МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Кафедра IT-инжиниринг

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой
PhD, доцент
_____ Т.С. Картбаев
«____» _____ 2019 г.

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Разработка мобильного путеводителя для путешественников по разным точкам мира

Специальность: 5В070400 – «Вычислительная техника и программное обеспечение»

Выполнила: Савина Т.Е. Группа: ВТ-15-1
Научный руководитель: ст. преп. Смагулова С.Е.

Консультанты:
по экономической части: к.э.н., профессор _____ Ж.Г. Аренбаева
«13» мая 2019 г.

по безопасности
жизнедеятельности: д.т.н., ст. преп. _____ Ш.Ш. Бекбасаров
«14» мая 2019 г.

по применению
вычислительной техники: ст. преп. _____ М.Н. Майкотов
«13» мая 2019 г.

Нормоконтролер: ст. преп. _____ АА. Айткazина
«15» мая 2019 г.

Рецензент: к.т.н., доцент _____ Б.С. Каленова
«____» _____ 2019 г.

Алматы 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Институт систем управления и информационных технологий

Кафедра IT-инжиниринг

Специальность 5В070400 – «Вычислительная техника и программное обеспечение»

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Савиной Тамаре Евгеньевна
Тема проекта: Разработка мобильного путеводителя для путешественников по разным точкам мира

Утверждена приказом по университету № 124 от «26» октября 2018 г.

Срок сдачи законченного проекта «24» мая 2019 г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): Руководство системы менеджмента качества на предприятии; международные стандарты ИСО-9001, данные преддипломной практики.

Перечень вопросов, подлежащих разработке в дипломном проекте, или краткое содержание дипломного проекта:
- аналитическая часть;
- проектная часть;
- практическая часть;
- экономическая часть;
- безопасность жизнедеятельности;
- приложение А. Техническое задание;
- приложение Б. Листинг программы;
- приложение В. Акт внедрения.

Перечень графического материала (с точным указанием обязательных чертежей): представлены 11 таблиц, 38 иллюстрации.

Основная рекомендуемая литература:

1 Лаптев Г.Д., Ладионенко М.А. и др. Компетентностный подход и роль дизайн-мышления в обучении инновационному предпринимательству.— М.: ТЕИС, 2010.

2 Каймин В.А. Информатика: Учебное пособие: Изд. 2-е. Издательство РИОР, 2007.

3 Варакин М.В. Разработка мобильных приложений под Android. УЦ «Специалист» при МГТУ им. Н. Э. Баумана, 2012.

4 Моргунов Е.П, Лузанова П.В. PostgreSQL. Основы SQL. – СПб.: БХВ-Петербург, 2018. – 336 с.

Консультации по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант	Сроки	Подпись
Экономическая часть	Аренбаева Ж.Г.	04.03. 2019 – 13.05. 2019	
Безопасность жизнедеятельности	Бекбасаров Ш.Ш.	04.02 – 13.05.19,	
Программное обеспечение	Майкотов М.Н.	02.05 - 14.05.2019	
Нормоконтролер	Айтказина А.А.	02.04 – 15.05.19	

ГРАФИК
подготовки дипломной проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Аналитическая часть	15. 01. 2019	
Проектная часть	30. 01. 2019	
Разработка мобильного приложения	15. 05. 2019	

Дата выдачи задания «29» октября 2018 г.

Заведующий кафедрой _____ Т.С. Картбаев

Научный руководитель проекта _____ С.Е. Смагулова

Задание принял к
исполнению студент _____ Т.Е. Савина

Андатпа

Бұл дипломдық жұмыс үшін басқа елдерге саяхаттауды жоспарлайтындарға арналған мобильді қосымша әзірленді, сонымен қатар бұл қосымша біздің туған еліміздің қонақтарына өте қолайлы. "Travel Mate" бағдарламасы әр түрлі елдер бойынша жолсілтеме, жақын іс-шаралар мен концерттер тізімі, белгілі бір орынға жету үшін көрікті және кеңестер бар. Сондай-ақ пайдаланушы өз саяхат жоспарын жасай алады. Өнім 8-нұсқадағы Java бағдарламалау тілінде 4.10.3 нұсқасы Android Studio сияқты бағдарламалық құралдардың көмегімен жасалған.

Жұмыс ортасы элементтерінің қызметкерлерге әсерін және олардың алдын алу жолдарын есепке алу және өнімді жобалаудың экономикалық тиімділігін есептеу қарастыру үшін қосымша пункттер болды.

Осы дипломдық жұмысты орындау қорытындысы әлеуетті, экономикалық негізделген, пайдалануға дайын бағдарламалық өнім болып табылады.

Аннотация

Для данной дипломноого проекта было выполнено мобильное приложение для тех, кто планирует свое путешествие в другие страны, а так же это приложение идеально подойдет гостям нашей родной страны. Приложение “Travel Mate” содержит путеводитель по разным странам, список ближайших мероприятий и концертов, достопримечательности и советы как добраться до определенного место назначения. А так же пользователь может содать свой собственный план путешествия. Мобильное приложение было разработано при помощи Android Studio версии 4.10.3. Данная программа написана на языке объектно-ориентированном языке Java.

Был произведен расчет экономической стоимости программного продукта, а так же был выявлен учет влияние внешних факторов на сотрудников и способы их устранения.

В данной дипломной работе представлен потенциальный, а так же экономический обоснованный, готовый к эксплуатации программаный продукт.

A n n o t a t i o n

During this thesis project a mobile application was developed for those who are planning their trip to other countries. This application is ideal for guests of our home country as well. The “Travel Mate” app contains guides to different countries, a list of upcoming events, sights and tips on how to get to a specific destination, as well as the creation of a user's personal travelling plan. The product was developed using software such as Android Studio version 4.10.3 with the help of Java programming language version 8.

Additional points for consideration were the calculation of the economic efficiency of product design and taking into account the influence of elements of the working environment on employees and ways to prevent accidents.

The conclusion of the implementation of this thesis is a potential, economically sound, also ready-to-use software product.

Содержание

Введение

В двадцать первом веке телефоны стали неотъемлемой частью жизни человека. Смартфон в наше время – это не простой способ для связи и коммуникации, это то, без чего человек не может обойтись в повседневной жизни. Для многих людей мобильные телефоны необходимы для работы, так как они очень функциональны: их можно использовать в качестве фотооаппарата, плеера, для нахождение нужной информации в Интернете, в смартфонах можно хранить различные данные, в независимости от их расширения. Грубо говоря, это небольшии копии компьютера, при этом их можно постоянно носить с собой, что очень удобно.

В наше время путешествие в различные страны или даже внутри своей страны набирают обороты с каждым годом. Путешествия, отдых за границей, далеки командировки стали неотъемлемой частью жизни человека. Данное приложение создано для упрощения жизни людей в качестве туристов, с помощью данного приложения пользователь сможет найти интересные достопримечательности, события, которые происходят в данной стране, а также отмечать на электронной карте, те места, которые он посетил.

Очень много людей путешествуют ради поиска новых эмоций, незабываемых ощущений и так же, чтобы увидеться новые красивые места, завести интересные знакомства. Так как число туристов растет с каждым днем, и эта обширная аудитория привлекает разработчиков мобильных приложений. Практически у каждого пользователя смартфона установлен какой-либо картографический сервис, который тоже можно отнести к туристическим приложениям. Но помимо простого определения метоположения с помощью GPS, необходимы более продвинутые решения, которые предлагают не просто маршрут до какого-то определенного места, но также пользователь сможет увидеть краткое описание, отзывы, фотографии и много другое. Так же нужно заметить, что подобные мобильные приложение подойдут не только для выезда в другие страны, но они идельно могут служить и тем людям, которые не выезжали из своего родного города.

Мобильные приложение в сфере туризма – это очень востребованный рынок ля разработчиков, но на данный момент качественных приложений по данной тематике не так много, как может показаться.

Находясь заграницей, да и сидя в родном городе, многим не хватает такого приложение. Данное приложение значительно упростит жизнь туриста, более того ответит на все его вопросы.

1 Тенденция создания мобильных приложений

В двадцать первом веке смартфоны стали обыденностью жизни человека. Если раньше основная задача мобильных телефонов было – установка связи с другим абонементом, то уже сейчас, смартфоны выполняют множество различных функций.

Уже сейчас мобильные телефоны стали основным устройством для многих разных активностей по всему миру.

Люди предпочитают использовать смартфоны для выхода в Интернет. По данным Google, количество поисковых запросов через мобильные телефоны превышает количество запросов через компьютер и ноутбук. Главные причины данного явление: мобильность, удобство использовать поиск в интернете в любое время и в любом месте. Как передает Ofcom, смартфоны уже заняли место ноутбуков, а также в Великобритании признали смартфоны как «самое важное устройство» для выхода в Интернет. Помимого этого, смартфоны составляют конкуренцию ТВ-индустрии. Все больше видео просматриваются в цифровом варианте, в основном за чем просмотров на различных видах смартфонов.

Подразумевается, что в двадцать первом веке смартфоны вышли на первое место по использованию потребителями.

Исходя из этого стоит отметить ценность мобильных приложений. Рынок мобильных приложений растет изо дня в день.

Сейчас люди проводят в среднем 6-7 часов в день, и вскоре время, проведенное в различных приложениях, будет насчитываться свыше триллиона часов.

Смартфоны являются важной частью повседневной жизни, многие компании прогназируют, что уже к 2021 году общее время, которое люди проводят в приложениях, превысит 3 триллиона часов.

Благодаря частому взаимодействию с различными приложениями, появляется возможность охватить и привлечь новых пользователей и поддержать интерес у уже существующих.

Во многих развитых странах на мобильных телефонах установлено более 70 мобильных приложений. Более того, каждый пользователь ежемесечно открывает и использует как минимум 40 приложений.

На рисунке 1.1 представлен график среднемесячного количества установленных и используемых приложений.

По данным исследования на рынках, средний показатель затрат времени на приложение в сутки выросло на 30% по сравнению с 2017 годом.

Использование на мобильных рынках растет. Уже сейчас поколение, которое работает на стационарных компьютерах попросту угасает.

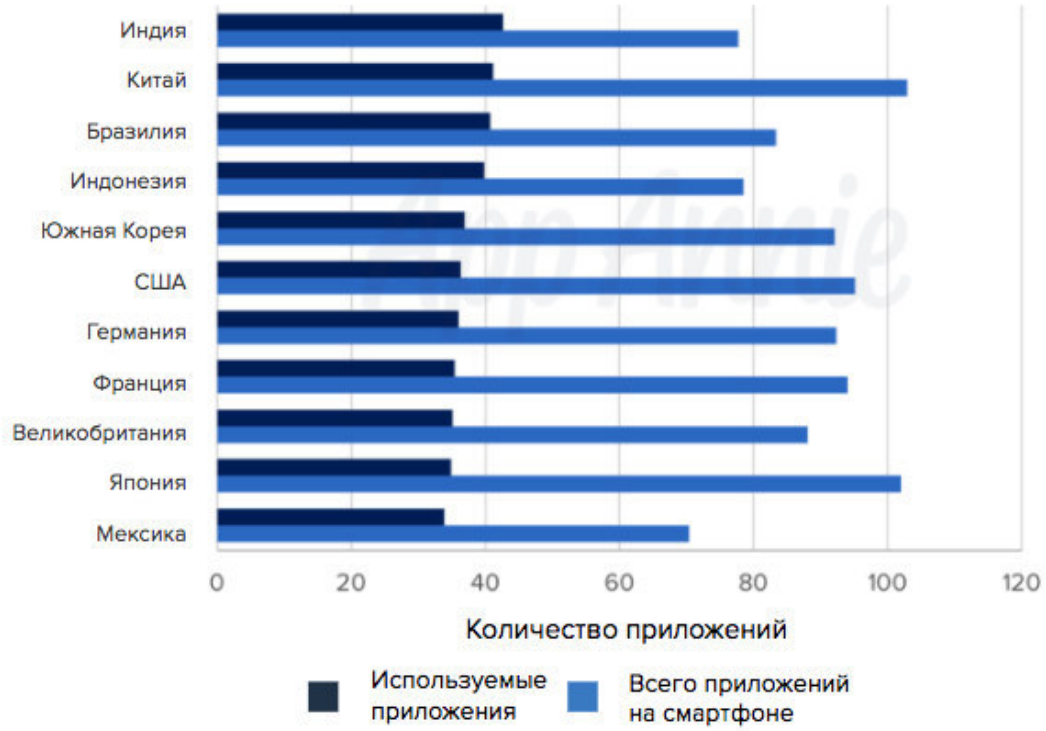

Рисунок 1.1 - Среднемесячное количество приложений

На рисунке 1.2 проиллюстрирован график среднего числа минут, проводимых в приложениях в день на выбранных рынках.

Рисунок 1.2 - График среднего числа минут, проводимых в приложениях

1.2 Назначение программного продукта

Уже сейчаст настал век информационных технологий. Большинство людей по всему миру уже не могу представить себе жизнь без мобильных устройств и приложений. Это так же касается сферы туризма.

Перелеты, путешествия, длительные командировки заграницу стали неотъемлемой частью жизни людей. И само собой разумеющейся, каждому туристу необходимо правильно составить план путешествия, посмотреть и выбрать места, обязательные для посещения. Поиск информации в интернете занимает слишком много времени, что не эффективно, а также неудобно при составлении плана путешествия. Для того, чтобы отлично ориентироваться и чувствовать себя как дома в чужой стране, на чужлом континенте, достаточно использовть свой смартфон.

Данное мобильное приложение отвечает по всем запросам туристов: обзор достопримечательностей, планирование распорядка поездки и кольтурной программы, а так же поик рестаранов или кафе, шоппинг, а так же в мобильном приложение содержится список всех актуальных мероприятий. Все перечисленное значительно упростит задачу, что очень удобно и сэкономит время на поиски.Достаточно просто скачать приложение на мобильный телефон. Забронировав отель, подобрать ресторан или кафе, посмотреть захватывающие места – все это можно сделать с помощью одного приложения в режиме онлайн. Отпадает необходимость искать всю это информацию через множество сайтов.

В начале пользователю предлагается зарегистрироваться, если же у пользователя есть аккаунт, то достаточно просто выполнить вход.

1.3 Обзор рынка мобильных приложений

На сегодняшний день развитие рынка мобильных приложений растет изо дня в день. По последним данным, в 2014 году, исходя из оценок специалистов, его стоимость составила примерно 75 млрд долларов. А в ожидании 2018 года предвещали поднятие до 130 млрд долларов. Напрашивается вывод, из данной усредненной статистики, что мобильных рынок является масштабным.

Доход от внутренних inn-app покупок является основным. А также внутренная реклама в самой среде приложения и сбор масштабных данных тоже приносят значительную чатсь прибыли. Самые используемыми и распространенныями категориями являются: социальные сети, различные рекламные сервисы, полезные приложения, направленные на целевую аудиторию, производительность и не только (рис. 1.3.1).

Целевые мобильные приложения – это такие приложение, которые направлены на определенную группу людей.

Юго-Восточная Азия и Латинская Америка являются наиболее развивающимеся и быстро растущими рынками. В то время как рынок Китая и Индии является, ка кни странно, уже устоявшимся.

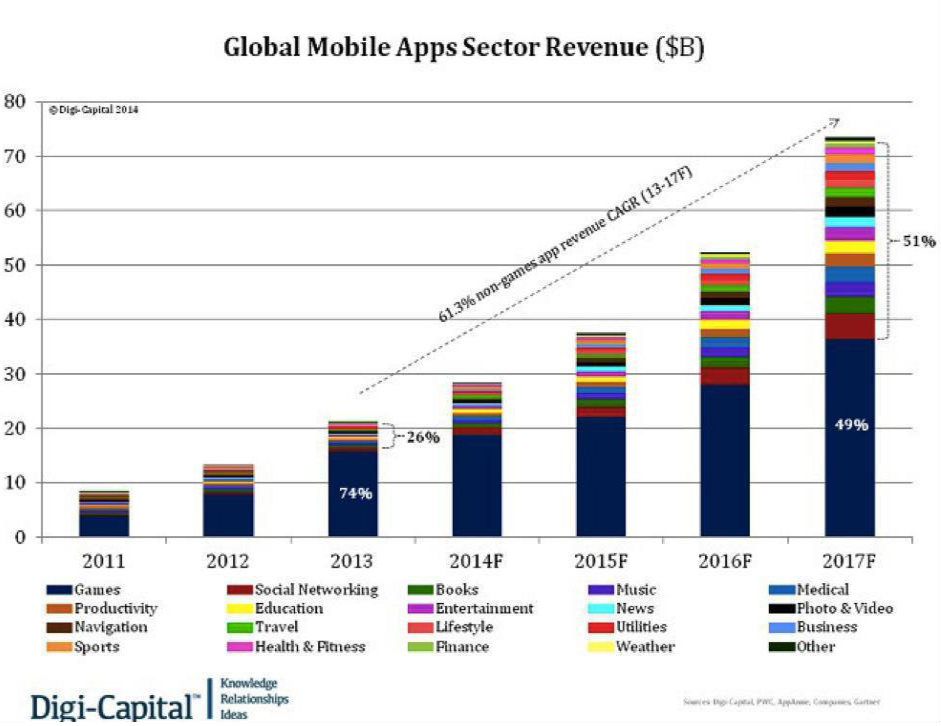

Рисунок 1.3.1 – Статистика роста рынка приложений на смартфон

Существует закономерность, которая проявляется при изучении данной статистики, каждый пятый разработчки, а если упоминать цифры, это примерно 2,1 млн человек во всем мире, разрабатывают мобильные приложения. Исходя из данных, которая представляет компания Apple, стоит добавить, то что в 2016 году во время конференции World Wide Developer Conference было сделано объявление о публикации около 1,45 млн приложение в AppStore, пользователи произвели скачивание 53 млн раз, а доход от реализации для разработчиков принес более 5,5 млрд долларов. В таблице 1.3.1 показан усредненный доход разработчика разных разных операционных систем.

Таблица 1.3.1 Количество задействованных в разработке кадров

Показатель	Компания Google	Компания Apple	Компания Microsoft
Потребителей	1200	1050	34
Мобильных приложений	968	1389	250
Программистов	350	475	96

Продолжение таблицы 1.3.1

Количество определенного показатели	Компания Google	Компания Apple	Компания Microsoft
Число download (загрузок) мобильных приложений (млрд.ра)	68	70	80
Прирост дохода разработки	1450	6160	350
Ср. Уровень дохода программиста	$13,000	$23,000	$4,525

Как показывают прогнозы, число скачивания мобильных приложений из разных категорий достигнет около 300 млрд в год, доходы будут приблизительно равны 98,6 млрд долларов.

По всему миру рыночная стоимость мобильных платежей увеличиться с 386 млрд до 967 млрд долларов к 2019 году. Данный небывалый рост популярности приложений на смартфонах обуславливается значительным увеличение самого предложения и спроса на мобильные устройства, планшеты и так далее.

Рисунок 1.3.2 – Статистика загрузок приложений Android и iOS

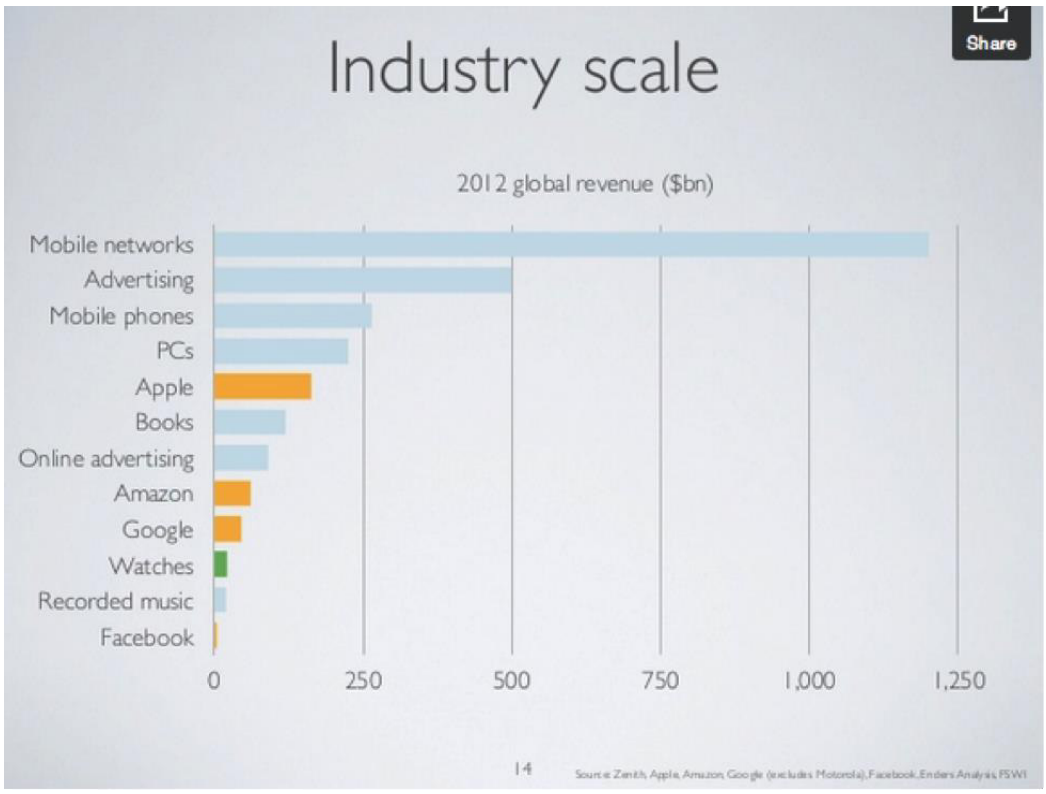

Рисунок 1.3.3 – Статистика по использованю приложений Android и iOS

Согласное диаграммам, которые представлены на рисунке 1.3.2 и 1.3.3 можно заметить рост по всех категориях мобильных приложений. Специалисты, которые исследуют рынок приложений на смартфоны, утверждают, что программные продукты на мобильные устройства преобразуются исходя из целевого маркетингового приложения в средство, предназначенное для растространения и передачи медиа информации. Объем охватывания контактов пользователя, их коэффициент и частота удержания заинтересованности в мобильном продукте производят сильно впечатление. Статистика по Соединенным Штатам Америки, где эксплуатация смартфонов значительно опережает потребление остальных медиа каналов, подверждает, что использование смартфонов составляет 86% от общего времени.

На сегодняшний день большая часть продуктов рынка мобильных приложений относят к продуктам B2C, из-за того, что B2B продукты находятся только на самом начале завхвата рынка. Но уже сейчас перспективы носят внушительный характер. Значительная часть специалистов по исследованию утверждают, что рыно приложений на смартфоны – это из самых многоообещающих digital-сфер.

Целевая а у д и т о р и я мобильных приложений на различные виды смартфонов. Приведем статистику, в Соединенных Штатаъ Америки около 79% пользователей смартфонов применяют их в качестве выхода в интернет. Смартфон это постоянный спутник любого человека, без него многие даже не выходят из дома.

Аналогичная ситуация происходит и в других развитых странах.

И сейчас многие отают предпочтению именно данному удобному девайсу, который может ответить на все запросы аудитории.

