Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ И. В. Евдокимов
подпись инициалы, фамилия
«___»_____ 2018г.

БАКАЛАВРСКАЯ РАБОТА

09.03.04 Программная инженерия
код и наименование специальности

Разработка мобильного приложения для курьерской службы предприятия «NinjaPizza»
тема

Руководитель	доцент кафедры «Информатика», канд. тех. н. должность, ученая степень	_____ подпись, дата	Тынченко В. В. фамилия, инициалы
Выпускник		_____ подпись, дата	Вельтер Д. В. фамилия, инициалы

Красноярск 2018

РЕФЕРАТ

Выпускная квалификационная работа по теме «Разработка мобильного приложения для курьерской службы предприятия «NinjaPizza»» содержит 41 страницу текстового материала, 29 рисунков, 11 использованных источников.

Цель выпускной квалификационной работы заключается в улучшении бизнес-процессов обработки и доставки заказов на предприятии «NinjaPizza» посредством разработки и последующего внедрения мобильного приложения.

Выпускная квалификационная работа содержит введение, 3 главы, заключение, список использованных источников.

Первая глава содержит общие сведения об анализе предметной области, обосновании внедрения программного средства, а также анализ похожих программных средств.

Вторая глава включает проектирование программного средства, описание архитектуры и используемых модулей при разработке.

Третья глава содержит описание функциональности и работы приложения.

Заключение содержит анализ полученных результатов.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На сегодняшний день, роль мобильных устройств в жизни людей занимает одно из первых мест. В 2018 году практически у каждого первого жителя современного города имеется личный мобильный телефон, а доля смартфонов при этом также занимает большую часть. Это обосновано тем, что мобильные устройства, помимо выполнения своей прямой обязанности совершать звонки имеет массу других функций. Они умеют делать фотографии, снимать видео, воспроизводить медиа-контент, выходить во всемирную сеть, а также с помощью всевозможных датчиков определять местоположения владельца, измерять пульс и так далее. В одном устройстве люди совмещают навигатор, фотоаппарат, серфинг интернета и социальных сетей.

Также, мобильные устройства все чаще используются на предприятиях. Это позволяет руководителям оптимизировать бизнес-процессы компании. Так, в фирмах такси, смартфоны уже давно заменили классические радио-рации.

В данной работе будет рассмотрен процесс оптимизации работы курьерской службы одной из Красноярских компаний по доставке еды.

Цель выпускной квалификационной работы заключается в улучшении бизнес-процессов обработки и доставки заказов на предприятии «NinjaPizza» посредством разработки и последующего внедрения мобильного приложения.

Для достижения поставленной цели требуется решить следующие задачи:

- провести анализ предметной области и обосновать необходимость внедрения программного средства для оптимизации работы курьерской службы предприятия;
- сформулировать постановку задачи автоматизации;
- исследовать рынок существующих программных продуктов аналогичного назначения;
- обосновать выбор средств реализации проекта;
- произвести проектирование и реализацию программного средства.

1 Аналитическая часть

1.1 Анализ предметной области

Компания «NinjaPizza» осуществляет доставку горячей еды, в частности, доставляет пиццу, салаты, закуски и так далее по всему городу Красноярску и прилегающим к нему населенным пунктам. Компания имеет три филиала по городу и обладает большим штатом сотрудников. Для автоматизации бизнес-процессов фирма имеет свою собственную CRM-систему. Она автоматизирует процесс оформления и последующего управления заказами, позволяет вести различные виды учета и статистику.

Для понимания общих бизнес-процессов работы компании были составлены диаграммы цепочки добавленного качества (VAD диаграмма) (рисунок 1.1) и диаграмма цепочки процесса, управляемой событиями (EPC диаграмма) (рисунок 1.2).

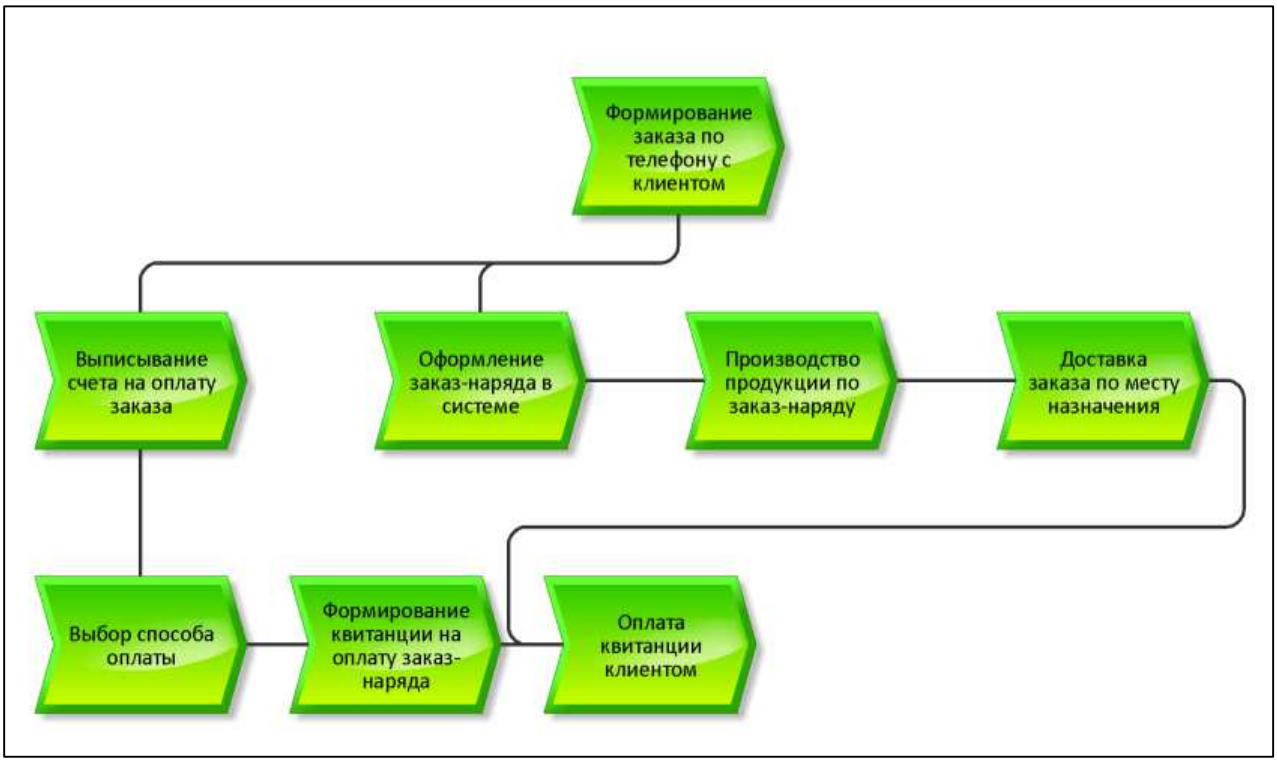

Рисунок 1.1 – VAD диаграмма

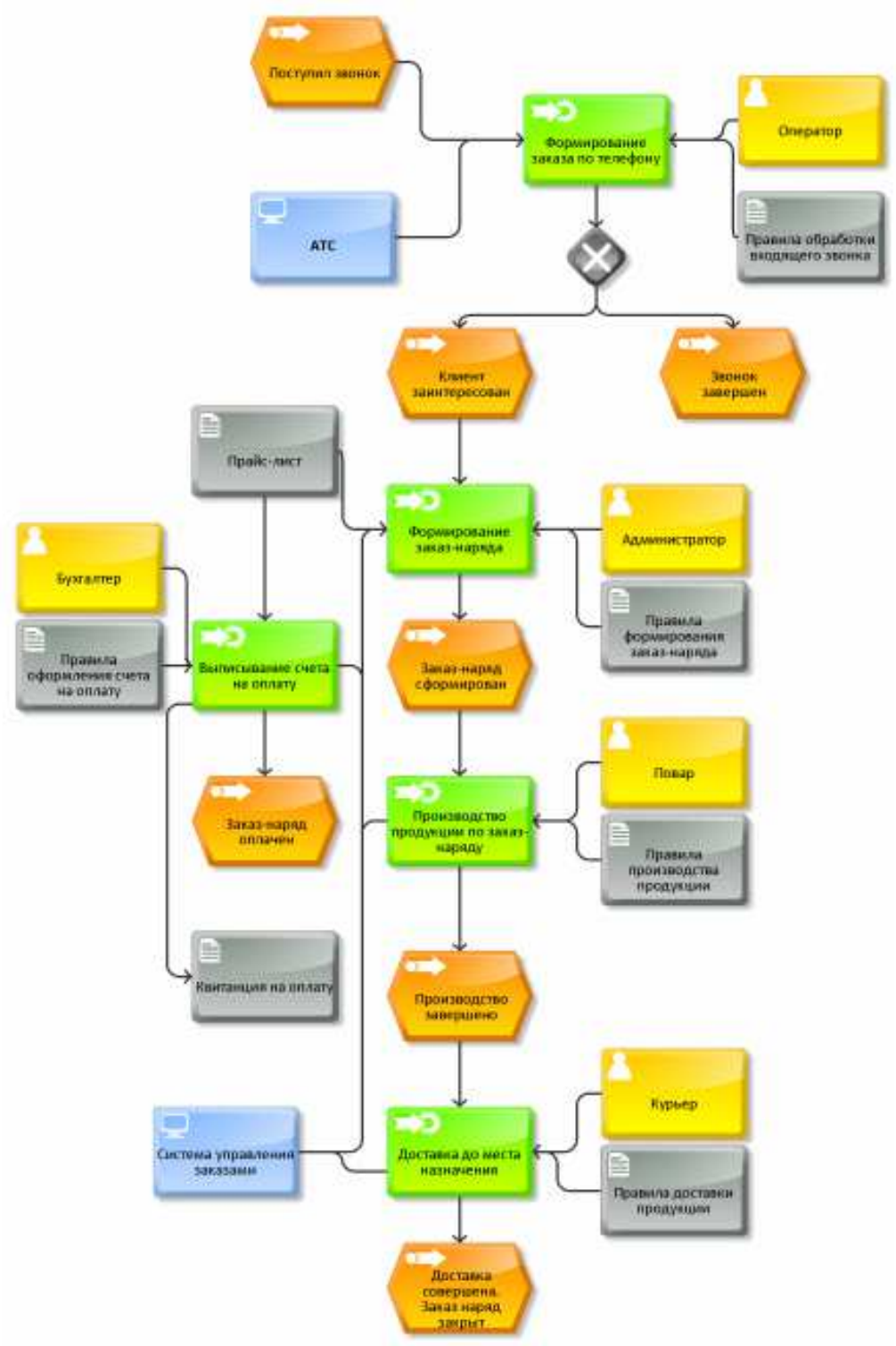

Рисунок 1.2 – EPC Диаграмма

Как видно из EPC диаграммы, система управлениями заказами участвует почти на каждом этапе производства заказа. В общем виде, касательно рассматриваемой предметной области, в системе имеется модель входящей

заявки, поступившей от клиента компании, который желает оформить заказ. У заказа на протяжении всего производства меняется его статус, а именно:

- в ожидании;
- в производстве;
- в печи;
- готов к отправке;
- в пути;
- доставлен;
- завершен.

Влиять на изменение статуса заказа могут операторы call-центра, администраторы, работники кухни, а также курьеры. По готовности заказа администратор отправляет данный заказ с курьером, попутно закрепив его в системе за определенным курьером. Курьер при доставке заказ-наряда по адресу должен отметить заказ в системе, тем самым изменив его статус с «В пути» на «Доставлен». Это необходимо для ведения статистики времени доставки и других важных показателей.

1.2 Обоснование необходимости внедрения программного средства

На данный момент в системе предусмотрена браузерная, не адаптированная под мобильные экраны версия, несущая одну единственную функцию об отметке заказа. При работе у курьера в пути возникает множество вопросов, которые необходимо решить, в частности:

- определить местоположение конечного адресата перед началом пути;
- вручную построить маршрут в навигаторе до конечного адресата;
- при необходимости связаться с клиентом, вручную набрав номер телефона;

– при необходимости связаться с администратором для решения возникших вопросов;

– при необходимости передать данные о заказе операторам call-центра, либо администратору для решения организационных или иных возникших вопросов;

– не забыть вовремя отметить заказ, чтобы не испортить статистику.

Решением всех этих проблем курьеру, зачастую, приходится разбираться в пути, что влечет за собой отвлечение внимания во время управления автомобилем. При плотном городском трафике это заметно сказывается на безопасности дорожного движения и может повлечь за собой серьезные последствия. Разрабатываемое приложение призвано сэкономить сотрудникам драгоценное время, автоматизировать большую часть выполняемых курьером действий, тем самым избавив его от частой ручной работы. За счет этого будет повышена производительность сотрудников благодаря экономии времени, а также снижению трудоемкости операций. Кроме того, будет повышена безопасность за рулем, так как водители будут меньше отвлекаться на выполнение ручной работы. Помимо этого, использование мобильного приложения исключит возможность введения курьером неверных данных в систему навигации, а также при дозвоне клиенту.

