override

public void onBindViewHolder(@NonNull ViewHolder holder, int position) {

Glide.with(context).load(cartItemList.get(position).getFoodImage()).into(holder.img\_cart\_item);

holder.txt\_food\_name.setText(new StringBuilder(cartItemList.get(position).getFoodName()));

holder.txt\_food\_price.setText(new StringBuilder("").append(cartItemList.get(position).getFoodPrice()+cartItemList.get(position).getFoodExtraPrice()));

holder.number\_button\_cart.setNumber(String.valueOf(cartItemList.get(position).getFoodQuantity()));

//Event

holder.number\_button\_cart.setOnValueChangeListener(new ElegantNumberButton.OnValueChangeListener() {

@Override

public void onValueChange(ElegantNumberButton view, int oldValue, int newValue) {

//Update db when click

cartItemList.get(position).setFoodQuantity(newValue);

EventBus.getDefault().postSticky(new UpdateItemInCart(cartItemList.get(position)));

}

});

}

@Override

public int getItemCount() {

return cartItemList.size();

}

public CartItem getItemAtPosition(int position) {

return cartItemList.get(position);

}

public class ViewHolder extends RecyclerView.ViewHolder{

private Unbinder unbinder;

@BindView(R.id.img\_cart\_item)

ImageView img\_cart\_item;

@BindView(R.id.txt\_food\_price)

TextView txt\_food\_price;

@BindView(R.id.txt\_food\_name)

TextView txt\_food\_name;

@BindView(R.id.number\_button\_cart)

ElegantNumberButton number\_button\_cart;

public ViewHolder(@NonNull View itemView) {

super(itemView);

unbinder = ButterKnife.bind(this,itemView);

}

}

}

**MenuAdapter.java**

package Adapters;

import android.content.Context;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageView;

import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;

import com.example.foodapp.R;

import org.greenrobot.eventbus.EventBus;

import java.util.List;

import EventBus.CategoryClick;

import Callbacks.RecyclerClickListener;

import Common.Common;

import Models.CategoryModel;

import butterknife.BindView;

import butterknife.ButterKnife;

import butterknife.Unbinder;

public class MenuAdapter extends RecyclerView.Adapter<MenuAdapter.ViewHolder> {

Context context;

List<CategoryModel> categoryModelList;

public MenuAdapter(Context context, List<CategoryModel> categoryModelList) {

this.context = context;

this.categoryModelList = categoryModelList;

}

@NonNull

@Override

public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

return new ViewHolder(LayoutInflater.from(context)

.inflate(R.layout.layout\_menu\_item,parent,false));

}

@Override

public void onBindViewHolder(@NonNull ViewHolder holder, int position) {

Glide.with(context).load(categoryModelList.get(position).getImage()).into(holder.menu\_image);

holder.menu\_name.setText(new StringBuilder(categoryModelList.get(position).getName()));

//Event

holder.setListener(new RecyclerClickListener() {

@Override

public void onItemClickListener(View view, int position) {

Common.categorySelected = categoryModelList.get(position);

EventBus.getDefault().postSticky(new CategoryClick(true,categoryModelList.get(position)));

}

});

}

@Override

public int getItemCount() {

return categoryModelList.size();

}

public class ViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {

Unbinder unbinder;

@BindView(R.id.menu\_image)

ImageView menu\_image;

@BindView(R.id.menu\_name)

TextView menu\_name;

RecyclerClickListener listener;

public void setListener(RecyclerClickListener listener) {

this.listener = listener;

}

public ViewHolder(@NonNull View itemView) {

super(itemView);

unbinder = ButterKnife.bind(this,itemView);

itemView.setOnClickListener(this);

}

@Override

public void onClick(View v) {

listener.onItemClickListener(v,getAdapterPosition());

}

}

@Override

public int getItemViewType(int position) {

if(categoryModelList.size() == 1){

return Common.DEFAULT\_COLUMN\_COUNT;

}else{

if(categoryModelList.size() % 2 ==0){

return Common.DEFAULT\_COLUMN\_COUNT;

}else{

return (position > 1 && position == categoryModelList.size() - 1) ? Common.FULL\_WIDTH\_COLUMN:Common.DEFAULT\_COLUMN\_COUNT;

}

}

}

}

**layout\_register.xml**

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:background="@android:color/white"

android:padding="8dp">

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

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

<EditText

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/edt\_name"

android:hint="Ваше имя"/>

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

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

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

<EditText

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/edt\_phone"

android:focusable="false"/>

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

</LinearLayout>

**layout\_order.xml**

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical"

android:padding="8dp"

android:background="@android:color/white">

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

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

<EditText

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/edt\_address\_order"

android:hint="Введите адрес"/>

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

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:textSize="22sp"

android:textStyle="bold"

android:textColor="@android:color/darker\_gray"

android:text="Способ доставки "/>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Доставка будет осуществлена по адресу который вы указали!"