При рассмотрении отдельных показателей, которые берут в расчет всех пользователей, основая их часть относится к возрасту от 20-40 лет. Данная целевая аудитория чаще всего находится в браке, проживают в городе или же пригородном районе, имеют свой доход и высшее образование.

Напрашивается вывод, что контингент пользователей состоит в основном из достаточно обеспеченных людей, образованных, получающие доходы выше, чем те, кто мобильные приложения не используют.

Это отличная стратегия компаниям для выдвижения медиа предложения.

Пользователи оценивают, в частности, два критерия – многозадачность и мобильность.

На рисунке 1.3.4 продемонстрированы демограчифеские показатели, которые влияют на использование смартфонов.

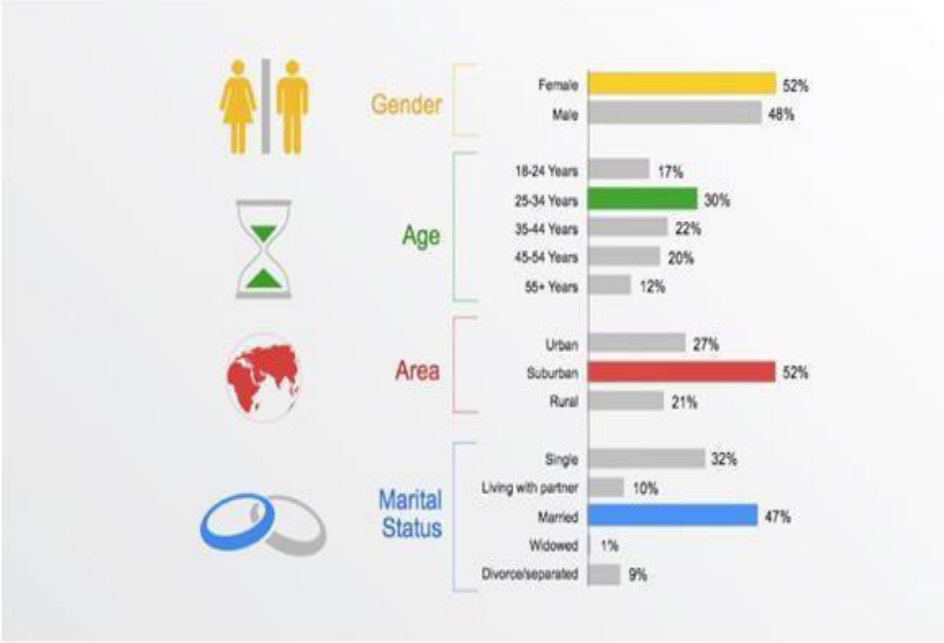

Рисунок 1.3.4 – Демографичская статистика по использованию мобильных устройств

Исходя из демографических показателей был предложен свой список пользователей мобильных предложений от компании Google.

Благодаря использованию приложений на смартфоны, что заметно при розничных продажах, пользовательская лояльность повышается. Уже не раз доказано, что лояльные пользователи имеют количество скачиваний гораздо выше, чем обычные, и их затраты на использование превышает до 3 раз количество времени, проведенное на мобильных версиях сайтов.

На рисунке 1.3.5 приведены социальные факторы, которые влияют на само использование различных смартфонов.

Показатели использования приложений в различных странах мире представлены на рисунке 1.3.6.

Рисунок 1.3.5 – Статистика влияния социальных факторов на использование мобильных устройств

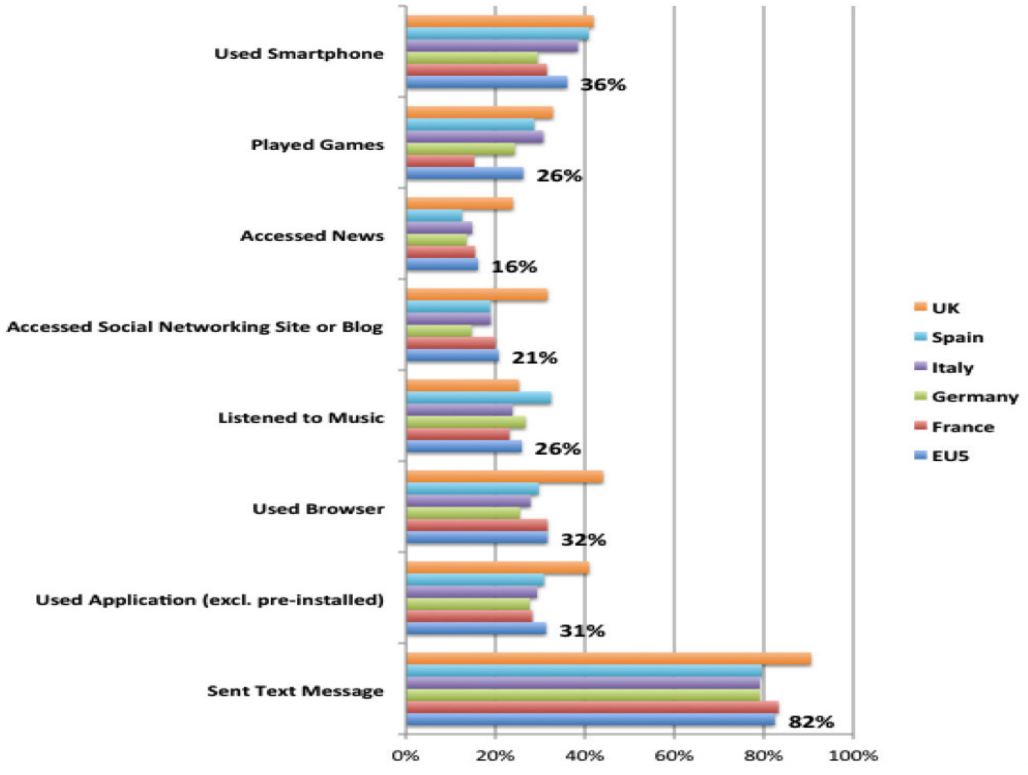

Рисунок 1.3.6 – Показатели использования приложений на смартфон в разных странах

Самая первая версия современного приложения на мобильное устройство была просто обычная оболочкая, только один интерфейс. Но уже в последующих версиях, мобильное приложение стало новым методом систем взаимодействия. Главная цель которой ответить на все запросы пользователя.

Как показывают результаты опроса, свыше ста разработчиков на разных операционных системах, в среднем тратят на разработку и частичную отладку готового программного продукта примерно 18 недель.

Конечно же разработка приложений не так предсказуема, если у разработчика уже есть опыт, но новые функции добавляют с каждым выпуском, требования продолжают изменяться, появляются различные исключения.

Поэтому выбор гибкой модели операционной системы является важным вопросом.

1.4 Анализ рынка устройств и операционных систем

На каждом компьютере установлена операционная система (ОС). Windows, OS X, macOS, Unix и Linux являются традиционными операционными системами. Даже если компьютер - ноутбук - и, следовательно, мобильный - он все еще работает под управлением одной из этих традиционных операционных систем. Однако это различие становится размытым, поскольку возможности планшетов начинают напоминать возможности ноутбуков.

Мобильные операционные системы - это те, которые предназначены специально для питания смартфонов, планшетов и носимых устройств, мобильных устройств, которые мы берем с собой куда угодно. Самыми популярными мобильными операционными системами являются Android и iOS, но другие включают BlackBerry OS, webOS и watchOS.

Обязательный этап перед разработкой собственного мобильное приложения: разработчику слудет проанализировать рынок устройств, и выявить наиболее конкуретноспособную и подходящую ОС.

Статистика продаж мобильных устройств в 2018 году приятно порадовала создателей смартфонов нового поколения.

На сягоднешний день существуют две крупнейшие мобильные операционные системы: ОС Andoroid и iOS. Они продолжают делить рынок (рис.1.4.1). Уже сейчас количество устройств, которые построены на базе ОС Android превышает миллиарда.

Исходя из международных данных Strategy Analytics примерно 67% всех пользователей мобильных устройств используют именно операционную системы Android. По последним данным, за полтора года продажи выросла болеем чем на 33%. В основном данный показатель был укреплен за счет продаж в таких странах как Индия и Китай.

В свою очередь показатели Apple были увеличены примерно 2 раза. Если в 2015 году доля рынка Apple составляла всего 19,1%, то сейчас она составляет 39,05%.

Разработчику обязательно необходимо знать статистику уже существующих приложений при выборе платформ, под которую как раз-таки будет разрабатываться само приложение.

Как позывают графики исследовательской компании Satista от февраля 2018 года, количество уже опубликованных приложений в двух магазинах конкурентов Apple и Google с июня 2009 по февраль 2018 (рисунок 1.4.2 и 1.4.3).

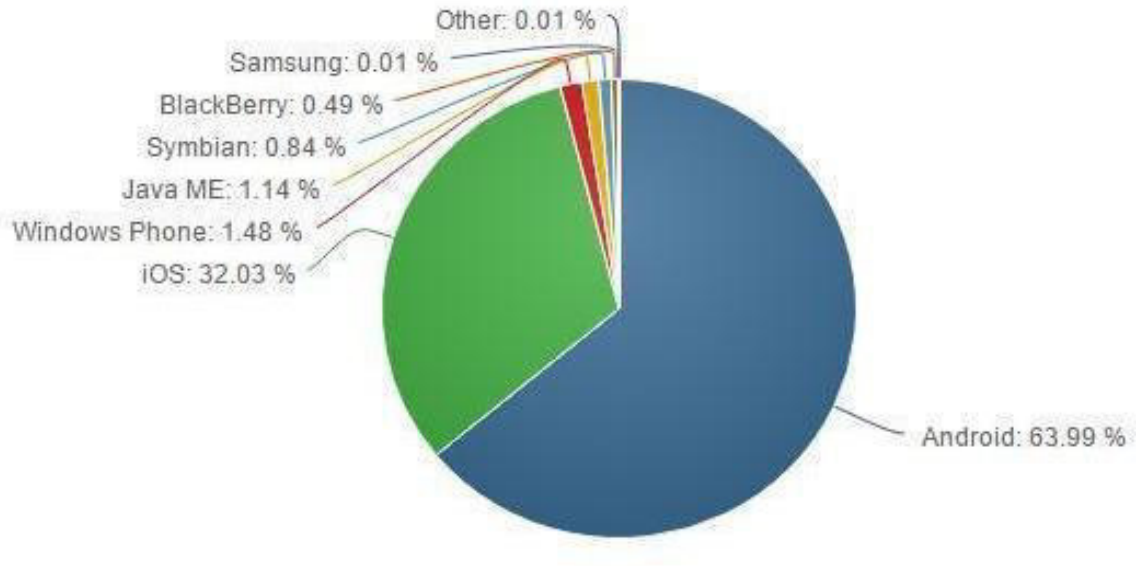

Рисунок 1.4.1 – Мировая доля рынка мобильных ОС

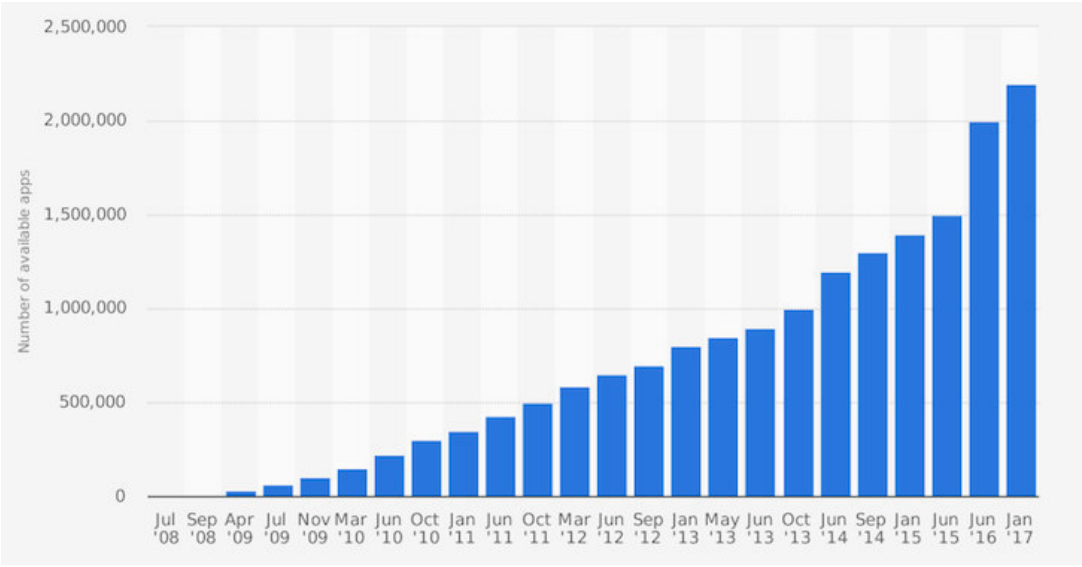

Рисунок 1.4.2 – Количество доступных приложений в AppStore

Исходя из приведенных выше анализов напрашивается вывод, что сейчас, на рынке конкируют в основном только две операционные системы. И это – iOS и Android. По итогу анализа видно, что именно жти две операционные системы следует выбирать под разработку мобильного приложения.
Так как сейчас ОС Android используют не только Samsung, но и такие компании как Huawei, Xiomi, Meizu, которые с каждым годом набирают все больше и больше популярности.

В то время как ОС iOS подходит только для смартфонов от компании Apple.

Поэтому выбор ОС Android является предпочтительнее, так как хватывает большую аудиторию пользователей самых разных возрастов.

У ОС Android есть собственные специальные программные особенности по сравнению с iOS. Так как их очень много, назовем лишь некоторые из них.

Начиная с восьмой платформы от Apple включена поддержка виджетов, но стоит заметить, что их реализация не стоит даже рядом с тем, что предлагает пользователю ОС от Google Android. В Android нет необходимости открывать каждый раз экран блокировки в поисках нужного виджета. В ОС Android их можно просто ставить куда вам удобно на рабочем столе и в это же время получать обновление без каких-либо лишних действий. А также прогноз погоды можно просмотреть просто, взглянув на экран без лишних манипуляций.

В Android и в iOS та же самая многозадачность сконструирована совсем по-разному, и включает свои плюсы и минусы. Но стоит заметить, что Google предлагает намного удобную и привычную для пользователей ПК систему. Именно она предлагает пользователю честную, хоть и затрачиваемую многозадачность.

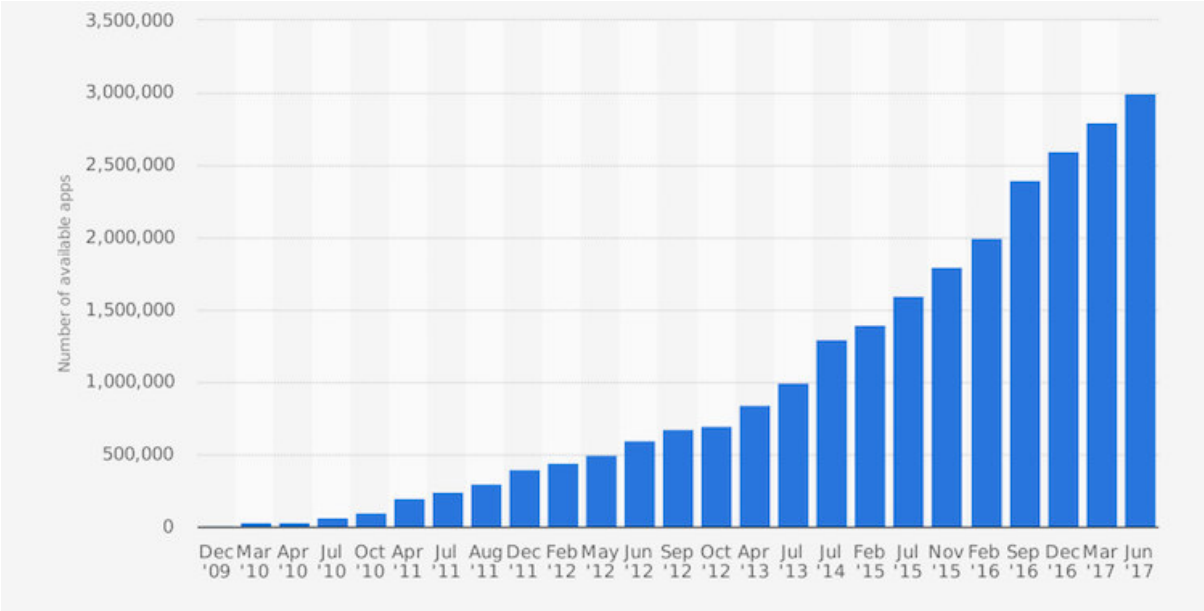

Рисунок 1.4.3 – Количество доступных приложений в Google Play

Гораздо чаще пользователи пользуются настройками Android, чем на IPhone и это общепринятый факт. Пользователи, которые используют ОС Android имеют возможность установить вид начального экрана с помощью различных тем и не только. Многие версии ОС от Google включает несколько вариантов, для того чтобы изменить цвета экранов, тему, вид и другие различные настройки.

Если пользователь хочет больше контроля над своим смартфоном, без необходимости внутруннего вскрытия ОС, то в этом случае, Android является лучшим решением.

На рисунке 1.4.4 приведена статистика пользователей по разным моделям ОС Android на 2018 год.

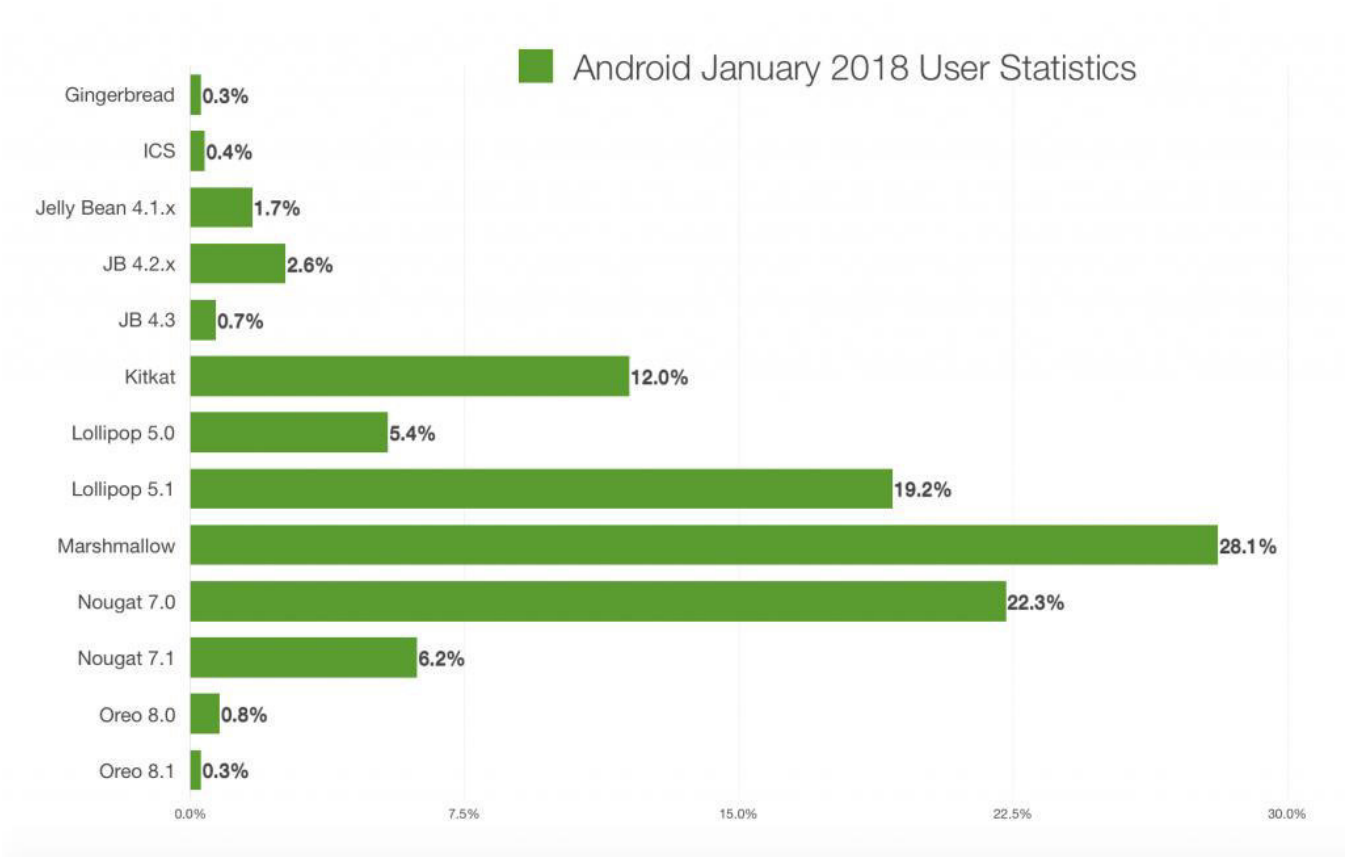

Рисунок 1.4.4 – Статистика пользователей Android

1.5 Обзор существующих аналогичных программных продуктов

В двадцать первом веке был сформирован новый рынок – ранок мобильных приложений.

С каждым днем количество мобильных приложений стремительно растет. Все программные продукты, касающиеся одной сферы и пытающиеся решить одни и те же проблемы, имеют множество сходств, но и множество различий.

По этой причине у одних программным продуктов может быть обширная аудитория заинтересованных пользователей, а другие же продукты, несмотря на схожие функции, могут так и остаться непризнанными.

Так же запрсы пользователей постоянно изменяются. Разработчики мобильных приложений должны подстраиваться под целевую аудиторию, но учитывать и определенные запросы других групп.

При разработке мобильного приложения необходимо учитывать потребности пользователи. Поддержание интереса и создание максимальных удобств для пользователя ставится в приоритете.

Был произведен поиск аналогов данного приложения. В ходе анализа был составлен топ 5 приложений для путешествий. Такой анализ позволит выявить сильные и слабые стороны приложения, какие функции пользуются популярностью и в процессе создания и развития могут быть добавлены.

1.5.1 TripAdvisor

В данном приложение главный акцент идет на сбор отзывов об отелях, ресторанах, достопримечательностях и т.д. Пользователь может просматривать отзывы и фотографии, оставленные другими пользователями, а также добавлять свои.

По мимо этого, в TripAdvisore пользователь может сравнивать цены на авиабилеты, получать ответы на конкретные вопросы о путешествиях на форумах.

Но главная функция приложения – база данных отзывов является и самым главным минусом. Не всегда стоит доверять отзывам.

Сама природа пользователей неоднозначна.

Современные маркетинговые исследования указывают на то, что в большинстве своем люди оставляют отзывы на отели, рестораны и достопримечательности в основном, если их опыт был резко позитивным или резко негативным**.**

Те, кто имел «средний» опыт проживания в каком-нибудь отеле, не будут иметь достаточно мотивации, чтобы об этом рассказать. Поэтому большая часть пользователей на сайтах, подобным TripAdvisor, оценивают место либо как великолепное, либо как отвратительное.

Причем негативные отзывы часто основаны на личных причинах, нежели на объективно плохом обслуживании.

Так же субъективностью отзывов вызывает сомнения.

Нужно понимать, что отзывы для отелей, ресторанов и достопримечательностей пишут совершенно разные люди. Мало кто об этом задумывается, но путешественники из разных стран отличаются совершенно особенным менталитетом.