1.3 Требования к программному средству

С учетом всех факторов к программному средству были предъявлены следующие требования:

– отображение общего списка заказов в сокращенном виде;

– отображение полной информации о заказе в отдельном окне;

– отображение конечного адреса на встроенной карте в окне заказа;

– построение маршрута в стороннем навигаторе по нажатию кнопки из приложения;

– возможность сделать звонок клиенту прямо из приложения;

– возможность отправки краткой информации о заказе в мессенджер WhatsApp;

– возможность сделать звонок администратору прямо из приложения;

– поддержка ночного режима в приложении при движении в темное время суток;

– ведение статистики выполненных заказов;

– создание, хранение и просмотр своих рабочих смен на неделю;

– уведомление пользователя при малом остаточном времени до доставки заказа;

– авторизация в приложении;

– оформление внешнего вида приложения в соответствии со стилисткой компании;

– иметь интуитивно-понятный удобный для пользователя интерфейс;

– обрабатывать исключительные ситуации, быть отказоустойчивым;

1.4 Анализ программных продуктов аналогичного назначения

В процессе поиска схожих программных продуктов были выделены два решения от разных производителей, а именно:

– мобильное приложение «ОПТИМУМ Курьер» от компании «CDC Центр Корпоративных Разработок» [1];

– мобильное приложение «Курьерская служба 2008» [2].

Приложение «ОПТИМУМ Курьер» имеет довольно устаревший пользовательский интерфейс, но обладает большим количеством функций, а именно:

– предоставление курьеру информации о точках доставки (клиентах) и запланированных маршрутах;

– передача накладных отгрузки на мобильное устройство курьера;

– отслеживание активностей курьера: посещения точки доставки (время посещения, точное место посещения, результат визита), изменения статуса доставки заказа («Доставлен», «Не доставлен», «Отказ от заказа», «Клиента нет дома»);

– обозначение на карте планового маршрута и текущего местоположения курьера;

– сканирование и распознавание QR- и штрих-кодов накладных;

– расчет количества денег к сдаче в кассу (считается по типу накладных и фактам отгрузки по ним товара);

– оперативный обмен текстовыми сообщениями с оператором (диспетчером);

– возможность набора телефонного номера из карточки клиента;

– возможность получения оплаты доставки банковской картой;

Второе приложение «Курьерская служба 2008» почти полностью повторяет функционал предыдущего рассмотренного решения.

Однако оба данных приложения работают в одном комплексе со своими системами. Приложение «ОПТИМУМ Курьер» работает в системе «ОПТИМУМ ГИС», а «Курьерская служба 2008» также является частью своей системы.

Проведя анализ данных программных решений и взяв во внимание тот факт, что предприятие работает в собственной CRM-системе, можно сделать вывод о том, что целесообразнее вести разработку собственного приложения, так как оба вышеупомянутых продукта, во-первых, требуют интеграции со своими сервисами, во-вторых слишком перегружены ненужным функционалом, который на рассматриваемом предприятии не используется или представлен в виде иных бизнес-процессов. Помимо всего этого, данные программные продукты не удовлетворяют некоторым требованиям, предъявленным к программному средству в пункте 1.3. А посредством создания собственного программного продукта можно учесть все необходимые требования и

подстроить его под уже существующий порядок ведения деятельности компании.

1.5 Выбор операционной системы

Проанализировав количество потенциальных пользователей будущего приложения были получены результаты, показывающие, какая операционная система установлена на устройствах. На рисунке 1.3 показана диаграмма используемых целевой аудиторией операционных систем в процентном соотношении.

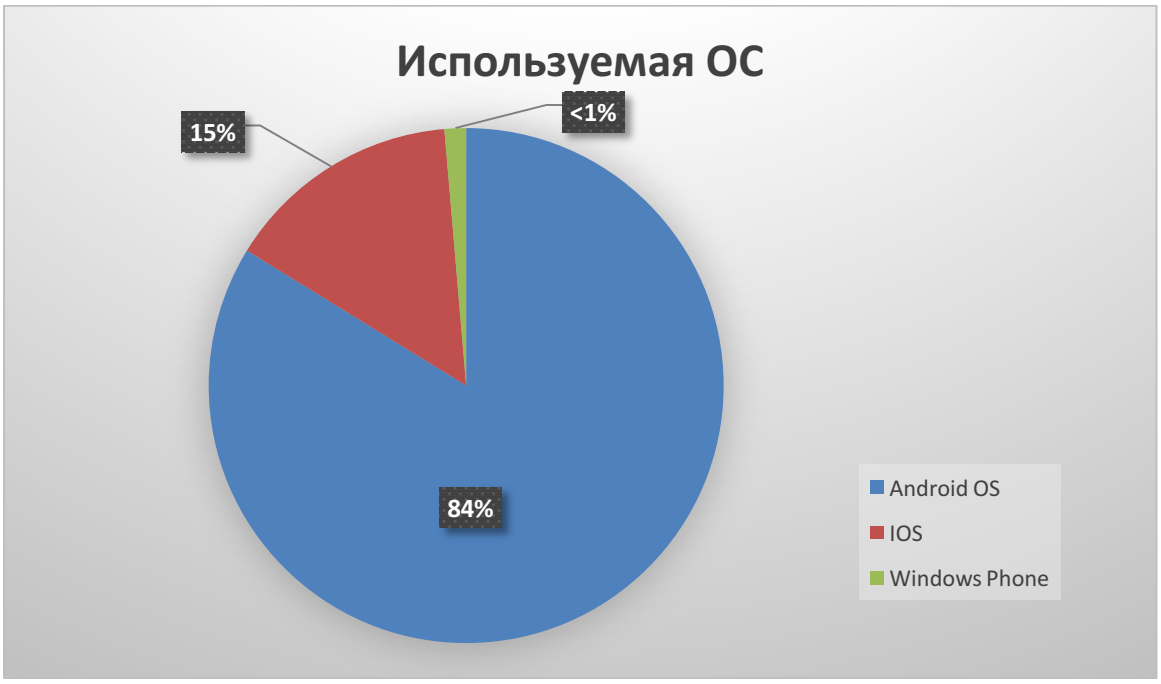

Рисунок. 1.3 – Соотношение операционных систем у будущей аудитории

Отсюда можно сделать вывод, что для первой версии приложения целесообразно вести разработку под операционную систему Android. Это охватит 84% будущих пользователей системы. Также этому способствует более финансово доступная разработка, чем в том случае, если бы она велась под операционную систему iOS. Для Windows Phone устройств разработка является крайне нецелесообразной, так как охватит меньше 1% аудитории. Для всех

остальных операционных систем, отличных от Android, будет предусмотрена максимально упрощенная браузерная версия, выполняющая основную функцию.

В качестве минимальной поддерживаемой версии Android была выбрана версия 5.0. Выбор данной версии рекомендован непосредственно самой компанией Google. Так, по ее мнению, при выборе данной минимальной поддерживаемой версии будет обеспечено покрытие 71.3% устройств на планете. На рисунке 1.4 представлен отчет Google о помощи выбора минимальной поддерживаемой версии Android.

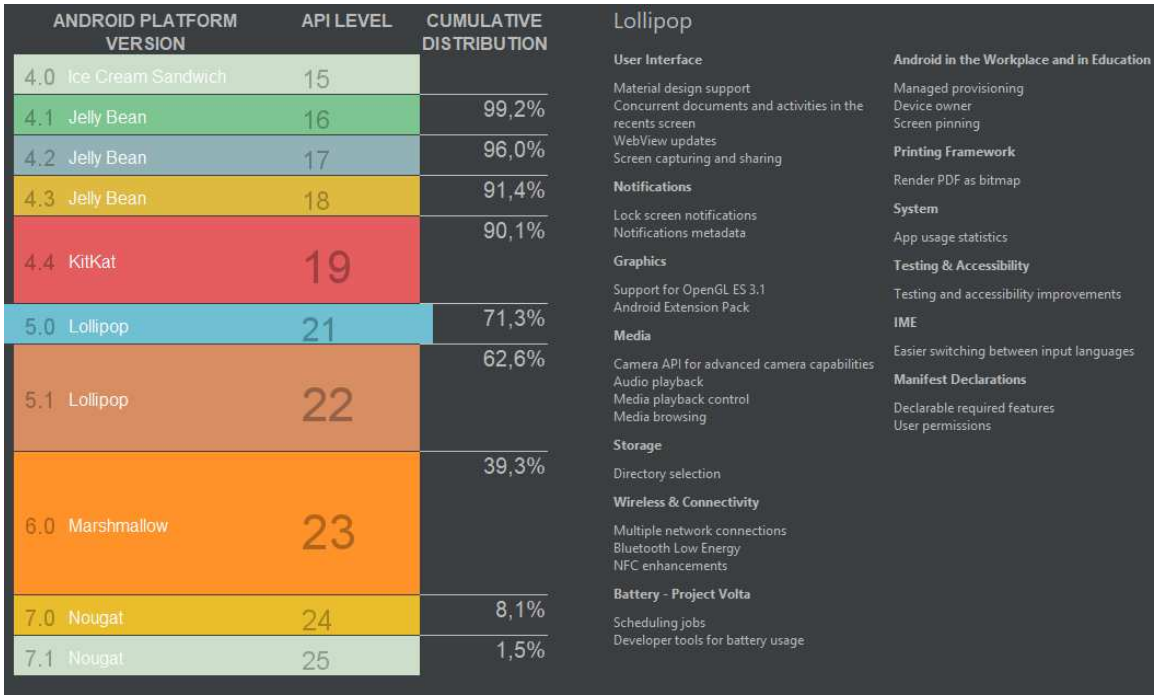

Рисунок 1.4 – Рекомендации Google по выбору версии ОС

Выбор более ранних версий, охватывающих более 90% устройств, не имеет смысла, так как вызовет некоторые проблемы при разработке, ограничит возможность использования современных библиотек и понизит уровень безопасности приложения.

В свете вышесказанного, рассмотрим схему работы Android приложений.

Приложения для операционной системы Android пишутся на языке программирования Java. Инструменты Android SDK (Software Development Kit – комплект разработки программного обеспечения) компилируют написанный код

– и все требуемые файлы данных и ресурсов - в файл APK – программный пакет Android, который представляет собой файл архива с расширением *.apk [3]. Файл *.apk является самостоятельным, в нем уже собраны и скомпилированы все необходимые библиотеки и ресурсы для полноценной работы. Данный пакет можно установить на любое поддерживаемое устройство, работающее на операционной системе Android.

Каждое приложение Android, установленное на устройстве, работает в собственной "песочнице" (изолированной программной среде):

– операционная система Android представляет собой многопользовательскую систему Linux, в которой каждое приложение является отдельным пользователем;

– по умолчанию система назначает каждому приложению уникальный идентификатор пользователя Linux (этот идентификатор используется только системой и неизвестен приложению); система устанавливает полномочия для всех файлов в приложении, с тем чтобы доступ к ним был разрешен только пользователю с идентификатором, назначенным этому приложению;

– у каждого процесса имеется собственная виртуальная машина (ВМ), так что код приложения выполняется изолированно от других приложений;

– по умолчанию каждое приложение выполняется в собственном процессе Linux. Android запускает процесс, когда требуется выполнить какой-либо компонент приложения, а затем завершает процесс, когда он больше не нужен, либо, когда системе требуется освободить память для других приложений.

Так, первый элемент приложений в ОС Android – это класс Activity и его производные. Если отбросить все нюансы, то можно сказать, что это окно приложения со всеми элементами представления (кнопками, бегунками, списками). По сути Activity представляет собой пользовательский интерфейс. Общая схема работы Activity или приложения в целом (если в нем имеется несколько activity) представлена на рисунке 1.5, в котором перечислены все

основные методы, вызываемые системой при первом создании activity, паузе, восстановлении, уничтожении и так далее.