android:textSize="18sp"

android:textStyle="italic"

android:id="@+id/txt\_address"/>

<TextView

android:text="Способ оплаты"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:textSize="22sp"

android:textStyle="bold"

android:textColor="@android:color/darker\_gray"/>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Оплата на месте получения заказа!"

android:textSize="18sp"

android:textStyle="italic"

android:id="@+id/txt\_payment"/>

</LinearLayout>

**layout\_menu\_item.xml**

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout

xmlns:android="http://schemas.android.com/apk/res/android"

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

android:layout\_width="match\_parent"

android:layout\_height="100dp"

app:cardElevation="4dp">

<ImageView

android:id="@+id/menu\_image"

android:scaleType="centerCrop"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

/>

<TextView

android:id="@+id/menu\_name"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_alignParentStart="true"

android:layout\_alignParentBottom="true"

android:layout\_gravity="center\_horizontal|center\_vertical"

android:background="@android:color/darker\_gray"

android:gravity="center"

android:text="Name of menu"

android:textColor="@android:color/white"

android:textSize="20sp" />

</RelativeLayout>

**layout\_cart\_item.xml**

<?xml version="1.0" encoding="utf-8"?>

<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"

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

android:orientation="vertical"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

<LinearLayout

android:padding="16dp"

android:orientation="horizontal"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

<ImageView

android:id="@+id/img\_cart\_item"

android:layout\_width="100dp"

android:layout\_height="100dp"

android:scaleType="fitXY"/>

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical">

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Cart"

android:id="@+id/txt\_food\_name"

android:textSize="14sp"

android:layout\_marginTop="8dp"

android:layout\_marginStart="8dp"

android:layout\_marginEnd="8dp"/>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/txt\_food\_price"

android:text="100.0"

android:textStyle="italic"

android:layout\_marginTop="8dp"

android:layout\_marginStart="8dp"

android:layout\_marginEnd="8dp"

android:textSize="12sp"

android:textColor="@android:color/black"

/>

<com.cepheuen.elegantnumberbutton.view.ElegantNumberButton

android:id="@+id/number\_button\_cart"

android:layout\_width="150dp"

android:layout\_height="40dp"

android:layout\_marginBottom="18dp"

android:layout\_marginTop="8dp"

android:layout\_marginLeft="8dp"

app:backGroundColor="@color/colorAccent"

app:initialNumber="1"

app:finalNumber="20"

app:textColor="@android:color/white"

app:textSize="8sp"

/>

</LinearLayout>

</LinearLayout>

</LinearLayout>

</androidx.cardview.widget.CardView>

**fragment\_food\_detail.xml**

<?xml version="1.0" encoding="utf-8"?>

<androidx.coordinatorlayout.widget.CoordinatorLayout

xmlns:android="http://schemas.android.com/apk/res/android"

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

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

>

<com.google.android.material.appbar.AppBarLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/app\_bar\_layout"

android:fitsSystemWindows="true"

android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">

<com.google.android.material.appbar.CollapsingToolbarLayout

android:layout\_width="match\_parent"

android:layout\_height="350dp"

android:fitsSystemWindows="true"

app:layout\_scrollFlags="scroll|exitUntilCollapsed"

app:expandedTitleTextAppearance="@android:color/transparent"

app:contentScrim="#0e0d0e"

android:id="@+id/collapsing">

<ImageView

android:id="@+id/image\_food\_detail"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:contentDescription="@null"

android:scaleType="centerCrop"

app:layout\_collapseMode="parallax" />

</com.google.android.material.appbar.CollapsingToolbarLayout>

</com.google.android.material.appbar.AppBarLayout>

<com.andremion.counterfab.CounterFab

android:id="@+id/btn\_cart"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

app:srcCompat="@drawable/ic\_shopping\_cart"

android:backgroundTint="@android:color/white"

android:elevation="6dp"

app:useCompatPadding="true"

app:pressedTranslationZ="12dp"

app:layout\_anchorGravity="bottom|right|end"

app:layout\_anchor="@id/app\_bar\_layout"/>

<androidx.core.widget.NestedScrollView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:id="@+id/nested\_scroll\_view"

android:clipToPadding="false"

app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical">

<androidx.cardview.widget.CardView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

app:cardElevation="5dp"

app:cardUseCompatPadding="true">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical">

