МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

РАБОТА ПРОВЕРЕНА
ЗАЩИТЕ
Рецензент
ЭВМ
_____
«___» _____2020 г.

ДОПУСТИТЬ К

Заведующий кафедрой

_____ Г.И. Радченко
_____ 2020 г.

Разработка мобильного приложения для автоматизации пусконаладочных работ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____ Д.В. Топольский
«___» _____2020 г.

Автор работы,
студент группы КЭ-222
_____ В.В. Кригер
«___» _____2020 г.

Нормоконтроллер,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___» _____2020 г.

Челябинск-2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой
ЭВМ
_____ Г.И. Радченко
«___»_____2020 г.

ЗАДАНИЕ
на выпускную квалификационную работу магистра
студенту группы КЭ-222
Кригеру Василию Владимировичу
обучающемуся по направлению
09.04.01 «Информатика и вычислительная техника»

Тема работы: «Разработка мобильного приложения для автоматизации пусконаладочных работ»

1. **Срок сдачи студентом законченной работы:** 1 июня 2020 г.

2. **Исходные данные к работе:** техническое задание:
   - — Язык программирования – Java.
   - — Платформа разработки – Android.
   - — Вся информация о базовых станциях, необходимая для работы, предоставляется заказчиком.
   - — В качестве компонента, отвечающего за использование карт, использовать Google Maps.

3. **Перечень подлежащих разработке вопросов:**
   – обзор литературы;
   – формирование требований к системе;
   – разработка архитектуры приложения;
   – разработка мобильного приложения;
   – тестирование и отладка.

4. **Дата выдачи задания:** 1 февраля 2020 г.

Руководитель работы _____/*Д.В.Топольский*

Студент _____/*В.В. Кригер*

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	01.03.2020	
Формирование требований к системе	01.04.2020	
Разработка архитектуры	15.04.2020	
Разработка мобильного приложения	01.05.2020	
Тестирование и отладка	15.05.2020	
Компоновка текста работы и сдача на нормоконтроль	24.05.2020	
Подготовка презентации и доклада	30.05.2020	

Руководитель работы _____/ *Д.В. Топольский*

Студент _____/*В.В. Кригер*

АННОТАЦИЯ

Кригер В.В. Разработка мобильного приложения для автоматизации пусконаладочных работ: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2020,, 64 с. 30 ил., библиогр. список – 32 наим.

В магистерской дисертации был произведен обзор литературы, выбраны компоненты для будущего приложения, выбраны функциональные и нефункциональные требования к системе, сформирована архитектура приложения и разработано мобильное приложение для автоматизации пусконаладочных работ. После тестирования приложения была произведена отладка.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Основной задачей отдела эксплуатации сети в ПАО «Мобильные ТелеСистемы» является поддержание базовых станций(БС) в рабочем состоянии. Базовая станция в радиосвязи — системный комплекс приёмопередающей аппаратуры, осуществляющей централизованное обслуживание группы оконечных абонентских устройств.

В настоящее время работники ПАО МТС тратят лишнее время для выполнения поставленных задач, из-за сложной специфики рабочего процесса. Для того, чтобы решить это проблему и увеличить скорость работы персонала, можно прибегнуть к помощи современных технологий. Таким образом, лучшее решение это - создать приложение, для сотрудников, которое, во-первых, лишит их необходимости, использовать стороннее программное обеспечение. И во-вторых, заберет часть обязанностей на себя. В настоящее время, существуют три основных типа приложений[1]:

- приложения для рабочего стола;
- мобильные;
- веб-приложения.

У каждого из вышеперечисленных типов есть свои плюсы и минусы. Исходя из того, что рабочие редко находятся в офисе, и большую часть времени занимаются обслуживанием базовых станций, было принято решение разрабатывать мобильное приложение.

Перед тем, как выбирать инструменты для будущего приложения, необходимо понять специфику работы пользователя. Для этого разделим пользователей на две категории:

1. Дежурный.
2. Работник.

У каждой группы пользователей свои требования.

У группы работников следующие требования:

1. Интуитивно понятный интерфейс.
2. Необходимо отображать поставленные задачи.
3. Должна быть возможность получить информацию о базовых станциях.
4. Связь с SMS сервером.
5. Необходимо иметь возможность получать и изменять текущую информацию о состоянии базовых станций.
6. Пользователь должен иметь возможность использовать карту, для нахождения одной или нескольких базовых станций.
7. Возможность получить информацию о базовой станции, к которой подключен телефон.
8. Необходимо выбирать дежурного из списка возможных дежурных.
9. Необходимо знать телефоны людей, обслуживающих базовые станции.
10. Необходима связь с энергетиками, отвечающими за текущие базовые станции.
11. Для более удобной работы необходимо наличие поиска.
12. Необходимо знать, от каких базовых станциях ключи находятся в офисе.
13. Рабочий должен иметь возможность отчитаться о проделанной работе, и видеть текущее состояние работ.

Дежурные – это работники, которое помимо основных обязанностей имеют обязанности дежурного. Таким образом, они имеют требования работников и ряд дополнительных требований:

1. Необходимо получать информацию о текущих инцидентах.
2. Инциденты должны быть классифицированы по степени важности.
3. Дежурный должен иметь возможность закрывать инциденты.

4. Информация о текущих и решенных инцидентах должна храниться, для возможности проверки работы дежурного.

Цели и задачи.

Цель, представленной выпускной квалификационной работы – разработать мобильное приложение для автоматизации пусконаладочных работ.

Для достижения поставленной цели, необходимо выполнить следующие задачи:

1. Произвести обзор литературы, для более четкого понимания предметной области.
2. Выбрать инструменты для разработки будущего приложения.
3. Определить требования.
4. Продумать архитектуру будущего приложения.
5. Разработать и реализовать мобильное приложение.
6. Произвести отладку приложения.

Актуальность.

Данное приложение является актуальным сразу по нескольким причинам. Во-первых, это заказ крупной компании, этот факт свидетельствует о том, что приложением будут пользоваться сотрудники, для которых оно разрабатывается. Во-вторых, в настоящий момент, невозможно найти приложения, которое выполняло бы поставленные цели.

1 ОБЗОР ЛИТЕРАТУРЫ

Перед тем как начать разрабатывать приложение, необходимо провести обзор литературы. Это позволит принять ряд решений, который повлияют на будущее приложение. Исходя из информации о проекте и типах приложений, было принято решение разрабатывать мобильное приложение. Для того, чтобы убедиться в правильности выбора, обратимся к современным исследованиям. Исходя из информации, предоставленной аналитическим агентством We Are Social и крупнейшей SMM-платформа Hootsuite можно будет убедиться в целесообразности разработки мобильного приложения.

На сегодняшний день мобильными устройствами пользуются 5.2 миллиардов человек. Это на 124 миллиона больше чем в прошлом году.

Рисунок 1 – Количество пользователей на различных устройствах в России.

По данным Digital 2020, количество интернет-пользователей в России превышает 118 миллионов. Это примерно 81% населения страны.

Мобильные устройства активно развиваются. В настоящее время

На мобильные телефоны теперь приходится больше половины времени, которое мы проводим в интернете — 50,1%.

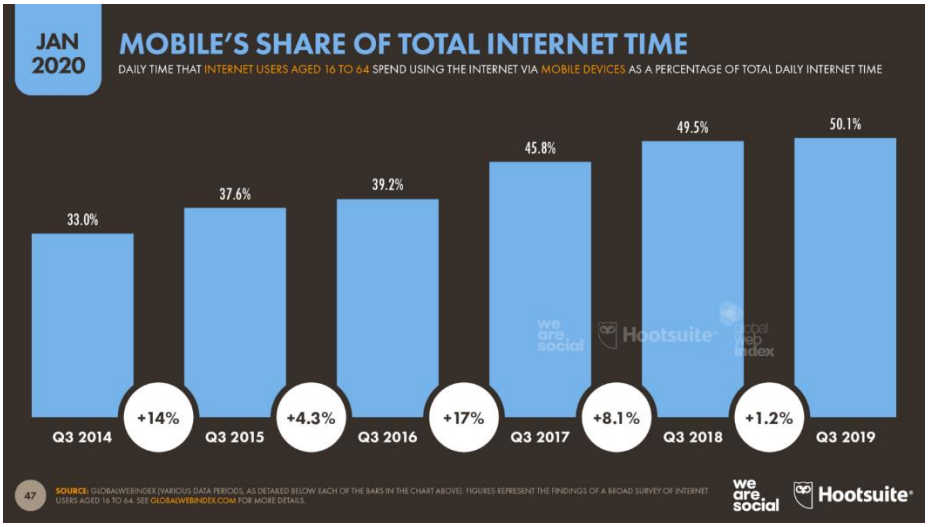

Рисунок 2 – Количество времени, проводимого в интернете.

Примерно 92% интернет-пользователей могут выходить в сеть с помощью мобильных устройств. При этом иногда использовать стационарные устройства, в связи со спецификой работ. Таким образом, более половины интернет пользователей в возрасте от 25 до 60 лет предпочитают ноутбуки и персональные компьютеры. Если обратиться к статистике, собранной компанией Statcounter, можно сделать вывод что количество запросов, обработанных с мобильных устройств, в настоящий момент является максимальным и продолжает расти. Но компьютеры также имеют значительное количество – 44 %.

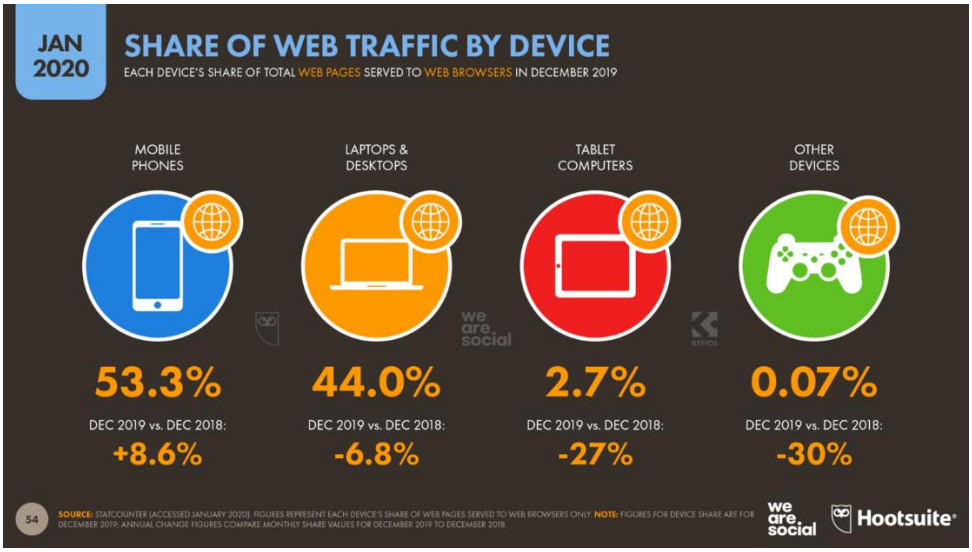

Рисунок 3 – Доля веб-трафика на устройстве.

Связано это в первую очередь с достоинствами мобильные устройств. Они занимают меньше места, и пользователи могут использовать их практически, где и когда угодно. Большинство интернет-магазинов переходят на мобильные приложения. Веб активно теряет позиции, а социальные сети помогают собирать ценные сведения для бизнеса[2].

Одними из ключевых факторов роста интернет-аудиториив этом году стали доступные смартфоны и недорогие тарифы на мобильный интернет. Необходимо учитывать и тот факт, что мобильные устройства в последнее годы активно развиваются. Если вспомнить прошлое десятилетие, пользователи не могли активно пользоваться интернетом и виной тому сразу несколько факторов: медленный интернет, неудобные браузеры, медленная работа самих телефонов[3].

Из всех этих данных можно сделать вывод, что большинство людей используют несколько разных устройств для выхода в интернет. Люди используют разные устройства в разное время и для разных целей, поэтому необходимо провести анализ предметной области перед выбором архитектуры будущего приложения.

Учитывая тот факт, что средний возраст в отделе обслуживания базовых станций составляет 25 лет, и люди, работающие с мобильным интернетом, являются активными пользователями сети. Можно обратить внимание еще на одно исследование. По информации App Annie представители поколения Z, люди в возрасте от 16 до 26 лет являются наиболее активными пользователями мобильных устройств. Они активнее представителей других поколений пользуются преимуществами современного мобильного мира: на 20% больше времени проводят в своих мобильных устройствах и на 30% чаще используют конкретные приложения[4].