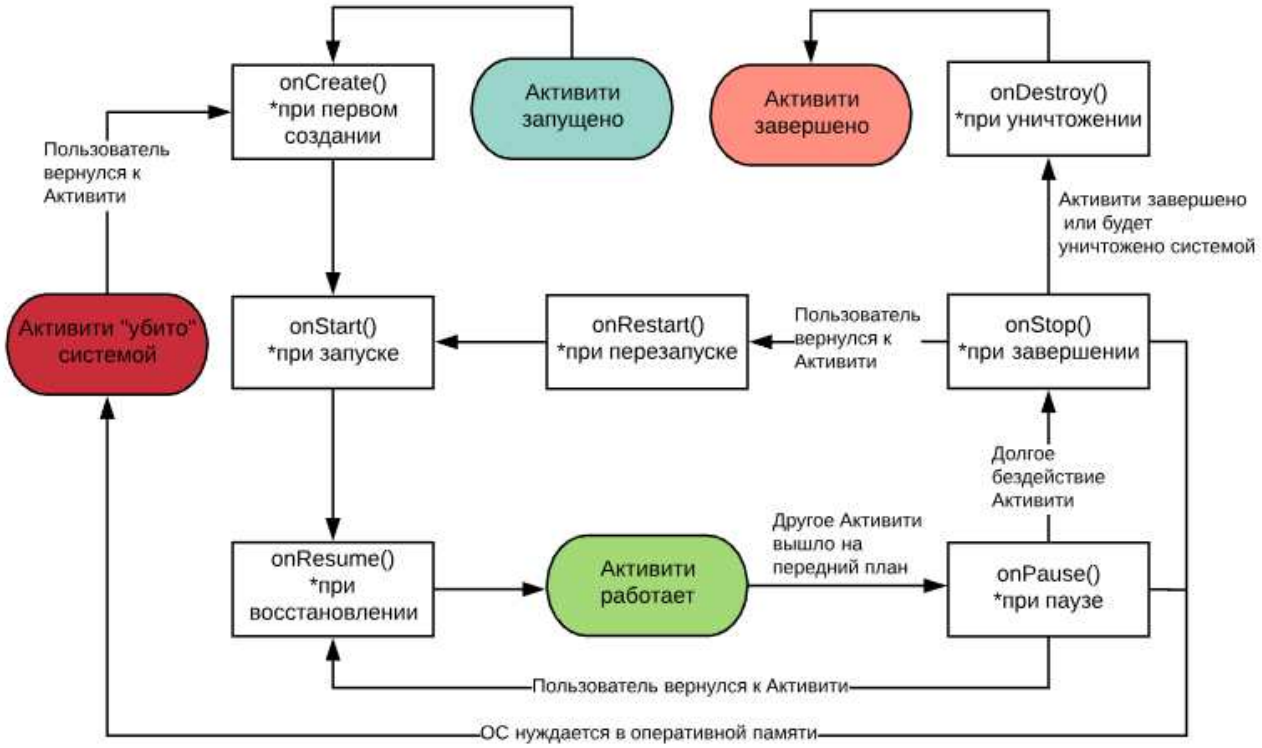

Рисунок 1.5 – Жизненный цикл Activity

Файл манифеста. Для запуска компонента приложения системе Android необходимо знать, что компонент существует. Для этого она читает файл AndroidManifest.xml приложения (файл манифеста). На рисунке 1.6 изображена структура файла манифеста. В этом файле, который должен находиться в корневой папке приложения, должны быть объявлены все компоненты приложения. Помимо объявления компонентов приложения, манифест служит и для других целей, среди которых:

- указание всех полномочий пользователя, которые требуются приложению, например, предоставление доступа к сети или разрешения на использование камеры;
- объявление минимального уровня API, требуемого приложению, с учетом того, какие API-интерфейсы оно использует;

- объявление аппаратных и программных функций, которые необходимы приложению, например, поддержка NFC или наличие фронтальной камеры;
- указание библиотек API, с которыми необходимо связать приложение (отличные от API-интерфейсов платформы Android), например, библиотеки сторонних разработчиков для реализации какой-либо функциональности.

Рисунок 1.6 – Структура файла манифеста

Объявление требований приложения. Существует огромное количество устройств, работающих под управлением Android, и не все они имеют одинаковые функциональные возможности. Чтобы приложение не могло быть установлено на устройствах, в которых отсутствуют функции, необходимые ему, важно четко определить профиль для типов устройств, поддерживаемых

приложением, указав требования к аппаратному и программному обеспечению в файле манифеста. Эти объявления по большей части носят информационный характер, система их не читает. Однако их читают внешние службы, например, Google Play, с целью обеспечения фильтрации для пользователей, которые ищут приложения для своих устройств.

Ресурсы приложения. Приложение Android состоит не только из кода — ему необходимы такие существующие отдельно от исходного кода ресурсы, как изображения, аудиофайлы и все, что связано с визуальным представлением приложения. Например, необходимо определять анимацию, меню, стили, цвета и макет пользовательских интерфейсов операций в файлах XML. Используя ресурсы приложения, можно без труда изменять его различные характеристики, не меняя код, а, кроме того, – путем предоставления наборов альтернативных ресурсов — можно оптимизировать свое приложение для работы с различными конфигурациями устройств (например, для различных языков или размеров экрана).

Для каждого ресурса, включаемого в проект Android, инструменты SDK задают уникальный целочисленный идентификатор, который может использоваться, чтобы сослаться на ресурс из кода приложения или из других ресурсов, определенных в XML. Например, если в приложении имеется файл изображения с именем «image.png» (сохраненный в папке «res/drawable/»), инструменты SDK сформируют идентификатор ресурса под именем «R.drawable.image», с помощью которого на изображение можно будет ссылаться и вставлять его в пользовательский интерфейс.

Один из наиболее важных аспектов предоставления ресурсов отдельно от исходного кода заключается в возможности использовать альтернативные ресурсы для различных конфигураций устройств. Например, определив строки пользовательского интерфейса в XML, вы сможете перевести их на другие языки и сохранить эти переводы в отдельных файлах. Затем по квалификатору языка, добавленному к имени каталога ресурса (например, res/values-fr/ для строк на французском языке) и выбранному пользователем языку система Android

применит к вашему пользовательскому интерфейсу строки на соответствующем языке.

1.5 Выбор средств разработки

Разработка мобильных приложений относительно новая и динамически развивающаяся область. Поэтому работа в этой области требует самых современных средств разработки. На сегодняшний день существует несколько инструментов разработки на языке Java для платформы Android. Одна из таких сред разработки называется Eclipse. Eclipse – это программная платформа с открытым исходным кодом, написанная на языке Java[4]. Основная цель ее создания – повышение производительности процесса разработки ПО. Большинство пользователей Eclipse используют данную платформу как интегрированную среду разработки Java IDE (Integrated Development Environment). Но ее возможности обширнее. Eclipse также располагает средой разработки плагинов PDE (Plugin Development Environment), которой заинтересуются в первую очередь желающие расширить сам Eclipse. В силу того, что эта программная платформа состоит полностью из плагинов, все разработчики инструментариев имеют возможность предложить собственные расширения к ней, предоставив пользователям цельную и последовательную интегрированную среду разработки.

Корпорация Microsoft также располагает собственным продуктом для разработки мобильных приложений. Называется она Xamarin. Данная среда является довольно мощным средством для написания приложений под лидирующие операционные системы на рынке, такие как iOS, Android и Windows Phone. Используя данную среду, разработчики имеют возможность создавать кроссплатформенные приложения для вышеперечисленных операционных систем. Разработка ведется на языке программирования C# с использованием платформы «.NET». Среда позволяет создавать единый пользовательских интерфейс для всех платформ с использованием библиотеки Xamarin.Forms.

В мае 2013 года корпорация Google представила собственную среду разработки под названием Android Studio [5]. С того времени, Google позиционировала данную среду как официальное средство разработки для платформы Android. Теперь, спустя почти 5 лет, получив большое количество крупных и небольших обновлений, Android Studio приобрела большую популярность среди разработчиков и отодвинула Eclipse на второй план. Остановимся на этой среде разработки и рассмотрим ее преимущества относительно других продуктов и в рамках разработки текущего проекта. Android Studio предназначена специально для разработки приложений под операционную систему Android, более того, данная среда разработана именно той компанией, которая владеет и поддерживает саму ОС Android. В связи с этим, IDE Android Studio имеет ряд преимуществ перед своими конкурентами, а именно:

- расширенный редактор макетов: WYSIWYG, способность работать с UI компонентами при помощи Drag-and-Drop, функция предпросмотра макета на нескольких конфигурациях экрана;
- сборка приложений, основанная на Gradle;
- Различные виды сборок и генерация нескольких *.apk файлов;
- Рефакторинг кода;
- Статический анализатор кода (Lint), позволяющий находить проблемы производительности, несовместимости версий и другое;
- Встроенный ProGuard и утилита для подписывания приложений;
- Шаблоны основных макетов и компонентов Android;
- Поддержка разработки приложений для Android Wear и Android TV;
- Встроенная поддержка Google Cloud Platform, которая включает в себя интеграцию с сервисами Google Cloud Messaging и App Engine.

Проанализировав вышеперечисленное, можно сделать вывод о том, что для данного решения, оптимальным вариантом будет использование среды разработки Android Studio, т.к. на данный момент времени первая версия

приложения будет выпускаться для операционной системы Android, а данная среда как раз хорошо оптимизирована на разработку программ именно для системы Android.

1.6.1 Среда разработки Android Studio

Android Studio — это интегрированная среда разработки (IDE) для работы с платформой Android, анонсированная 16 мая 2013 года на конференции Google I/O. Первая стабильная версия 1.0 была выпущена в декабре 2014 года, тогда же прекратилась поддержка плагина Android Development Tools (ADT) для Eclipse.

Android Studio, основанная на программном обеспечении IntelliJ IDEA от компании JetBrains, - официальное средство разработки Android приложений. Данная среда разработки доступна для Windows, OS X и Linux.

На рисунках 1.7 и 1.8 показаны рабочие области редактора текстовых файлов и редактора макетов интерфейса, а также общий вид и компоновка панели инструментов разработчика в среде Android Studio.

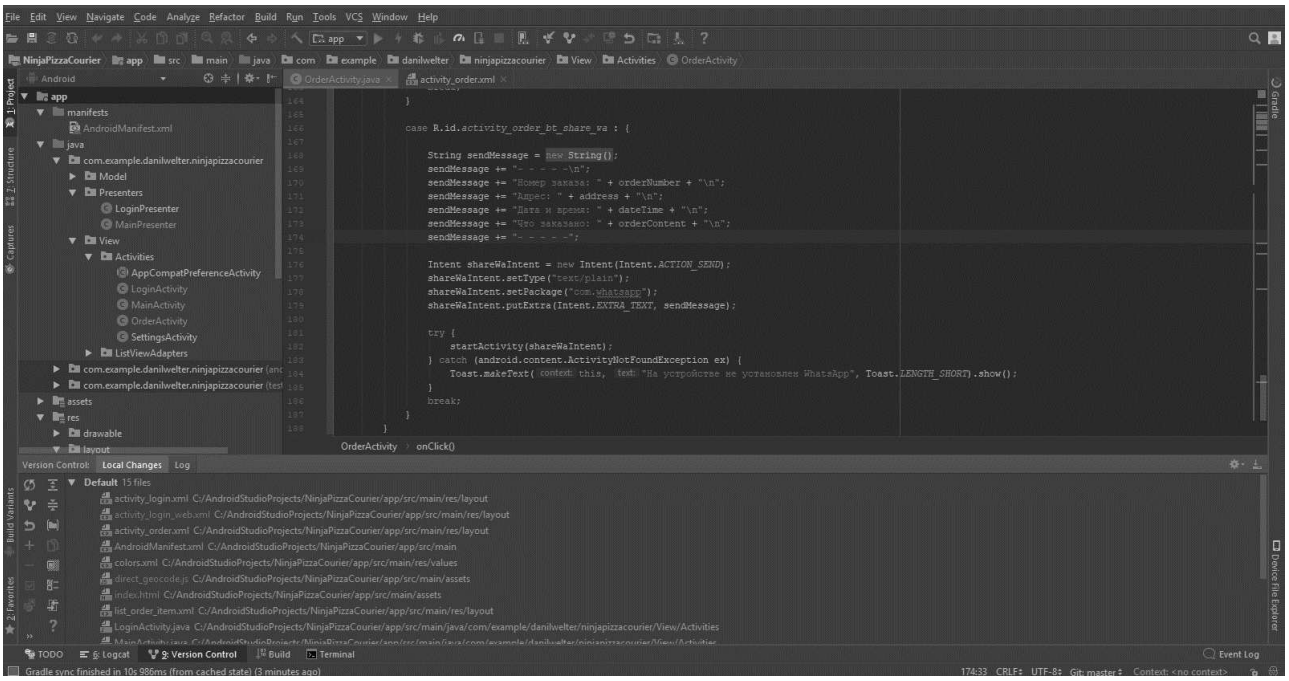

Рисунок 1.7 – Окно редактора текстовых файлов в Android Studio

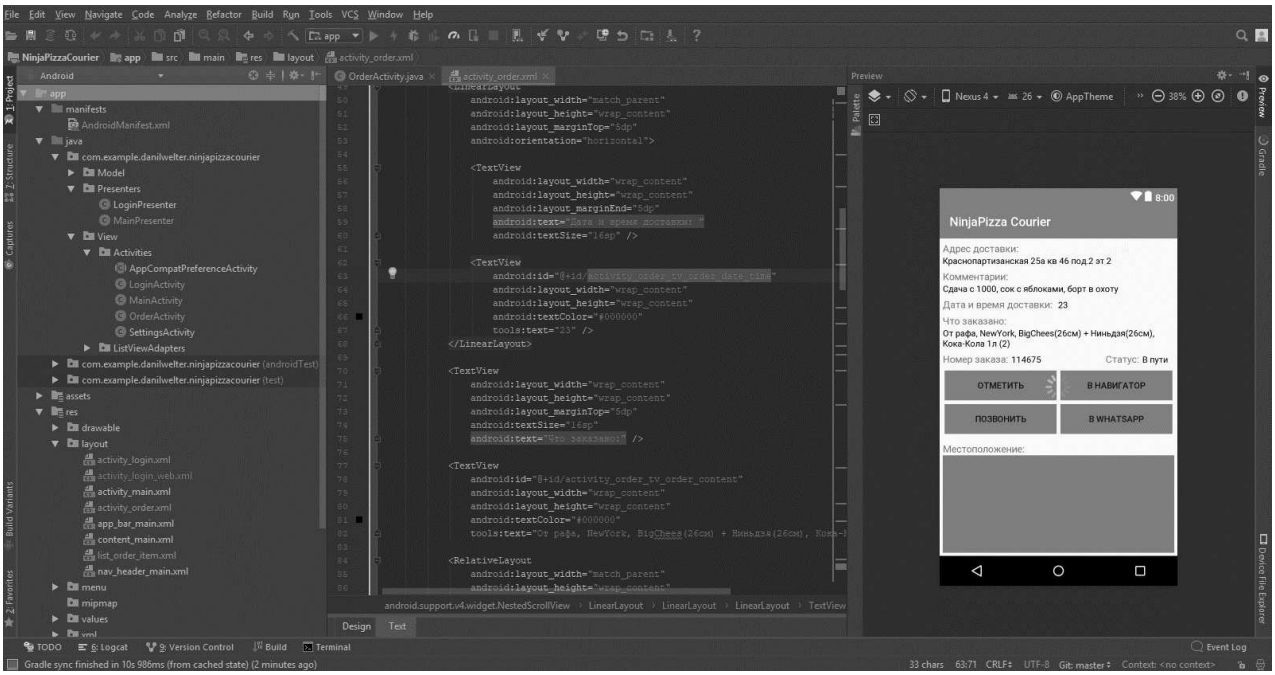

Рисунок 1.8 – Окно редактора макетов разметки пользовательского интерфейса

Рабочая область в Android Studio имеет широкий диапазон пользовательских настроек, включая цветовую гамму, расположение элементов, возможность изменения и задания собственных горячих клавиш, так что данная IDE легко настраивается под каждого отдельно взятого разработчика. Также присутствует синхронизация с Google-аккаунтом, которая позволяет использовать все преимущества облачных сервисов Google.