На рисунке 1.5.1 изображено мобильное приложение TripAdvisor.

1.5.2 Yuggler

Популярное приложение на AppStore для тех, кто путешествует с детьми по Европе и США. Возможность выбрать место для семейного досуга по заданным параметрам — интересы, пол, возраст, удаленность от вашего местопребывания. Найдете и детскую площадку, и музей робототехники.

Плюсы данного приложения: просмотр советов от родителей и пользователей. Позволяет делать закладки, добавлять в избранное и использовать напоминания, чтобы не пропустить самое интересное. Рекомендует варианты развлечений с учетом акций и скидок.

Но также существуют и минусы: направленность YUGGLER очень узка, с помощью данного приложения можно найти развлечение только для детей и различные парки. На этом все. То есть если вы захотите найти интересные достопримечательности, кафе поблизости – в этом случае это приложение вам никак не поможет. Так же стоит учесть GPS порядочно сажает батарею смартфона, а еще нужно быть постоянно в интернете, так что придется купить местную симку.

То есть данное приложение имеет ограничения и удобно только для определенного таргета пользователей.

На рисунке 1.4.2 представлено данное мобильное приложение.

1.5.3 FOURSQUARE

FOURSQUARE - техническим языком, то это социальная сеть с возможностью геопозиционирования.

С помощью данного приложение можно найти подходящие кафе, рестораны и пункты быстрого питания.

Приложение FOURSQUARE имеет приятный, но совсем не понятный дизайн, чтобы полностью понять работу данного приложения понадобится время или же объяснения.

Так же FOURSQUARE является социальной сетью, в которой можно оставлять свои сообщения с привязкой к географическим координатам. Главный минус данного приложения – никакой конфиденциальности. Если же пользователь хочет просто найти на карте интересное кафе, при поиске будут все время выходить различные уведомление, что кто-то когда-то написал об этом кафе. Очень неудобно и как-то даже назойливо. Так же это приложение рассчитано только для мест питания, никаких достопримечательностей, карт, развлечений.

Данное предложение имеет ограниченный спрос, оно не отвечает всем запросам пользователя.

На рисунке 1.4.3 представлено данное мобильное приложение.

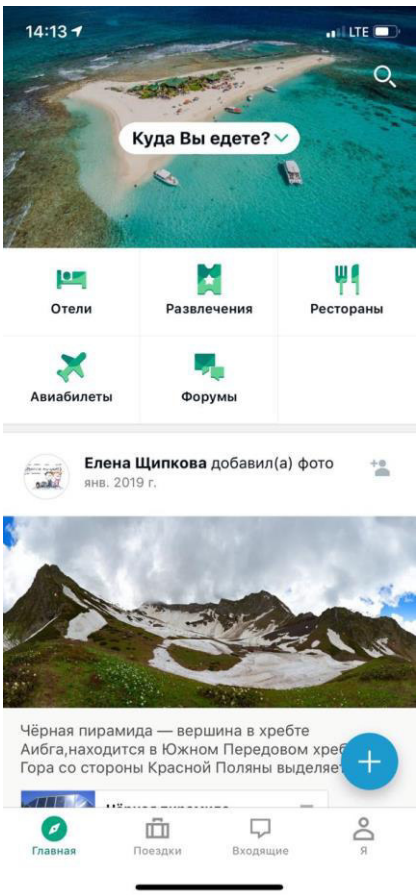

Рисунок 1.5.1 – Приложение TripAdvisor

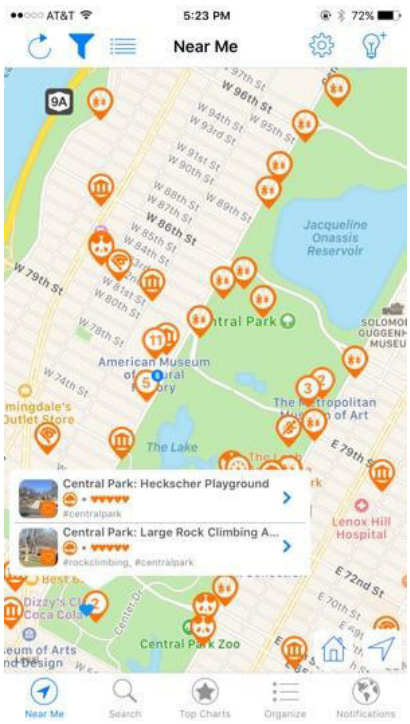

Рисунок 1.5.2 – Приложение Yuggler

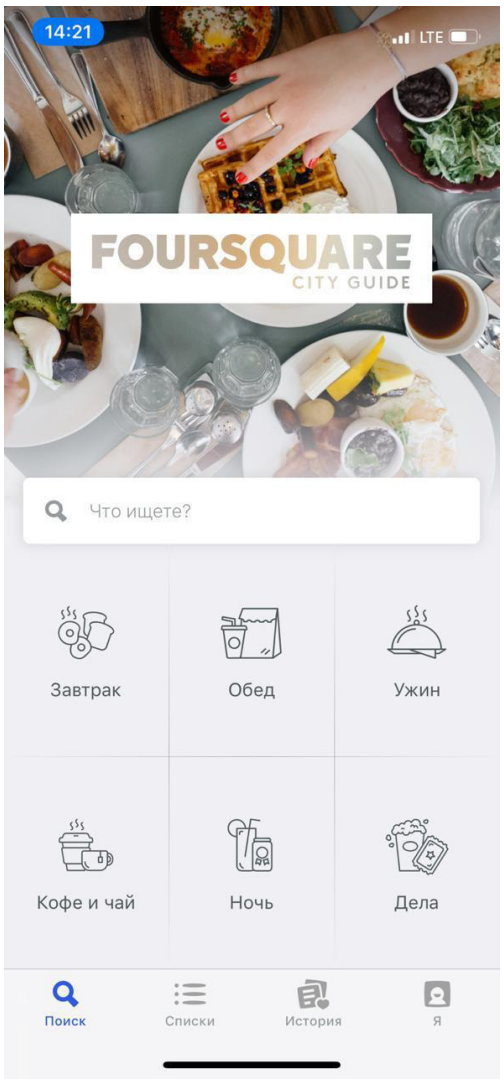

Рисунок 1.4.3 – Приложение Foursquare

1.6 Постановка цели и задач

Проект состоит в том, чтобы исследовать реализацию индивидуального путеводителя по городу на платформе Android. Данное мобильное приложение должен содержать некоторые функции, такие как:

– автоматическая локализация и навигационная поддержка;
– извлечение информации из точек интереса;
– настройка напоминаний;
– добавление отзывов;
– поддержка связи и так далее.

Кроме того, проект должен изучить современные инновационные технологии как можно больше и поставить как строительные блоки, такие технологии как Google Map, Google Calendar и их адаптация в разрабатываемом приложении.

Результатом проекта станет демонстрация прототипа путеводителя по городу.

2 Описание и обоснование выбора средств и технологий разработки

В данном подразделе идет пояснение о том, какие средства и технологии разработки мобильного приложения по теме дипломного проекта были взяты, раскрытие их основных достоинств и недостатков.

2.1 Обзор и сравнение платформ разработки

Для данного анализа были выбраны несколько самых популярных и отличных платформ для создания мобильных приложений. А именно:

– Appery;

Appery - это облачный конструктор мобильных приложений, который можно использовать для создания приложений для Android или iOS. Он включает в себя Apache Cordova (Phone Gap), Ionic и jQuery Mobile с доступом к встроенным компонентам. Поскольку компоновщик работает в облаке, его не нужно устанавливать или загружать, и быстро начать работу. Конструктор приложений Appery включает визуальный редактор, использующий компоненты перетаскивания для создания пользовательского интерфейса. Appery автоматически генерирует код для любых компонентов, которые разработчик может вставлять. Разработчик может подключиться к любому REST API и использовать его в своем приложении, а также мгновенно добавить облачную базу данных и серверную часть в свое приложение, если нужно хранить данные.

– Mobile Roadie;

Mobile Roadie - это создатель приложений, который позволяет любому создавать и управлять своими приложениями для iOS или Android. Более того, здание происходит очень наглядно. Платформа поддерживает все типы мультимедиа с автоматическим импортом ключевых слов RSS, Twitter или Google News и автоматически обновляемой фан-стенкой для общения с пользователями в режиме реального времени. Разработчик может точно просмотреть свое приложение через бэкэнд Mobile Roadie, так же, как пользователи будут смотреть на своих устройствах.

– TheAppBuilder;

TheAppBuilder предоставляет набор приложений для сотрудников, клиентов, мероприятий и брошюр с двумя доступными подходами. Это может быть платформой, если разрабатывается приложение для компании. Существует возможность создать приложение с помощью интерактивного инструментария, и либо предоставленное обучение, либо само TheAppBuilder будут работать самими программистами, чтобы определить и построить структуру приложения и наполнить его исходным содержимым.

- MobioOne;

MobiOne Studio - это инструмент на базе Windows для создания кроссплатформенных мобильных приложений для платформ iOS и Android.

Кроссплатформенные приложения MobiOne построены на основе популярной модели гибридного нативного приложения HTML5 с использованием инфраструктуры Cordova / PhoneGap с открытым исходным кодом.

– Genymotion Android Emulator.

Это один из первых эмуляторов под ОС Android, который включае в себя готовые и настроенные образы Android (x86 с OpenGL).

Среда разработки Eclipse имеет поддержку толького одноуровневого проекта, который требуется для самого приложения, но дополнительным вторым проектом, только же для тестирования, если же вы используете официальную платформу для тестирования Android-модулей.

Модуль AndMoreToole (или же в сокращении ADT) в свою чередь не требует какого-либо внешнего инструмента построения. Таа как он уже содержит все необходимое для создание разных типо приложения для ОС Android. Он содержит только два файла проекта, которые необходимо хранить в системе управления версиями исходного кода. А именно project и classpath.

Так же осуществлен контроль над каталогами файлов: setting, но он сильно изменяется в ходе тестирования и отладки, поэтому он может быть легко проигнорирован. В среде Eclipse существует даже специальный интерфейс прикладного программирования, который предназначен для манипулирования структурой проекта.

Исходя из того, что существуют только два файла, был подведен итог, то что о взломе проекта путем просто редактирования файлов конфигурации не может быть и речи.

Более того, среда разработки Eclipse поддерживает инструмент сборки Maven, которые оснащен модулем M2E (Maven Eclipse) и M2E-Android. НО для разработки любого приложение на базе Eclipse необходимы оба.

2.2 Выбор средств и технологий

2.2.1 Android studio

Для разработки мобильного приложения была в качестве среды разработки была выбрана Android Studio.

Платформа Android studio – специальная интегрированная среда разработки (IDE), которая направлена специально на работу с платформой Android. Анонс данной IDE был проведен 16 мая 2013 года на тематической конференции Google.

Android Studio базируется на программном обеспечении IntelliJIDEA от фирмы JetBrains. Стоит заметить, что Android Studio является официальным средством разработки под Android приложения. Так же данная среда разработки доступна для всех операционных систем: Windows, MacOS и Linux.

Главные особенности Android Studio приведены ниже.

После каждого обновления версией Android Studio появляются различные новые функции. На сегодняшний день доступны и пользуются растущей популярностью следующие функции:

– обширный редактор макетов: WYSIWYG;
– рефактор написанного кода;
– работа с многими UI компонентами с помощью Drag-and-Drop;
– предпросмотр макета на выбранных конфигурациях экрана;
– множество видов сборок и генерация apk файлов;
– ProGuard;
– специальная утилита для подписывания приложений;
– шаблоны основных макетов;
– шаблоны компонентов Android;
– Gradle;
– новеший эмулятор и подключение реального смартфона;
– единая среда, где вы можете разрабатывать для всех устройств Android;
– Instant Run для внесения изменений в ваше работающее приложение без создания нового APK;
– улучшенная версия Java 8.

2.2.2 Gradle

Именно Android Studio использует быстро растущую систему сборки Gradle. Которая основывается на концепциях Apache Ant и Apache Maven, но при этом также вводит Groovy DSL (Domain-Specific Language), предназначающий для сценариев, которые открывают множество возможностей автоматизации, таких как загружая бета-версию. apk в TestFlight для тестирования. Gradle поддерживает инкриметнальные сборки и определяет какие компененты дерева сборки не были изменены, а какие задачи, которые зависят от этих частей, не должны быть перезапущены. Он был специально разработан для расширяемях многопроектных сборок.

2.2.3 Язык программирования – Java

Язык программирования Java является типизированным оъектно-ориентированным языком программирования. Он был разрбаотан компанией Sun Microsystems, уже позднее был приобретен компанией Oracle. И на сегодняшний день, данный проект пренадлежит OpenSource. А также распространяется по специальной лицензии GPL.

В OpenJDK вносят свою лепту большие фирмы, такие как — Oracle, RedHat, IBM, Google, JetBrains. Данные крупные компании разрабатывают собственные сборки JDK. Как заявляет фирма Oracle — отличия меж OpenJDK и OracleJDK буквально отсутствует, за исключением лицензии, отрисовки шрифтов в Swing и кое-каких библиотек, на которые лицензия GPL не

распространяется. Приложения Java, как правило, транслируются в особый байт-код, в следствие чего они работают на любой архитектуре ПК с помощью, конечно, виртуальной машины, которая основана на Java.

Основные плюсы Java:

– простота;

Синтаксис языка берет свое начало и унаследован от языка программирования C++. Данный язык трудно назвать простям языком программирования, особенно на фоне Python и Groove. Однако тогда эфолюциоонай вид позволил привлечь внимение разработчиков языка программирования СИ;

– надёжность;

Надежность Java обеспечивается двумя принципами:

– ООП. Принцип объектно-ориентированного программирования, в частности иерархия наследования, увеличивает процент читаемости кода, а также снижает количество ошибок;

– строгая типизация. Конечно приходится выполнять внушительный объем работы разработчику, но при этом данные интерпретируются в однозначном режиме;

Более того, изначально в Java существовал запрет на прямой доступ к памяти, что тоже повышало уровень надежности. Но конечно, опытные разработчики придумали несколько лазеек, например, можно привести бэкдор sun.misc.Unsafe, который успешно обходит этот запрет.

– безопасность;

В Java была сохранена общая форма конструкций, в особенности по сравнению с C++, но был формально лишен двух потенциальных угроз: множественного наследования и указателей. Конечно на деле обе функции были сохранены, но уже представлены в расстраиваемом языке программирования в другом виде: используются значения вместо указателей, а в множественном наследовании играют важную роль интерфейсы, а не классы. Такая особенность Java практически исключает возможный риск от невнимательных ошибок программиста.

– удобство;

Главная концепция Java использует слоган «Написан раз, запущен везде». Что означает независимость от используемой операционной системы или установленного программного обеспечения. С помощью транслирования в байт-код виртуальной машины это можно достигнуть.

Эта особенность языка Java необходимо как раз-таки для android. Так как смартфоны Android оснащены разнообразием производителей, моделей телефонов, а также разные характеристики – все это может оказать негативное влияние на работу приложений, если бы данного универсрального инструмента

– производительность;

Java имеет ряд особенностей, одна из них как раз-таки связана с трансляцией в байт-код, что сказывается наилучшим образом на

производительность готовых продуктов. Однотипные программы на Java немного уступаются по скорости программам, которые написаны нс C/C++, но превосходят при этом JavaScript, Ruby, Python.

– развитая экосистема.

За все годы существования Java был дополнен десятками IDE и фреймворками, а также сотнями сообщества и различных форумов, множествами библиотек и плагинов.

Каждый язык программирования по-своему уникален. Но при этом каждый имеет свои недостатки и плюсы. Сами особенности Java не носят революционнай характер, но при всем этом, они фундаментальны. Именно это отличает хороший язык программирования.

Малое количество технологий актуальны более 20 лет. Java входит в это число. В 2018 году язык программирования Java занял четвертое место, уступив только: JavaScript, HTML и SQL. По результатам StackOverflow Java занимает 17 место в списке самых популярных технологий. Java как раз-таки тот самый язык программирования, который проверен годами и так же любим многими разработчиками.

Java является основным языком для написания android-приложений. И это не смотря на рост Kotlin. Стоит заметить, что все разработчики, которые пишут на Java, легко могут стать Android-программистами. Хоть и Android Sudio использует SDK вместо JDK, но написание кода осуществляется именно на Java.

2.2.4 Картографический сервис Google

Сервис Google Maps возможно использова как основу для различных сторонних сервисов.

Более того Google создали API (Application programming interface) специально для Google Maps. Главная цель, которой – это привлечение программистов к интеграции картам Google Maps в разрабатываемы мобильные приложения.

Существует отличительная черта для казахстанских и российских программистов, использующие Google Maps API от других служб, спроектированные по той же аналогии. В главное особенность – это, то что Google Maps находит поселения России по их русским названиям.

Под управление операционной системы Android существует бесплатная версия приложения Google Maps для всех смартфонов. В данном приложение содержатся: голосовой поиск, Street view, навигатор и не только.

Что бы использовать данный сервис, нужно зайти на сайт из определенного файла google_maps_api.xml и выбрать необходимый API.

Фрагмент сайта представлен на рисунке 2.2.1.

Следующее действие после выбора необходимого API является потребность включение сервиса и заполнение данных учета, это нужно, чтобы получиться ключ, которых используется в проекте (рисунок 2.2.2).

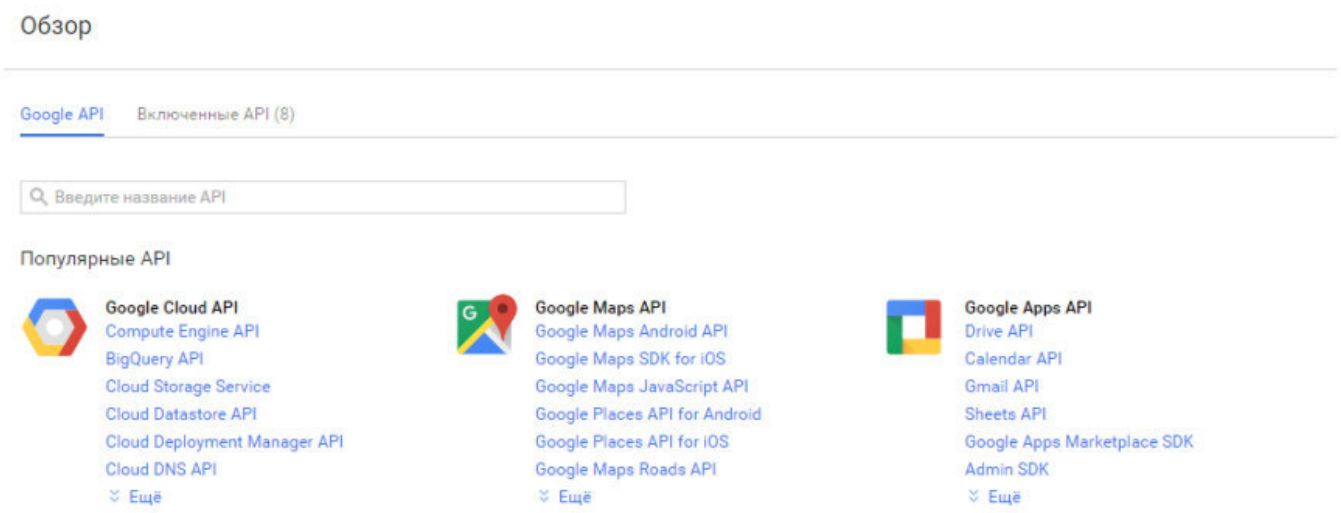

Рисунок 2.2.1 – Сайт console.developers.google.com

После выполнения всех действий необходимо вставить ключ API в тот же используемый файл. Заполнение учетных данных представленно на рисунке 2.2.3.

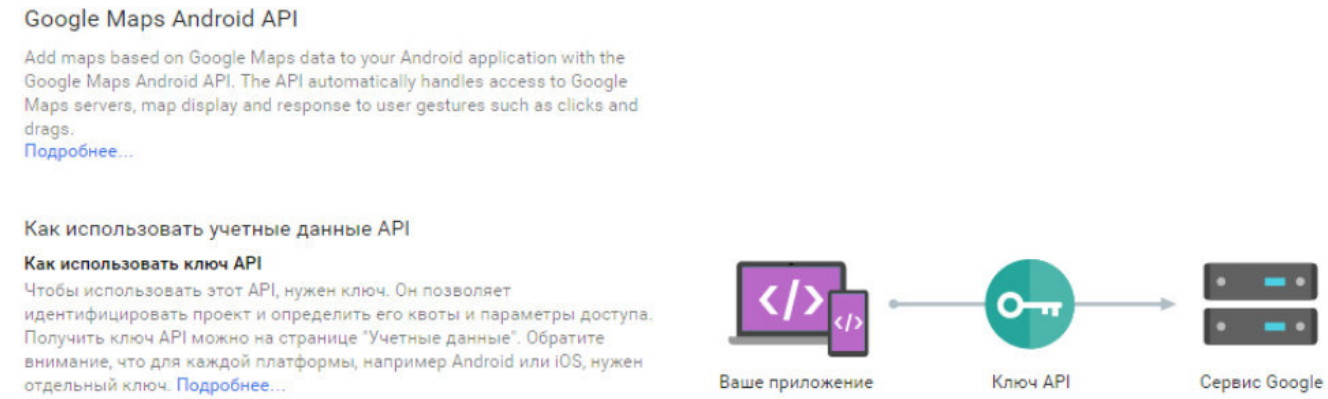

Рисунок 2.2.2 – Подключение Google Maps Android API

Так же гланым преимуществом является то, что компания Google являют официальную поддержку браузеров Google Chrome, Internet Explorer, Mozilla Firefox и Safari во всех операционных системах.

Google поддерживает доступ к мобильной версии сайта для платформ смартфона с различных устройств под управлаением ОС Android, iOs, Symbian, Windows Phone и не только.