Таким образом, можно сделать вывод, что решение разрабатывать мобильное приложение является полностью правильным, в связи со спецификой приложения и ситуацией на рынке приложений.

1.1 ОБЗОР АНАЛОГОВ

Явных аналогов у данного приложения не существует, это связанно со спецификой работы. Вероятнее всего аналоги существуют у других сотовых компаний, но являются частью корпоративной тайны и их нет в свободном доступе.

Таким образом, для нахождения аналогов необходимо произвести декомпозицию задач. После этого, можно будет найти аналоги, отвечающие за конкретные задачи.

Поиск аналогов был проведен в Google Play. Google Play[6] (прежнее название — Android Market) — магазин приложений, игр, книг, музыки и фильмов компании Google и других компаний, позволяющий владельцам устройств с операционной системой Android устанавливать и приобретать различные приложения. Из всех подобранных аналогов была сформирована таблица 1.

Таблица 1 – Обзор существующих решений

Приложение	Функция	Оценка	Особенности
Any.do[5]	Планирование задач, управление проектом	4.4	Удобный планировщик, есть многопользовательский режим
Trello[6]	Планирование задач, управление проектом	4.5	Удобный планировщик, есть многопользовательский режим
Todoist[7]	Планирование задач, управление проектом	4.6	Минималистичный планировщик, нет возможности обмениваться задачами с другими пользователями.
2ГИС[8]	Использование карт	4.5	Удобный навигатор, достаточно полная информация о городе
Яндекс навигатор[9]	Использование карт	4,0	Удобный навигатор, есть проблемы с актуальностью информации
Pulse sms[10]	Отправка SMS сообщений	4.4	Удобный SMS менеджер, требует настройки
Handcent Next SMS[11]	Отправка SMS сообщений	4.1	SMS менеджер, объединяет множество месенжерей.

Рассмотрим аналоги более подробно, чтобы оценить возможность замены будущего проекта на группу приложений.

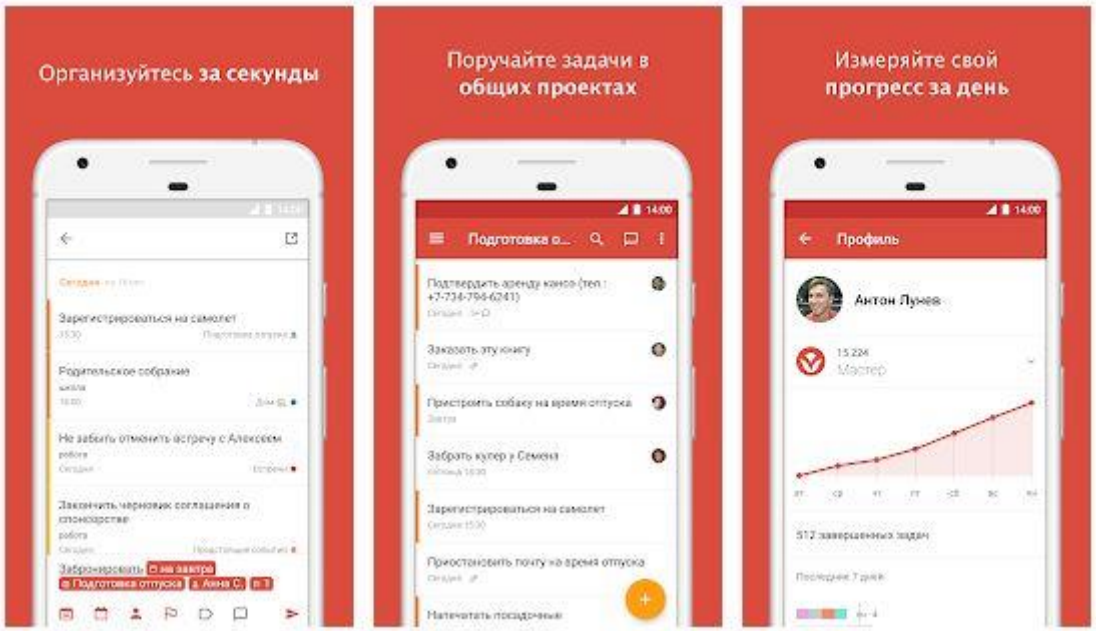

Рисунок 4 – Интерфейс Todoist.

Todoist— таск-менеджер, предназначенный как для личного, так и для командного использования. Сервис поддерживает русский язык.

Todoist позволяет создавать подзадачи, объединять задачи в проекты, поддерживает приоритезацию. В платных версиях можно получать напоминания в самом сервисе или по электронной почте, настраивать тему, добавлять собственные метки к задачам, прикреплять файлы, комментировать, оценивать выполненные задачи.

В версии «Бизнес» можно создавать команду и поручать выполнение задач сотрудникам, обмениваться комментариями, работать с шаблонами проектов.

Плюсы:

- быстрая запись задачи, например, «сделать отчет #Работа», и Todoist автоматически создаст задачу в проекте «Работа» с напоминанием каждый месяц;
- создание повторяющихся сроков выполнения – например, каждый второй понедельник;

- выделение самых важных дел на каждый день с помощью цветовых уровней приоритета;
- синхронизация с другими устройствами;
- доступ к задачам для других зарегистрированных пользователей.

В платной версии можно добавлять комментарии к своим задачам, отправлять их по электронной почте, получать напоминания на почту или через SMS, экспортировать задачи в календарь Google, Outlook или iCal, использовать поиск и просматривать статистику своей продуктивности по дням и проектам[12].

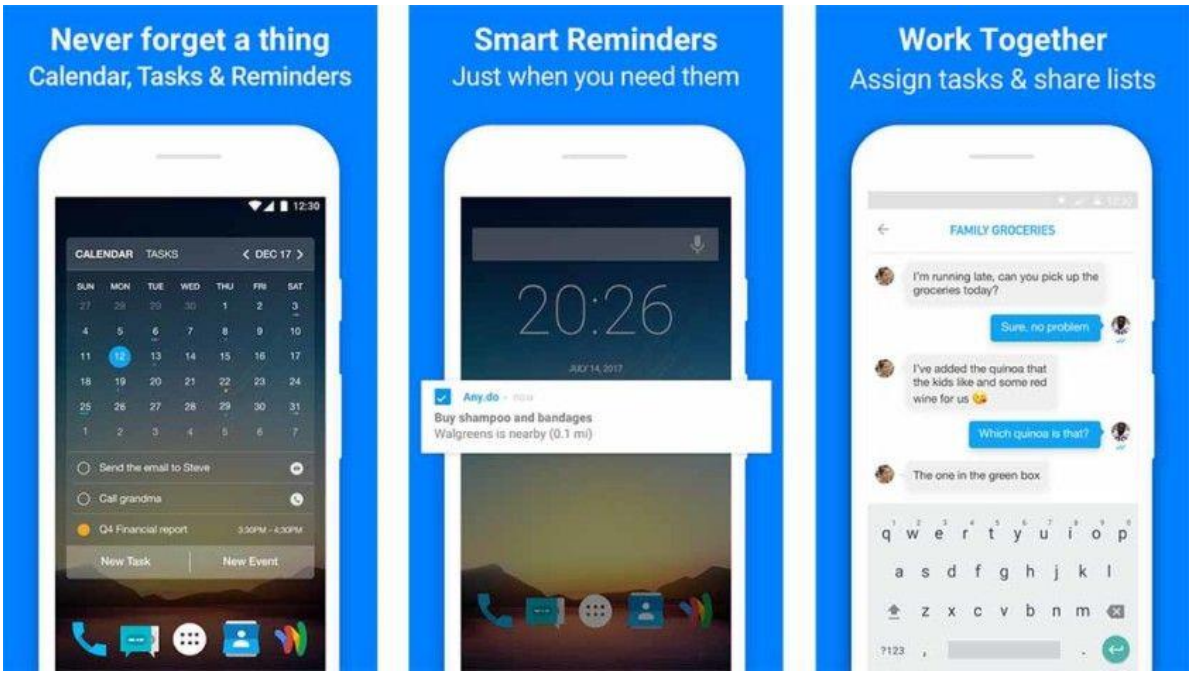

Рисунок 5 – Интерфейс Any. Do.

Any. Do. Сервис для управления задачами и делами, предназначенный для личного использования. Позволяет формировать списки дел и задач, устанавливать напоминания, поручать задачи другим пользователям. Сервис бесплатен, но есть платная премиум-версия стоимостью $2,99 в месяц[13].

Плюсы:

- голосовое добавление задач;

- синхронизация с календарем телефона;
- задачу можно разделить с другими пользователями, если они зарегистрированы в приложении;
- дополнительное Chrome-приложение, позволяющее работать со списком задач на компьютере;
- функция Moment помогает быстро организовать задачи, указав на ошибки в их расстановке.

Преимущества платной версии:

- повторяющиеся задачи
- настраиваемые темы
- неограниченный размер файлов;
- неограниченный доступ к Any.do Moment

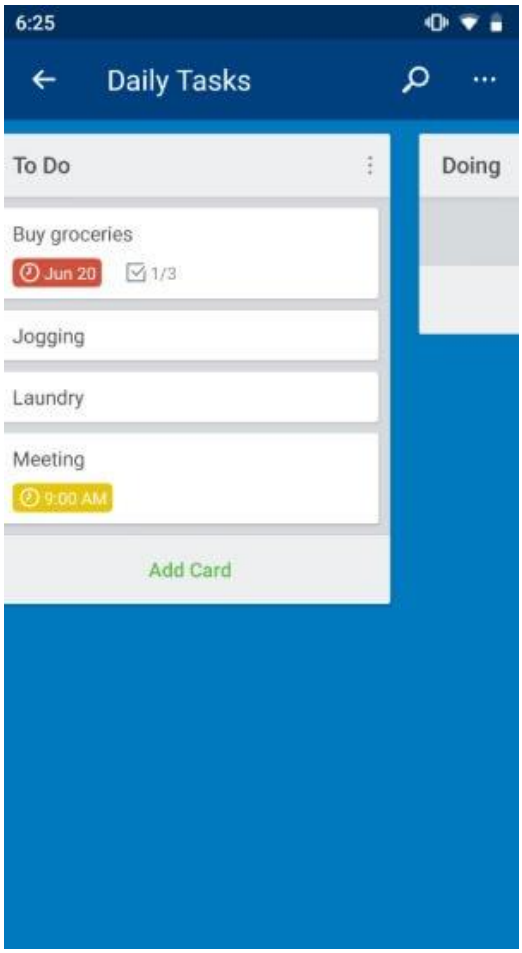

Рисунок 6 – Интерфейс Trello

Trello представляет собой канбан-доску со списками карточек, которую можно использовать для управления задачами как лично, так и в команде. В бесплатной версии можно интегрировать в Trello только один сервис, а размер вложенного файла не может превышать 10 МБ.

В платных версиях доступны дополнительные инструменты для командной работы, а также предлагается повышенный уровень безопасности (например, двухфакторная аутентификация). Сервис поддерживает русский язык.

В карточке можно обмениваться комментариями с другими пользователями, прикреплять участников, добавлять метки, использовать чек-лист.

В сервис можно интегрировать Jira, «Google Диск», Dropbox, Evernote, Slack, GitHub, GitLab и ещё несколько десятков сервисов[14].

Плюсы:

- создавать доски для проектов;
- использовать самостоятельно или приглашать коллег для распределения задач;
- изменять рабочие процессы в зависимости от проекта;;
- добавлять чек-листы на карточки
- обмениваться комментариями с коллегами;
- прикреплять файлы из Google Диска и Dropbox;
- работать в автономном режиме и автоматически синхронизировать карточки при подключении.

С помощью данных приложений пользователь может планировать задачи, делиться задачами с другими пользователями, отслеживать прогресс выполнения задач. При этом специфика постановки задачи внутри компании

обязывает использовать SMS сообщения. Таким образом пользователь должен будет вручную вбивать поставленные задачи в планировщик задач, что замедляет процесс выполнения работы.

Следующая функция, рассмотренная в обзоре, это использование карт.

Приложение Яндекс.Навигатор – качественный, удобный и бесплатный навигатор для Андроид. Он не требует много аппаратных ресурсов мобильного девайса, легок в управлении и занимает не много места на диске. Умеет прокладывать маршруты с учетом плотности трафика, загруженности дорог, аварий и перекрытий улиц.