1.6.2 Диспетчер виртуальных устройств AVD (Android Virtual Device)

Так как разработка ведется под мобильные приложения, то нам необходимо каким-либо образом тестировать эти самые приложения. Здесь имеется два варианта – либо подключать физическое устройство к компьютеру с использованием режима отладки, либо использовать виртуальную машину. В первом случае мы имеем возможность тестировать свои приложения на реальном устройстве, но как показывает практика, зачастую, тесты проводятся на одном или двух-трех физических устройствах. Этого набора часто не хватает для проведения тестирования, так как мы имеем всего несколько комбинаций

диагонали экрана, его разрешения и версии операционной системы. Здесь к нам на помощь приходит диспетчер виртуальных устройств AVD. С его помощью мы можем эмулировать запуск операционной системы Android на Windows или других системах, на которых ведется разработка. Данная эмуляция помогает нам создать почти любую конфигурацию желаемого устройства. Мы можем запускать почти любую комбинацию, а именно, устанавливать желаемое разрешение, диагональ экрана, версию операционной системы, добавлять или убирать поддержку всевозможных датчиков.

На данный момент предлагается несколько решений от разных производителей программного обеспечения, но мы подробнее остановимся на встроенном эмуляторе в среду разработки Android Studio. Для того, чтобы настроить эмулятор, необходимо открыть соответствующие настройки в Android Studio, а именно, меню Virtual Device Configurator. Он имеет вид, продемонстрированный на рисунке 1.9.

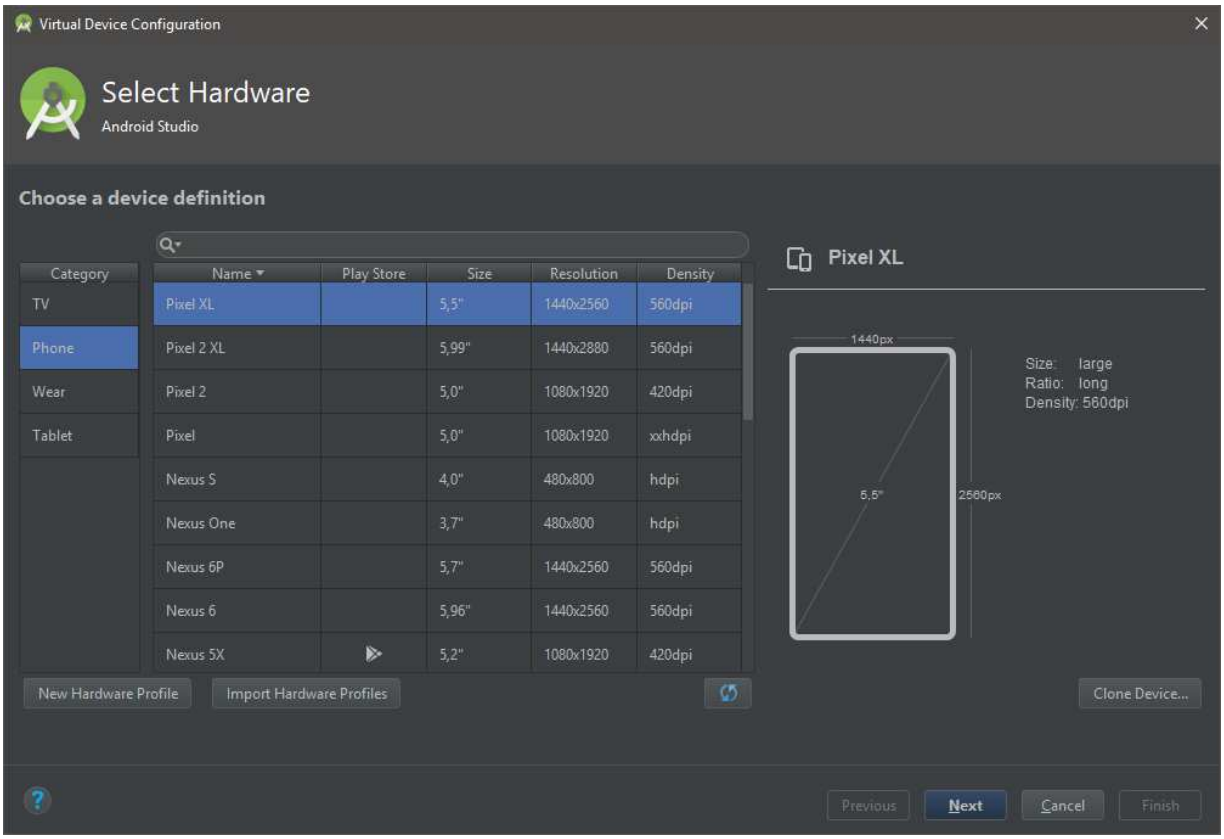

Рисунок 1.9 – Окно конфигуратора виртуальных устройств в Android Studio

Как видно из скриншота, в данном разделе мы можем выбрать уже предустановленные разработчиками IDE конфигурации устройств. Помимо эмуляции смартфонов, здесь также можно запустить эмулятор планшета, наручных часов и телевизора под управлением Android OS. Также можно создать свою собственную конфигурацию, нажав на кнопку «New hardware profile». Окно конфигурации собственного устройства изображено на рисунке 1.10.

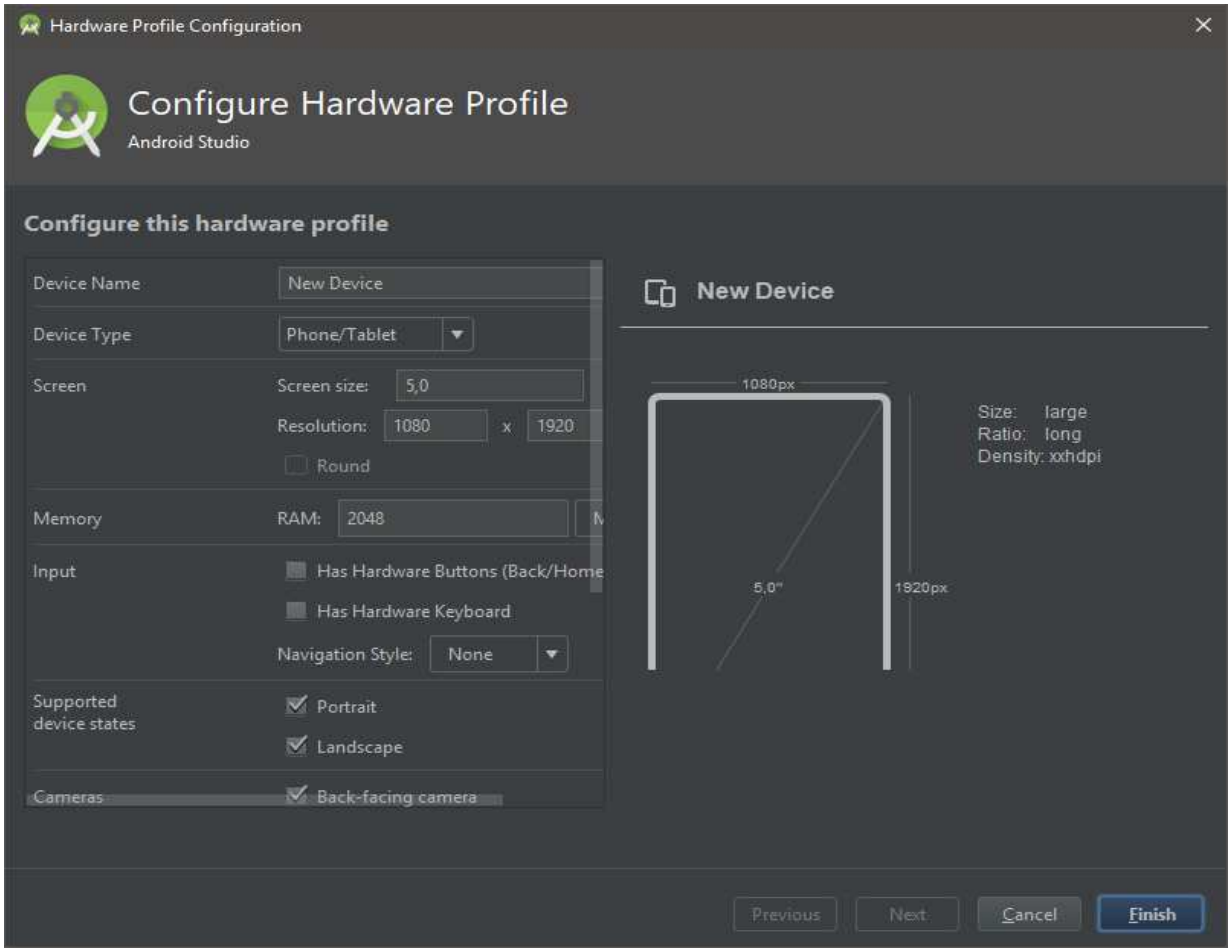

Рисунок 1.10 – Окно конфигурации собственного устройства.

Здесь, как видно из рисунка, можно задавать диагональ экрана, его разрешение, тип устройства (смартфон, планшет, наручные часы, или устройство Android Auto), объем выделяемой оперативной памяти, поддерживаемую ориентацию экрана, поддержку фронтальной и тыловой камер, различные сенсоры и датчики.

1.6.3 Шаблон проектирования MVP для Android ОС

Для реализации архитектуры проекта был использован шаблон проектирования MVP. MVP - шаблон проектирования пользовательского интерфейса, который был разработан для облегчения автоматического модульного тестирования и улучшения разделения ответственности в презентационной логике (отделения логики от отображения) [6]:

- Модель (англ. Model) - хранит в себе всю бизнес-логику, при необходимости получает данные из хранилища.
- Вид (англ. View) - реализует отображение данных (из Модели), обращается к представителю за обновлениями.
- Представитель (англ. Presenter) - реализует взаимодействие между моделью и представлением.

На рисунке 1.11 представлен принцип работы шаблона MVP.

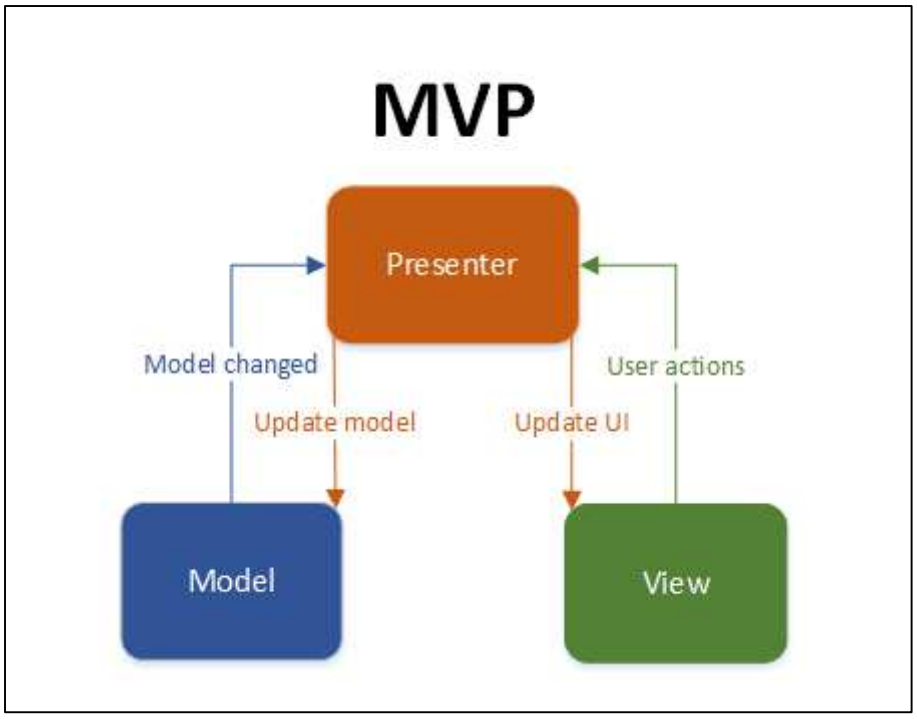

Рисунок 1.11 – Шаблон проектирования MVP

Обычно экземпляр Вида (Представления) создаёт экземпляр Представителя, передавая ему ссылку на себя. При этом Представитель работает с Видом в абстрактном виде, через его интерфейс. Когда вызывается событие представления, оно вызывает конкретный метод Представителя, не имеющего ни параметров, ни возвращаемого значения. Представитель получает необходимые для работы метода данные о состоянии пользовательского интерфейса через интерфейс Вида и через него же передаёт в представление данные из модели и другие результаты своей работы.