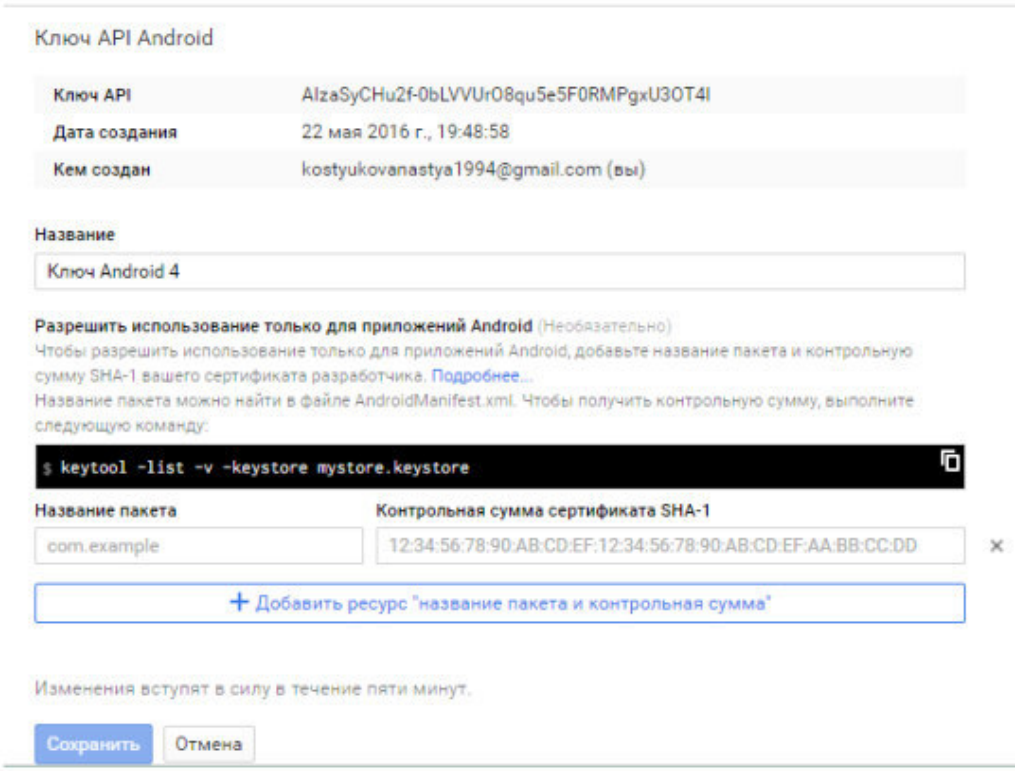

Рисунок 2.2.3 – Заполнение учетных данных

2.2.5 JSON

JSON является текстовым форматом дя обмена данными, который основана на JavaScript. JSON легко читается разработчиками. Данный формат был создан Дугласом Крокфордом. Хоть данный формат происходит от JavaScript, но он считается независимым от языка. JSON может использоваться почти со всеми язаками программрования. Для многих языков подготовлен готовый код, с помощью которого происходит создания и обработка различных данных.

Также есть возможность вставки рабостопособных функций JavaScript.

JSON считается общепринятым форматом для обмена данными приложений по типу клиент-сервер. Он считается универсальным форматом, который обеспечивает обмен данных. Многие приложения содержат систему, в которую должны входить web-клиент, android и клиент. Так как у всех технологий присущ разный язык программирования. Поэтому серверу необходимо посылать ответ в формате JSON, а клиенты в свою очередь приводят ответы к необходимому формату.

Объектом в JSON является неупорядоченным набором пар ключ/значение, который представлен на рисунке 2.2.4.

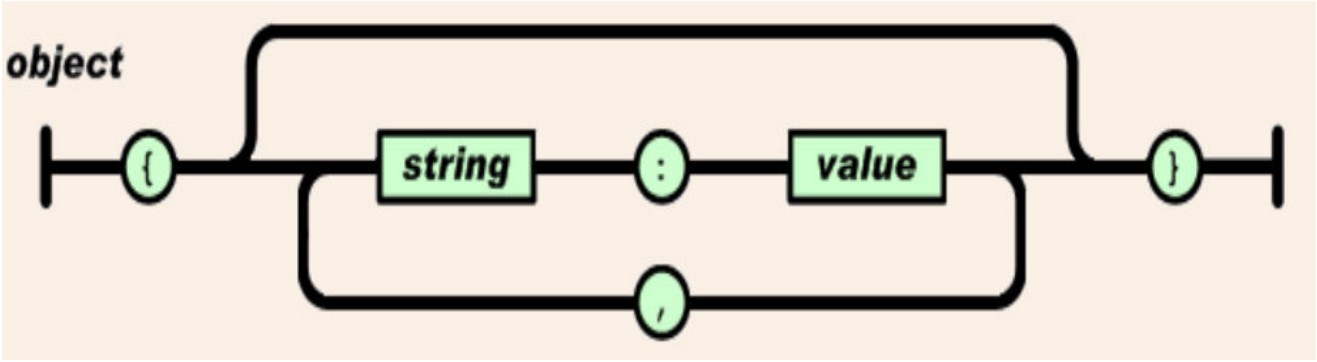

Рисунок 2.2.4 – Структура объекта

Массив же в JSON является упорядоченной коллекцией значений. Структура массива представлена на рисунке 2.2.5.

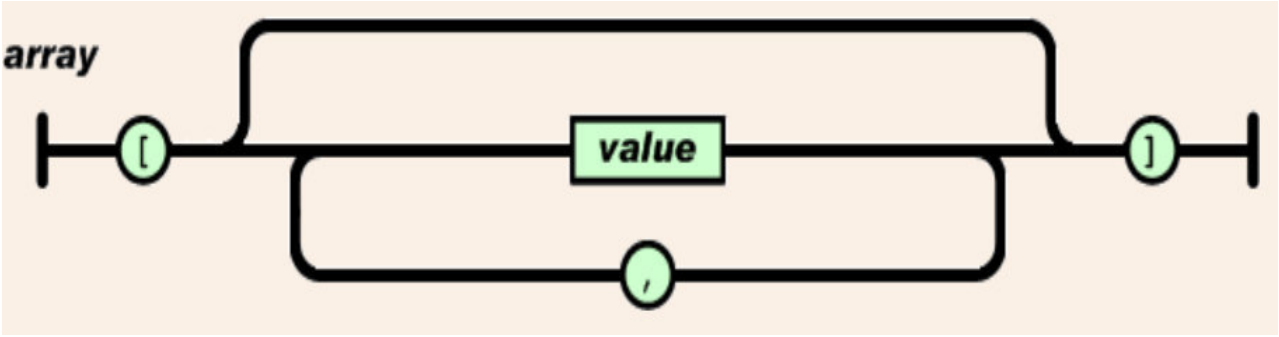

Рисунок 2.2.5 – Структура массива

2.2.6 SQLite

SQLite — система управления базами данных(СУБД).

SQLite была разработана компанией Oracle Corporation.

Большинство разработчиков используют SQLite в качестве сервера, к которому локальные или же удаленные клиенты обязаны обратиться.

Однако библиотека внутреннего сервера входит в сам дистрибутив, который позволяет включить SQLite в многие автономные программы.

СУБД SQLite поддерживает огромное количество типов таблиц, что обеспечивает гибкость данной СУБД. У пользователей есть выбор: таблицы типа MyISAM, которые имееют поддержку полнотекстового поиска, или же таблицы InnoDM, которые поддерживают транзакции на определенном уровне.

Специальный тип таблиц EXAMPLE поставляется вместе с СУБД SQLite, который демонстрирует различные принципы для создания новых типов таблиц. СУБД SQLite имеет открытую архитектуру и GPL-лицензирование, и поставляется для демонстрации различных принципов, которые предназначены для создания новых типов таблиц.

2.3 Описание информационной базы

SQLite - это внутрипроцессная библиотека, которая реализует автономный, безсерверный, транзакционный механизм базы данных SQL.

Файлы базы данных SQLite - это рекомендуемый формат хранения Библиотеки Конгресса США. Думайте о SQLite не как о замене Oracle, а как о замене fopen ().

SQLite - это компактная библиотека.

SQLite обычно работает быстрее, чем больше памяти освобождается. Тем не менее, производительность, как правило, довольно хорошая, даже в условиях нехватки памяти. В зависимости от того, как он используется, SQLite может быть быстрее, чем прямой ввод-вывод файловой системы.Так же SQLite можно использоваться на выделенных машинах с обширными массивами данных (некоторые превышают несколько гигабайтов), так и на встраиваемых системах. Именно SQLite выбрали для реализации хранения данных в мобильных приложениях, оно является стандартным решением, предлагаемым и поддерживаемым ОС Android.

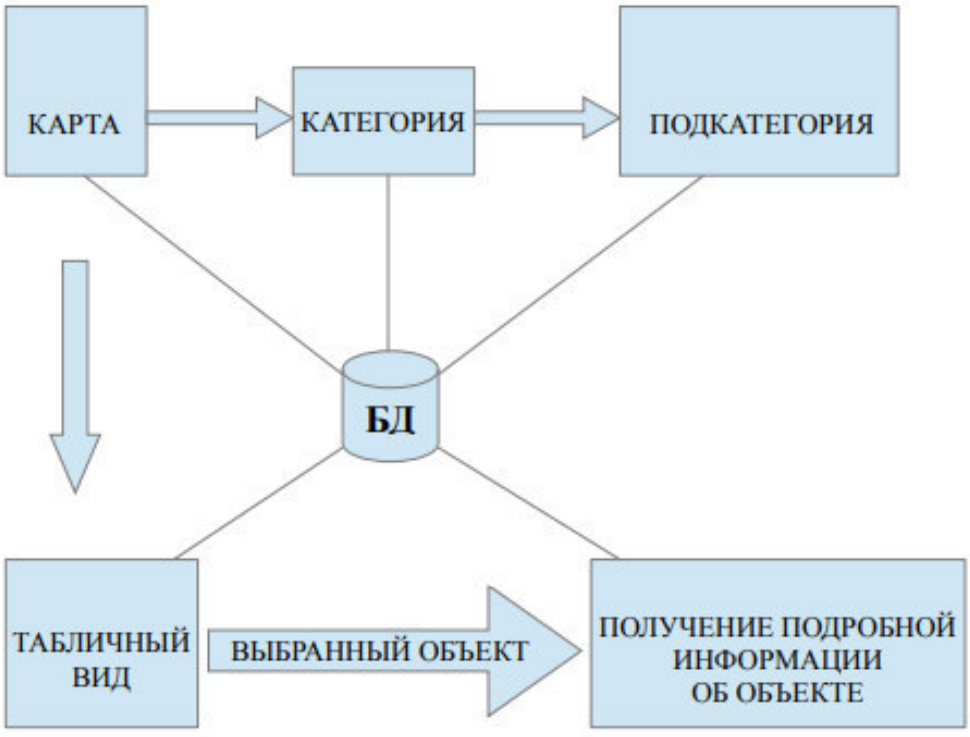

Рисунок 2.3.1 – Схема движения потоков данных

На рисунке 2.3.1 представлена диаграмма потоков данных.

Пользователь выбирает необходимую категорию, в дальнейшем категория обращается напряму к базе данных с помощью SQL-запросов. При этом идет загрузка нужных объектов. Сама же Activity в данном случае играет

роль текстового поля. Для ее заполнения запрашивается запрос к базе данных о выгрузке объектов, уже потом идет заполнение.

В случае когда необходимо получить следующую информацию, то нужно загрузить другую активити, которая содержит нужную информацию, куда и передается его ID с объектами.

Активности связаны между собой, что позволяет легко переходить между ними.

2.3.1 Описание структуры базы данных

На ринсунке 2.3.2 представлена диаграмма, которая демонстрирует связи и поля существующих таблиц.

В информационной базе сущетсвуют 6 таблиц.

В таблице «Город» содержится ключ города, наименование города, континент расположения, история, факты города.

В таблице «Достопримечательности» содержится описание памятных мест, которые есть в городе.

В таблице «Места питания» имеется содержание всех кафе и ресторанов, а также уличных забегалок в выбранном городе.

В таблицу «Отели» идет сбор всех отелей в выбранном городе, описание отеля, а также оценка самого отеля, по мнению пользователей.

В таблице «Магазины» хранится информация и адрес магазинов города.

А также в таблицу «Поездка» пользователь сохраняет выбранные места для орагнизации своего путешествия.

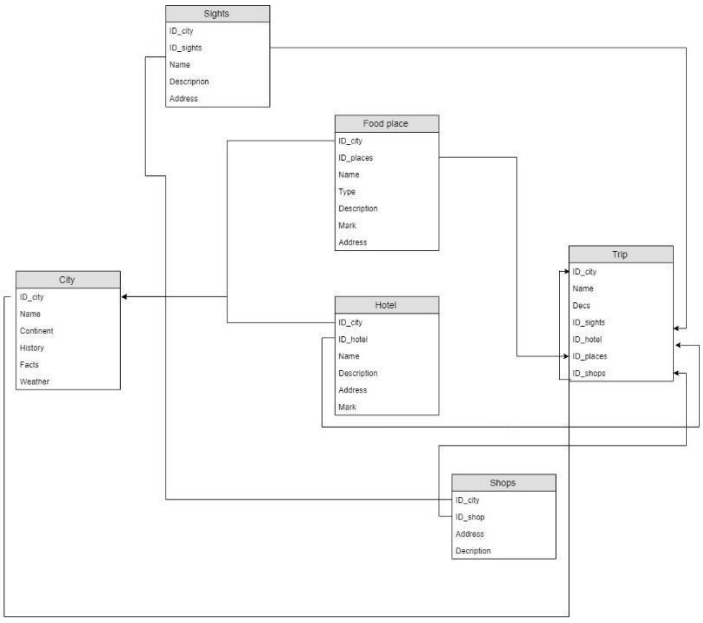

Рисунок 2.3.2 – Диаграмма связей баз данных

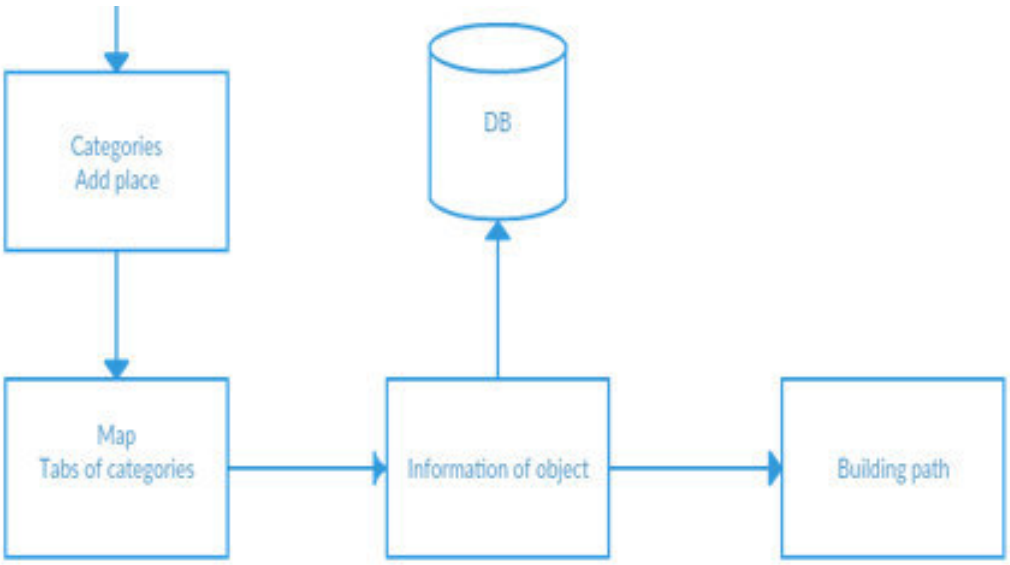

Рисунок 2.3.3 – Схема взаимодействия фрагментов приложения

На рисунке 2.3.3 представлена диаграмма взаимодействия фрагментов разрабатываемого программного продукта. Здесь проиллюстрирована демонстрация выбора объектов из соответствующей категории.

2.4 Разработка UML диаграмм

UML предназначен для поддержания процессов моделирования ПС, который основан на объектно-ориентированном подходе, организации взаимосвязи концептуальных, а также программных понятий. Кроме того, UML отражает проблемы, связанные с масштабированием сложных систем. Все модели, написанные на UML, используют на всех этапах жизненного цикла ПС. Начинают, как правило с бизнем-анализа и заканчивают сопровождением системы.

Абстрактная модель самой системы и подсистем создается при описании работы приложения. Это и есть UML-модель. При написании работы приложения приводят к примеру диаграмму компонентов, то есть диаграмму отдельных функция для разрабатываемого ПО.

Вся основа мобильного приложения состоит из Activity. Они являются основой Android приложения. По этим данным, схема работы разрабатываемого программного продукта имеет вид схемы связей, которая как раз таки осуществляется между различными Activity.

На рисунке 2.4.1 продемострирована UML диаграмма Activity, которая наглядно демонстрирует работу всего разрабатываемого мобильного приложения.

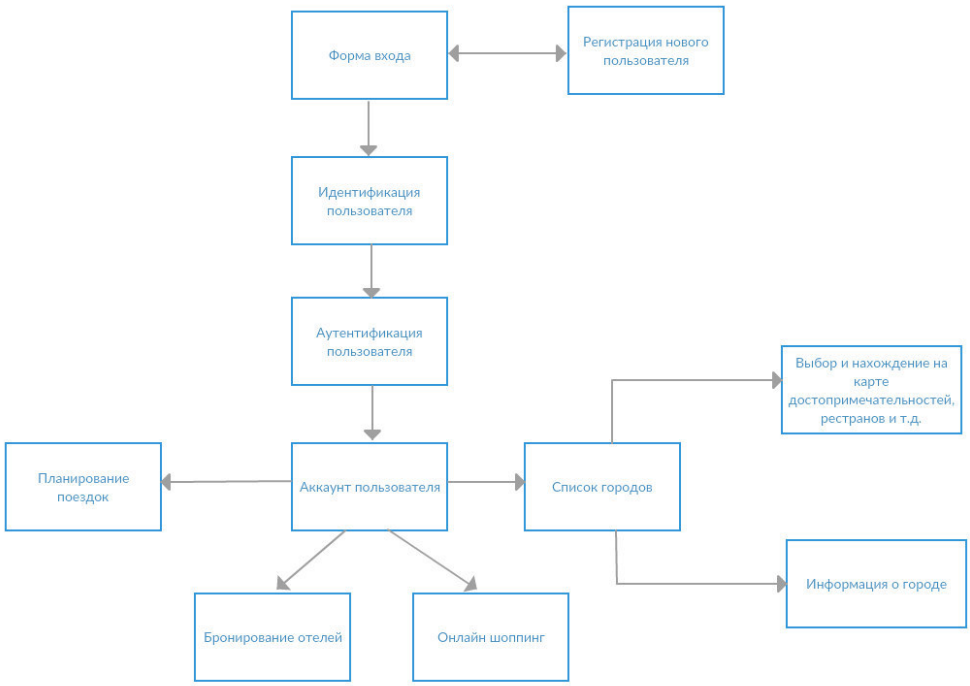

Рисунок 2.4.1 - Диаграмма активностей приложения

2.4.1 Диаграмма классов приложения

При моделировании программных систем наиболее чаще используют диаграмму классов. Диаграмма классов не оотображает динамическое поведение самих объектов, скорее она является одиной из форм статического проектирования. Как раз-таки классы, интерфейсы, связи между ними отображаются в диаграммах классов.

Основным строительным блоком ПС как раз-таки является класс.

Класс – это основной строительный блок ПС.

Каждому классу присуще свое название, атрибусы и операции. Класс на диаграмме изображается в виде примоугольника и разделен на 23 области. В верхней части содержится название данного класса, в средней идет описание атрибутов, в нижней – операции, который выполняет данный объект.

Все мобильные приложения состоят из активностей и фрагментов. Схемой представления мобильных приложений под ОС Android как раз-таки являются активности.

Класс – это экран пользовательного интерфейса.

Отдельная форма приложения это – активити.

Мобильные приложение под Android могут состоять из нескольких активнойстей, которые переключается между собой во время выполнения работы приложения, что наглядно представлено на рисунке 2.4.2.

Каждая активити выполняет свою роль в приложении. В каждой активности мобильного приложения представлены определенные данные, но они так же могут ссылаться друг на друга.

В таблице 2 описаны классы, которые представлены на диаграмме.

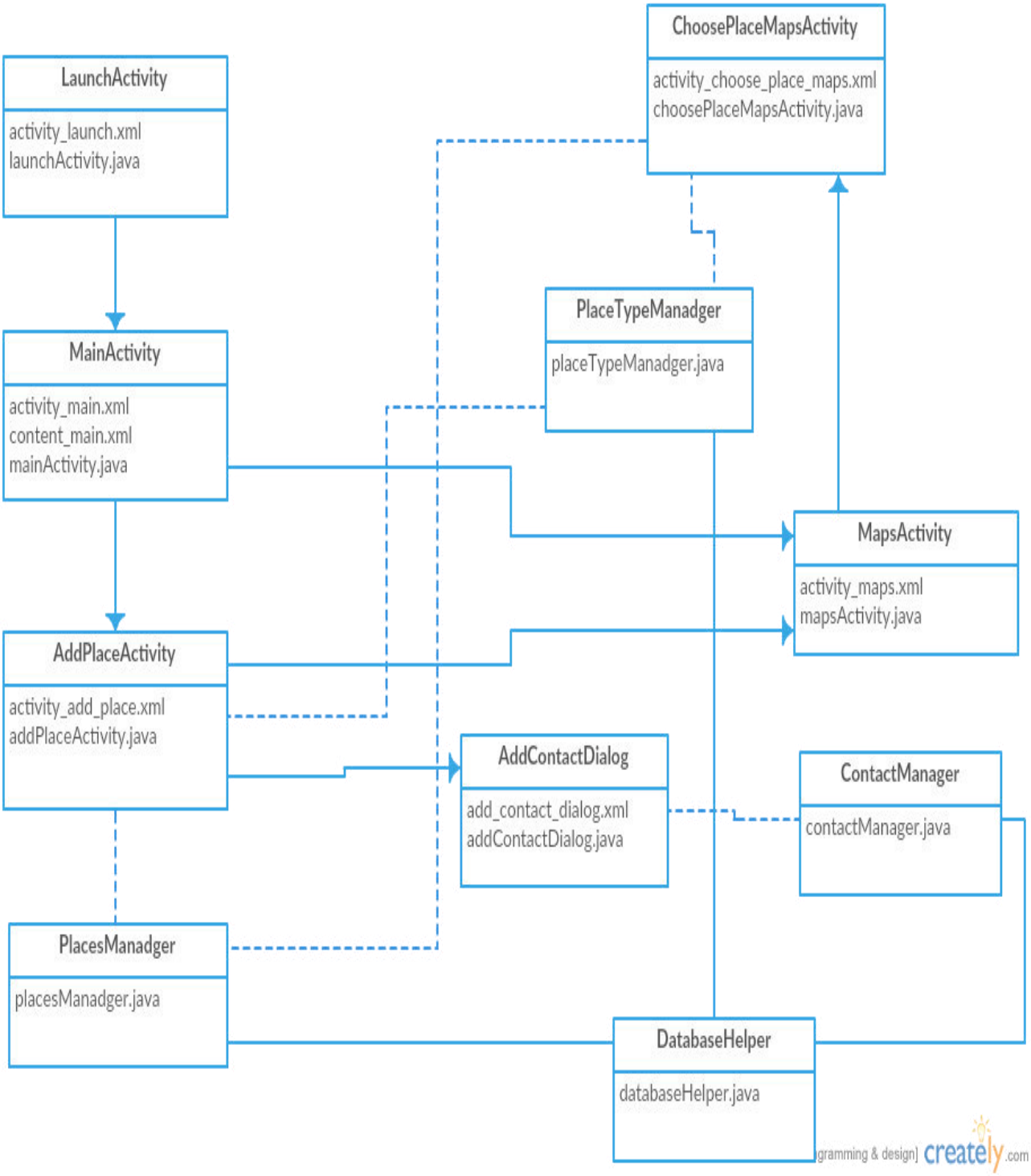

Рисунок 2.4.2 – Связи между активностями и классами приложения

Таблица 2.4.1 – Описание классов

Тип компонента	Название	Описание класса
Пользовательский интерфейс	activity_launch	Представление для отображения самого первого экрана в виде приветствия

Продолжение таблицы 2.4.1

Тип компонента	Название	Описание класса
	content_main	Представление для отображения домашнего экрана
	activity_add_place	Предсталение для отображения активности для добавления объекта на карту
	add_contact_dialog	Представление для отображения окна для добавления информации
	activity_maps	Представление для отображения географической карты
	activity_choose_place_maps	Представление для отображения подкатегорий на карте и информации об объекте
Взаимодействие с пользователем	launchActivity	Класс с функцией для перехода на домашний экран
	mainActivity	Класс с функциями выбора категории и перехода на другую активность для добавления объекта
	addPlaceActivity	Класс с функциями добавления объекта на карту и информации о нем
	choosePlaceMapsActivity	Класс с функциями выбора места, просмотра информации и построение маршрута
	addContactDialog	Класс с функцией добавления контактной информации объекта
	mapsActivity	Класс для работы с картами Google
Взаимодействие с базой данных	contactManager	Класс для работы с контактными данными
	databaseHelper	Абстрактный класс для работы с бд
	placeTypeManager	Класс для работы с данными о типе объекта
	placeManager	Класс для работы с данными об объекте. Режим работы, описание, тип и т.п.