Рисунок 7 – Интерфейс Яндекс Навигатора

Интерфейс у мобильной версии Яндекс Навигатора хорошо структурирован и нагляден. При передвижении на авто по проложенному маршруту отображает информацию о приблизительном оставшемся до пункта назначения времени и расстоянии до конечной точки. Если по пути нужно заехать на СТО, заправку или автомойку, на карте Яндекс Навигатора можно указать дополнительный пункт - и маршрут будет оптимально пересчитан с учетом промежуточного пункта. Можно включать и отключать слой с

отображением пробок и направлений путем нажатия одной единственной кнопки с желтым сигналом светофора.

К особенностям навигатора относится ночной gps-режим для Андроид, где карта отображается в темных тонах, не отвлекая внимание водителя во время езды в позднее время и не дает глазам сильно устать при рассмотрении объектов. Кстати, набор доступных для поиска POI-объектов действительно впечатляет. Наряду со ставшими уже привычными ресторанами, банкоматами и кинотеатрами, можно задействовать слои с отображением находящихся неподалеку пунктов связи, почтовых отделений, салонов и множества других интересных мест[15] .

2 ГИС на сегодняшний момент концептуально не отличается от Яндекс Навигатора. Но с точки зрения навигатора проигрывает в функционале.

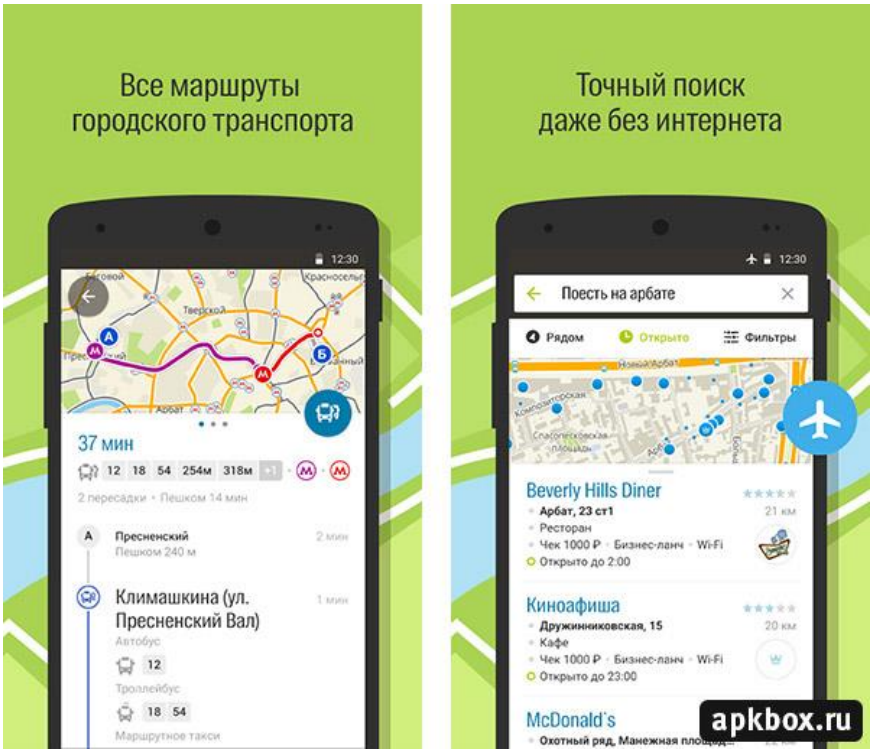

Рисунок 8 – 2ГИС

Таким образом, оба навигатора справляются с поставленной задачей и могут помочь пользователю доехать до базовой станции. Но стоит учитывать

тот факт, что расположение базовых станций – информация статическая. И таким образом было бы логичнее использовать карту с заранее известными объектами, чтобы сократить время на поиск.

Третья функция, это отправка SMS сообщений. В качестве первого аналога было рассмотрено приложение Pulse sms. Pulse – приложение и веб-сервисе одновременно, с помощью которого можно общаться со всем миром. При этом сообщения шифруются и хранятся в фирменном облаке. Pulse – это первый кроссплатформенный SMS-мессенджер, то есть имеет синхронизируемые клиенты для смартфона, планшета, компьютера, умных часов, а также веб-версию. Но, за одновременное использование приложения на нескольких устройствах сразу придется оформить платную подписку[16].

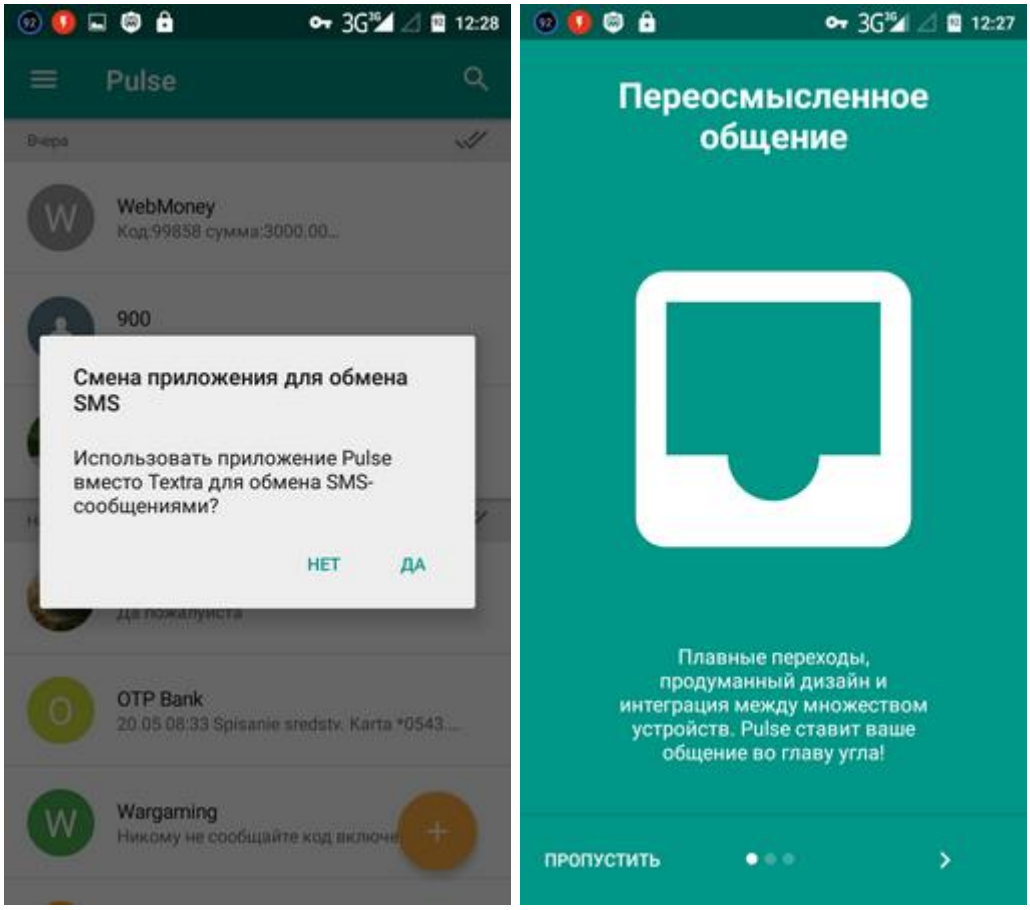

Рисунок 10 – Pulse sms

Handcent Next SMS сход по функционалу, преимуществам и недостаткам с Pulse sms, нет необходимости рассматривать его отдельно.

1.2 АНАЛИЗ ТЕХНОЛОГИЙ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

Рынку мобильных приложений уже больше десяти лет, однако он до сих пор бурно развивается. Спрос на создание мобильных приложений со стороны компаний постоянно растёт, и он всё ещё заметно превышает предложение, что приводит к постоянному удорожанию разработки. Одно из решений в удешевлении этого процесса — кроссплатформенная разработка, когда один и тот же код используется на всех платформах[17].

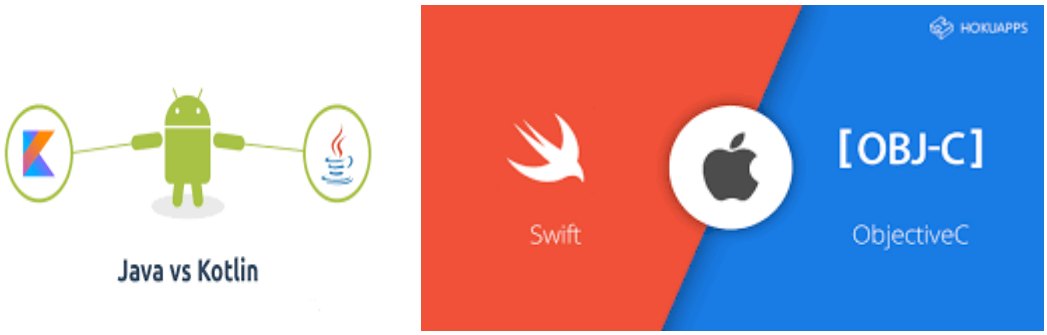

Рисунок 9 – Естественные языки

Если разработчики в процессе написания приложения пользуются принятым для конкретной платформы языком программирования, будь то Objective-C[18] и Swift[19] для iOS или Java[20] или Kotlin[21] для Android, такое приложение будет называться нативным (от англ. native — родной, естественный) (рисунок 9).

Такие мобильные приложения пишутся на языках программирования, утвержденных разработчиками программного обеспечения под каждую конкретную платформу, а потому органично встраиваются в сами операционные системы. Приложения загружаются через магазины приложений (App Store, Google Play и т.д.) и соответствуют требованиям этих магазинов.

Главное преимущество нативных приложений – то, что они оптимизированы под конкретные операционные системы, а значит работают

корректно и быстро. Также они имеют доступ к аппаратной части устройств, то есть могут использовать в своем функционале камеру смартфона, микрофон, акселерометр, геолокацию, адресную книгу, плеер и т.д. Можно настроить получение push-уведомлений. Еще один плюс – экономный расход ресурсов телефона (батарея, память).

Нативные приложения могут полностью или частично работать и при отсутствующем интернет-соединении, поэтому пользователи менее зависят от качества связи и могут пользоваться приложением там и тогда, когда им это удобно.

Разумеется, написание такого продукта требует от разработчика владение специальными знаниями и умениями для работы в конкретной среде разработки. Как следствие стоимость таких приложений гораздо выше в силу их трудоемкости и того, что под каждую платформу приходится писать отдельное приложение на другом языке.

Неестественные языки

Кроссплатформенные приложения пишутся сразу для нескольких платформ на одном языке, отличном от нативного. Как такой код может работать на разных устройствах? Тут тоже есть два подхода.

Первый заключается в том, что на этапе подготовки приложения к публикации он превращается в нативный для определённой платформы с помощью транспилера. Фактически один кроссплатформенный язык программирования «переводится» на другой.

Второй — в том, что к получившемуся коду добавляется определённая обёртка, которая, работая уже на устройстве, на лету транслирует вызовы из неродного кода к родным функциям системы.

Предполагается, что большая часть такого кода может переносится между платформами — очевидно, что, например, логика совершения покупок,

сохранения товара в корзину, просчёта маршрута для такси, написания сообщения в мессенджер не меняется в зависимости о того, Android у клиента или iOS. Нужно лишь доработать UI и UX для платформ, но сейчас, в определённых пределах, даже это можно объединить — например, менюгамбургер активно используется как на Android, так и на iOS. Так что даже внесений исправления в интерфейс для того, чтобы приложение отвечало духу и букве нужной платформы — вопрос желания, необходимой скорости и качества разработки.

Преимущества:

– стоимость и скорость разработки. Так как кода надо писать заметно меньше, то и стоимость работ снижается;

– возможность использовать внутренние ресурсы компании. Как мы покажем дальше, кроссплатформенную разработку мобильных

– приложений зачастую можно осуществить силами уже существующих у вас программистов.

Недостатки:

- ограничения объема хранимых в мобильном приложении данных, от которых зависит как часто приложению придется что-то докачивать через интернет (html5 – 50 Мб);
- невозможность использовать общий поиск мобильного телефона;
- невозможность узнать тип сетевого соединения (GPRS, 3G, LTE, WiFi – зависит от фреймворка);
- ограничения, накладываемые браузером операционной системы на доступ к акселерометру, гироскопу, геопозиционированию, видеозахвату;
- невозможность явной работы с файловой системой (создание и управление файлами и папками);

— ненативный интерфейс, который выглядит одинаково на всех операционных системах, или его приходится реализовывать отдельно под каждую из платформ;
— необходимость создавать различные компоненты с нуля (к примеру, выезжающее меню или анимацию), тогда как в нативных приложениях быстрее и проще использовать готовые компоненты;
— сложнее процесс оптимизации под различные размеры экранов устройств, чем у нативных приложений.