1.7 Сравнение и анализ картографических сервисов

В связи с тем, что в приложении необходимо использовать карты, то следует провести анализ существующих решений на рынке и выбрать максимально подходящий сервис с учетом требований программного средства. На сегодняшний день, можно выделить три основных сервиса, способных помочь в решении поставленной задачи. Это сервисы 2ГИС, технологии Яндекс и Google [7][8][9]. Кратко рассмотрим каждую из них, проведем анализ и выберем наиболее подходящий вариант. В таблице 1.1 приведен сравнительный анализ вышеперечисленных сервисов.

Таблица 1.1 – Сравнительный анализ картографических сервисов

Критерий	2ГИС	Яндекс.Карты	Google.Maps
Покрытие	Покрытие в городах присутствия	Лучшее покрытие в России	Лучшее покрытие мира
Детализация объектов	Наилучшая детализация в городах присутствия	Приемлемая детализация в России	Хорошая детализация по миру. В России могут отсутствовать некоторые области
Детализация на уровне зданий	Детализация крупных торговых центров	Нет	Крупные торговые центры
Режим 3D	Да	Да	Да

Окончание таблицы 1.1

Критерий	2ГИС	Яндекс.Карты	Google.Maps
Ночной режим	Да	Да	Да
Актуализация гео-справочной информации	Ежемесячно	Нет данных	Нет данных
Возможность вызова навигатора с параметрами адреса из стороннего приложения	Нет	Да	Да

2ГИС предоставляет JavaScript API, с помощью которого на устройстве можно отобразить, используя компонент WebView, карту и метку с адресом на ней. У Яндекса схожая реализация с 2ГИС. Google предлагает использование обыкновенного API, что избавляет от использования компонента WebView. В реализуемом проекте потребуется использовать технологию прямого геокодирования, поскольку исходными данными является строковый параметр в виде названия улицы, дома и так далее.

Проведя опрос среди будущих пользователей, какие навигаторы они используют при работе, были получены следующие данные, показанные на рисунке 1.12

Рисунок 1.12 – Используемый навигатор при работе

Проанализировав вышеперечисленные данные, можно сделать вывод о том, что использование сервисов Google принесет неудобства будущим пользователям системы, так как они уже привыкли работать с другими сервисами. Можно заметить, что согласно диаграмме, совсем никто не использует сервисы Google карт в своей работе. Это связано с тем, что детализация объектов на карте Google в городе Красноярске недостаточна для полноценной навигации курьеров. Часто, Google карты не могут найти тот или иной адрес, в составе которого присутствуют литеры, строения и тому подобное.

В системе будет предусмотрено два места, где будут использоваться картографические сервисы, а именно, область на экране с картой, непосредственно в самом приложении, в которой будет отображена метка с конечным адресом доставки для наглядного понимания местоположения, относительно районов города, а также кнопка, по нажатию которой будет вызываться уже приложение навигатора для построения маршрута до конечной точки, если это необходимо. В связи с этим было решено остановиться на внедрении Яндекс карт в приложение, так как данный сервис пользуется популярностью среди потенциальных пользователей, а также приложение Яндекс.Навигатор имеет возможность запускаться извне с указанными параметрами конечной точки, в отличии от 2ГИС, который не имеет данной возможности.

1.7.1 Использование картографических сервисов Яндекс

Приложение будет использовать три сервиса Яндекс, а именно, Яндекс.Карты, Яндекс.Геокодер и Яндекс.Навигатор. Яндекс карты предоставляют JavaScript API для использования своих функций. Карты нужно для того, чтобы оценить конечное местоположение адресата прямо в приложении, не тратя время на переходы из одного приложения в другое. Яндекс.Навигатор будет запускаться из приложения с заранее подготовленными параметрами и сразу построит маршрут до необходимой точки. Сервис Геокодер

необходим для того, чтобы использовать прямого геокодирование, так как изначально исходная информация поступает в виде строкового значения адреса. Функция прямого геокодирования осуществит преобразование текстового адреса в абсолютные координаты в виде долготы и широты. Это необходимо для установки места на карте и построения маршрута в навигаторе.

Яндекс предоставляет обширную удобочитаемую документацию на русском языке с примерами и встроенной «песочницей». Это заметно облегчает написание функционала

2 Проектирование программного средства

2.1 Диаграммы вариантов использования

Диаграммы вариантов использования (Use Case Diagram) применяются для наглядного описания работы пользователя с приложением. На рисунке 2.1 приведена общая диаграмма вариантов использования [10].

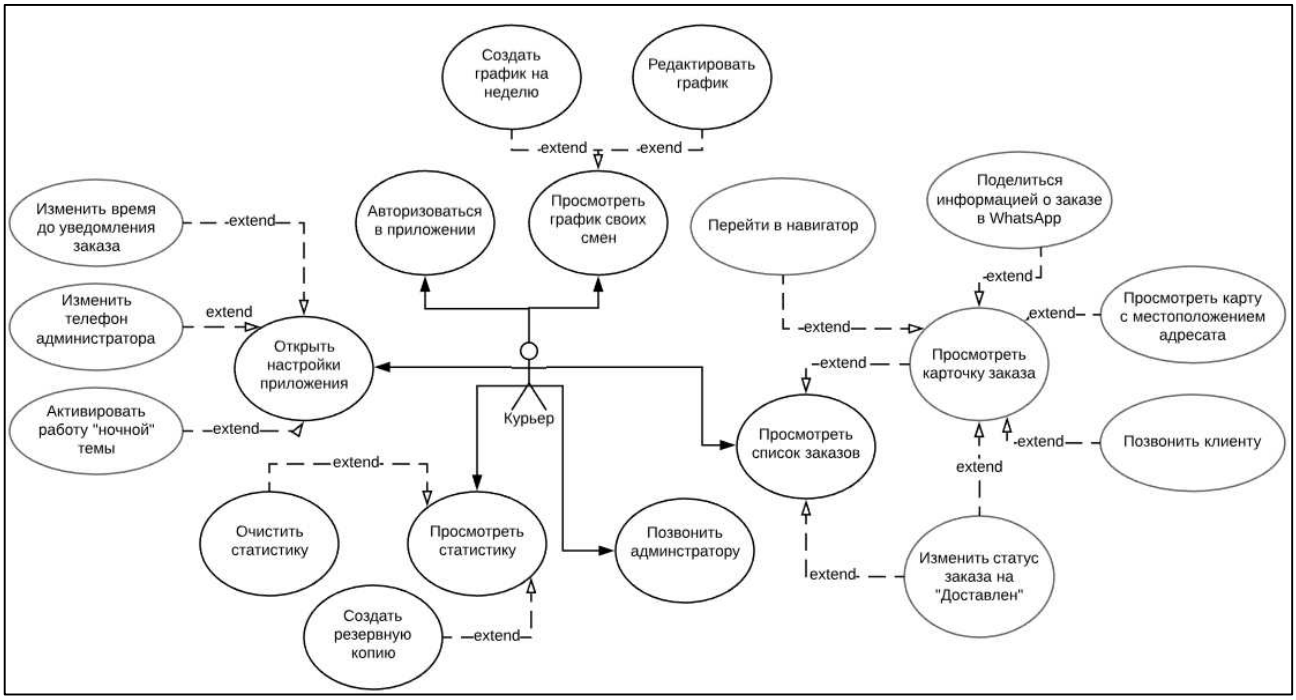

Рисунок 2.1 – Диаграмма вариантов использования

В представленной выше диаграмме вариантов использования фигурируют 4 основных варианта:

– авторизоваться в приложении с помощью логина и пароля;

– просмотреть график своих смен. Данная функция является некими структурированными заметками, которые пользователь, при желании, может добавлять или редактировать, чтобы не забыть свой график в будущем;

– позвонить администратору. Данный вариант использования предполагает звонок на номер администратора, указанного в настройках приложения;

– просмотреть статистику. Здесь пользователь может просмотреть статистику своей работы, а именно, количество выполненных заказов за смену. Данный вариант расширен еще двумя: очистить статистику или создать резервную копию;

– открыть настройки приложения. Пользователь может работать с настройками приложений, а именно, изменить время до уведомления заказа, изменить или добавить телефон оператора, а также активировать режим ночной темы;

– просмотреть список заказов. Данный вариант использования является основным. Он расширен еще двумя вариантами: изменить статус заказа на «Доставлен» и просмотреть карточку заказа. Последний вариант расширяется еще несколькими, а именно, переход в сторонний навигатор для ведения по маршруту, отправка информации о заказе в WhatsApp, просмотр карты с местоположением адресата прямо из приложения, дозвон клиенту, если это необходимо, а также изменение статуса заказа на «Доставлен».

2.2 Архитектура программного продукта

Разрабатываемое приложение является клиент-серверным. Для своей работы ему необходимо получать данные об активных заказах курьера с удаленного сервера. Общая схема работы клиент серверных приложений представлена на рисунке 2.2.

Сначала клиент посылает серверу http-запрос, далее, если сервер может распознать этот запрос, он при необходимости обращается к базе данных, которая вернет серверу запрос. Сервер преобразует данные в необходимый для клиента формат JSON и отправляет ему.

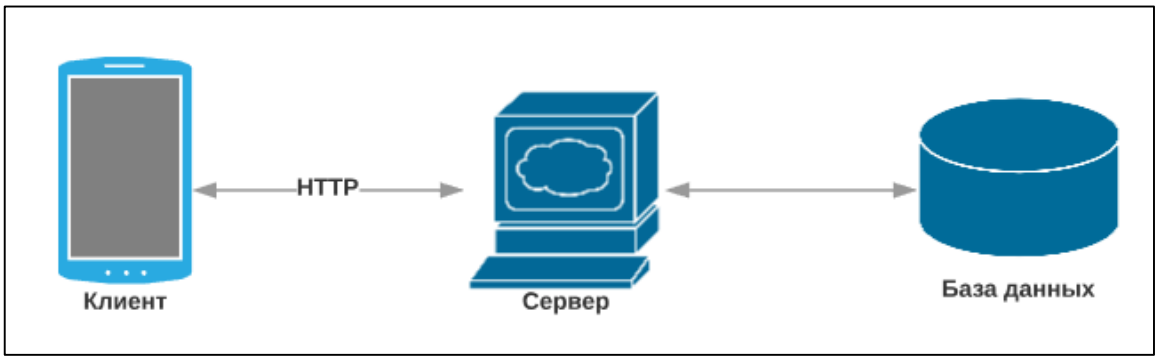

Рисунок 2.2 – Классическая схема работы клиент-серверного приложения

Касательно остальных данных, таких как статистика клиента и график смен, они будут хранится на устройстве в локальной базе данных SQLite [11].

2.2.1 Модель, обмен и хранение данных

При запросе на сервер клиент получит ответ в виде массива активных заказов на указанного в параметре запроса идентификатора курьера. Данная модель представлена на рисунке 2.3.

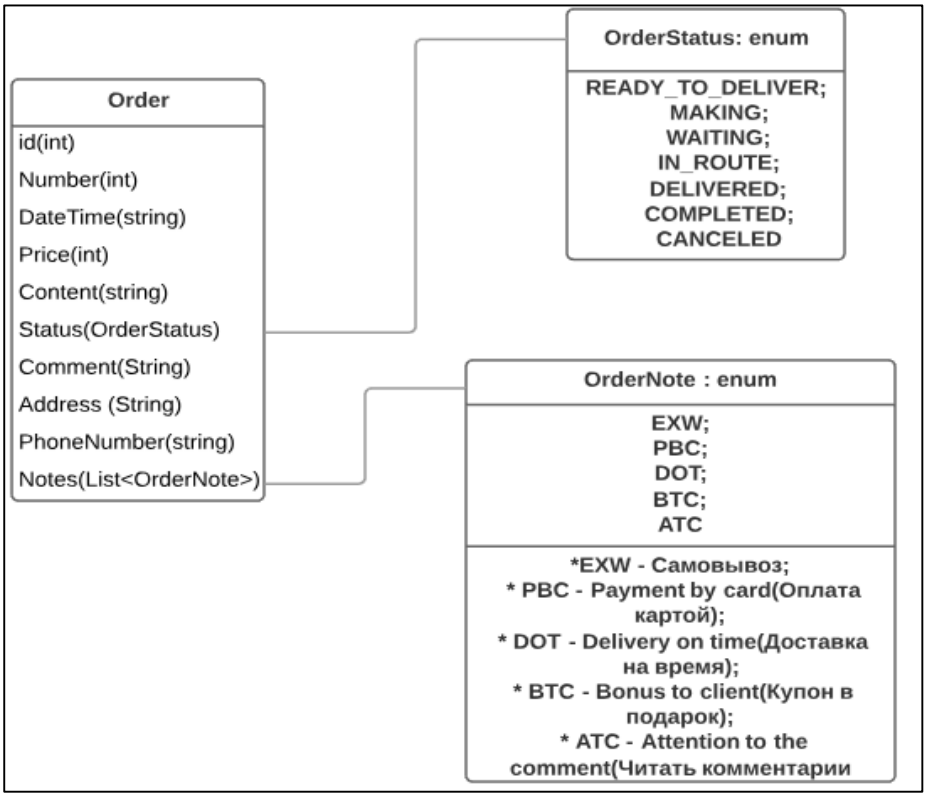

Рисунок 2.3 – Модель получаемого через HTTP-запрос заказа

Схема получения списка заказов приведена на рисунке 2.4.