2.4.2 Диаграммы вариантов использования

Описания взаимоотношения и зависимости между несколькими группами вариантов использования, а также действующих лиц, которые принимают участия в процессе.

Отображение проекта и описание внутренних устройств – все это не входит в диагрмму использования. Данная диаграмма предназначена для упрощения взаимодействия с потенциальными пользователями системы, то есть с клиентами и необходимо для определения характеристик системы.

Диаграмма вариантов использования не указывает применяемые в ходе разработки методы, она просто указывает на то, что система должна делать.

С системой в диаграммах вариантов использования взаимодействует актант (внешний объект). Диаграмма вариантов использования для мобильного приложения представлена на рисунке 2.4.3.

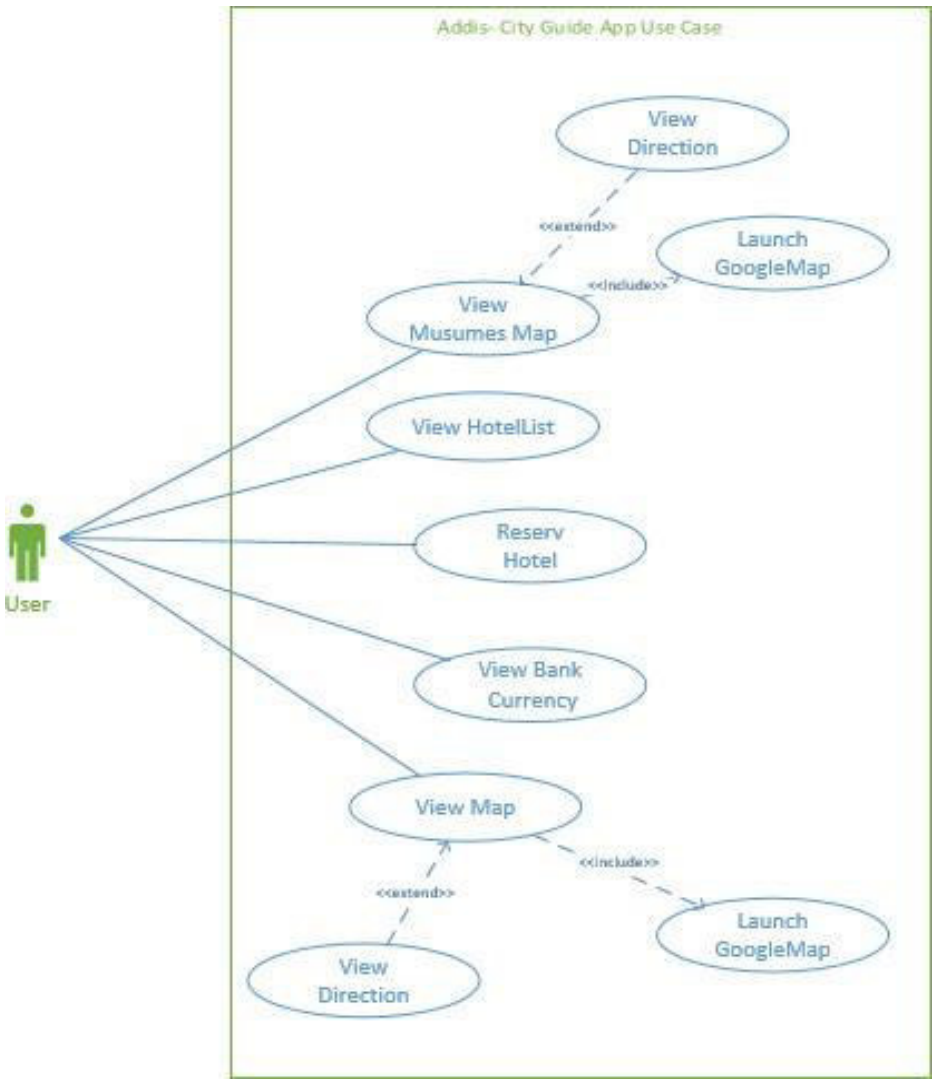

Рисунок 2.4.3 – Диаграмма использования

Из рисунка 2.4.3 видно, что при загрузке приложения пользователю предоставляются следующие возможности:

- выбор категории «Еда»;
- выбор категории «Развлечения»;
- выбор категории «Достопримечательности»;
- выбор категории «Отели»;
- просмотр карты.

3 Проектирование программного продукта

3.1 Структура мобильного приложения

Мобильные приложения под Android бывают разными. Но скелет самих приложений остается прежним.

Абсолютное каждое приложение запускает с своего процесса.

В Android устроены четыре типа компонентов:

– Activities;
– Broadcast receivers;
– Content providers;
– Services.

Кроме того, большую роль в мобильных приложениях Android играют объекты Intetn.

Механизмом для описания одной или нескольких операций, например, выбор фотографии или же видео, отправка письма, переход по ссылке и так далее, называется Intent. Intent необходим для запуска другой Activity.

3.1.1 Activities

Activity в системе управляются как стеки действий. Когда запускается новое Activity, оно обычно помещается на вершину текущего стека и становится текущим Activity. Предыдущее Activity уходит в фоновый режим. На экране может быть один или несколько стеков Activity.

Activity имеет четыре состояния:

- если Activity находится на переднем плане экрана (в самой верхней позиции самого верхнего стека), оно активно или выполняется. Обычно это действие, с которым в данный момент взаимодействует пользователь;
- если Activity потеряло фокус, но все еще представляется пользователю, оно отображается. Это возможно, если новое не полноразмерное или прозрачное действие сфокусировано поверх деятельности, другое занятие имеет более высокую позицию в многооконном режиме или само действие не может быть сфокусировано в текущем оконном режиме. Такая Activity полностью активна (она сохраняет всю информацию о состоянии и членах и остается прикрепленной к оконному менеджеру);
- если Activity полностью скрыто другим действием, оно останавливается или скрывается. Он по-прежнему сохраняет всю информацию о состоянии и членах, однако он больше не виден пользователю, поэтому его окно скрыто, и оно часто будет уничтожено системой, когда потребуется память в другом месте;
- система может отбросить Activity из памяти, либо попросив его завершить, либо просто уничтожив его процесс, сделав его уничтоженным. На диаграмме, которая изображена на рисунке 2.1 показаны важные пути состояний Activity.

Квадратные прямоугольники представляют методы обратного вызова, которые можно реализовать для выполнения операций, когда действие перемещается между состояниями. Цветные овалы - это основные состояния, в которых может быть активность.

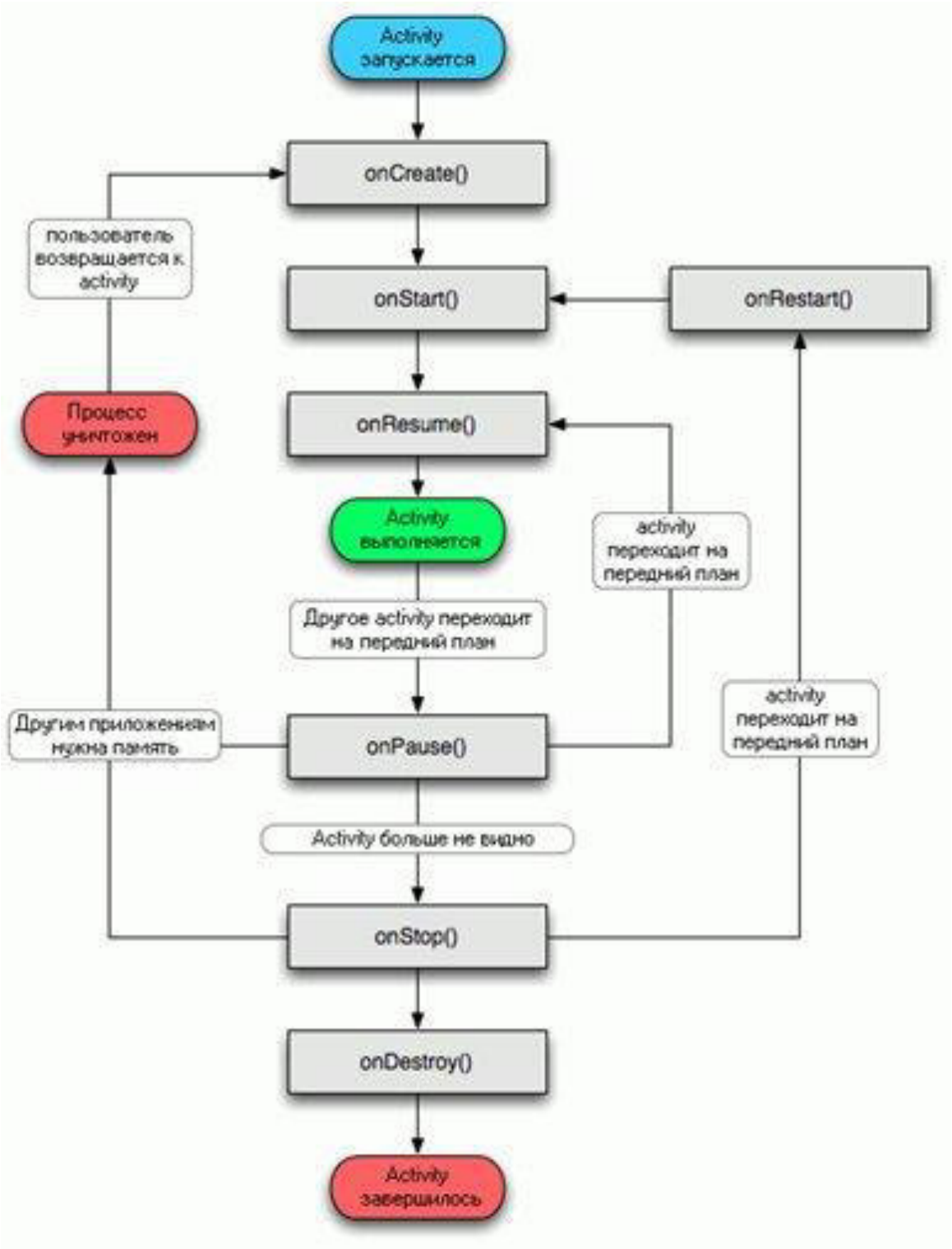

Рисунок 2.1 - Жизненный цикл Activity

3.1.2 Типы процессов в мобильных Android-приложениях

Важно, чтобы разработчики приложений понимали, как различные компоненты приложения (в частности, Activity, Service и BroadcastReceiver) влияют на время жизни процесса приложения.

Неправильное использование этих компонентов может привести к тому, что система погубит процесс приложения во время выполнения важной работы.

- процесс переднего плана - тот, который требуется для того, что пользователь делает в настоящее время. Различные компоненты приложения могут по-разному рассматривать содержащий его процесс на переднем плане;
- видимый процесс выполняет работу, о которой в данный момент знает пользователь, поэтому его уничтожение окажет заметное негативное влияние на взаимодействие с пользователем;
- Сервисный процесс - это тот, который содержит Сервис, запущенный методом startService (). Хотя эти процессы не видны непосредственно пользователю, они обычно делают вещи, которые интересуют пользователя (такие как фоновая загрузка или выгрузка сетевых данных), поэтому система всегда будет поддерживать такие процессы, если не будет достаточно памяти для сохранения всех передний план и видимые процессы.
- Кэшируемый процесс - это тот, который в данный момент не нужен, поэтому система может убить его по своему усмотрению, когда требуется память в другом месте. В нормально функционирующей системе это единственные процессы, связанные с управлением памятью: хорошо работающая система будет всегда иметь в наличии несколько кэшированных процессов (для более эффективного переключения между приложениями) и регулярно убивать самые старые из них по мере необходимости. Только в очень критических (и нежелательных) ситуациях система может достичь точки, в которой все кэшированные процессы будут уничтожены, и она должна начать уничтожение сервисных процессов.

Приоритет процесса также может быть увеличен на основе других зависимостей, которые имеет процесс.

Например, если процесс A связан с Сервисом с флагом Context.BIND_AUTO_CREATE или использует ContentProvider в процессе B, то классификация процесса B всегда будет по крайней мере так же важна, как и процесс A.

3.1.3 Services

Сервис - это компонент приложения, который может выполнять длительные операции в фоновом режиме и не предоставляет пользовательский интерфейс. Другой компонент может работать в режиме фона.

Более того, определенный компонент связывается с Service, взаимодействует с ним, выполняет определенные процессы между Service.

Кроме то проводит транзакции, воспроизводит прослушивание музыки, выполняет ввод-вывод из файла. Все это происходит из фонового состояния.

Кроме того, компонент может связываться со службой, взаимодействовать с ней и даже выполнять межпроцессное взаимодействие (IPC). Например, сервис может обрабатывать сетевые транзакции, воспроизводить музыку, выполнять файловый ввод-вывод или взаимодействовать с поставщиком контента, все из фонового состояния.

Это три разных типа услуг:

- передний план;

Служба переднего плана выполняет некоторые операции, которые заметны для пользователя. Например, аудио приложение будет использовать сервис переднего плана для воспроизведения звуковой дорожки. Службы Foreground должны отображать Уведомление. Службы Foreground продолжают работать, даже когда пользователь не взаимодействует с приложением.

- фон;

Фоновая служба выполняет операцию, которая не замечена пользователем напрямую. Например, если приложение использует сервис для сжатия своего хранилища, это обычно будет фоновый сервис.

- граница.

Привязка идет только тогда и только тогда, когда приложение само связывается с ним используя метод bindService (). Далее предлагается интерфейс по типу клиент-сервер, который взаимодействует с Service. Но если к нему привязывает уже другой компонент, Service прекращает свою работу. Одновременно могут связывать один или несколько компонентов, но если они связываются и включаются все, то Service уничтожается.

3.1.4 Broadcast receivers

Некоторые вещания прослушиваются и регистрируются.

Broadcast receivers проходят реализацию путем extand класса Android и уже потом идет переопределение особенного метода onReceive ().

Вещательные приемники реализуются путем расширения класса Android BroadcastReceiver и переопределения метода onReceive ().

Это достигается путем ссылки на строку действия намерения вещания. Когда обнаружена соответствующая широковещательная передача, вызывается метод onReceive () получателя широковещательной передачи, после чего у метода есть 5 секунд для выполнения любых необходимых задач перед возвратом.

Важно отметить, что приемник вещания не обязательно должен быть запущен все время.

В случае обнаружения соответствующего намерения система времени выполнения Android автоматически запускает широковещательный приемник перед вызовом метода onReceive ().

Компонент Broadcast Receivers получает и реагирует на трансляцию объявления. Например, когда низкий уровень заряда батареи низкий, информация необходима для информировать пользователей. Все приемники расширяют базовый класс BroadcastReceiver.

3.2 Характерные черты дизайна мобильных Android приложений

Material Design стал детищем Google в середине 2014 года под кодовым названием «Quantum Paper» и представляет собой новый подход «чернилами и пером».

С помощью Material Design цель состоит в том, чтобы обеспечить высококачественный результат на всех платформах, предоставляя пользователям контроль над четко обозначенными, приятными на вид компонентами, которые ведут себя как объекты реального мира.

В отличие от изображения предметов, имеющих отношение к культуре (например, корзин для мусора), в скейоморфизме, Material Design включает в себя применение основных естественных законов из физического мира, главным образом касающихся освещения и движения.

Идея состоит в том, что, имитируя физический мир, мы снижаем когнитивные нагрузки пользователей благодаря внимательному отношению к макету, визуальному языку и библиотеке шаблонов, максимизируя предсказуемость и устраняя двусмысленность.

Концепция «дизайна» Material Design служит системой для наложения элементов и анимации.

Это также позволяет более персонализированный опыт. В качестве примера была приведена функция показа последователей в Твиттер (рисунок 3.2.1).

Для Material Design очень важно соответствовать ожиданиям пользователей относительно того, как компоненты должны вести себя.

Например, объекты на экране более правдоподобны, если они следуют законам.

Эффективное применение материалов требует понимания следующих принципов:

– метафора материала;
– принимая понятие тактильной реальности.

Специальный дизайн, чтобы предложить пользователям атрибуты, которые они находят знакомыми, как материальные средства сразу узнаваемые функции реального мира, такие как кнопки, должны мгновенно позволить пользователям увидеть, что делать. Поверхности и швы / края должны дополнять их, предлагая подсказки. Используйте реалистичный свет / затенение, чтобы разделить пространство дизайна и показать движущиеся части.

Рисунок 3.2.1 – Функция показа последователей в Твиттер

Используя адаптивный дизайн – разработчик убеждается, что иерархия, цвета, значки и пространственные пропорции одинаковы для всех устройств благодаря адаптивному выбору версии.

Смелые, графические и удачно подобранные элементы придают смысл и радуют глаз.

Превосходное использование цветов и пробелов, четкие изображения и большая типография по всему экрану помогают пользователям погрузиться в процесс.

Критически важно сочетать смелый, осознанный выбор дизайна с тонкостью в отношении их естественной реакции на вводимые пользователем данные. Например, если пользователь выбрасывает выбранный элемент в группу объектов, последний должен отодвинуться, приспосабливая его. В Material Design разработчики стараются не позволять объектам совершать волшебные вещи, такие как исчезновение или перемещение самостоятельно. Вместо этого программисты переводят знакомство ограниченного физического мира на экраны, которые имеют неограниченный потенциал.

Бесконечная прокрутка YouTube иллюстрирует это.

В мае 2018 года Google выпустила пересмотренную версию, чтобы исправить серьезную проблему - первоначальные рекомендации носили ограничительный характер, подчеркивая функциональность над стилем. Поскольку приложения, созданные в соответствии с ними, выглядели одинаково, многие производители приложений не любили Material Design. Google должен был сбалансировать согласованность Material Design с возможностью дифференциации, чтобы предоставить дизайнерам возможность адаптировать его к потребностям бренда. Версия 2 содержит не только новые

рекомендации, но и набор инструментов (в том числе новые пакеты значков и редактор тем материала), которые можно использовать для настройки дизайна.

Таким образом, разработчик мобильных приложений может отрегулировать эстетику в соответствии с присутствием бренда в какой-либо организации, опираясь на вечные природные законы.

Пример Material Design представлен на рисунке 3.2.2.

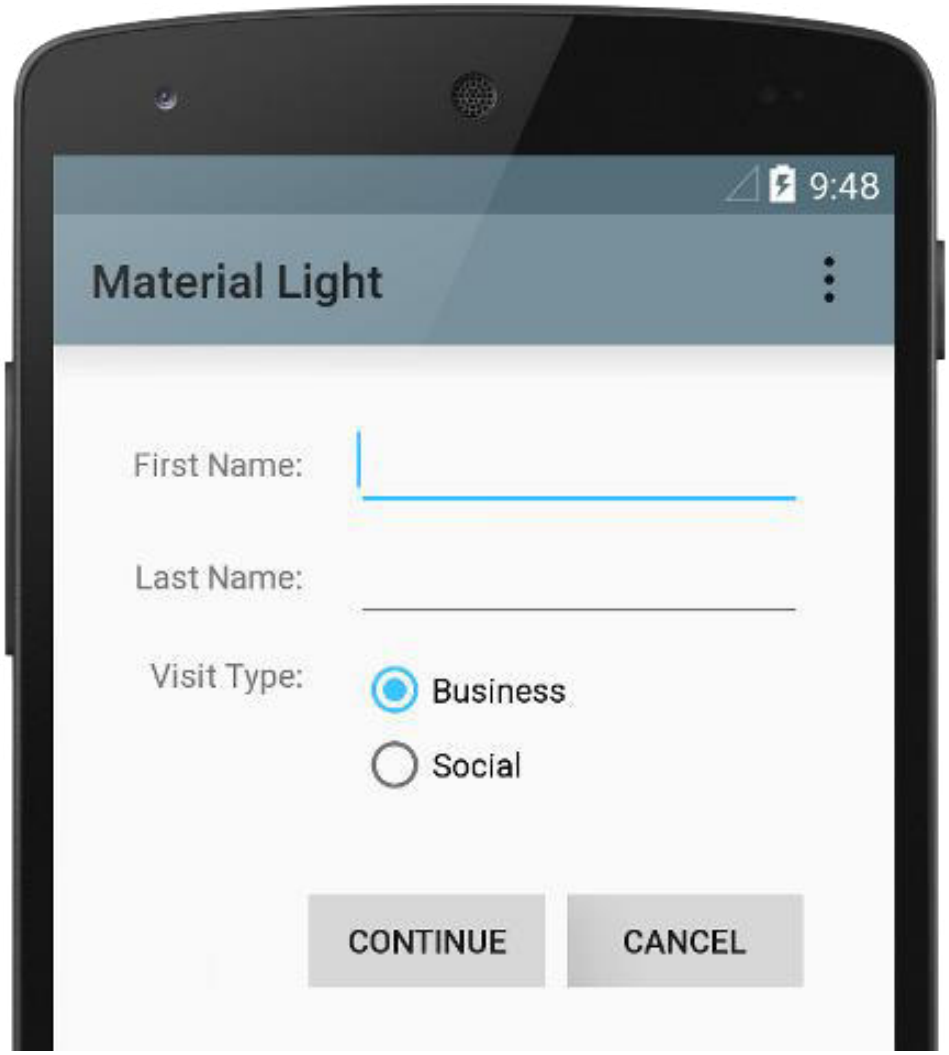

Рисунок 3.2.2 – Пример Material Design

3.3 Результат реализации программного продукта

Результатом разработки программного продукта является готовое мобильное приложение на базе ОС Android. Функции мобильного приложения были выполнены согласно поставленным целям и задачам. На рисунках нижу продемонстрирован готовый программный продукт.

При первоначальной установке приложения пользователю необходимо войти в аккаунт (рисунок 3.3.1). Если же пользователь использует данное приложение впервые, то необходимо зарегистрироваться (рисунок 3.3.2).