Популярные платформы и инструменты кроссплатформенной разработки

Как мы написали выше, есть два подхода — превращение кода в нативный на этапе сборки или добавление определённой обёртки, транслирующей вызовы к системе и от неё.

Cordova и PWA — два инструмента, работающие как раз в идеологии обёртки (рисунок 6).

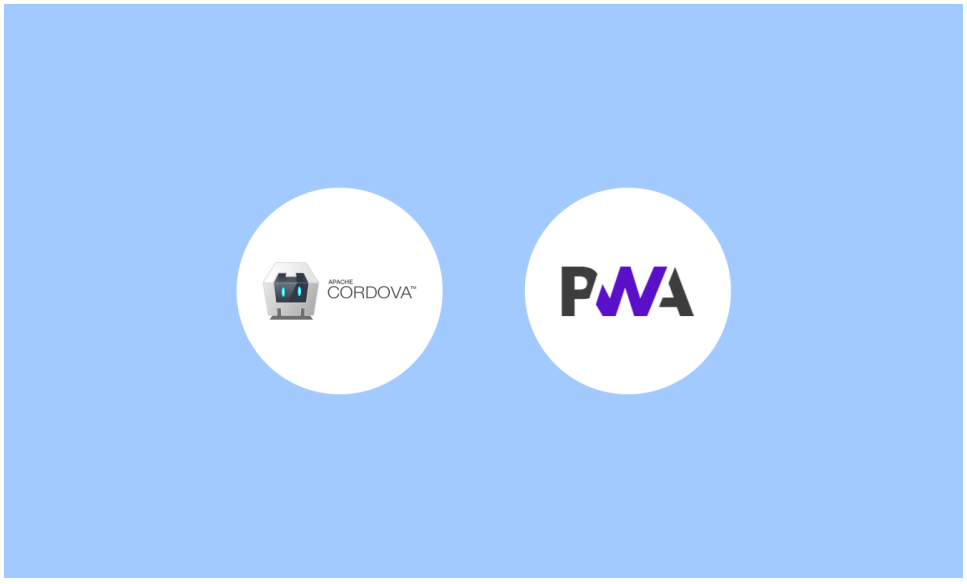

Рисунок 10 – Cordova и PWA

Cordova и HTML5

Одно из самых популярных направлений в кроссплатформенном программировании, которое часто по-народному называют PhoneGap. Фактически создаётся мобильный сайт, который «оборачивается» небольшим платформенным кодом, транслирующим вызовы от системы к приложению и обратно.

Все недостатки и достоинства тут выражены как нигде ярко. Вы можете использовать веб-разработчиков (HTML, CSS и JavaScript как основные технологии) и за месяц или даже пару недель сделать первую версию приложения за относительно небольшие деньги. Да, она будет подтормаживать в работе, возможно, в ней будет не совсем точная геолокация, но она будет работать на всех устройствах и позволит вам, как минимум, протестировать спрос со стороны клиентов на мобильных устройствах.

Для такого подхода создано огромное количество фреймворков, но все они делают фактически одно и тоже. Различие между ними в том, что Cordova (PhoneGap) не задаёт ограничений и шаблонов на логику и UI для вашего HTML5-проекта, а фреймворки оперируют собственными готовыми UIэлементами, имитирующими мобильные платформы, и своей логикой разработки. В качестве примера такого подхода можно указать: Ionic Framework — обёртка; Framework7, Mobile Angular UI, Sencha Touch, Kendo UI — интерфейсные фреймворки.

PWA

Модная технология от Google — это те же самые веб-приложения, но за счёт использования определённых технологий (в первую очередь это так называемые Service Worker — работающие в фоновом режиме скрипты, и Web App Manifest — описание веб-приложения в понятном для мобильной системы виде) они без обёртки из PhoneGap могут работать как нативные. Они могут

устанавливаться на домашний экран в обход магазина приложений, работать в офлайне, работать с пуш-уведомлениями, с нативными функциями.

Проблема в том, что не все платформы даже сейчас поддерживают эти «определённые технологии». В первую очередь это касается Apple, которой, видимо, очень не нравится возможность распространять приложения в обход App Store.

Учтя все недостатки HTML5-решений, многие компании создали инструменты, которые позволяют писать код на одном, не нативном, языке, а он 28 потом транслируется в нативный. Так убивается два зайца одновременно: кодовая база получается одна, а приложения получаются максимально близки к нативному. Далее речь пойдет о платформах без веб-разработки (Рисунок 7).

Рисунок 11 – Платформы без веб-разработки Xamarin

Платформа компании Microsoft. Используется стандартный для Enterpriseразработки язык программирования C#[22], кроссплатформенная среда разработки — Visual Studio. На выходе — нативные приложения для iOS, Android и Windows. Правда, относительно большого размера.

React Native

Платформа от Facebook[23] — приложения пишутся на JavaScript и с использованием CSS-подобных стилей. Интерфейс получается родной, а код интерпретируется уже на платформе, что придаёт ему нужную гибкость. 29

Будучи относительно молодой платформой, React Native пока очевидно (хоть и не катастрофически) страдает от недостатка средств разработки и документации.

Flutter

Естественно, не мог обойти тему кроссплатформенной разработки Android и iOS-приложеий и такой гигант, как Google. Flutter, пока, правда, существующий только в бета-версии, исповедует отличный от React Native и Xamarin подход. Он не превращает исходный код в нативный, который выполняется платформой, а на самом деле рисует окно на экране смартфона и отрисовывает все элементы сам. В качестве языка используется «фирменный» Dart, который Google создал как усовершенствованную версию JavaScript.

У этого есть как преимущества (например, внешне идентичные интерфейсы), так и недостатки (например, перерисовка интерфейса требует определённых затрат памяти и процессорного времени).

Платформа быстро развивается и Google вкладывает в это много сил и средств. Но по сравнению с Flutter даже React Native кажется вполне устоявшейся и впечатляющей экосистемой.

1.3 АНАЛИЗ СОВРЕМЕННОЙ ТЕХНОЛОГИИ РЕАЛИЗАЦИИ БАЗ ДАННЫХ

Большинство современных мобильных приложений используют базы данных для хранения различной информации. где будет хранится информация. Разберем какие системы управления базами данных существуют, а их преимущества и недостатки[24].

SQLite

SQLite – легко встраиваемая в приложения база данных. Так как это система базируется на файлах, то она предоставляет довольно широкий набор инструментов для работы с ней, по сравнению с сетевыми СУБД. При работе с этой СУБД обращения происходят напрямую к файлам (в этих файлах хранятся данные), вместо портов и сокетов в сетевых СУБД. Именно поэтому SQLite очень быстрая, а также мощная благодаря технологиям обслуживающих библиотек.

Преимущества SQLite:

- файловая структура – вся база данных состоит из одного файла, поэтому её очень легко переносить на разные машины;
- используемые стандарты – данная база данных использует sql, но со своими особенностями;
- отличная при разработке и тестировании – в процессе разработки приложений часто появляется необходимость масштабирования. SQLite предлагает всё что необходимо для этих целей, так как состоит всего из одного файла и библиотеки написанной на языке C.
- с точки зрения скорости работы является максимально быстрой благодаря тому, что размещается в памяти мобильного телефона.

Недостатки SQLite:

- отсутствие системы пользователей – более крупные СУБД включают в свой состав системы управления правами доступа пользователей. Обычно применения этой функции не так критично, так как эта СУБД используется в небольших приложениях;

— отсутствие возможности увеличения производительности – опять, исходя из проектирования, довольно сложно выжать что-то более производительное из этой СУБД.

MySQL

MySQL – это самая распространенная полноценная серверная СУБД. MySQL очень функциональная, свободно распространяемая СУБД, которая успешно работает с различными сайтами и веб приложениями. Обучиться использованию этой СУБД довольно просто, так как на просторах интернета вы легко найдете большее количество информации. Также, существует огромное количество различных плагинов и расширений, облегчающих работу с системой. Несмотря на то, что в ней не реализован весь SQL функционал, MySQL предлагает довольно много инструментов для разработки приложений. Так как это серверная СУБД, приложения для доступа к данным, в отличии от SQLite работают со службами MySQL.

Преимущества MySQL:

— простота в работе – установить MySQL довольно просто. Дополнительные приложения, например GUI, позволяет довольно легко работать с БД;
— богатый функционал – MySQL поддерживает большинство функционала SQL;
— безопасность – большое количество функций, обеспечивающих безопасность, которые поддерживается по умолчанию;
— масштабируемость – MySQL легко работает с большими объемами данных и легко масштабируется;
— скорость – упрощение некоторых стандартов позволяет MySQL значительно увеличить производительность.

Недостатки MySQL:

Известные ограничения – по задумке в MySQL заложены некоторые ограничения функционала, которые иногда необходимы в особо требовательных приложениях;

Проблемы с надежностью – из-за некоторых способов обработки данных MySQL (связи, транзакции, аудиты) иногда уступает другим СУБД по надежности;

PostgreSQL

Является самым профессиональным из всех трех рассмотренных СУБД. Она свободно распространяемая и максимально соответствует стандартам SQL. PostgreSQL или Postgres стараются полностью применять ANSI/ISO SQL стандарты своевременно с выходом новых версий.

От других СУБД PostgreSQL отличается поддержкой востребованного объектно-ориентированного и/или реляционного подхода к базам данных. Например, полная поддержка надежных транзакций, т.е. атомарность, последовательность, изоляционность, прочность (Atomicity, Consistency, Isolation, Durability (ACID).) Благодаря мощным технологиям Postgre очень производительна. Параллельность достигнута не за счет блокировки операций чтения, а благодаря реализации управления многовариантным параллелизмом (MVCC), что также обеспечивает соответствие ACID. PostgreSQL очень легко расширять своими процедурами, которые называются хранимые процедуры. Эти функции упрощают использование постоянно повторяемых операций.

Хотя PostgreSQL и не может похвастаться большой популярностью в отличии от MySQL, существует довольно большое число приложений, облегчающих работу с PostgreSQL, несмотря на всю мощность функционала. Сейчас довольно легко установить эту СУБД используя стандартные менеджеры пакетов операционных систем.

Достоинства PostgreSQL:

- открытое ПО, соответствующее стандарту SQL – PostgreSQL – бесплатное ПО с открытым исходным кодом. Эта СУБД является очень мощной системой; – Большое сообщество – существует довольно большое сообщество, в котором вы запросто найдёте ответы на свои вопросы;
- большое количество дополнений – несмотря на огромное количество встроенных функций, существует очень много дополнений, позволяющих разрабатывать данные для этой СУБД и управлять ими;
- расширения – существует возможность расширения функционала за счет сохранения своих процедур;
- объектность – PostrgreSQL это не только реляционная СУБД, но также и объектно-ориентированная с поддержкой наследования и много другого.

Недостатки PostgreSQL:

- производительность – при простых операциях чтения PostgreSQL может значительно замедлить сервер и быть медленнее своих конкурентов, таких как MySQL;
- популярность – по своей природе, популярностью эта СУБД похвастаться не может, хотя и присутствует довольно большое сообщество; – Хостинг – в силу вышеперечисленных факторов иногда довольно сложно найти хостинг с поддержкой этой СУБД.

1.4 ВЫВОДЫ

Исходя из вышесказанного, можно сделать вывод что использование комплекса приложений, которые могли бы заменить разрабатываемое приложение имеет ряд серьезных недостатков. Во-первых, каждый из аналогов проигрывает оригиналу по ряду причин. Во-вторых, пользователю придется переключаться между приложениями, и это замедлит работу.

После детального обзора возможных платформ и социального опроса будущих пользователей было принято решение разрабатывать нативное приложение под Android. В данном случае нативное приложение имеет ряд преимуществ, которые были описаны ранее и при этом не имеет недостатков, так как у пользователей нет телефонов с операционной системой IOS. В качестве языка программирования выбран – Java. На сегодняшний день это один из двух наиболее популярных языков для разработки на Android. Самое большое преимущество языка Kotlin является его фреймворки и библиотеки, которые разрабатываются конкретно под него. Но в данном проекте, весь необходимый функционал можно реализовать на Java.

В качестве баз данных будут использоваться SQLite и MSSql. Специфика будущего приложения подразумевает наличие как базы данных, расположенной на сервере, так и базы данных, встраиваемой в приложение.

2 ФОРМИРВОАНИЕ ТРЕБОВАНИЙ К СИСТЕМЕ

Для реализации данной системы необходим следующий набор подсистем:

1. Клиентское приложение с графическим интерфейсом, написанное под ОС Android.
2. SQLite база данных, которая будет хранить информацию об инцидентах и работах каждого пользователя.
3. MSSql база данных, которая будет хранить информацию с комментариями (недоделками).