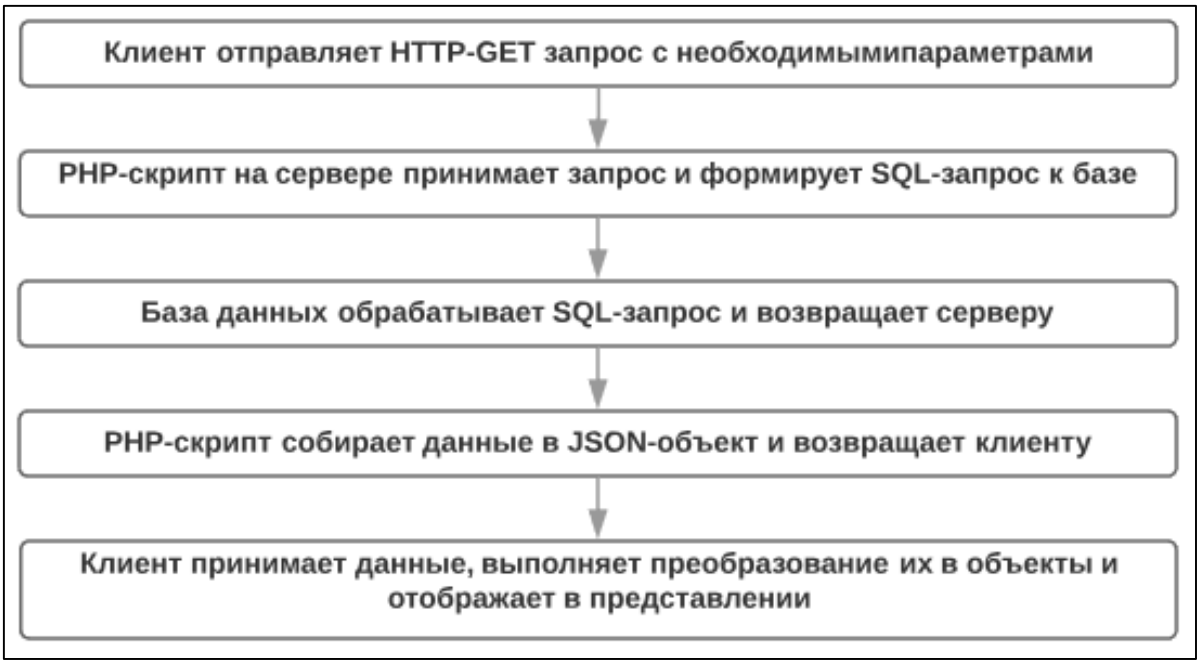

Рисунок 2.4 – Процесс получения данных с сервера

Для работы с данными, связанными со статистикой выполненных заказов и графиком работы курьеров, была реализована встраиваемая СУБД SQLite. Ее движок не является отдельно работающим процессом, с которым взаимодействует программа, а представляет собой библиотеку, с которой программа компонуется, и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций библиотеки SQLite. Это позволяет уменьшить накладные расходы, время отклика и упрощает программу. Схема базы данных для реализации данного функционала представлена на Рисунок 2.5.

Таблица users содержит в себе информацию о пользователях системы на случай, если устройство авторизуется под новым курьером, данные не исказятся и сохранятся. Также это необходимо для того, чтобы иметь возможность расширения приложения в будущем, если будет производится сбор данных в облако. Таблица имеет поля с уникальным идентификатором, логином

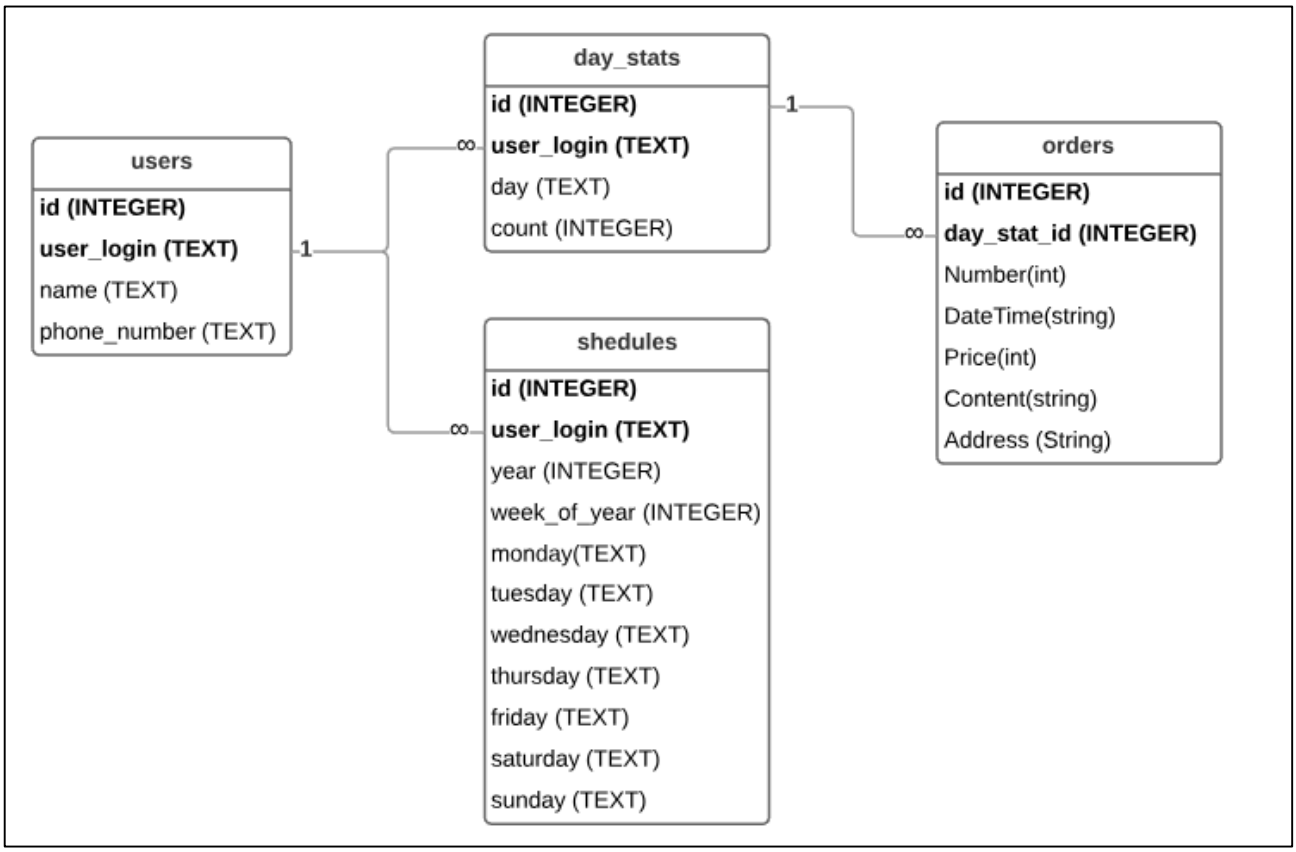

Рисунок 2.5 – Модель базы данных

пользователя, его полным именем и телефонным номером.

Таблица day_stats содержит информацию о совершенных заказах курьера в указанный день. Имеет поля id, user_login – id пользователя, day – день в строковом формате датирования и count – количество совершенных заказов.

Таблица shedules содержит в себе расписания не неделю. По связующему полю user_login можно получить расписания указанного пользователя. Здесь расписания разбиты понедельно.

Таблица orders представляет собой модель заказа. Она требуется для таблицы day_stats. По связующему полю day_stat_id можно получить список заказов, относящихся к определённому дню.

2.2.3 Интеграция картографических сервисов

Для того, чтобы карты отображались в устройстве, их необходимо определить в макете представления. Работа этого сервиса реализована через JavaScript API, следовательно, необходимо определить на разметке компонент WebView, позволяющий загружать веб-страницы прямо в приложении. Начиная с версии Android 5.0, компонент WebView использует отдельное приложение System WebView, которое может обновляться из магазина приложений, вне зависимости от обновлений системы. Компонент WebView необходимо расположить в активности с карточкой заказа, где отображена вся информация о заказе. Для того, чтобы карты отобразились в активности, необходимо в коде программы определить компонент WebView и сослаться на файл макета. Далее необходимо инициализировать настройки WebView – WebSettings и разрешить поддержку выполнения скриптов JavaScript, так как для работы карт на странице подключается сторонняя библиотека. Для подключения библиотеки в html-страницу прописывается скрипт со ссылкой на «api-maps.yandex.ru». Далее настройка производится с помощью собственного JavaScript файла с использованием методов из подключенной выше библиотеки. Для начала использования, необходимо вызвать функцию, которая гарантирует, что API загрузилось, прежде чем начнет исполнятся код, который её использует. В качестве примера на рисунке 2.6 приведен простейший пример вызова карт с указанными координатами и масштабом.

```
var myMap;

//Дождемся готовности API
ymaps.ready(init);

function init () {
    // Создание экземпляра карты и его привязка к контейнеру
    myMap = new ymaps.Map('map', {
        // При инициализации карты обязательно нужно указать
        // её центр и коэффициент масштабирования.
        center: [56.01, 92.85], // Красноярск
        zoom: 10
    });
}
```

Рисунок 2.6 – Создание карты с указанными параметрами

Результатом данного скрипта будет являться карта, продемонстрированная на рисунке 2.7

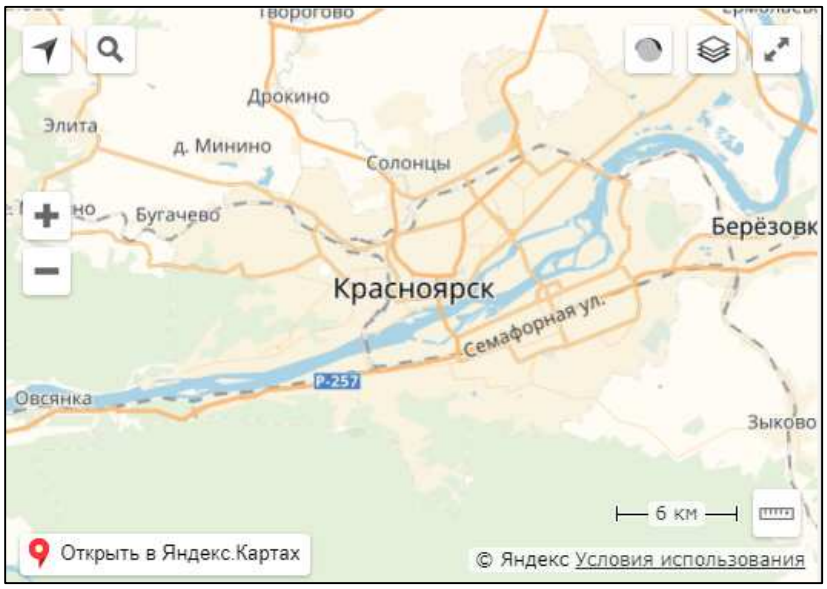

Рисунок 2.7 – Результат выполнения скрипта

Методы прямого геокодирования используются в том случае, когда необходимо, имея адрес в строковом формате, получить координаты. Данные методы также встроены в основной API и не требуют дополнительных библиотек. Геокодирование в приложении необходимо для того, чтобы по адресам заказов, представленных в строковом формате, получать географические координаты для последующей подстановки их в метод, который отображает точку на карте. Данная функция вызывается через метод geocode(). Она принимает несколько параметров, самый главный, это адрес в строковом представлении, затем параметр boundedBy – он определяет область карты, в которой, предположительно, может находиться объект. Это самые важные параметры в данном случае. Чтобы геокодер работал правильно, то есть показывал улицы именно в Красноярске, а не в каком-нибудь другом городе, а входная строка подается именно без названия города, то необходимо изначально отцентрировать карту на городе Красноярске, а затем в параметр boundedBy передать эти значения.

2.3 Пользовательский интерфейс

Пользовательский интерфейс системы состоит из набора файлов макета формата *.xml К каждому активити прилагается файл разметки, а также такие файлы используются при описании элементов списка, всевозможных меню и настроек. Список макетов располагается в директории «../res/layout». Для данного проекта определены следующие макеты, представленные на рисунке 2.8.