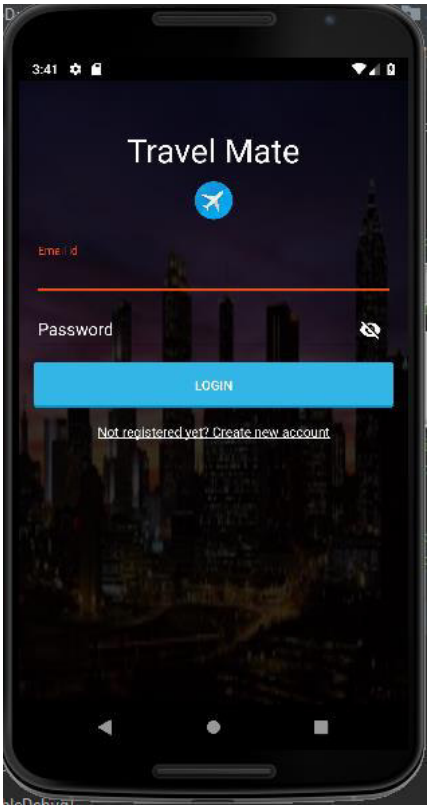

Рисунок 3.3.1 – Авторизация пользователя

Ниже представлен фрагмент кода для авторизации пользователя:

```
@Override
public void rememberUserInfo(String token, String email) {
   SharedPreferences.Editor editor = mSharedPreferences.edit();
   editor.putString(USER_TOKEN, token);
   editor.putString(USER_EMAIL, email);
   editor.apply();}
@Override
public void checkUserSession() {
   if (mSharedPreferences.getString(USER_TOKEN, null) != null) {
 Intent intent = MainActivity.getStartIntent(LoginActivity.this);
      startActivity(intent);
      finish();    }}
                    if (responseCode == HttpsURLConnection.HTTP_CREATED
&& res.equals(successfulMessage)) {
                  mView.openLogin();
                  mView.setLoginEmail(email);
                  mView.showMessage("signup succeeded! please login");
               } else {
```

```
                    mView.showMessage(res);
                }
                mView.dismissLoadingDialog();
            } catch (Exception e) {
                e.printStackTrace();
                mView.showError()}     }); } });}
        @Override
public void checkUserSession() {
   if (mSharedPreferences.getString(USER_TOKEN, null) != null) {
 Intent intent = MainActivity.getStartIntent(LoginActivity.this);
       startActivity(intent);
       finish();    }}
@Override
public void openLogin() {
   log.setVisibility(View.VISIBLE);
   sig.setVisibility(View.GONE);}
public void setLoginEmail(String email) {
   email_login.setText(email);}
public void showMessage(String message) {
   Snackbar snackbar = Snackbar
          .make(findViewById(android.R.id.content), message,
```

Метод onRepsonse для регистрации нового пользователя:

```
          @Override
          res = Objects.requireNonNull(response.body()).string();
          final int responseCode = response.code();
          mhandler.post(() -> {
             try {
                String successfulMessage = "\"Successfully registered\"";
                if (responseCode == HttpsURLConnection.HTTP_CREATED &&
res.equals(successfulMessage)) {
                    mView.openLogin();
                    mView.setLoginEmail(email);
                    mView.showMessage("signup succeeded! please login");
                } else  mView.
        showMessage(res);}
```

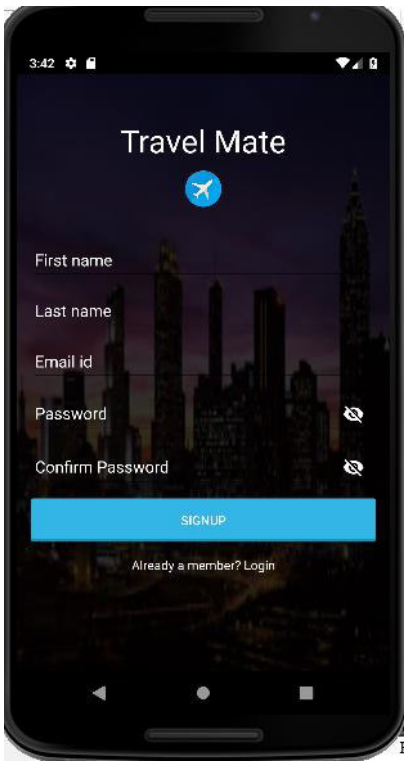

Рисунок 3.3.2 – Окно регистрации пользователя

Рисунок 3.3.3 – Список городов

На рисунке представлен список городов. При нажатии на выбранный город идет переход не более подробную информацию о выбранном городе:

погода в данный момент, история города, рестораны, достопримечательнсоти, шоппинг центры (рисунок 3.3.3).

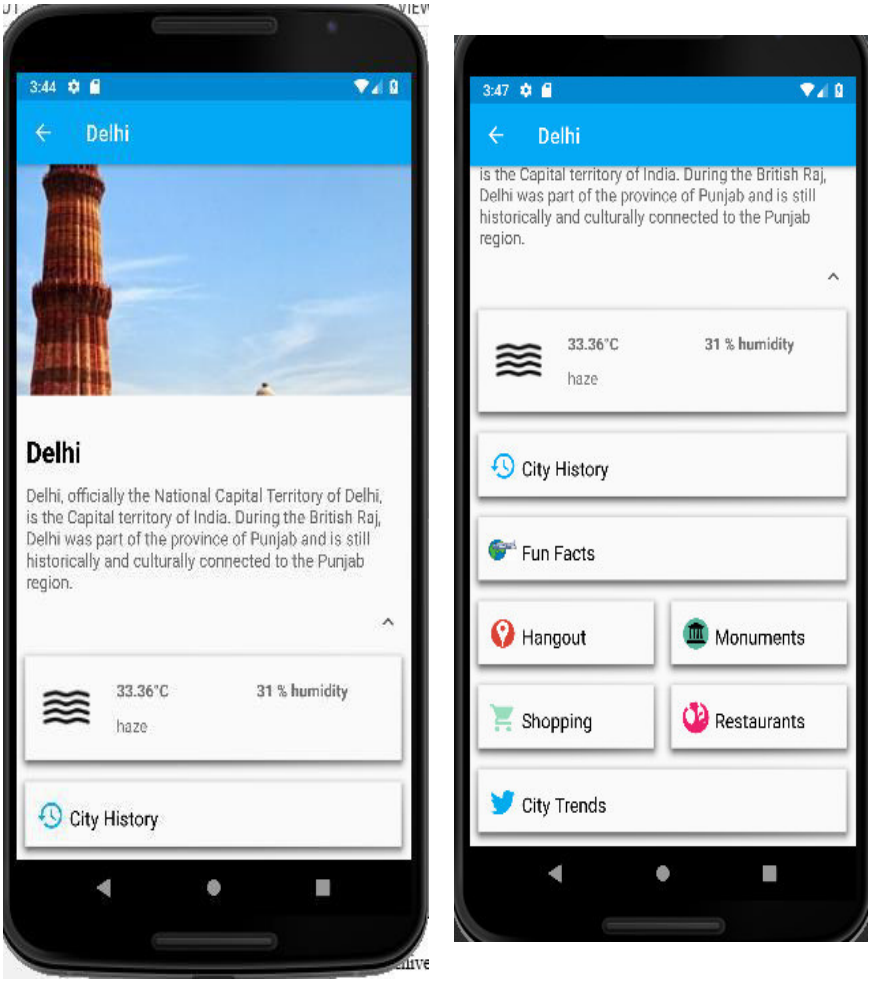

Рисунок 3.3.4 – Обзор выбранного города

Код для получения истории выбранного города:

```
public void fetchHistory() {
animationView.playAnimation();
Handler handler = new Handler(Looper.getMainLooper());
String uri;
uri = API_LINK_V2 + "get-city-information/" + mCity.getId();
Log.v("EXECUTING", uri);
        .url(uri)
        .build();
{
        Log.e("Request Failed", "Message : " + e.getMessage());
        handler.post(() -> networkError());
    }
```

После выбора города, пользователь может посмотреть список ресторанов, кафе (рисунок 3.3.5). Так же в активити «Monuments» содержатся

перечень достопримечательностей, которые вынесены на карту города (рисунок 3.3.6).

Рисунок 3.3.5 – Список ресторанов

Ниже представлен фрагмент кода для активити «Restaurants»:

```
private void getRestaurantItems() {
    String uri = API_LINK_V2 + "get-all-restaurants/" + mCity.getLatitude() + "/" + mCity.getLongitude();
    Log.v("executing", "URI : " + uri);
String uri = API_LINK_V2 + "get-all-restaurants/" + mCity.getLatitude() + "/" + mCity.getLongitude();
    Log.v("executing", "URI : " + uri);
```

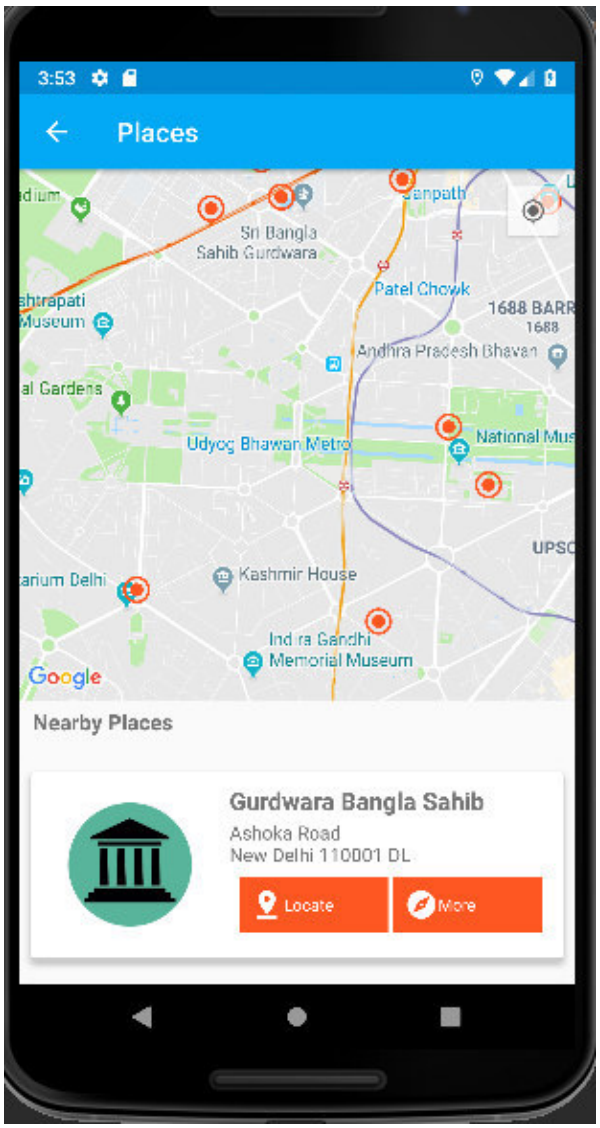

Рисунок 3.3.6 – Перечень достопримечательностей

```
//Set up client
OkHttpClient client = new OkHttpClient();
  {
    @Override
    public void onFailure(Call call, IOException e) {
        mHandler.post(() -> networkError());
        Log.v("Request Failed", "Message : " + e.getMessage());
    }

          OkHttpClient client = new OkHttpClient();
//Execute request
Request request = new Request.Builder()
        .header("Authorization", "Token " + mToken)
        .url(uri);
```

4 Экономическая часть

В данной дипломной работе описан проект по разработке мобильного приложения для путешествия в другие страны.

Целью данного проекта является – обеспечение всей необходимой информации туристам. Благодаря данному приложению пользователь сможет увидеть список всех достопримечательностей, а также сохранить в списки. Интерфейс программы достаточно прост и удобен для использования данного приложения.

В данном разделе дипломной работы производится расчет экономической составляющей по разработке мобильного приложения отражающие временные, трудовые, и финансовые затраты.

4.1 Трудоемкость разработки приложения

Для того, что бы определить трудоемкость разработки программного продукта необходимо прежде всего составить в ходе разработки мобильного приложения перечень всех необходимов этапов, и какие виды работ на определенном этапе исполнялись.

Трудоемкость программного продукта отражена в таблице 4.1.1.

Таблица 4.1.1 - Распределение работ при разработки приложения

Этап разработки Android-приложения	Вид работы	Трудоемкость разработки, чел.× ч.
Формулировка цели для создания ПП	Выбор цели	2 ч
	Обоснование выбора разработки программного продукта	3 ч
Анализ рынка	Исследование рынка	5 ч
	Анализ существующих ПО	4,5 ч
Алгоритмизация	Описание алгоритмов и различных процессов, составление схемы	3,5 ч
Работа по созданию и управлению БД	Создание и доведение БД до рабочего состояния	40 ч
Разработка графического дизайна	Создание дизайна программного продукта	50 ч
Создание программы	Создание функциональной части ПП	160 ч
	Тестирование ПП	40 ч

Продолженние таблицы 4.1.1

Этап разработки Android-приложения	Вид работы	Трудоемкость разработки, чел.× ч.
Отладка программного продукта	Исправление сбоев в функциональной части	30 ч
	Корректировка или оптимизация дизайна ПП	20 ч
Итоговая трудоемкость разработки мобильного приложения:	358 ч	

4.2 Расчет затрат на разработку Android-приложения

В данном этапе рассчитываются затраты на необходимые материальные ресурсы, которые в свою очередь, делятся на основные и дополнительные, иногдв их еще называют вспомагательными затратами.

Определяем общую сумму затрат на различные материальные ресурсы (Зм), по формуле, которая представлена ниже:

$$З_м = \sum_{i=1}^{n} Р_i * Ц_i \quad (4.1)$$

Где $Р_i$ - расход i-го вида материального ресурса, натуральные единицы;

$Ц_i$ - цена за единицу i-го вида материального ресурса, тг;

i - вид материального ресурса;

n - количество видов материальных ресурсов.

$$З_{бумага} = 1 \times 1000 = 1000 \text{ тг} \quad (4.1.1)$$

$$З_{катридж} = 1 \times 984 = 984 \text{ тг} \quad (4.1.2)$$

$$З_{общие} = 1000 + 984 = 1984 \text{ тг} \quad (4.1.3)$$

Расчет затрат на материальные ресурсы представлены в таблице 2.

Таблица 4.1.2 - Расход на материальные ресурсы

Использованный ресурс	Единица измерения	Кол.израсход. материала	Цена за единицу, тг	Сумма, тг
Бумага А4	Шт	1	1000	1000
Катридж	Шт	1	984	984
ИТОГО расход на материалы составляет:				1 984

Так как для разработки программного продукта используется техника, которая требует некоторых расходов.

n - количество электрооборудования.

$$З_{э\ ноутбук} = 0,09 \times 0,7 \times 358 \times 18,32 = 413,2 \text{ тенге} \quad (4.2.1)$$

$$З_{э\ принтер} = 0,304 \times 0,7 \times 15 \times 18,32 = 58,5 \text{ тенге} \quad (4.2.2)$$

$$З_{э\ smartphone} = 0,012 \times 0,7 \times 40 \times 18,32 = 6,2 \text{ тенге} \quad (4.2.3)$$

$$З_{э\ общие} = 413,2+6,2+58,5=478,5 \text{ тенге} \quad (4.2.4)$$

Необходимо учесть расходы по оплате труда всех работников, которые вовлечены и заняты разработкой самого программного продукта.

Таблица 4.1.3 - Расходы на электроэнергию

Наименование оборудования	Паспортная мощность, кВт	Коэффициент использования мощности	Время работы оборудования для разработки ПП, ч	Цена электроэнергии, тг/кВт*ч	Сумма, тг
Ноутбук Apple MacBook Air 13 (MQD32) 128Gb 2017	0,09	0,7	358	18,32	413,2
Смартфон Samsung	0,012	0,7	40	18,32	6,2

Продолжение таблицы 4.1.3

Оборудование	Паспортная мощность, кВт	Коэффициент использования мощности	Время работы оборудования, ч	Цена электроэнергии, тг/кВт*ч	Сумма , тг
Принтер KYOCERA FS-1040	0,304	0,7	15	18,32	58,5
ИТОГО расход на электроэнергию составляет:					478,5

Затраты на оплату труда приведены в таблице 4.

Таблица 4.1.4 - Затраты на оплату труда

Категория работника	Трудоемкость разработки, чел.×ч	Часовая ставка, тг/ч	Сумма, тг
Программист	358	892,86	319 644
Руководитель проекта	10	1071,43	10 710
Иготовая сумма на оплату труда:			330 324

Общая сумма затрат на оплату труда ($З_{ТР}$) определяется по формуле:

$$З_{тр} = \sum_{i=1}^{n} ЧС_i * Т_i \quad (4.3)$$

где $ЧС_i$ – ставка в час i-го работника, тг;

$Т_i$ - трудоемкость разработки приложения, чел.×ч;

i – категория работника;

n – количество работников, занятых разработкой мобильного приложения.

Часовая ставка служащего:

$$ЧС_i = \frac{ЗП_i}{ФРВ_i} \quad (4.4)$$

$$ЧС_i = \frac{ЗП_i}{ФРВ_i} = \frac{150\,000}{168} = 892{,}86$$

$$ЧС_{рук} = \frac{ЗП_{рук}}{ФРВ_{рук}} = \frac{180\,000}{168} = 1071{,}43$$

ФРВ – фонд рабочего времени сотрудника, который составляет примерно 168 часов в месяц.

Трудоемкость разработки мобильного приложения под Android определяется по данным таблицы 4.1.1.

Социальный налог и медстраховка составляет 11% от затрат на оплату труда всех служащих ($З_{тр}$), занятых разработкой мобильного приложения. Так же необходимо учесть, что пенсионные отчисления не облагаются социальным налогом (ставки указаны на 2019 год).

$$Зпо = Зтр \times 0{,}1 = 319\,644 \times 0{,}1 = 31\,964{,}4 \quad (4.4.1)$$

$$Зтр\,(с\ п.о.) = Зтр - Зпо = 319\,644 - 31\,964{,}4 = 287\,679{,}6 \quad (4.4.2)$$

$$Нс = Зтр(с\ п.о.) \times 0{,}11 = 287\,679{,}6 \times 0{,}11 = 31\,645 \quad (4.4.3)$$

Расчет затрат на оплату труда для руководителя проекта.

$$Зпо = Зтр \times 0{,}1 = 10\,710 \times 0{,}1 = 1\,071 \quad (4.4.4)$$

$$Зтр\,(с\ уч\ п.о.) = Зтр - Зпо = 10\,710 - 1\,071 = 9\,639 \quad (4.4.5)$$

$$Нс = Зтр(с\ уч\ п.о.) \times 0{,}11 = 9\,639 \times 0{,}11 = 1\,060{,}29 \quad (4.4.6)$$

Денежное возмещение износа оборудования включается в статью «Амортизационные расходы».

Амортизационные отчисления приведены в таблице 5.

Общая сумма амортизационных отчислений определяется по формуле:

$$З_{АМ} = \sum_{i=1}^{n} \frac{Ф_i * Н_{А_i} * Т_{НИРi}}{100 * Т_{Эфi}} \quad (4.5)$$

где $Ф_i$ - стоимость i-го ОФ, тг;

$Н_{Ai}$ - годовая норма амортизации i-го ОФ, %;

$Т_{НИРi}$ - время работы i-го ОФ за весь период разработки мобильного приложения, ч;

$Т_{Эфi}$ - эффективный фонд времени работы ОФ за год, ч;

i – вид ОФ;

n - количество ОФ.

Годовые нормы полезного использования оборудования принимаются от 3 до 10 лет. Исходя из этого, срок полезного использования ноутбук Apple Macbook составляет – 5 лет, срок использование компбютерной мышки – 3 года, а срок использования принтера – 7 лет.

Годовые амортизационные фонды составляют:

$$\text{Н}_{Ai} = \frac{100}{T_{Ni}}, \quad (4.6)$$

где T_{Ni} - возможный срок использования i-го ОФ, год;

$$\text{Н}_{\text{Ai}} = \frac{100}{5} = 20\ \%$$

$$\text{Н}_{\text{Ai}} = \frac{100}{3} = 33{,}3\ \%$$

$$\text{Н}_{\text{Ai}} = \frac{100}{7} = 14{,}3\ \%$$

Далее идет подсчет затрат на амортизационные отчисления для каждого использованнового входе разработки оборудования:

$$\text{З}_{\text{АМ1}} = \frac{350\,000 \times 20 \times 358}{100 \times 1970} = 12\,721 \text{ тенге} \quad (4.5.1)$$

$$\text{З}_{\text{АМ2}} = \frac{25\,000 \times 33{,}3 \times 358}{100 \times 1970} = 1\,513 \text{ тенге} \quad (4.5.2)$$

$$\text{З}_{\text{АМ3}} = \frac{64\,000 \times 14{,}3 \times 15}{100 \times 1970} = 70 \text{ тенге} \quad (4.5.3)$$

$$\text{З}_{\text{АМ4}} = \frac{60\,000 \times 20 \times 40}{100 \times 1970} = 244 \text{ тенге} \quad (4.5.4)$$

$$\text{З}_{\text{АМобщие}} = 12\,721 + 1\,513 + 70 + 244 = 14\,548 \text{ тенге}$$

Таблица 4.1.5 - Амортизация ОФ

Наименование оборудования и ПО	Стоимость оборудования и ПО, тг	Годовая норма амортизации, %	Эффективный фонд времени работы оборудования и ПО, ч/год	Время работы оборудования и ПО для разработки ПП, ч	Сумма, тг
Ноутбук Apple MacBook Air 13 (MQD32) 128Gb 2017	350 000	20	1970	358	12 721

Продолжение таблицы 4.1.5

Наименование оборудования и ПО	Стоимость оборудования и ПО, тг	Годовая норма амортизации, %		Эффективный фонд времени работы оборудования и ПО, ч/год	Время работы оборудования и ПО для разработки ПП, ч	Сумма, тг
Принтер лазерный HP P1102	64 000	14,3		1970	15	70
Смартфон Samsung	60 000	20		1970	40	244
Беспроводная компьютерная мышь Microsoft	25 000	33,3	1970	358	1513	Беспроводная компьютерная мышь Microsoft
	Итого по амортизационным фондам					14 548

Так же были произведены затраты на аредную плату помещения. Данный вид затрат входит в категорию «Прочие затраты».

Таблица 4.1.6 – Арендная плата

Площадь помещения, $м^2$	Стоимость 1 $м^2$	Стоимость за 1 месяц, тг	Длительность аренды (в месяцах)	Сумма, тг
35	3800	133 000	1	133 000

Затраты на интернет приведены в таблице 4.1.7

Таблица 4.1.7 – Дополнительные затраты

Цена за 1 месяц, тг	Количество месяцев	Сумма, тг
4 500	1	4 500

В таблице 4.1.8 составлена смета всех затрат на разработку мобильного приложения.