2.1 ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

В данном подразделе определим какой функционал будет в мобильном приложение. Для этого воспользуемся диаграммой прецедентов (вариантов использования).

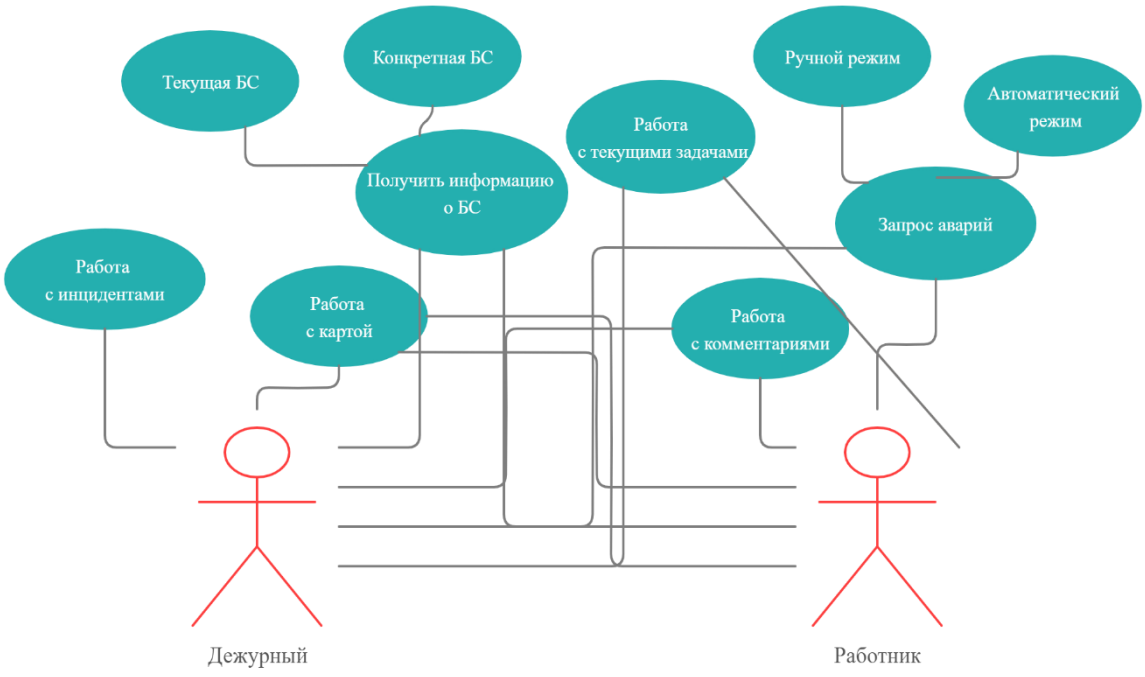

Рисунок 12 – Диаграмма вариантов использования мобильного приложения

На Рисунке 12 изображена диаграмма вариантов использования мобильного приложения.

Как говорилось ранее, отличие дежурного от пользователя, в том что дежурной кроме основной работы занимается инцидентами. Таким образом можно сформировать функциональные требования:

- работа с картой – пользователи должны уметь работать с картой. Им необходимо выводить на карту различные типы базовых станций – текущую, все аварии, и все базовые в радиусе 35 километров от ближайшей базовой станции;
- получить информацию о БС – информация о базовых бывает краткая и полная. Также необходимо запоминать номера телефонов обслуживающих организаций;
- текущая БС – базовая станция, к которой в данный момент подключен телефон пользователя;
- конкретная БС – базовая станция, выбранная пользователем в приложении;
- работа с текущими задачами – необходимо брать в работу и отчитываться о выполнении задач, поставленных руководством. Для этого они должны быть выведены на главный экран;
- запрос аварий – пользователь должен иметь актуальную информацию о авариях, чтобы правильно распределить рабочее время;
- ручной режим запроса авария – пользователь должен иметь возможность сам запросить аварию у сервера и получить ответ от него;
- автоматический режим запроса аварий – каждый час аварии будут запрашиваться у сервера автоматически.

2.2 НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

- приложение должно быть написано под ОС Android версии 6.0 и выше;
- необходимо использовать базы данных для хранения информации о базовых станциях и работах;
- в качестве карт необходимо использовать Google Maps;
- пользователь должен отправлять и принимать произвольные и шаблонные sms сообщения;
- пользовательский интерфейс мобильного приложения разрабатывается с учетом пожеланий будущих пользователей.

3 РАЗРАБОТКА АРХИТЕКТУРЫ

На сегодняшний день существует четыре вида архитектуры мобильных приложений – это MVC, MVI, MVP и MVVM[25]. У каждой из них есть как свои преимущества, так и свои недостатки.

Существует мнение, что поскольку ка каждый проект — уникален, то нет идеальной архитектуры. Для того, чтобы выбрать архитектуру будущего приложения предлагаю ознакомится с существующими.

Model View Controller

У архитектуры MVC есть два варианта: **контроллер-супервизор (supervising controller)** и **пассивное представление (passive view)**.

Контроллера-супервизор не популярен при мобильной разработке.

Архитектуру MVC можно охарактеризовать двумя пунктами:

- представление(View) — это **визуальная проекция модели**
- контроллер — это **соединение между пользователем и системой**

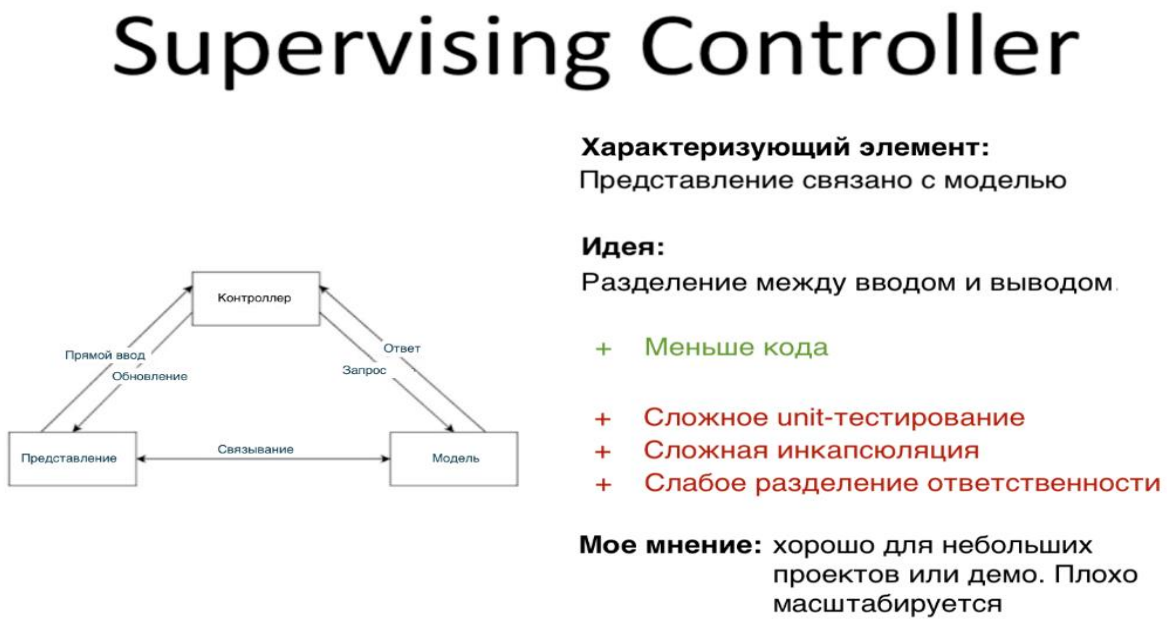

Рисунок 13 – Архитектура MVC

Представление отвечает за ту часть проекта, с которой взаимодействует пользователь. Контроллер отвечает за логику приложения, в нем описываются все необходимые для работы методы. Модель хранит в себе данные.

Model View Presenter

Данная архитектура облегчает unit-тестирование, **презентер (presenter)** прост для написания тестов, а также может многократно использоваться, потому что представление может реализовать несколько интерфейсов.

Если говорить о корректности создания интерфейсов, то MVP и MVC необходимо рассматривать как основные идеи, а не паттерны разработки.

Рисунок 14 – Архитектура MVP

Model View View-Model

Существует другой способ биндинга: вместо привязывания представления к интерфейсу, мы привязываем элементы представления к параметрам view-модели — такая архитектура называется MVVM. Когда мы меняем параметры в модели, в разметку тоже вносятся изменения.

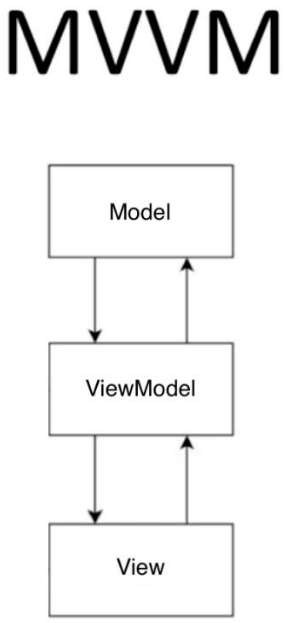

Рисунок 15 –Архитектура MVVM

Итогом применения паттерна MVVM является функциональное разделение приложения на три компонента, которые проще разрабатывать и тестировать, а также в дальнейшем модифицировать и поддерживать.

Model View Intent

MVI – это такой шаблон проектирования, в котором Модель (Model) является активным компонентом, принимающим на вход Намерения (Intents) и производящая Состояния (State). Представление (View) в свою очередь принимает Модели Представления (View Model) и производит те самые Намерения. Состояние преобразуется в Модель Представления при помощи функции-трансформера (View Model Mapper). Схематически шаблон MVI можно представить следующим образом:

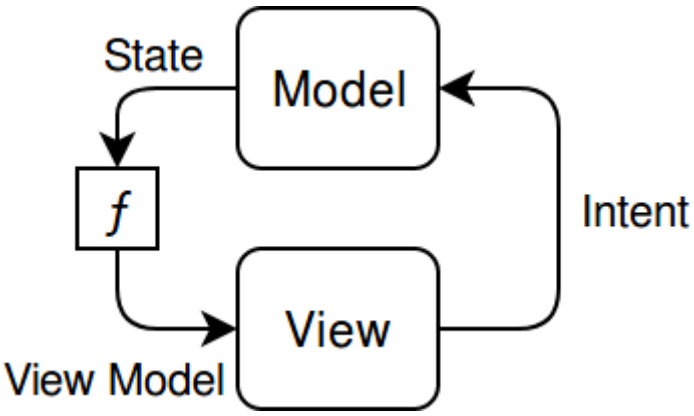

Рисунок 16 – Архитектура MVI

Представление нужно для отображения данных. Данные для каждого Представления группируются в Модель Представления (View Model) и обычно представляются в виде data-класса.

В MVIDroid Представление не производит Намерения напрямую. Вместо этого оно производит События Представления (UI Events), которые затем преобразуются в Намерения при помощи функции-трансформера.

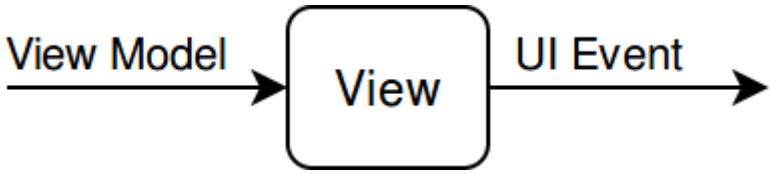

Рисунок 17 – Функция трансформер MVI

В библиотеке понятие Модели немного расширено, здесь она производит не только Состояния, но и Метки (Labels). Метки используются для общения Моделей между собой. Метки одних Моделей могут быть преобразованы в Намерения других Моделей при помощи функций-трансформеров. Схематически Модель можно представить так:

Рисунок 18 – Схематическое представление Модели

3.1 АРХИТЕКТУРА БУДУЩЕГО ПРИЛОЖЕНИЯ

Главный плюс MVI – это MVI тот факт, что шаблон может значительно улучшить масштабируемость.

Таким образом, на основании вышеперечисленной информации была разработана архитектура будущего приложения.