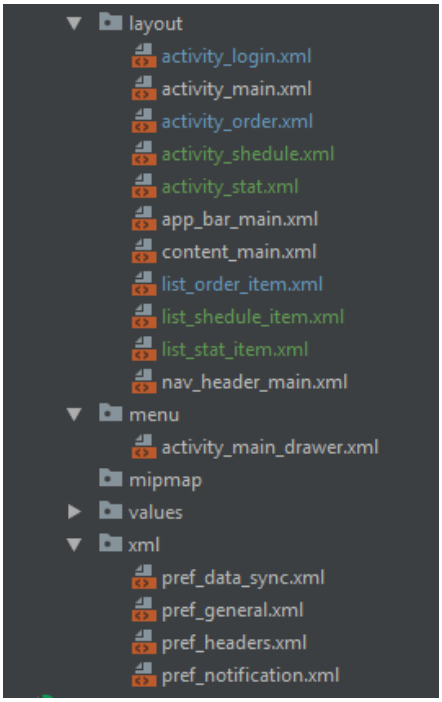

Рисунок 2.8 – Макеты пользовательского интерфейса в приложении

Основной интерфейс приложения, открывающийся при запуске приложения, выполнен в формате NavigationDrawerActivity. Это довольно популярный шаблон макета. Все приложения Google-сервисов выполнены в данном стиле. Макет достаточно удобный, интуитивно понятый и позволяет обеспечить довольно быструю навигацию по элементам меню. Его суть заключается в выдвижном боковом меню, откуда доступны основные функции приложения. Пример такого меню представлен на рисунке 2.9, здесь меню находится в активном состоянии.

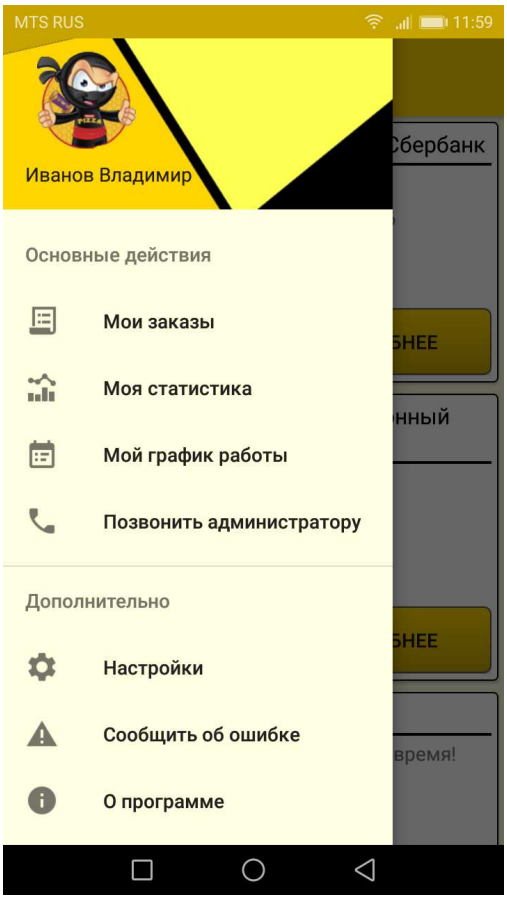

Рисунок 2.9 – Активное меню NavigationDrawer

Для работы с заказами был предусмотрен следующий интерфейс. Если пользователю не нужна дополнительная информация, то ему достаточно будет сокращенного списка с заказами, в котором кратко отображена самая необходимая информация о заказе и имеется кнопка для изменения статуса заказа на «Доставлен». Если пользователь желает увидеть больше информации о заказе, ему будет предложено по кнопке «Подробнее» перейти в карточку заказа. Там ему будут доступны все действия, которые он может выполнить со своим заказом.

Активность со статистикой реализована в виде списка, элементами которого являются дни или недели, содержащие в себе информацию о выполненных заказах. В разделе с графиком работы будет представлен соответствующий список. Раздел с настройками реализован стандартными инструментами с использованием SettingsActivity.

3 Описание работы приложения

3.1 Инсталляция приложения на устройство

Для того, чтобы установить приложение на устройство, необходимо иметь смартфон под управлением операционной системы Android версии не ниже 5.0, способный выходить в сеть интернет, а также должна быть поддержка системы позиционирования GPS. Инсталляция происходит запуском apk файла на устройстве.

3.2 Работа с приложением

3.2.1 Авторизация в приложении

При первом запуске приложения пользователю предлагается выполнить вход с помощью своего логина и пароля. На рисунке 3.1 изображен экран входа в систему.

Рисунок 3.1 – Экран входа в систему

После ввода своих учетных данных, пользователь при желании может поставить галку «Оставаться в системе», чтобы при следующем запуске приложения не приходилось заново авторизовываться. Свои учетные данные курьер может получить у администратора.

3.2.2 Боковое меню приложения

При переходе на основной экран приложения, проведя пальцем от левого края экрана, можно вызвать боковое меню приложения, продемонстрированного на рисунке 3.2.

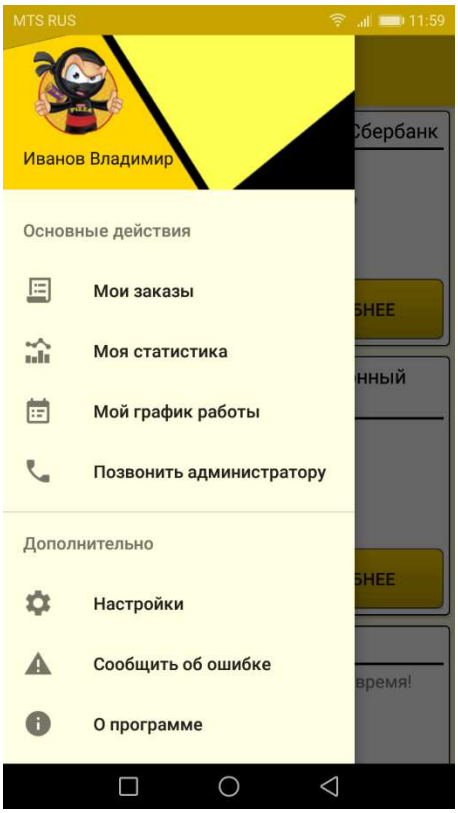

Рисунок 3.2 – Боковое меню приложения

Данное навигационное меню является основным в приложении. Отсюда можно перейти в раздел со списком заказов, со статистикой и графиком работы. Также отсюда можно осуществить звонок администратору. В разделе «Дополнительно» пользователь может перейти в настройки приложения,

сообщить о найденной ошибке через форму обратной связи с разработчиком или можно посмотреть информацию о программе, ее версию.

3.2.3 Работа с настройками

На данный момент в приложении имеется несколько настроек. Они разделены на две категории: «Основные» и «Уведомления». Раздел настроек изображен на рисунке 3.3.

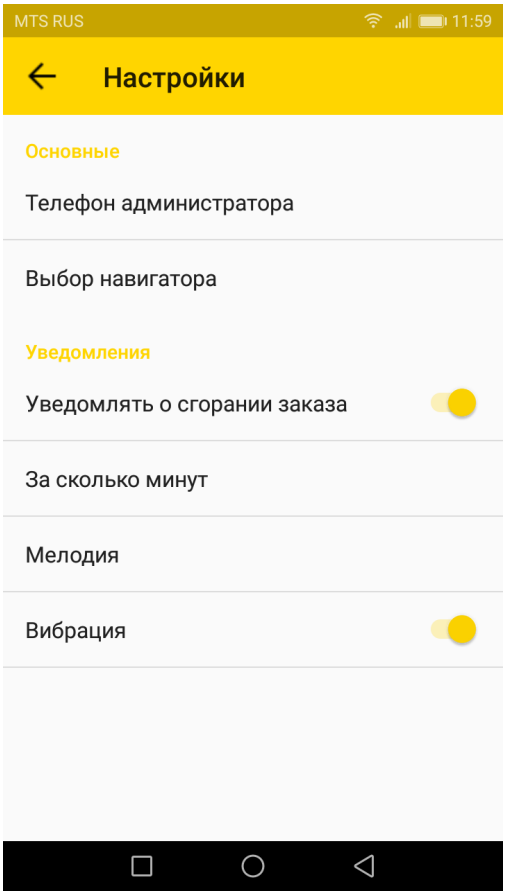

Рисунок 3.3 – Настройки приложения

В основных настройках расположена форма ввода телефона администратора, по которому будет совершаться вызов из главного меню приложения, а также выбор приложения для навигации. Данное меню выбора представлено на рисунке 3.4.

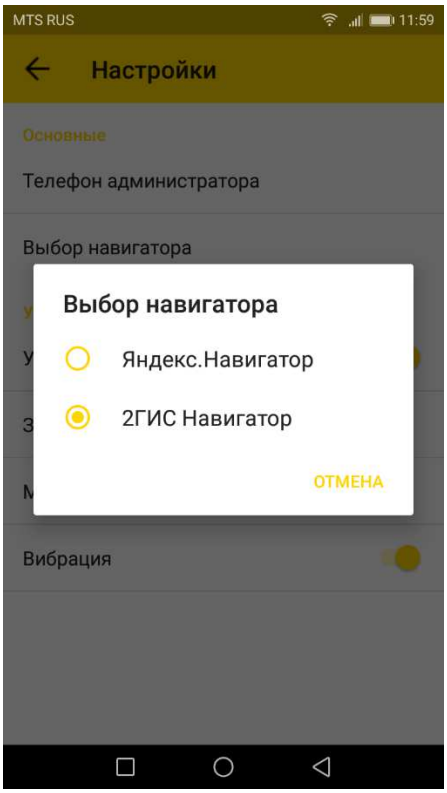

Рисунок 3.4 – Меню выбора навигатора

В разделе уведомлений располагаются настройки оповещения о сгорании заказа. Данное уведомление можно включить, выбрать количество минут, за которые пользователя необходимо уведомить о сгорании заказа, а также выбрать мелодию и активировать вибрацию.

3.2.4 Работа со списком заказов

При входе в приложение список активных заказов выводится по умолчанию. Также, данный список можно вызвать из бокового меню приложения. Для обновления списка заказов пользователю необходимо потянуть вниз от верхней части экрана. Данный способ обновления реализован на многих современных приложениях. На рисунке 3.5 продемонстрирован список заказов в сокращенном виде.

Рисунок 3.5 – Окно со списком заказов

Под каждым заказом расположено две кнопки. Первая кнопка отвечает за изменение статуса заказа на «Доставлен», вторая кнопка позволяет перейти в карточку заказа для подробной работы с ним. В окне списка заказов отображается сокращенная информация о заказе: его адрес, комментарии и количество минут до доставки.

При переходе в карточку заказа по нажатию кнопки «Подробнее» пользователю открывается экран, представленный на рисунке 3.6.

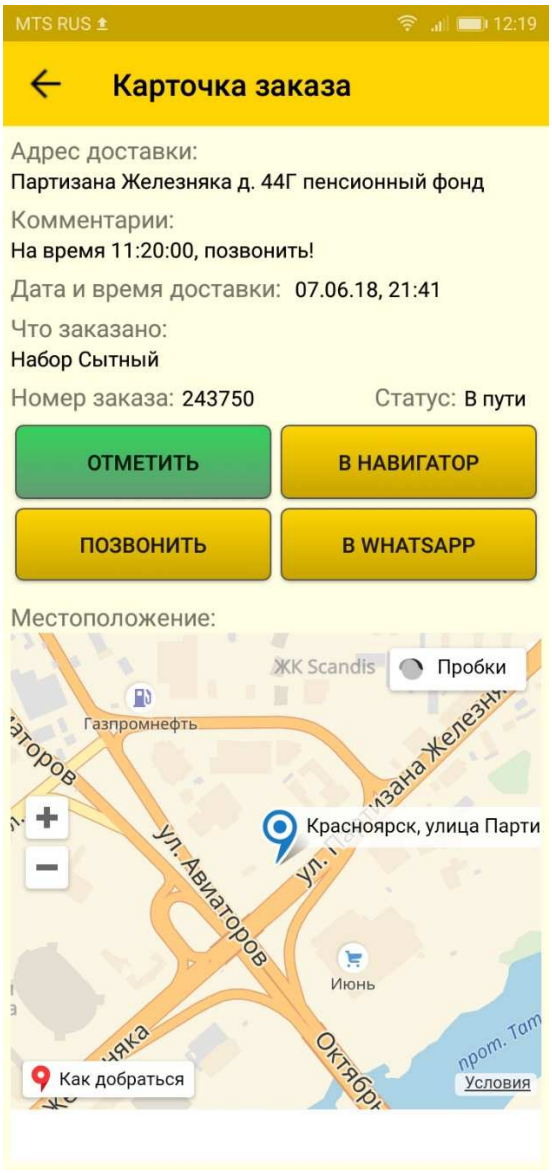

Рисунок 3.6 – Окно с карточкой заказа

В данном разделе отображена полная информация о заказе и ряд кнопок, выполняющих различные действия над текущим заказом. Кнопка «Отметить» также отвечает за изменение статуса заказа, кнопка «Позвонить» выполняет дозвон клиенту, кнопка «В навигатор» запускает внешнее приложение навигатора с навигацией до указанного адреса. Кнопка «В WhatsApp» формирует удобочитаемую строку с информацией о заказе и отправляет отформатированный текст через мессенджер WhatsApp. Внизу экрана отображен фрагмент карты, отцентрированный в районе адреса доставки, с необходимым масштабом, и в центре расположена метка, указывающая на конкретный дом.