Таблица 4.1.8 - Смета затрат на разработку мобильного приложения

Статьи затрат программного обеспечения	Сумма, тг
1. Материальные затраты, в том числе: - материалы - электроэнергия	2 465,5
2. Затраты на оплату труда.	330 324
3. Отчисления на социальные нужды.	32 705,29
4. Амортизация основных фондов.	14 548
5. Прочие затраты.	137 500
Итогая смета разработки мобильного приложения	517 542,79

Структура себестоимости разработки Android-приложения представлена на рисунке 4.1

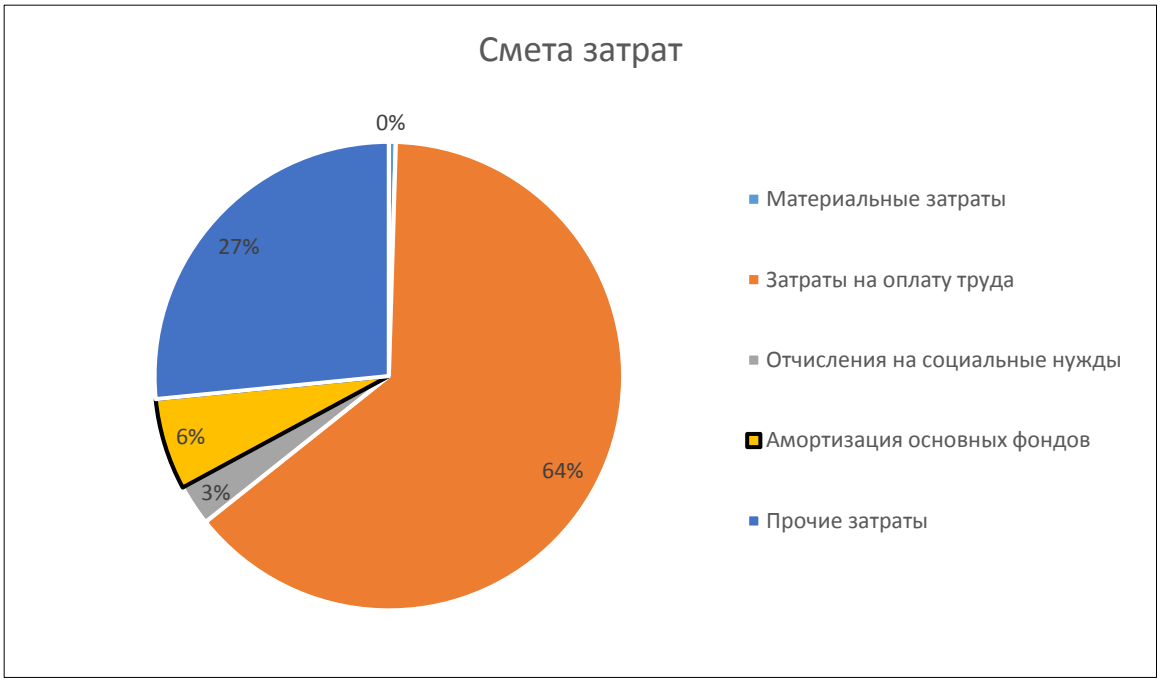

Рисунок 4.1. Смета затрат ПП

4.3 Определение договорной цены мобильного приложения

Величина договорной цены мобильного Android-приложения устанавливается с учетом эффективности, качества и сроков ее выполнения на уровне, отвечающем экономическим интересам клиента и самого исполнителя. Договорная цена ($Ц_д$) для прикладных ПП рассчитывается по формуле:

$$Ц_д = З_{ПП} \times \left(1 + \frac{P}{100}\right), \quad (4.7)$$

где $З_{ПО}$ - затраты на разработку мобильного приложения (из таблицы 4.6), тг;

Р - средний уровень рентабельности разрабатываемого продукта, который составляет 25% .

$$Ц_д = З_{ПО} \times \left(1 + \frac{P}{100}\right) = 517\,542{,}79 \times (1 + 0{,}25)$$
$$= 517\,542{,}79 + 129\,385{,}6975 = 646\,928{,}4875 \text{ тенге}$$

Затем следует определить цену реализации (с учетом налога, принятым Налоговым кодексом РК и равным 12%).

Цена реализации с учетом НДС рассчитывается по формуле:

$$Цр = Цд + Цд \times НДС$$

$$Ц_р = 646\,928{,}4875 + 646\,928{,}4875 \times 0{,}12 = 724\,559{,}906 \text{ тенге}$$

Таким образом:

– себестоимость программного продукта составляет - 517 542,79 тенге;
– прибыль составляет - 129 385,6975 тенге;
– договорная цена - 724 559,906.

5 Производственная безопасность

Главной целью данной дипломной работы является создание мобильного Android-приложения для основной таргетной аудитории – туристов и просто тех, кто выезжает в различные страны. Так же пользователи, приехавшие из других стран в Казахстан, выбрав город их пребывания, могут ознакомиться со всеми достопримечательности и прекрасными местами нашей родной страны, благодаря данному приложению, которое для удобства пользователей написано на нескольких языках.

5.1 Анализ условий труда

Рабочее место - это некоторая часть пространства, в котором инженер, в нашем случае именно программист, осуществляет некую трудовую деятельность, и в следствие чего проводит большую часть рабочего времени.

Если рабочее место сотрудника должно быть организовано по всем правилам, в этом случае производительность труда инженера возрастает с 8 до 20 процентов.

Работа над данным проектом осуществляется в офисе, который находится по адресу Айтеке би, 29. В данном офисе работаю 3 служащих. Так как работа происходит в офисе, где единственное устройство – ноутбук, то проблемы с шумом исключаются.

Так же в данном помещении установлен настенный кондиционер ALMACOM ACH-24 AS, который обеспечивает хороший воздухообмен, поддерживает приемлемую и комфортную температуру. Из этого следует сделать вывод, то, что в офисе, где происходит основная работа над программным продуктом, проблем с шумом и вентиляцией не наблюдаются.

Но стоит заметить, что окна здания выходят на Север и в большей части офис находится в тени, из-за чего возникают проблемы с освещением.

В комнате поставлены 3 светильника ЛД65 и имеетя одно окно площадью 7 $м^2$.

Недостаток освещение неблагоприятно влияет на работу программиста, в данном разделе были проведены расчеты по решению данной проблемы.

5.2 Расчет освещения

5.2.1 Расчет естественного освещения

Недостаточная освещенность и пониженная контрастность могут повлечь за собой напряженность зрительного анализатора, что в дальнейшем может привести к ухудшению зрения.

В данном случае, работа предполагает собой чтение, письмо и работу за компьютером, из этого следует, что освещенность, которая необходима для выполнения работ в данном помещении составляет 500 лк.

Необходимо просчитать площадь боковых световых проемов офиса, которая предназначена для создания нормируемой освещенности на рабочих местах.

Тип помещения – офис.

Характеристики помещения:

длина помещения равна $L = 7$ м,

ширина помещения равна $B = 5$ м,

высота помещения равна $H = 3$ м.

Высота рабочей поверхности составляет $h_{\text{рп}} = 0{,}725$, окна начинаются с высоты $h\text{рп} = 0{,}725$ м, высота самого окна $h_0 = 1{,}5$м.

Офис находится в городе Алматы (в IV часовом поясе). Световой климат пояса - IV северной широты и южнее.

Рабочие места расположены в $1_{\text{рт}} = 0{,}5$ м, от наружной стены помещения, где проектируются оконные проемы. Минимальная освещенность будет в точке, отстоящей на расстояние 4 м от оконного проема.

Общую площадь окон S_0, м^2, можно определить по формулам 5.1 и 5.2

$$\mathrm{S_o} = \frac{\mathrm{S_o} * \mathrm{e_N} * \mathrm{K_з} * \eta_o * \mathrm{K_{зд}}}{100 * \tau_0 * \mathrm{r}} \tag{5.1}$$

где S_n- площадь помещения;

$$\mathrm{S}_n = L * B = 7 \times 5 = 35 \text{ м}^2; \tag{5.2}$$

$e_н$ – нормируемое значение КЕО;

$К_з$ –коэффициент запаса;

m_N - коэффициент светового климата;

Учитывая заданный световой пояс, ориентация световых проемов направлена на Север, определим по формуле 5.3.

$$e_x^{IV} = e_x \times m \times c \tag{5.3}$$

где m = 0,7;

$c = 0{,}75$;

$en - 1{,}2$.

$$e_x^{IV} = 1{,}2 \times 0{,}7 \times 0{,}75 = 0{,}63$$

Находим коэффициент $k_3 = 1{,}2$ (учебные помещения, лаборатории, конструкторские бюро).

τ_0- общий коэффициент светопропускания,
равный

$$\tau_0 = \tau_1 \times \tau_2 \times \tau_3 \times \tau_4;$$

$\tau_1 = 0{,}5$ (пустотелые стеклянные блоки);

$\tau_2 = 0{,}6$ (деревянные двойные раздельные переплеты);

$\tau_3 = 0{,}8$ (железобетонные фермы и арки);

$\tau_4 = 1$ (регулируемые жалюзи и шторы);

$\eta_0 -$ световая характеристика окон.

$$\tau_0 = 0{,}8 * 0{,}6 * 0{,}5 * 1 = 0{,}24$$

Определим по формуле 5.4

$$L = B - 1$$

$$h_{\text{расч}} = h_{\text{но}} + h_0 - h_{\text{рп}} \qquad (5.4)$$

$$L = 5 - 1 = 4 \text{ м}$$

$$\frac{L}{l} = \frac{L}{B-1} = \frac{7}{4} = 1{,}75$$

$$h_{\text{расч}} = 0{,}8 + 1{,}5 - 0{,}75 = 1{,}57$$

$$\frac{B}{h_{\text{расч}}} = \frac{5}{1{,}57} = 3{,}18$$

По таблице 5.2 определим $\eta 0 = 10{,}5$.

$r_1 -$ коэффициент, который учитывае рост КЕО;

$$\frac{B}{h_{\text{расч}}} = \frac{0{,}5}{4} = 0{,}125$$

Где ρср $= 0{,}5$ – средник коэффициент отражение в офисе. Во внимание принимается только односторонее боковое освещение.

Поблизости нет высоток и зданий, которые затеняют. Поэтому их коэффициент не берется во внимание и равен 1.

$$S_{\text{сп}} = \frac{35 \times 1{,}2 \times 10{,}5 \times 1 \times 0{,}63}{100 \times 0{,}24 \times 1{,}05} = 11{,}025 \text{ м}^2$$

$S_{\text{сп}} = 11{,}025$ м^2, означает, что в помещение должно быть окно с площадью 11,025 м^2 .

$$l_{ok} = \frac{S_o}{h_{ok}} = \frac{11{,}025}{3} = 3{,}675 \qquad (5.5)$$

В помещение имеется только одно окно площадью 7 м^2 вместо нужных 11,025 м^2 . Исходя из этого, можно сделать вывод, что в помещение не хватает естественного освещение и есть потребность в использование искусственного освещение.

Так как основаня цель – создание благоприятных условий труда в помещение с площадью S=35 м^2.

5.2.2 Расчет искусственного освещения

В качестве светильника был взят ЛД65. Длина светильника 1514 мм, мощность – 65 ВТ. В помещении имеются 3 светильника.

Найденный разряд работы зрения – V.

Нормируемая освещённость в офисе составляет 500 лк.

Схема освещенности представлена на рисунке 5.1

В таблицу 5.1 изображены технические характеристики используемых светильников.

Данный выбор светильников зависит только от выбора самого заказчика.

Таблица 5.1– Технические характеристики ЛД65

тип лампы	Мощность, вт	напряжение, в	световой поток, лм	длина L, не более, мм	диаметр D, мм	тип цоколя
ЛД 65	65	110	3750	1514	38	G13d

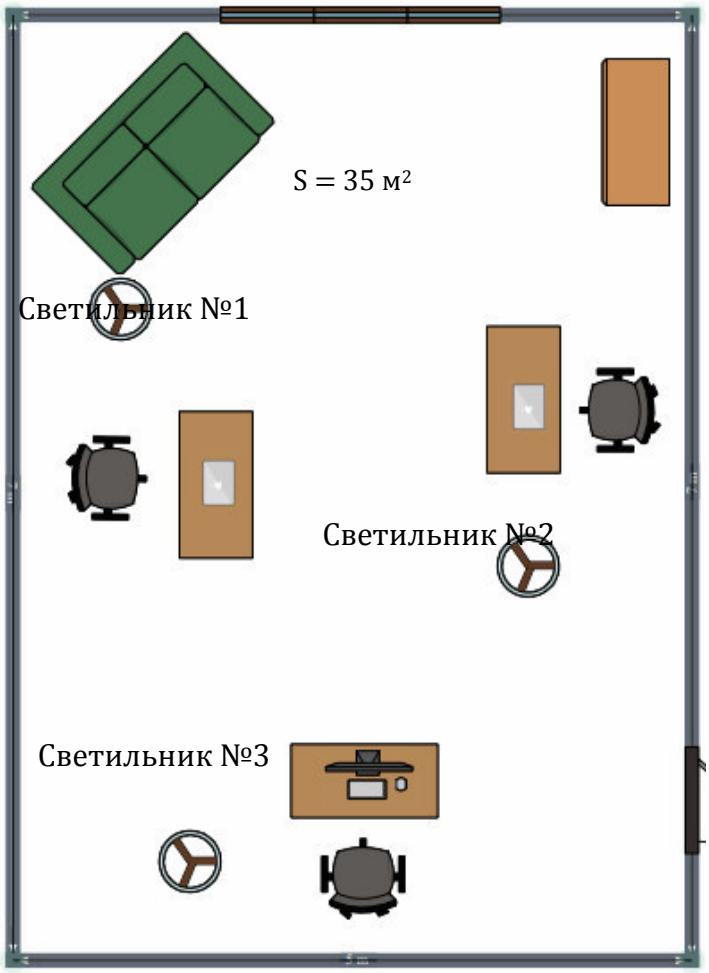

Рисунок 5.1 – Схема расчета освещенности

Для начала идет проверка соответсивия минимальной заданной освещенности при имеющихся 3 светильниках.

$$E = \frac{N \cdot n \cdot \Phi_л \cdot \eta}{k_z \cdot S_{oc} \cdot Z}$$

Где S_{OC} −площадь помещения;

k_z − коэффициент запаса;

N – количество светильников;

Z –коэффициент неравномерности освещения, Z = 1,4;

n – количество ламп в светильнике;

$Ф_л$ – световой поток выбранной лампы, Фл = 3570 лм

η – коэффициент использования, $\eta = 65\%$

$$E = \frac{3 \cdot 2 \cdot 3750 \cdot 0{,}65}{1{,}4 \cdot 35 \cdot 1{,}2} \approx 250 \text{ лк}$$

При 3 светильниках минимальная освещенность равняется примерно 250 лк, что не соотвествует условиям труда. В следствие чего возникает необходимость, увеличить количество светильников для обеспечения приемлемой освещенности. Для этого необхоимо произвести расчеты для рекострукции.

Ниже прредставлены коэффициенты отражения от потолка стен и пола:

$$\rho\text{пот} = 70\ \%$$

$$\rho\text{ст} = 50\ \%$$

$$\rho\text{пол} = 30\ \%$$

Вычислим высоту подвеса светильника над рабочей поверхностью по формуле 5.7:

$$H = h - hp - hc \qquad (5.7)$$

где: h_c – расстояние от светильника до перекрытия, h_c = 0,1 м;

h_p – высота рабочей поверхности над полом, h_p = 0,7 м;

h – высота помещения, h=3 м.

$$H = 3 - 0{,}1 - 0{,}7 = 2{,}2 \text{ м}$$

Расстояние между рядами светильников:

$$L_b = \lambda \cdot H \qquad (5.8)$$

$$L_b = 0{,}8 \cdot 2{,}2 = 1{,}76 \text{ м}$$

Расстояние между светильниками:

$$L_a = L_b - 0{,}1 = 1{,}76 - 0{,}1 = 1{,}66 \text{ м}$$

Расстояние от стены до ближайшего светильника, когда работа у стены не проводится, определяем по формуле 5.9

$$l_1 = (0{,}4 \div 0{,}8) \cdot L \tag{5.9}$$

$$l_1 = 0{,}6 \cdot 1{,}76 = 1{,}056 \text{ м}$$

Определяем индекс помещения по формуле 5.10

$$i = \frac{L \cdot B}{h_p \cdot (L + B)} \tag{5.10}$$

$$i = \frac{4 \cdot 2}{2{,}175 \cdot (4 + 2)} = 1{,}319$$

Коэффициент использования в данном случае равен $\eta = 65\%$, коэффициент запаса равен $k_з = 1{,}2$

Определим количество люминесцентных ламп по формуле 5.11

$$N = \frac{E \cdot k_z \cdot S_{OC} \cdot Z}{n \cdot \Phi_л \cdot \eta} \tag{5.11}$$

Где S_{OC} – площадь помещения;

k_z – коэффициент запаса;

E – заданная минимальная освещенность, Е = 500 лк;

Z – коэффициент неравномерности освещения, Z = 1,4;

n – количество ламп в светильнике;

$\Phi_л$ – световой поток выбранной лампы, Фл = 3570 лм

η – коэффициент использования, η = 65%

$$N = \frac{500 \times 1{,}2 \times 35 \times 1{,}4}{2 \times 3750 \times 0{,}65} \approx 6$$

В результате получим, что для создания нормируемой освещенности 500 лк в офисе необходимо поставить 6 люминесцентных ламп серии ЛД с минимальной мощностью 65 Вт. При начальном анализе офиса данное

требование не соответствовало реальному расположению дел. В офисе имело всего 3 светильнка, то есть они давали на весь офис всего 50 процентов света. По санитарным мерам это не допустимо.

Для решения данной проблемы следует провести реконструкцию офиса и добавить 3 люминесцентных ламп серии ЛД-65 для обеспечения необходимого освещения в помещении. На рисунке 5.2 изображен офис после реконструкции.

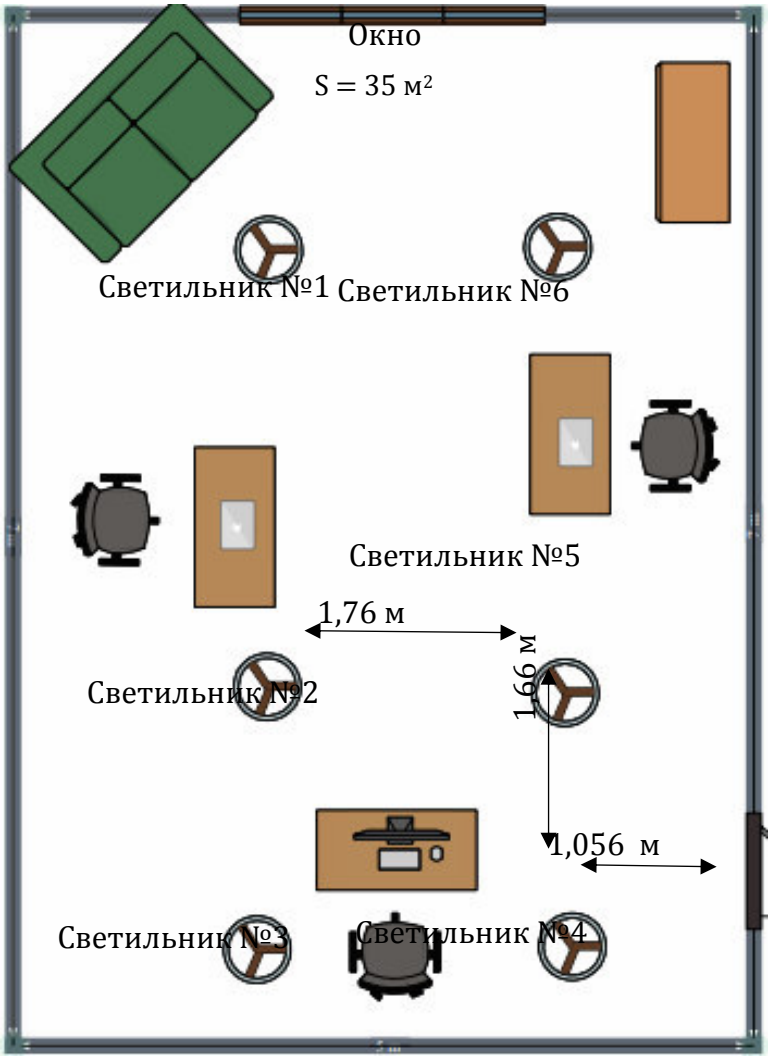

Рисунок 5.2 - Схема после реконструкции

Заключение

Данный дипломный проект раскрывает вопросы о недосточной информационной осведомленности по многим разным странам. Актуальность данного проекта состоит в том, что были раскрыты недостатки уже существующих мобильных приложений в сфере туризма и путешествий, а так же было предложено решение для устранения данных проблем.

В ходе выполнения дипломной работы было разработано мобильное приложение для путешественников в различные точки мира, спроектированное на платформе Android.

Данное приложение подойдет абсолютно для любого человека, кто планирует поездку в другие города и страны. Также она подойдет для туризма внутри родной страны.

Был создан приятный интерфейс, а также учтены все потребности туриста в путешествие.

При разработки мобильного приложения на базе ОС Android были разаботаны такие функции как:

– найдены и проанализированы приложение, разработанные по той же аналогии, выявлены недостатки и плюсы;

– было создано приложение, которое реализует следующие функции:

– нахождение подробной информации об истории города, достопримечательностей, список рестаранов и т.д;

– использование Google Maps для определения географический координакт локаций и текущего месторасположения пользователя;

– поиск в базе данных, а так же полная визуализация на map-картах Google основных объектов выбранной Activity;

– произведена отладка приложения как и на реальном устройстве, так и на стандартных эмуляторах.

Список литературы

1 Пайлон Д. UML версия 2 для программистов. Издательство Питер, 2012. (Дата обращения: 05.04.2019)

2 Котляров В.П. Основной фундамент для тестирования программного обеспечения. Издательство Бином, 2009. (Дата обращения: 08.04.2019)

3 https://ru.wikipedia.org/wiki

4 Основы компьютерного моделирования// Программа компьютерного моделирования BPwin (AllFusion Process Modeler). URL: http://bourabai.kz/cm/bpwin.htm (дата обращения: 22.02.2019)

5 Студенческая библиотека онлайн// Обеспечение работы веб-сайта. URL: https://studbooks.net/2215545/informatika/obespechenie_raboty_sayta (дата обращения 25.02.2019)

6 Ресурс для IT-специалистов// Visual Studio Code – редактор кода для Linux, OSX и Windows. URL: https://habr.com/ru/company/microsoft/blog/262523/ (дата обращения: 01.03.2019)

7 Ткаченко В. Обучение в Интернет// Структура программного обеспечения ПК. URL: https://www.lessons-tva.info/edu/e-inf1/e-inf1-3-1.html (дата обращения: 03.03.2019)

8 Абдимуратов Ж.С., Мананбаева С.Е. Безопасность жизнедеятельности. Методические указания к выполнению раздела «Расчет производственного освещения» в выпускных работах для всех специальностей. Бакалавриат – Алматы: АИЭС, 2009. – 20 с.

9 Естественное и искусственное освещение. Государственные нормативы в области архитектуры, градостраительства и строительства. СНиП РК 2.04. - 05.2002.

10 Косьяненко М.А Способы разработки мобильных приложений одновременно для ос android и ios / Инноватика-2018: сб. материалов XIV Международной школы-конференции студентов, аспирантов и молодых ученых (26-27 апреля 2018 года) / под ред. А.Н. Солдатова, С.Л. Минькова. - Томск, STT, 2018 (в печ.)

11 Гриффитс Д., Гриффитс Д. Head First. Программирование для Android [Текст]. – СПб.: Питер, 2016. – 704 с.: ил. – (Серия «Head First O’Reilly»).

12 SQLite vs MySQL vs PostgreSQL: compare beetwen two systems [Электронный ресурс]. – URL: http://devacademy.ru/posts/sqlite- vs-mysql-vs-postgresql/ (дата обращения: 21.02.2019).