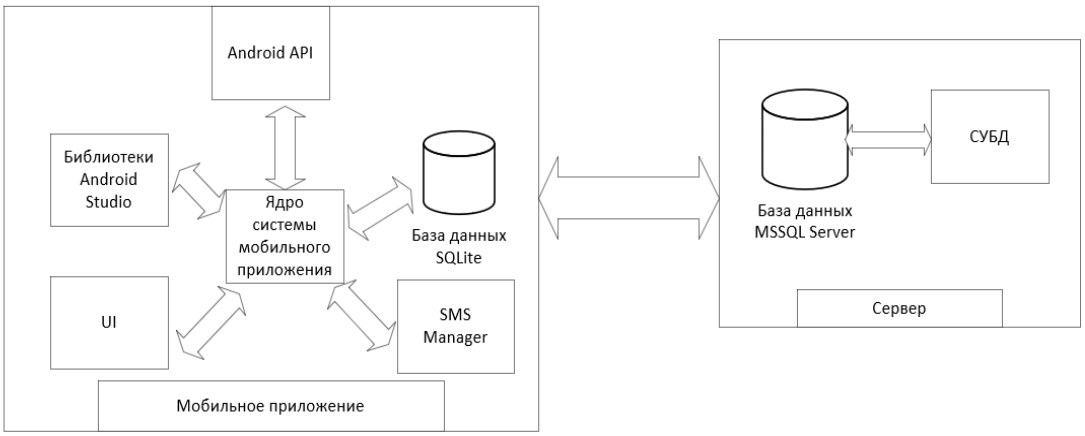

Рисунок 19 – Архитектура приложения

На Рисунке 13 показана схема компонентов мобильного приложения, которая формирует ее архитектуру. Проект необходимо разделить на клиент и сервер. В качестве клиента будет выступать мобильное приложение. Рассмотрим каждый компонент подробнее:

База данных MSSQL Server – база данных, которая находится на удаленном сервере. Необходима для того, чтобы все пользователи обменивались актуальной информацией.

СУБД - компонент, отвечающий за взаимодействие других компонентов серверной части с базой данных.

База данных SQLite– это локальная база данных, под управлением СУБД SQLite. В этой базе данных содержится информация о работах и инцидентах.

Библиотеки Android Stuido - Пакет, взаимодействующий с функциями операционной системы с помощью Java.

Android API - пакет, предназначенный для взаимодействия с функциями общими для всех мобильных устройств.

SMS Manager – компонент отвечающий, за перехват, обработку и отправку SMS сообщений.

UI – графический интерфейс пользователя.

Ядро системы мобильного приложения - ядро, обеспечивающее взаимодействие между всеми компонентами мобильного приложения.

3.2 ОПИСАНИЕ ДАННЫХ

Принято разделять данные на статические и динамические. Статические данные не изменяются в процессе выполнения программы. Динамические же наоборот изменяются. БД принято использовать для динамических данных.

Если посмотреть все данные, которые потребуются нам для работы, их можно разделить на статические и динамические. Вся информация о базовых станциях, является статической, так как не изменяется в процессе работы, и очень редко изменяется в реальности.

Для обработки статической информации не целесообразно использовать базы данных. Вместо этого создадим в папке Resource новый XML файл и выгрузим в него всю имеющуюся информацию. После чего с помощью регулярного выражения распределим данные по массивам и переменным.

Часть информации, которую нельзя хранить в ресурсах лучше хранить в SQLite[26]. Эту информацию нет необходимости использовать на других устройствах. Рассмотрим информацию, которая хранится в SQLite[27].

Список значений:

— NumberBts(int) – номер базовой станции;

— ShortName(String0 – короткое название базовой станции;

— NumberIncident(int) – номер инцидента;

— TimeIncident(int) – время инцидента;

— StatusIncident(int) – статус инцидента, по нему определяется инцидент или работа;

— MessOfRequistAlarms(String) – текст запрашиваемой смс;

— DateOfRequistAlarms(Date) – дата, когда запрашивался инцидент;

— NumberOfAlarms(int) – номер комментария(недоделки);

— Priority(int) – приоритет инцидента, число от 1 до 5;

— StatusRequestAlarms(String) – статус запроса аварии;

— StatusOfEnergy(boolean) – флаг, запоминает, отправлялось ли сообщение энергетикам;

— StatusOfLowPower(boolean) – флаг, запоминает, понижалось или нет мощность на базовой станции;

— LonCoordinate(double) – координата широты;

— LatCoordinate(double) – координата долготы;

— Message(String) – полный текст сообщения;

— DateOfSla(DateTime) – время, до которого нужно успеть закрыть инцидент.

Динамические данные в проекте – это комментарии, написанные пользователями к базовым станциям (“недоделки”) и информация о текущих работах, инцидентах.

Поскольку пользователи должны иметь доступ не только к своим, но и чужим комментариям необходимо будет использовать базу данных MSSql, расположенную на удаленном сервере. Приложение будет взаимодействовать с БД с помощью **Java Persistence Query Language[28**].

JPQL — платформенно-независимый объектно-ориентированный язык запросов, являющийся частью спецификации Java Persistence API (JPA).

JPQL используется для написания запросов к сущностям, хранящимся в реляционной базе данных. JPQL во многом похож на SQL, но в отличие от последнего, оперирует запросами, составленными по отношению к сущностям JPA, в отличие от прямых запросов к таблицам базы данных.

Рассмотрим информацию, которая хранится на сервере.

Список значений MSSql:

- Id(Int) – поле идентификатора (первичный ключ);
- NumberBS(Int) – номер базовой станции;
- Comment(String) – комментарий оставленный пользователем(недоделка);
- Date(date) – дата, когда был оставлен комментарий;
- Who(String) – Фамилия человека, оставившего комментарий.

4 РЕАЛИЗАЦИЯ

4.1 РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА

Для того, чтобы ознакомиться с функционалом приложения рассмотрим его возможности.

Работа приложения, для нового пользователя начинается с ввода пароля.

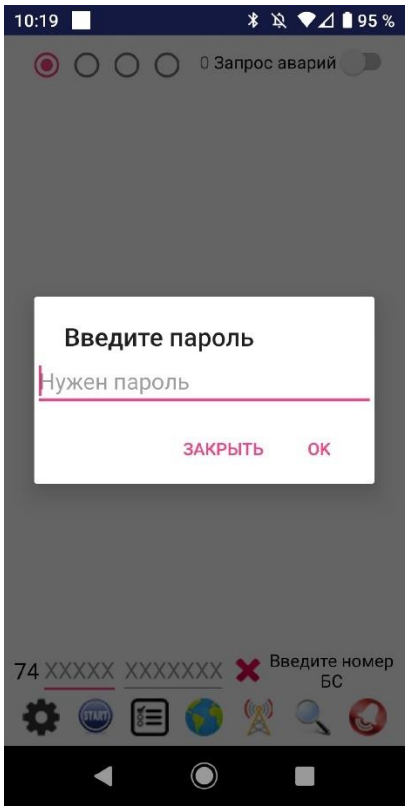

Рисунок 20 – Диалоговое окно с вводом пароля

Если пароль не правильный, приложение будет закрыто. Если пароль правильный, пользователь начинает работу с приложением. На рисунке 21 вы можете увидеть главное меню приложения.

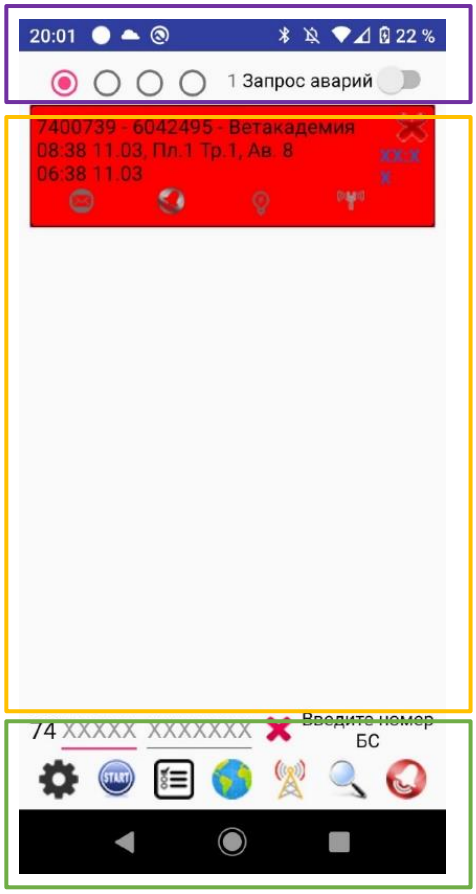

Рисунок 21 – главное меню приложения

Согласно современным требованиям к разработке мобильных приложений принято разделять интерфейс на три части. В верхней части интерфейса(фиолетовый цвет) располагаются кнопки навигации. Центральная часть главного меню(Желтый цвет) является рабочей областью В ней располагаются работы, активные и завершенные инциденты, а также избранные базовые станции. Нижняя часть главного меню(зеленый цвет) отвечает за основной функционал приложения. В ней расположены панель поиска, информация о выбранной базовой и кнопки. Рассмотрим по подробнее функционал каждого элемента:

Счетчик аварий – это число, которое показывает количество работ или инцидентов.

Запрос аварий – представляет собой кнопку типа switch. Данная кнопка имеет два режима. Если кнопка включена аварии будут запрашиваться автоматически.

Навигационные элементы – представляют собой группу кнопок типа RadioButton. В зависимости от активной кнопки меняется

центральная(рабочая) часть приложения. Первая кнопка отвечает за активные инциденты. Вторая за завершенные инциденты. Третья за выбранные базовые станции, выполняет роль закладок Четвертная кнопка отвечает за активные работы. Результат работы кнопок навигации показан на рисунке 22:

Рисунок 22 – Работа кнопок, отвечающих за навигацию

Центральная часть главного меню (рабочая область) выводит либо работы, либо инциденты. Инциденты бывают красные(срочные) и серые (в активном ожидании). Рассмотрим более подробно действие, которые можно совершать в рабочей области приложения. Для этого рассмотрим рисунок 23:

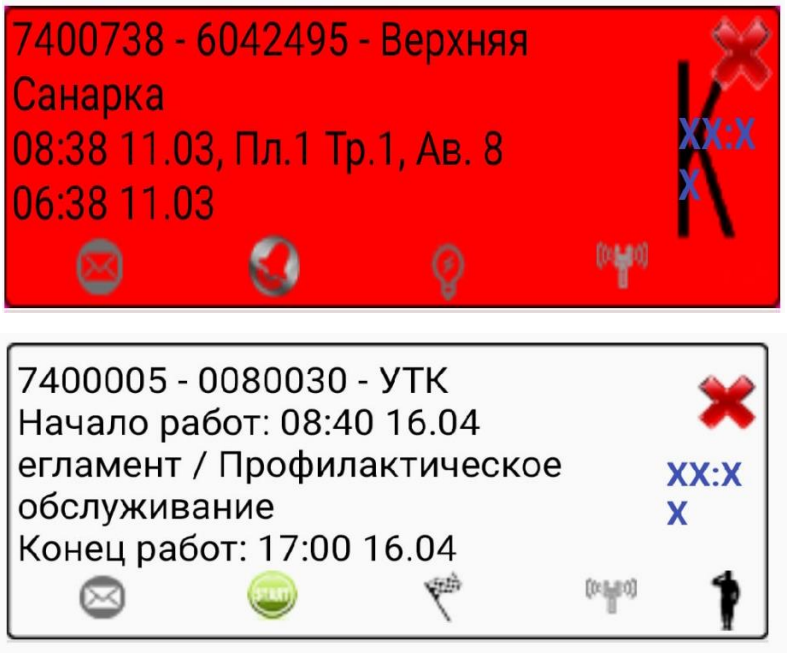

Рисунок 23 – Отличие инцидента от работы

На рисунке 23 изображены инцидент и работа. Часть функционала схожа, но часть различается. Рассмотрим основные возможности:

- кнопка с изображением конверта – показывает полное смс сообщение, полученное пользователем. Оно необходимо для уточнения интересующей информации;
- кнопка с изображением звонка – позволяет отключить уведомления для данного инцидента;
- кнопка с изображением лампочки – имеет два режима работы. Если пользователь сообщил энергетикам о проблеме на станции, он может включить лампочку и таким образом запомнить, что проблема связанна с электричеством;
- кнопка с изображением сотовой вышки – имеет два режима работы. Если пользователь переключил станцию в режим уменьшенного энергопотребления, данный маркер напоминает ему об этом;

- большая буква К – информирует о том, что ключи от базовой станции находятся у дежурного;
- XX:XX – позволяет выставить время. Работает в режиме напоминания пользователю о каком-либо событии;
- кнопка START – взять в работу аварию. Отправляет смс на сервер, в которой указывает номер станции. Таким образом пользователь сигнализирует системе о начале работ;
- кнопка с изображением флага – закончить работу. Отправляет смс на сервер. Пользователь сигнализирует о завершении работ;
- иконка солдата – имеет три режима работы. Меняет цвет в зависимости от прогресса выполнения задачи.