<TextView

android:id="@+id/food\_name"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="8dp"

android:padding="12dp"

android:text="Food name"

android:textColor="@color/colorPrimary"

android:textSize="20sp"

android:textStyle="bold"/>

<LinearLayout

android:id="@+id/layout\_price"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal">

<ImageView

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

app:srcCompat="@drawable/ruble"

/>

<TextView

android:id="@+id/food\_price"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="9"

android:text="1 000"

android:textColor="@color/colorPrimary"

android:textSize="18sp"

android:textStyle="bold"/>

</LinearLayout>

<com.cepheuen.elegantnumberbutton.view.ElegantNumberButton

android:id="@+id/number\_button"

android:layout\_width="100dp"

android:layout\_height="30dp"

android:layout\_marginLeft="8dp"

android:layout\_marginTop="8dp"

android:layout\_marginBottom="18dp"

app:backGroundColor="@color/colorAccent"

app:textColor="@android:color/white"

app:finalNumber="20"

app:initialNumber="1"

app:textSize="8sp" />

</LinearLayout>

</androidx.cardview.widget.CardView>

<androidx.cardview.widget.CardView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

app:cardElevation="5dp"

app:cardUseCompatPadding="true">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical">

<TextView

android:id="@+id/food\_description"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="12dp"

android:lineSpacingMultiplier="1.5"

android:padding="12dp"

android:text="Description"

android:textColor="@android:color/black"

android:textSize="14sp"/>

</LinearLayout>

</androidx.cardview.widget.CardView>

<androidx.cardview.widget.CardView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:padding="8dp"

app:cardElevation="5dp"

app:cardUseCompatPadding="true">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical"

android:padding="8dp">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/size"

android:textSize="22sp"/>

<RadioGroup

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:weightSum="3"

android:id="@+id/rg\_group\_size"/>

</LinearLayout>

</androidx.cardview.widget.CardView>

<androidx.cardview.widget.CardView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

app:cardElevation="5dp"

app:cardUseCompatPadding="true">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical"

android:padding="8dp">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/addon"

android:textSize="22sp"/>

<ImageView

android:id="@+id/img\_add\_addon"

android:layout\_width="30dp"

android:layout\_height="30dp"

android:layout\_gravity="center\_vertical"

android:layout\_marginLeft="4dp"

android:layout\_marginTop="4dp"

app:srcCompat="@drawable/ic\_add\_box"

android:tint="@color/colorAccent"/>

</LinearLayout>

<com.google.android.material.chip.ChipGroup

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/chip\_group\_user\_selected"/>

</LinearLayout>

</androidx.cardview.widget.CardView>

</LinearLayout>

</androidx.core.widget.NestedScrollView>

</androidx.coordinatorlayout.widget.CoordinatorLayout>

**fragment\_cart.xml**

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout

xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="8dp"

android:background="@color/backgorund\_color">

<androidx.recyclerview.widget.RecyclerView

android:layout\_above="@+id/group\_place\_holder"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:visibility="gone"

android:id="@+id/recycler\_cart"/>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerInParent="true"

android:text="В корзине ничего нет"

android:id="@+id/txt\_empty\_cart"

android:textSize="35sp"

android:layout\_marginTop="8dp"

android:layout\_marginStart="8dp"

android:layout\_marginEnd="8dp"

/>

<androidx.cardview.widget.CardView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:id="@+id/group\_place\_holder">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical"

android:padding="8dp"

android:layout\_margin="8dp">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/txt\_total\_price"

android:layout\_marginStart="8dp"

android:layout\_marginTop="8dp"

android:layout\_marginEnd="8dp"

android:textSize="24sp"

android:text="Всего: 0,0"/>

<com.google.android.material.button.MaterialButton

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Оформить заказ"

android:id="@+id/btn\_order"/>

</LinearLayout>

</androidx.cardview.widget.CardView>

</RelativeLayout>

**activity\_home.xml**

<?xml version="1.0" encoding="utf-8"?>

<androidx.drawerlayout.widget.DrawerLayout xmlns:android="http://schemas.android.com/apk/res/android"

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

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/drawer\_layout"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:fitsSystemWindows="true"

tools:openDrawer="start">

<include

layout="@layout/app\_bar\_home"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" />

<com.google.android.material.navigation.NavigationView

android:id="@+id/nav\_view"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_gravity="start"

android:background="@android:color/white"

android:fitsSystemWindows="true"

app:headerLayout="@layout/nav\_header\_home"

app:menu="@menu/activity\_home\_drawer" />

</androidx.drawerlayout.widget.DrawerLayout>