13 Android Developers. CameraDevice [Электронный ресурс]. – https://developer.android.com/reference/android/hardware/camera2/CameraDevice. ht ml. (дата обращения: 27.04.2019)

14 Android Developers. Connecting to the Network [Электронный ресурс]. – https://developer.android.com/training/basics/network- ops/connecting.html (дата обращения: 07.04.2019).

15 Android Developers. LocationListener [Электронный ресурс]. – https://developer.android.com/reference/android/location/LocationListener.html. (дата обращения: 07.03.2019).

16 Android Developers. Motion Sensors [Электронный ресурс]. – (https://developer.android.com/guide/topics/sensors/sensors_motion.html#sensors-motion-linear). (дата обращения: 15.05.2019).

17 Обзор среды программирования Android Studio [Электронный ресурс]. – URL: https://www.intuit.ru/studies/courses/12643/1191/lecture/21980? page=3 (дата обращения: 9.02.2019).

18 Прототип: что это и зачем он нужен? [Электронный ресурс]. – URL: https://infoshell.ru/blog/prototip-chto-jeto-i-zachem-on-nuzhen/ (дата обращения: 11.02.2019).

19 Android Developers. Sensors Overview [Электронный ресурс]. – https://developer.android.com/guide/topics/themes/study/sensors/sensors_overview.html.(дата обращения: 07.04.2019).

20 Мобильные приложения требуют хостинга [Электронный ресурс]. – URL: http://blog.filanco.ru/2012/11/28/mobilnye-prilozheniya-trebuyut-xostinga/ (дата обращения: 26.03.2019).

Приложение А
(обязательное)

Техническое задание

1. Общие требования:

– Наименование разрабатываемой системы:

– Разработка мобильного путеводителя для путешественников по разным точкам мира

– Цель разработки:

– Удовлетворение основных потребности пользователей в качестве туристов. А именно: обзор достопримечательностей, планирование путешествия, а также быстрый поиск нахождения мест питания и проживания, шоппинг, посещение различных коцертов и мероприятий – все это значительно упростит отдых для турист и сэкономит время поисков;

– Предлагаемые технологии для разработки системы (на выбор разработчика):

– Android Studio;
– GenyMotion;
– MobiOne;
– Android SDK;
– TestIphone;
– Предлагаемые языки и технологии программирования:
– Python;
– Ruby;
– Kotlin;
– Java;
– C#;
– PyGame.
– Kivy;
– Общий объем программной части системы, Мб
– Не более 50 Мб.

2. Технические требования:
– Платформа работы приложения: Android
– Приложение Android
– Верстка Android Книжная: Да
– Верстка Android Альбомная: Адаптивная от книжной
– Совместимость с Android: Android 4.4. и старше
– Верстка планшет Книжная: Адаптивная от телефона
– Верстка планшет Альбомная: Адаптивная от телефона
– Сервер
– Совместимый вебхостинг на базе Apache2+PHP5+MySQL

3. Экономические требования:

– Расчет стоимости системы и стоимости разработки программного обеспечения (подлежит обсуждению):

– Стоимость готового продукта 725 000 тг;

– Стоимость разработки 518 000тг.

– Целевая аудитория и области применения:

.

Приложение Б
(обязательное)

Листинг программы

```
Token " + mToken)
.url(uri)
.build();
//Setup callback
client.newCall(request).enqueue(new Callback() {
@Override
public void onFailure(Call call, IOException e) {
mHandler.post(() -> networkError());
Log.v("Request Failed", "Message : " + e.getMessage());
}
@Override
public void onResponse(Call call, final Response response) throws IOException {
final String res = Objects.requireNonNull(response.body()).string();
mHandler.post(() -> {
try {
JSONArray array = new JSONArray(res);
Log.v("Response",75 res );
for (int i = 0; i < array.length(); i++) {
JSONObject object = array.getJSONObject(i);
String imageUrl = object.getString("restaurant_image");
String name = object.getString("restaurant_name");
String address = object.getString("address");
String ratings = object.getString("aggregate_rating");
String votes = object.getString("votes");
String restaurantURL = object.getString("restaurant_url");
int avgCost = object.getInt("avg_cost_2");
restaurantItemEntities.add(
new RestaurantItemEntity(imageUrl, name, address, ratings,
votes, avgCost, restaurantURL));
JSONObject object = array.getJSONObject(i);
String imageUrl = object.getString("restaurant_image");
String name = object.getString("restaurant_name");
String address = object.getString("address");
String ratings = object.getString("aggregate_rating");
String votes = object.getString("votes");
String restaurantURL = object.getString("restaurant_url");
int avgCost = object.getInt("avg_cost_2");
String restaurantURL = object.getString("restaurant_url");
int avgCost = object.getInt("avg_cost_2");
restaurantItemEntities.add(
new RestaurantItemEntity(imageUrl, name, address, ratings,
votes, avgCost, restaurantURL));
JSONObject object = array.getJSONObject
```

```
public class RestaurantsActivity extends AppCompatActivity implements
RestaurantsCardViewAdapter.OnItemClickListener {
    @BindView(R.id.restaurants_recycler_view)
    RecyclerView mRestaurantsOptionsRecycleView;
    @BindView(R.id.animation_view)
    LottieAnimationView animationView;

    private City mCity;
    private Handler mHandler;

private String mToken;


private SharedPreferences mSharedPreferences;
    public List<RestaurantItemEntity> restaurantItemEntities = new ArrayList<>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_restaurants);
        ButterKnife.bind(this);

        Intent intent = getIntent();
        mCity = (City) intent.getSerializableExtra(EXTRA_MESSAGE_CITY_OBJECT);
        mHandler = new Handler(Looper.getMainLooper());
        mSharedPreferences = PreferenceManager.getDefaultSharedPreferences(this);
        mToken = mSharedPreferences.getString(USER_TOKEN, null);
        getRestaurantItems();
        Objects.requireNonNull(getSupportActionBar()).setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        setTitle(mCity.getNickname());
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == android.R.id.home)
            finish();
        return super.onOptionsItemSelected(item);
    }

    @Override
    public void onItemClick(int position) {
        Intent browserIntent = new Intent(Intent.ACTION_VIEW,
Uri.parse(restaurantItemEntities.get(position).getURL()));
        startActivity(browserIntent);
```

```
        Intent intent = new Intent(context, RestaurantsActivity.class);

 return intent;
    }

    private void getRestaurantItems() {

        String uri = API_LINK_V2 + "get-all-restaurants/" + mCity.getLatitude() + "/" +
mCity.getLongitude();
        Log.v("executing", "URI : " + uri);

        //Set up client
        OkHttpClient client = new OkHttpClient();
        //Execute reques
                    for (int i = 0; i < array.length(); i++) {
                       JSONObject object = array.getJSONObject(i);
                       String imageUrl = object.getString("restaurant_image");
                       String name = object.getString("restaurant_name");
                       String address = object.getString("address");
                       String ratings = object.getString("aggregate_rating");
                       String votes = object.getString("votes");
                       String restaurantURL = object.getString("restaurant_url");
                       int avgCost = object.getInt("avg_cost_2");
                       restaurantItemEntities.add(
                             new RestaurantItemEntity(imageUrl, name, address, ratings,
                                   votes, avgCost, restaurantURL));

    animationView.setVisibility(View.GONE);
                    RestaurantsCardViewAdapter restaurantsCardViewAdapter
                         = new RestaurantsCardViewAdapter(RestaurantsActivity.this,
                         restaurantItemEntities, RestaurantsActivity.this);
                    RecyclerView.LayoutManager mLayoutManager = new
LinearLayoutManager(RestaurantsActivity.this);
                    mRestaurantsOptionsRecycleView.setLayoutManager(mLayoutManager);
                    mRestaurantsOptionsRecycleView.setItemAnimator(new DefaultItemAnimator());
                    mRestaurantsOptionsRecycleView.setAdapter(restaurantsCardViewAdapter);
                } catch (JSONException e) {
                    networkError();
                    e.printStackTrace();
                    Log.e("ERROR : ", "Message : " + e.getMessage());
                }    }); }
        });    }


    animationView.setVisibility(View.GONE);
                    RestaurantsCardViewAdapter restaurantsCardViewAdapter
                         = new RestaurantsCardViewAdapter(RestaurantsActivity.this,
                         restaurantItemEntities, RestaurantsActivity.this);
```

```
                RecyclerView.LayoutManager mLayoutManager = new
LinearLayoutManager(RestaurantsActivity.this);
                mRestaurantsOptionsRecycleView.setLayoutManager(mLayoutManager);
                mRestaurantsOptionsRecycleView.setItemAnimator(new DefaultItemAnimator());
                mRestaurantsOptionsRecycleView.setAdapter(restaurantsCardViewAdapter);
            } catch (JSONException e) {
                networkError();
                e.printStackTrace();
                Log.e("ERROR : ", "Message : " + e.getMessage());
            }   }); }
    });   }

import io.github.project_travel_mate.destinations.CityFragment;
import io.github.project_travel_mate.friend.FriendsProfileActivity;
import io.github.project_travel_mate.friend.MyFriendsFragment;
import io.github.project_travel_mate.login.LoginActivity;
import io.github.project_travel_mate.mytrips.MyTripsFragment;
import io.github.project_travel_mate.notifications.NotificationsActivity;
import io.github.project_travel_mate.travel.TravelFragment;
import io.github.project_travel_mate.utilities.AboutUsFragment;
import io.github.project_travel_mate.utilities.UtilitiesFragment;
import io.github.tonnyl.whatsnew.WhatsNew;
import io.github.tonnyl.whatsnew.item.WhatsNewItem;import android.view.MenuItem;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;

import com.github.juanlabrador.badgecounter.BadgeCounter;
import com.squareup.picasso.Picasso;

import io.github.project_travel_mate.destinations.CityFragment;
import io.github.project_travel_mate.friend.FriendsProfileActivity;
import io.github.project_travel_mate.friend.MyFriendsFragment;
import io.github.project_travel_mate.login.LoginActivity;
import io.github.project_travel_mate.mytrips.MyTripsFragment;
import io.github.project_travel_mate.notifications.NotificationsActivity;
import io.github.project_travel_mate.travel.TravelFragment;
import io.github.project_travel_mate.utilities.AboutUsFragment;
import io.github.project_travel_mate.utilities.UtilitiesFragment;
import io.github.tonnyl.whatsnew.WhatsNew;
import io.github.tonnyl.whatsnew.item.WhatsNewItem;
import okhttp3.Call;
import static utils.Constants.API_LINK_V2;
import static utils.Constants.SHARE_PROFILE_USER_ID_QUERY;
import static utils.Constants.USER_DATE_JOINED;
import static utils.Constants.USER_EMAIL;
import static utils.Constants.USER_ID;
import static utils.Constants.USER_IMAGE;
import static utils.Constants.USER_NAME;
```

```
import static utils.Constants.USER_STATUS;
import static utils.Constants.USER_TOKEN;
import static utils.DateUtils.getDate;
import static utils.DateUtils.rfc3339ToMills;
import static utils.WhatsNewStrings.WHATS_NEW1_TEXT;
import static utils.WhatsNewStrings.WHATS_NEW1_TITLE;

/**
 * Launcher Activity; Handles fragment changes;

    private SharedPreferences mSharedPreferences;
    private int mPreviousMenuItemId;
    private String mToken;
    private DrawerLayout mDrawer;
    private Handler mHandler;
    private static final String travelShortcut = "io.github.project_travel_mate.TravelShortcut";
    private static final String myTripsShortcut = "io.github.project_travel_mate.MyTripsShortcut";
    private static final String utilitiesShortcut = "io.github.project_travel_mate.UtilitiesShortcut";

        mSharedPreferences = PreferenceManager.getDefaultSharedPreferences(this);
        mToken = mSharedPreferences.getString(USER_TOKEN, null);
        mPreviousMenuItemId = R.id.nav_city; // This is default item
        mHandler = new Handler(Looper.getMainLooper());
        // To show what's new in our application
        WhatsNew whatsNew = WhatsNew.newInstance(
                new WhatsNewItem(WHATS_NEW1_TITLE, WHATS_NEW1_TEXT));
        whatsNew.setButtonBackg

round(ContextCompat.getColor(this, R.color.colorPrimaryDark));
        whatsNew.setButtonTextColor(ContextCompat.getColor(this, R.color.white));
        whatsNew.presentAutomatically(this);
        String action =  getIntent().getAction();
        if (Intent.ACTION_VIEW.equals(action)) {
            showProfile(getIntent().getDataString());
        }
        //Initially city fragment
        Fragment fragment;
        FragmentManager fragmentManager = getSupportFragmentManager();
        fragment = CityFragment.newInstance();
        fragmentManager.beginTransaction().replace(R.id.inc, fragment).commit();
        NavigationView navigationView = findViewById(R.id.nav_view);
        navigationView.getMenu().getItem(0).setChecked(true);
        getRuntimePermissions()
        mDrawer = findViewById(R.id.drawer_layout);
```

```
    String emailId = mSharedPreferences.getString(USER_EMAIL,
getString(R.string.app_name));
    fillNavigationView(emailId, null);

    getProfileInfo();
    if (getIntent() != null && getIntent().getAction() != null) {
       switch (getIntent().getAction()) {
          case travelShortcut:
             fragment = TravelFragment.newInstance();
             fragmentManager.beginTransaction().replace(R.id.inc, fragment).commit();
             break;
          case myTripsShortcut:
             fragment = MyTripsFragment.newInstance();
             fragmentManager.beginTransaction().replace(R.id.inc, fragment).commit();
             break;
          case utilitiesShortcut:
             fragment = UtilitiesFragment.newInstance();
             fragmentManager.beginTransaction().replace(R.id.inc, fragment).commit();
             break;        }
    }   }



    getProfileInfo();
    if (getIntent() != null && getIntent().getAction() != null) {
       switch (getIntent().getAction()) {
          case travelShortcut:
             fragment = TravelFragment.newInstance();
             fragmentManager.beginTransaction().replace(R.id.inc, fragment).commit();
             break;
          case myTripsShortcut:
             fragment = MyTripsFragment.newInstance();
             fragmentManager.beginTransaction().replace(R.id.inc, fragment).commit();
             break;
          case utilitiesShortcut:
             fragment = UtilitiesFragment.newInstance();
             fragmentManager.beginTransaction().replace(R.id.inc, fragment).commit();
             break;        }
    }   }
```

```
int id = item.getItemId();
      Fragment fragment = null;
      FragmentManager fragmentManager = getSupportFragmentManager();

      switch (id) {
         case R.id.nav_travel:
            fragment = TravelFragment.newInstance();
            break;

         case R.id.nav_mytrips:
            fragment = MyTripsFragment.newInstance();
            break;

         case R.id.nav_city:
            fragment = CityFragment.newInstance();
           break;

         case R.id.nav_utility:
            fragment = UtilitiesFragment.newInstance();
            break;
         case R.id.nav_about_us:
            fragment = AboutUsFragment.newInstance();
            break;
         case R.id.nav_signout: {
            //set AlertDialog before signout
            ContextThemeWrapper crt = new ContextThemeWrapper(this, R.style.AlertDialog);
            AlertDialog.Builder builder = new AlertDialog.Builder(crt);
            builder.setMessage(R.string.signout_message)
                  .setPositiveButton(R.string.positive_button,
                        (dialog, which) -> {
                           mSharedPreferences
                                 .edit()
                                 .putString(USER_TOKEN, null)
                                 .apply();
                           Intent i = LoginActivity.getStartIntent(MainActivity.this);
                           startActivity(i);
                           finish();
                        })
                  .setNegativeButton(android.R.string.cancel,
                        (dialog, which) -> {

                        });
            builder.create().show();
            break;
         }
         case R.id.nav_myfriends :
            fragment = MyFriendsFragment.newInstance();
```

```
            break;
        case R.id.nav_settings :
            fragment = SettingsFragment.newInstance();
            break;
    }

    if (fragment != null) {
        fragmentManager.beginTransaction().replace(R.id.inc, fragment).commit();
    }

    mDrawer.closeDrawer(GravityCompat.START);
    mPreviousMenuItemId = item.getItemId();
    return true;
}

private void getRuntimePermissions() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        if (ContextCompat.checkSelfPermission(MainActivity.this,
                Manifest.permission.CAMERA)
                != PackageManager.PERMISSION_GRANTED) {
            requestPermissions(new String[]{Manifest.permission.CAMERA,

            }, 0);
        }     }    }
public void onClickProfile(View view) {
    Intent in = ProfileActivity.getStartIntent(MainActivity.this);
    startActivity(in);
}

public static Intent getStartIntent(Context context) {
    Intent intent = new Intent(context, MainActivity.class);
    return intent;
}
private void fillNavigationView(String emailId, String imageURL) {

    // Get reference to the navigation view header and email textview
    View navigationHeader = navigationView.getHeaderView(0);
    TextView emailTextView = navigationHeader.findViewById(R.id.email);
    emailTextView.setText(emailId);

    ImageView imageView = navigationHeader.findViewById(R.id.image);
    Picasso.with(MainActivity.this).load(imageURL).placeholder(R.drawable.icon_profile)
            .error(R.drawable.icon_profile).into(imageView
```

Продолжение приложения Б

```
    // to fetch user details
    String uri = API_LINK_V2 + "get-user";
```

```
        Log.v("EXECUTING", uri);

        //Set up client
        OkHttpClient client = new OkHttpClient();
        //Execute request
        Request request = new Request.Builder()
                .header("Authorization", "Token " + mToken)
                .url(uri)
                .build();{
                final String res = Objects.requireNonNull(response.body()).string();
                mHandler.post(() -> {
                   try {
                      JSONObject object = new JSONObject(res);
                      String userName = object.getString("username");
                      String firstName = object.getString("first_name");
                      String lastName = object.getString("last_name");
                      String status = object.getString("status");
                      int id = object.getInt("id");
                      String imageURL = object.getString("image");
                      String dateJoined = object.getString("date_joined");
                      Long dateTime = rfc3339ToMills(dateJoined);
                      String date = getDate(dateTime);

String fullName = firstName + " " + lastName;
                      mSharedPreferences.edit().putString(USER_NAME, fullName).apply();
                      mSharedPreferences.edit().putString(USER_EMAIL, userName).apply();
                      mSharedPreferences.edit().putString(USER_DATE_JOINED, date).apply();
                      mSharedPreferences.edit().putString(USER_IMAGE, imageURL).apply();
                      mSharedPreferences.edit().putString(USER_STATUS, status).apply();
                      mSharedPreferences.edit().putString(USER_ID, String.valueOf(id)).apply();
                      fillNavigationView(fullName, imageURL);
                   } catch (JSONException e) {
                      e.printStackTrace();
                      Log.e("ERROR : ", "Message : " + e.getMessage())  }
                });

    }

    @Override
    protected void onResume() {
        super.onResume();
        fillNavigationView(mSharedPreferences.getString(USER_NAME,
getString(R.string.app_name)),
                mSharedPreferences.getString(USER_IMAGE, null));
        invalidateOptionsMenu();
    }
```

```
  void showProfile(String data) {
     Uri uri = Uri.parse(data);
     String userId = uri.getQueryParameter(SHARE_PROFILE_USER_ID_QUERY);
     String myId = mSharedPreferences.getString(USER_ID, null);
     Log.v("user id", userId + " " + myId);
     if (userId != null) {
        int id = Integer.parseInt(userId);
        if (!userId.equals(myId)) {
           Intent intent = FriendsProfileActivity.getStartIntent(MainActivity.this, id);
           startActivity(intent);
        } else {
           Intent intent = ProfileActivity.getStartIntent(MainActivity.this);
           startActivity(intent);
        }
     }
  }



public void updateNotificationsCount(Menu menu) {
     String uri;
     uri = API_LINK_V2 + "number-of-unread-notifications";
     Log.v("EXECUTING", uri);

     //Set up client
     OkHttpClient client = new OkHttpClient();
     //Execute request
     Request request = new Request.Builder()
           .header("Authorization", "Token " + mToken)
           .url(uri)
           .build();


        @Override
        public void onResponse(Call call, final Response response) throws IOException {
           final String res = Objects.requireNonNull(response.body()).string();

           mHandler.post(() -> {
              try {
                 JSONObject object = new JSONObject(res);
                 int mNotificationCount = object.getInt("number_of_unread_notifications");
                 if (mNotificationCount > 0) {
                    BadgeCounter.update(MainActivity.this,
                          menu.findItem(R.id.action_notification),
                          R.drawable.ic_notifications_white,
                          BadgeCounter.BadgeColor.RED,
                          mNotificationCount);
                 }
```

```
                } catch (JSONException e) {
                    e.printStackTrace();
                    Log.e("ERROR : ", "Message : " + e.getMessage());
                }
            });
                }
            } catch (JSONException e) {
                e.printStackTrace();
                Log.e("ERROR : ", "Message : " + e.getMessage());
            }
            })
            Intent intent = NotificationsActivity.getStartIntent(MainActivity.this);
            startActivity(intent);
            return true;

        return super.onOptionsItemSelected(item);
    }
}


import okhttp3.Request;
import okhttp3.Response;
import utils.TravelmateSnackbars;

import static utils.Constants.API_LINK_V2;
import static utils.Constants.READ_NOTIF_STATUS;
import static utils.Constants.USER_TOKEN;

public class NotificationsActivity extends AppCompatActivity implements
SwipeRefreshLayout.OnRefreshListener,
        TravelmateSnackbars {

    @BindView(R.id.animation_view)
    LottieAnimationView animationView;
    @BindView(R.id.notification_list)
    ListView listView;
    @BindView(R.id.swipe_refresh)
    SwipeRefreshLayout swipeRefreshLayout
    private String mToken;
    private Handler mHandler;
    ArrayList<Notification> notifications;
    private NotificationsAdapter mAdapter;
    private MaterialDialog mDialog;
    private Menu mOptionsMenu;
    boolean allRead = false;
    private static final int SECOND_MILLIS = 1000;
    private static final int MINUTE_MILLIS = 60 * SECOND_MILLIS;
    private static final int HOUR_MILLIS = 60 * MINUTE_MILLIS;
```

Приложение В
(обязательное)

Акты внедрения

dasm/ms
in affiliation with MINDSHARE

Исх. № 037 «13» мая **20**19**г**

Акт внедрения мобильного приложения на ОС Android «Travel Mate»

Настоящий акт свидетельствует, что мобильное приложение «Travel Mate», разработанное Савиной Т.Е., внедрен в ТОО «DASM/MS».

Процесс внедрения проходил с 6 мая по 10 мая 2019 года.

Заявленные характеристики системы предполагали наличие следующих функциональных возможностей:

- авторизация пользователя;
- список городов с подробной информацией;
- привязка Google карт к мобильному приложению;
- способность пользователя планировать путешествия;
- адаптивность мобильного приложение.

В ходе опытной эксплуатации программного обеспечения подтверждено, что оно обладает всеми заявленными возможностями.

На момент подписания настоящего Акта система установлена и эксплуатируется на мобильном телефоне марки Android.

Директор
ТОО «DASM/MS» **Пак Ю.В.**

050010, Республика Казахстан, г. Алматы, Айтеке би, 34/29, 5 этаж, Тел/факс: + 7 (727) 313 13 91, 315 16 91
050010, Republic of Kazakhstan, 34/29 Aitike bi str., Almaty, Tel/fax: + 7 (727) 313 13 91, 315 16 91