Рассмотрим нижнюю часть главного меню.

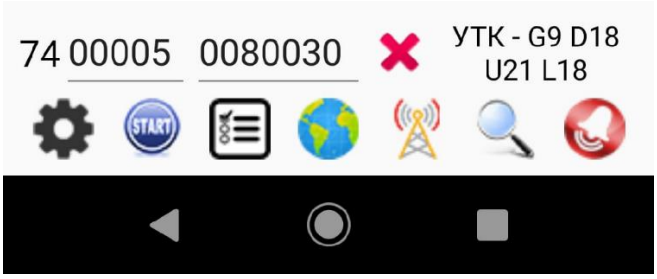

Рисунок 24 – Нижняя часть приложения

Опишем каждый элемент рисунка 24 подробней. В левом верхнем углу можно увидеть панель поиска. 74 – номер региона. Пять чисел после – номер базовой станции. Восемь чисел после – номер аварии. Выбранная пользователем авария автоматически заносится в эту часть экрана. При нажатии на номер станции или аварии можно использовать ручной ввод. Красный крестик сбрасывает значения.

Часть рабочей области от красного крестика информирует пользователя о нагрузке станции. В ней перечислены типы антенн(G9, D18, U21, L18). Эта информация помогает рабочему принять решение об отключении приоритетной нагрузки. При нажатии на эту область вылезет

текстовое окно. В нем пользователь может посмотреть номера телефонов людей, ответственных за станцию.

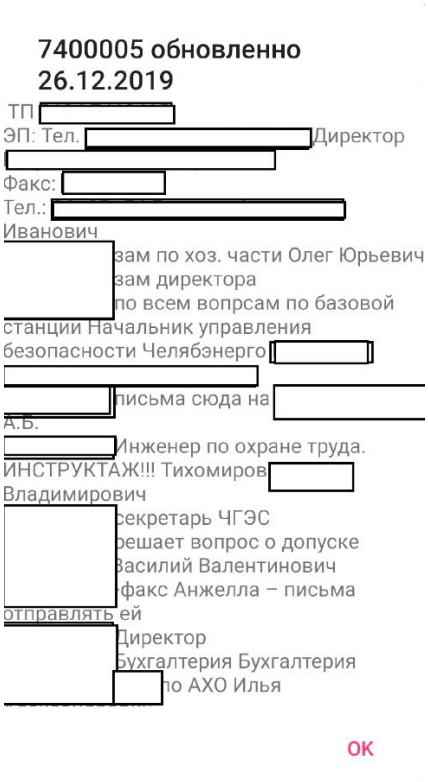

Рисунок 25 – Текстовое окно с информацией

Обращаясь, к рисунку 24, рассмотрим работу кнопок.

Кнопка номер 1 с изображением шестеренки. При нажатии на нее появляется окно, с различным текстом. Результат работы показан на рисунке 26:

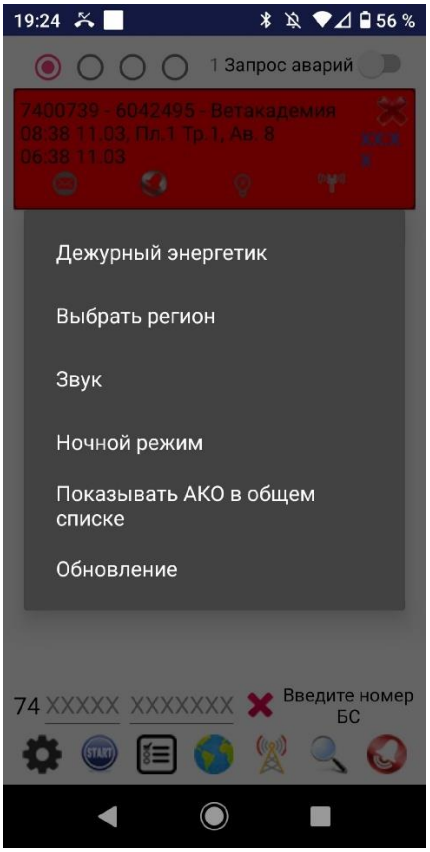

Рисунок 26 – Результат работы кнопки номер 1

Каждый текстовый элемент является активным и при нажатии на него происходят определенное действие:

- дежурный энергетик – при нажатии на этот элемент, пользователь может выбрать дежурного энергетика из списка;
- выбрать регион – пользователь может выбирать регион, в котором работает. Данное приложение имеет возможность работать в Сочи и Челябинской области;
- звук – пользователь выбирает какие уведомления отключить.
- ночной режим – активируется режим, в котором пользователь не получает уведомления с 12 00 до 8 00;
- АКО в общем списке – инцидента в активном ожидании загружаются совместно с простыми инцидентами. Если после загрузки необходимо оставить только обычные инциденты, пользователь может нажать эту кнопку;

- обновление – присылает запрос на сервер, где хранится APK файл. Если версии разные, происходит обновление программы. Функция ручного обновления.

Кнопка номер 2 с изображением слова START. При нажатии на кнопку появляется окно с текстом. Результат работы показан на рисунке 27:

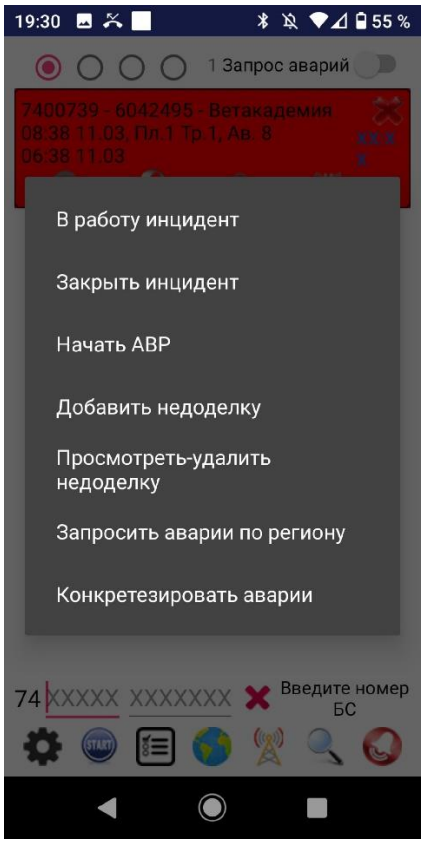

Рисунок 27 – Результат работы кнопки номер 2

Рассмотрим более подробно каждый элемент всплывающего меню:

- в работу инцидент – отправляет смс сообщение, которое сигнализирует о том, что пользователь ознакомился с предстоящей работой;
- закрыть инцидент – открывает окно, с заранее заготовленными шаблоны. При нажатии на шаблон, текст отправляется и сигнализирует о закрытии аварии;
- начать АВР – отправить смс, сигнализирующую о физическом начале работ. Отправляется, когда работник приехал на базовую станцию;

- добавить недоделку – пользователь вводит номер базовой станции и текст недоделки. После чего она отправляется на сервер;
- просмотреть-удалить недоделку – показывает все имеющиеся недоделки по выбранной базовой станции. После просмотра можно удалить недоделку, если информация не актуальна;
- запросить аварии по региону – запрос аварий у sms сервера. Сервер присылает все аварии разделенные на три группы – аварии по питанию, аварии по температуре, аварии по сигнализации;
- конкретизировать аварии – позволяет конкретизировать информацию о авариях, полученных по региону.

Кнопка номер 3 с изображением списка. Запрашивает инциденты в активном ожидании. Это инциденты, которые невозможно выполнить в ближайшее время. Они сохраняются, я пользователь может получить информацию о них, если необходимо.

Кнопка номер 4 с изображением планеты Земля. Открывается диалоговое окно с выбором. Пользователь может вывести на карту конкретные базовые станции, все базовые в радиусе 35 км от пользователя или все активные аварии. После выбора режима работы открывается карта. Подробнее на рисунке 28:

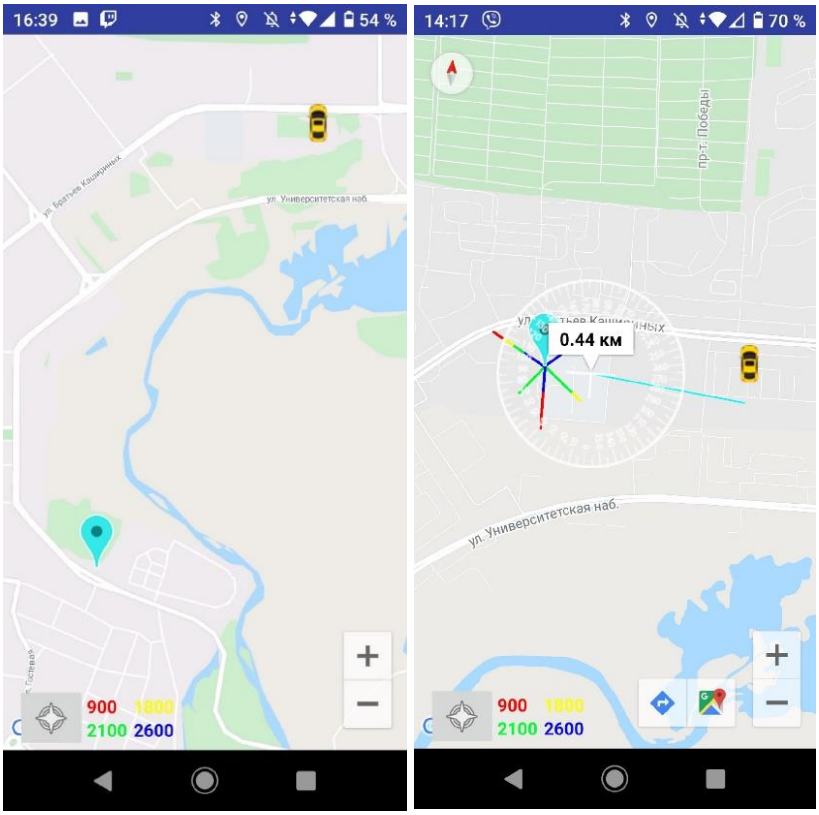

Рисунок 28 – Работа класса MapsActivity

На карте пользователь видит выбранные базовые станции и свое местоположение. При нажатии на базовую станцию рисуются линии направления антенн и линии радиорелейного пролета. При нажатии и удержании можно поставить точку и соединить ее со второй точкой. Таким образом можно измерить расстояние и посмотреть точное значение углов.

Кнопка номер 5 с изображением сотовой вышки. Определяет базовую станцию, к которой пользователь подключен в данный момент. Пользователь получает всплывающее сообщение с информацией. Результат работы можно увидеть на рисунке 29:

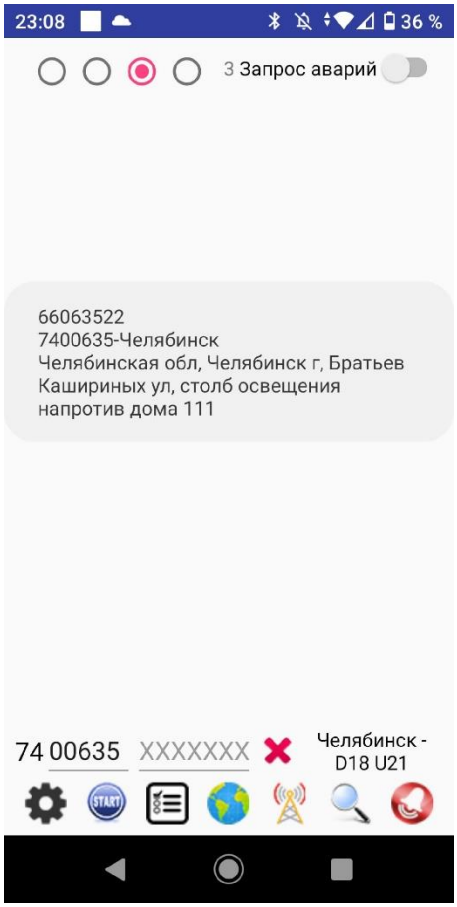

Рисунок 29 – работа кнопки номер 5

Кнопка номер 6 с изображение лупы. Осуществляет функцию поиска. Во всплывающем окне пользователю легче будет найти интересующую его базовую станцию.