3.2.5 Статистика пользователя

При нажатии в боковом меню на кнопку «Моя статистика» пользователю откроется экран со статистикой. Там приведено общее количество заказов за последний месяц и количество заказов за последнюю неделю. Ниже представлен список с карточками, в которых показано количество заказов, выполненных в определенный день. Данный экран отображен на рисунке 3.8

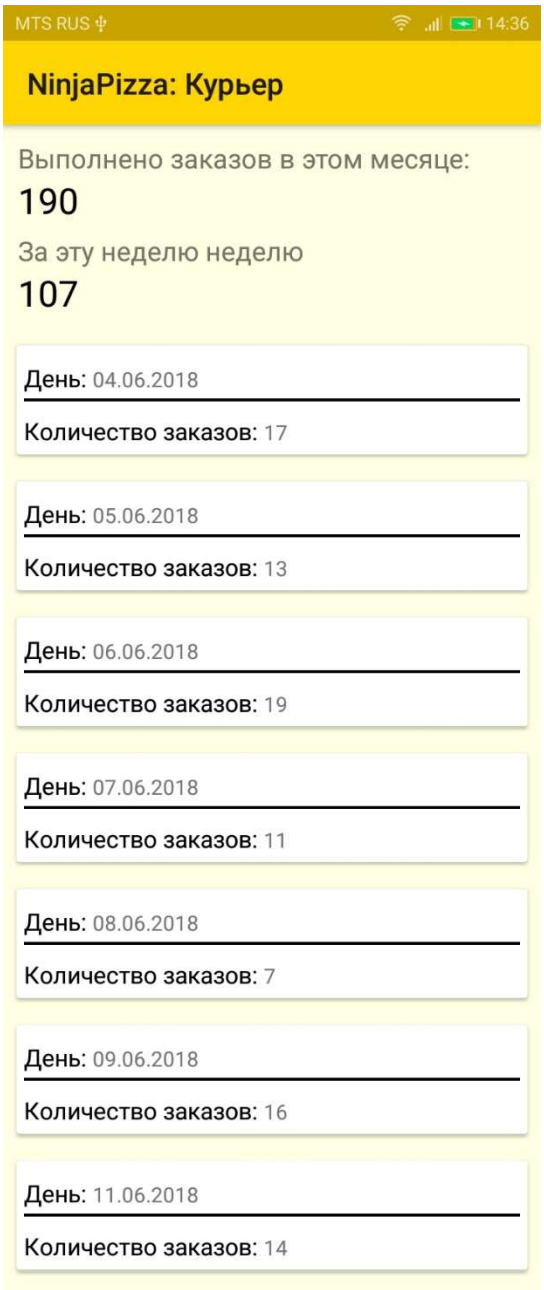

Рисунок 3.8 – Окно со статистикой

3.2.6 График смен

При выборе пункта «Мой график работы» в основном меню пользователю откроется список с его сменами понедельно. При нажатии на кнопку со знаком «+» откроется окно создания новой смены. Данный раздел подобен обыкновенным заметкам, однако он структурирован именно под хранение графика смен. Данное окно изображено на рисунке 3.9.

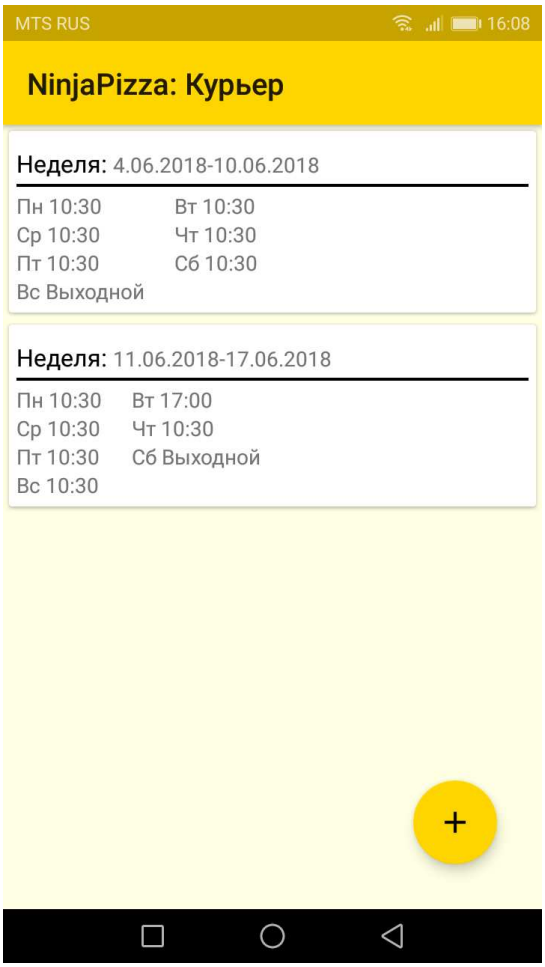

Рисунок 3.9 – Окно со списком смен

ЗАКЛЮЧЕНИЕ

Цель данной выпускной квалификационной работы состояла в улучшении бизнес-процессов обработки и доставки заказов на предприятии «NinjaPizza» посредством разработки и последующего внедрения мобильного приложения.

При выполнении работы были получены следующие результаты.

Произведен анализ предметной области, в ходе которого рассмотрен бизнес-процесс обработки заказов и доставки их клиентам, осуществляемый службой доставки исследуемого предприятия. Анализ показал, что указанный бизнес-процесс слабо автоматизирован и может быть улучшен посредством реорганизации самого процесса и внедрения программной системы, автоматизирующей функции курьера службы доставки. Исследование рынка готовых программных продуктов аналогичного назначения позволило сделать вывод о целесообразности собственной разработки, для чего был произведен выбор средств реализации мобильного приложения, а также сторонних модулей, необходимых для решения задач автоматизации.

На основании аналитического этапа проектирования выполнялись работы по логическому моделированию всех аспектов будущей системы, а также ее физической реализации. Приложение предназначено для операционной системы Android и разработано в среде разработки Android Studio. Система имеет клиент-серверную архитектуру и реализована с использованием шаблона проектирования MVP, а также использует модули для работы с Яндекс-картами, класс прямого геокодирования и библиотеку Retrofit для получения данных с сервера.

Разработанное приложение обладает всеми объявленными в требованиях функциями, а именно, позволяет отмечать статус заказа как «Доставлен», осуществлять звонки клиентам и администраторам прямо из приложения, показывает встроенную карту с предустановленным конечным местоположением на экране с заказом, а также позволяет переходить в навигатор с предустановленным маршрутом.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Мобильное приложение ОПТИМУМ Курьер [Электронный ресурс]: СиДиСи Центр Корпоративных Разработок, 2018 г. // Режим доступа: http://www.cdc.ru/

2. Мобильное приложение курьера для Android [Электронный ресурс]: Меасофт, 2018 г. // Электронная энциклопедия. – Режим доступа: http://wiki.courierexe.ru

3. Android SDK [электронный ресурс]: Википедия, 2018 г. // Электронная энциклопедия. – Режим доступа: https://ru.wikipedia.org.

4. Eclipse(IDE) [электронный ресурс]: Национальная библиотека имени Баумана, 2018г. // Электронная энциклопедия – Режим доступа: https://bmstu.wiki

5. Android Studio (IDE) [электронный ресурс]: Google Developers, 2018 г. // Google для разработчиков. – Режим доступа: https://developer.android.com/studio

6. Model-View-Presenter [электронный ресурс]: Википедия, 2018 г. // Электронная энциклопедия. – Режим доступа: https://ru.wikipedia.org

7. 2ГИС API [электронный ресурс]: 2ГИС, 2018 г. // Электронная документация. – Режим доступа: http://api.2gis.ru

8. API карт [Электронный ресурс]: Яндекс, 2018 г. // Электронная документация. – Режим доступа: https://tech.yandex.ru

9. Maps Android API [электронный ресурс]: Google, 2018 г. // Электронная документация. – Режим доступа: https://developers.google.com/

10. Диаграмма вариантов использования [Электронный ресурс]: ИНТУИТ, 2018г. // Национальный Открытый Университет. – Режим доступа: https://www.intuit.ru

11. SQLite [Электронный ресурс]: Национальная библиотека им. Н. Э. Баумана, 2018г. // Электронная энциклопедия. – Режим доступа: https://ru.bmstu.wiki/SQLite

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ И. В. Евдокимов
подпись инициалы, фамилия
«18» июня 2018г.

БАКАЛАВРСКАЯ РАБОТА

09.03.04 Программная инженерия

код и наименование специальности

Разработка мобильного приложения для курьерской службы предприятия «NinjaPizza»

тема

Руководитель — доцент кафедры «Информатика», канд. тех. н. (должность, ученая степень) _____ (подпись, дата) Тынченко В. В. (фамилия, инициалы)

Выпускник _____ (подпись, дата) Вельтер Д. В. (фамилия, инициалы)

Красноярск 2018

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Кафедра «Информатика»
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ И. В. Евдокимов
подпись инициалы, фамилия
«18» июня 2018г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Вельтер Данилу Вячеславовичу
фамилия, имя, отчество

Группа КИ14-17Б Направление (специальность) 09.03.04
подпись, дата код

Программная инженерия
наименование

Тема выпускной квалификационной работы:
Разработка мобильного приложения для курьерской службы предприятия «NinjaPizza»

Утверждена приказом по университету: № 7834/с от 2018-05-30

Руководитель ВКР: В.В. Тынченко, доцент кафедры «Информатика», канд. тех. н.
инициалы, фамилия, должность, ученая степень

Исходные данные для ВКР: информация о бизнес-процессе обработки заказов в курьерской службе предприятия «NinjaPizza»

Перечень разделов ВКР:
введение, аналитическая часть, проектирование системы, описание работы приложения, заключение, список использованных источников

Перечень графического материала: презентационные слайды PowerPoint.

Руководитель ВКР
подпись Тынченко В. В.
фамилия, инициалы

Задание принял к исполнению
подпись Вельтер Д. В.
фамилия, инициалы студента

«30» мая 2018 г.

ОТЗЫВ РУКОВОДИТЕЛЯ

на выпускную квалификационную работу студента группы КИ14-17Б,
обучающегося по направлению подготовки
09.03.04 Программная инженерия,

Вельтер Данила Вячеславовича

на тему «Разработка мобильного приложения для курьерской службы предприятия «NinjaPizza»

Актуальность темы исследования обусловлена потребностью компании «NinjaPizza» г. Красноярска в разработке и внедрении мобильного приложения, предназначенного для использования курьерами службы доставки.

Содержание ВКР полностью соответствует заданию.

В ходе выполнения работы автором1 было проведено исследование предметной области, обоснована целесообразность автоматизации ряда функций курьера службы доставки, сформирован набор основных требований к проектируемой системе, выполнен сравнительный анализ существующего программного обеспечения аналогичного назначения, произведен выбор средств реализации, выполнены и описаны основные этапы моделирования данных и программного обеспечения.

Разработанное мобильное приложение для ОС Android «NinjaPizza Courier» реализует требуемую функциональность, обладает качественным интерфейсом, обеспечивает необходимую производительность.

Текст бакалаврской работы изложен лаконично, грамотно, хорошо структурирован. Все технические решения описаны с достаточной степенью детализации и проиллюстрированы графическим материалом.

В ходе выполнения ВКР Вельтер Д.В. проявил самостоятельность, ответственность, творческую инициативу.

Разработанное мобильное приложение внедрено в компании «NinjaPizza», что подтверждается актом о внедрении и свидетельствует о практической значимости результатов данной ВКР.

Считаю, что представленная работа соответствует всем требованиям, предъявляемым к выпускным квалификационным работам, и заслуживает оценки «отлично», а студент Вельтер Д.В. – присвоения квалификации бакалавра по направлению подготовки 09.03.04 Программная инженерия.

Доцент каф. «Информатика»,
канд. техн. наук, доцент /В.В. Тынченко/

Акт о внедрении программного обеспечения

Мобильное приложение для курьеров службы доставки «Ni

Настоящий акт свидетельствует о том, что мобильное прило ОС Android «NinjaPizza Courier», предназначенное для испо курьерами службы доставки «NinjaPizza» и разработанное Вельтер Вячеславовичем внедрено в компании «NinjaPizza».

К приложению были предъявлены следующие требования:

– отображение общего списка заказов в сокращенном виде;

– отображение полной информации о заказе в отдельном окн

– отображение конечного адреса на встроенной карте в окне з

– построение маршрута в стороннем навигаторе по нажатию к из приложения;

– возможность сделать звонок клиенту прямо из приложения;

– возможность отправки краткой информации о заказе в мессенд WhatsApp;

– возможность сделать звонок администратору прямо приложения;

– работа на ОС Android с версии 5.0;

– уведомление пользователя при малом остаточном времени д доставки заказа.

В ходе эксплуатации программы подтверждено, что мобильное приложение обладает всеми заявленными возможностями. На момент подписания настоящего акта приложение используется компанией «NinjaPizza».

Заместитель директора Малёнова Д. Е.