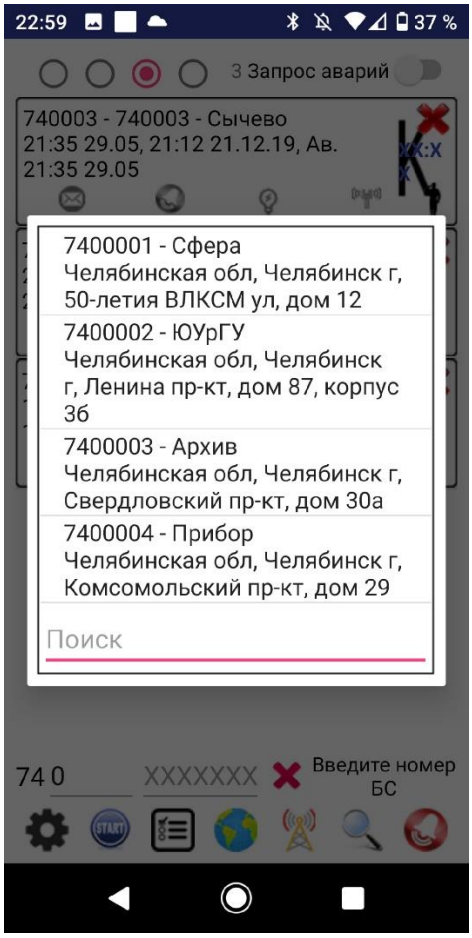

Рисунок 30 – Работа кнопки номер 6

Кнопка номер 7 с изображением звоночка – запрос информации об аварии у sms сервера. Ручное управление.

4.2 РЕАЛИЗАЦИЯ КЛАССОВ

Мобильное приложение разделено на несколько классов, каждый из которых отвечает за определенный функционал. Каждый класс имеет свои методы и переменные. С листингом каждого класса можно ознакомиться в приложении к пояснительной записке.

В пункте 4.1 была рассмотрена работа не всех классов, так как некоторые из них работают в фоновом режиме. Опишем их функционал:

AlarmReciever – класс, позволяющий каждый час запрашивать аварии у sms сервера.

RecieveSms – класс, отвечающий за перехват сообщений.

BsData – класс, который работает с базой данных SQLite.

FtpReciver – класс, отвечающий за работу Ftp протокола. Он необходим для подключения к серверу и проверки версии программы.

JSEEProvider – класс, необходимый для создания безопасного интернет соединения.

MapsActivity – класс для работы с Google maps. У данного класса есть свой Activity. Результат его работы был рассмотрен ранее.

Notify – класс, отвечающий за отправку уведомлений. Уведомления приходят в “шапку” телефона. Уведомления бывают разных типов. Например, напоминают о времени выполнения работ.

ProcessingSMS – класс, отвечающий за парсинг входящего sms сообщения.

SendSMS – класс, отвечающий за отправку sms сообщений.

SreenOn – класс позволяющий включать экран пользователя на несколько секунд.

SreenOnReciver – перехватывает уведомления(Notify). Для того, чтобы включить экран в момент получения уведомления.

RequestAlarm – класс, отвечающий за напоминание пользователю о работах и инцидентах. Каждый час запрашивает у SMS сервера незавершённые работы и инциденты и сравнивает их с пользовательскими. В случае отсутствия отчета, напоминает пользователю о том, что необходимо закрыть инцидент.

AddNedodelki – класс, отвечающий за работу с MSSql сервером. Отсылает запросы на сервер и получает ответы. Запросы отсылаются с помощью JPQL запроса.

5 ТЕСТИРОВАНИЕ И ОТЛАДКА

5.1 МЕТОДОЛОГИИ ТЕСТИРОВАНИЯ

Приложение на Android чаще всего тестируют автоматически с помощью внутреннего тестирования Google Play Console. Данный сервис составляет отчет о тестирование[29] после того, как установочный файл APK[30] был загружен в раздел Версии приложения. Но данный вариант не возможен, потому что нет возможности загрузить приложение в GooglePlay. Google запрещает приложениям отправлять SMS сообщения.

Таким образом необходимо будет использовать ручное тестирование.

Ручное тестирование программного обеспечения– это процесс проверки ПО, выполняемый специалистами вручную. Это значит, что для его проведения не используются какие-либо специальные автоматизированные средства[31].

Программное обеспечение проверяется инженерами по тестированию, которые берут на себя роль конечных пользователей, моделируют ситуации в соответствии с тестовыми сценариями и фиксируют результат. Задача ручного тестирования программного обеспечения — выявить любое поведение, отличающееся от ожидаемого пользователем. Это важный этап обеспечения качества ПО, который направлен на тщательное исследование программного кода и выявление ошибок в работе систем.

При ручном функциональном тестировании (РФТ) проверка различных функций ПО осуществляется тест-кейсами. Основная цель РФТ — определить, насколько разработанное программное обеспечение соответствует функциональным требованиям, то есть способно ли оно при определенных условиях решать задачи, необходимые пользователям.

Также тестированием занимались будущие пользователи. На различных этапах разработки пользователи получали бета версии приложения. Это так называемое тестирование [32]. Благодаря нему удалось решить некоторые проблемы и сделать интерфейс более удобным. После каждого этапа Юзабилити тестирования была произведена отладка приложения.

В результате ручного тестирования формируются кейсы. Первостепенная задача тестирования – проверить соответствует ли приложение функциональным требованиями. Результаты ручного тестирования представлены в таблице 2.

Таблица 2 – Результаты тестирования

Требование	Ожидаемый исход	Реальный исход	Вывод
Работа с картой	Пользователь должен уметь выводить на карту различную информацию.	На рисунке 28 видно, что карта работает исправно. Карта поддерживает дополнительные функции, которые не были описаны раньше.	Компонент, отвечающий за работу с картой, работает исправно.
Получение информации о БС	Пользователь должен уметь получать различного рода информацию о базовых станциях.	На рисунках 25, 23, видно, что пользователь может получить как информацию о базовых станциях, так и информацию о номерах телефонах лиц, связанных с БС.	Функция получения информации работает исправно.
Работа с текущими задачами	Пользователь должен видеть поставленные задачи, брать их в работу, отчитываться о их выполнении.	На рисунках 22,23 видно, что пользователь получает как инциденты, так и аварии. Есть возможность переключения между ними. Также пользователь может работать с задачи автономно, брать в работу и закрывать их.	Функция получения и работы с задачами работает исправно.

Продолжение таблицы 2

Требование	Ожидаемый исход	Реальный исход	Вывод
Запрос аварий.	Пользователь должен уметь запрашивать аварии как в ручном, так и в автоматическом режимах.	Включение и отключение автоматического режима происходит благодаря кнопки в верхней части экрана (запрос аварий). Пользователь может вручную запрашивать интересующую его аварию с помощью крайней правой кнопки в нижней части меню.	Два типа запроса аварий работают исправно.

Приложение успешно прошло все тесты. Из результатов тестирования можно сделано вывод – приложение функционирует правильно.

ЗАКЛЮЧЕНИЕ

По итогам выпускной квалификационной работы было разработано приложение для автоматизации пуско-наладочных работ в соответствии с требованиями заказчика. Благодаря обзору литературы и аналогов были сделаны выводы по концепции будущего приложения. После чего были определены требования для будущей системы, учитывающие интересы двух групп пользователей. Был произведен анализ современных архитектур мобильного приложения и разработана собственная архитектура будущего приложения. Приложение было реализовано, протестировано и отлажено. В настоящий момент приложение активно используется в городе Челябинске, Челябинской области и городе Сочи.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Типы и примеры приложений. – http://ipkey.com.ua/faq/984-application.html. – Дата обращения: 07.05.2020.

2 Вся статистика интернета на 2020 год — цифры и тренды в мире и в России. – https://www.web-canape.ru/business/internet-2020-globalnaya-statistika-i-trendy/. – Дата обращения: 07.05.2020.

3 Статистика интернета 2017-2018 в мире и в России. – https://www.webcanape.ru/business/internet-2017-2018-v-mire-i-v-rossii-statistika-i-trendy. – Дата обращения: 07.05.2020.

4 Отчет App Annie: весь рынок мобильных приложений за 2019 год. – https://qmobi.agency/blog/app-annie-2018-2019/. – Дата обращения: 08.05.2020.

5 Приложение в Google Play – Any.Do. – https://play.google.com/store/apps/details?id=com.anydo . – Дата обращения: 08.05.2020

6 Приложение в Google Play – Trello. – https://play.google.com/store/apps/details?id=com.trello . – Дата обращения: 08.05.2020

7 Приложение в Google Play – Totoist. – https://play.google.com/store/apps/details?id=com.todoist . – Дата обращения: 08.05.2020

8 Приложение в Google Play – 2ГИС. – https://play.google.com/store/apps/details?id=ru.dublgis.dgismobile&hl=ru . – Дата обращения: 08.05.2020

9 Приложение в Google Play – Яндекс Навигатор. – https://play.google.com/store/apps/details?id=ru.yandex.yandexnavi . –Дата обращения: 08.05.2020

10 Приложение в Google Play – Pulse Sms. – https://play.google.com/store/apps/details?id=xyz.klinker.messenger . – Дата обращения: 08.05.2020

11 Приложение в Google Play – Handcent Next SMS. – https://play.google.com/store/apps/details?id=com.handcent.app.nextsms . – Дата обращения: 08.05.2020

12 Топ 10 приложений планировщиков. - https://androidinsider.ru/obzory-prilozhenij/top-10-prilozhenij-planirovshhikov-pod-android.html#todoist. – Дата обращения: 08.05.2020.

13 22 мобильных приложения для управления проектами. – https://spark.ru/startup/smsaero/blog/39785/22-mobilnih-prilozheniya-dlya-upravleniya-proektami. –Дата обращения:09.05.2020

14 40 сервисов для управления задачами и проектами. – https://vc.ru/services/50333-40-servisov-dlya-upravleniya-zadachami-i-proektami. – Дата обращения: 10.06.2020

15 Обзор Навигатор от Яндекс. –https://softdroid.net/navigator-android-karty-marshruty#2. – Дата обращения:10.05.2020

16 Выбор SMS-мессенжера для Android устройств. – https://overclockers.ru/lab/show/84944/vybiraem-sms-messendzher-dlya-android-ustrojstv-pulse-sms-handcent-next-sms-a-takzhe-itogi-rassmotreniya-pyati-prilozhenij#2. – Дата обращения:10.05.2020

17 На чём писать кроссплатформенные приложения. – https://livetyping.com/ru/blog/na-chem-pisat-krossplatformennye-prilozhenija. – Дата обращения:10.05.2020

18 About Objective-C. – https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/P rogrammingWithObjectiveC/Introduction/Introduction.html. – Дата обращения: 10.05.2020.

19 Swift – Apple (RU). – https://www.apple.com/ru/swift. – Дата обращения: 10.05.2020.

20 Java – Объектно-ориентированный язык программирования / Хабр. – https://habr.com/ru/hub/java. – Дата обращения: 10.05.2020.

21 Kotlin Programming Language. – https://kotlinlang.org. – Дата обращения: 10.05.2020.

22 Обзор языка C#. – руководство по C#. – https://docs.microsoft.com/ruru/dotnet/csharp/tour-of-csharp. – Дата обращения: 10.05.2020.

23 Facebook – Выполните вход или зарегистрируйтесь. – https://www.facebook.com/. – Дата обращения: 10.05.2020.

24 Обзор существующих баз данных. – https://devacademy.ru/article/sqlite-vs-mysql-vs-postgresql. – Дата обращения: 10.05.2020.

25 Паттерны разработки: MVC vs MVP vs MVVM vs MVI. – https://habr.com/ru/post/344184/. – Дата обращения: 12.05.2020.

26 Руководство по SQLite. – https://proglib.io/p/sqlite-tutorial. – Дата обращения: 12.05.2020.

27 Добавление, удаление и обновление данных в SQLite. – https://metanit.com/java/android/14.2.php. – Дата обращения: 12.05.2020.

28 JavaPersistenceQueryLanguage. – https://ru.wikipedia.org/wiki/Java_Persistence_Query_Language. Дата обращения: 12.05.2020.

29 Отчеты о тестировании – Справка – Play Console. – https://support.google.com/googleplay/androiddeveloper/answer/7002270#sources. – Дата обращения: 18.05.2019.

30 Что такое APK-файлы на Android и зачем они нужны? | AndroidLime. – https://androidlime.ru/apk-files. – Дата обращения: 18.05.2019.

31 Ручное тестирование. – https://www.appline.ru/services/testing/ruchnoe-testirovanie. – Дата обращения: 18.05.2019.

32 Юзабилити тестирование. – https://ru.wikipedia.org/wiki/Юзабилити-тестирование. – Дата обращения: 18.05.2019.