МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

кафедра Компьютерных технологий

«Допущен к защите»
Заведующий кафедрой Куралбаев З. К.
д. ф-м. н., проф
(Ф.И.О., ученая степень, звание)
_____ «____» _____ **20__ г.**
(подпись)

ДИПЛОМНЫЙ ПРОЕКТ

На тему: "Разработка мобильного приложения прогноза футбольных ставок"

Специальность 5В070400 ВТиПО

Выполнил (а) Мингазов Э. Р. ПО-12-2
(Фамилия и инициалы) группа

Научный руководитель Зубкір С.Б. доцент, к.ф.-м.н.
(Фамилия и инициалы, ученая степень, звание)

Консультанты:

по экономической части:
Бекишева А.Ч., к.э.н., доцент
(Фамилия и инициалы, ученая степень, звание)
_____ «05» 05 20 16 г.
(подпись)

по безопасности жизнедеятельности:
Приходько М.Г. д.х.н., профессор
(Фамилия и инициалы, ученая степень, звание)
_____ «11» 04 20 16 г.
(подпись)

по применению вычислительной техники:
Зубкір С.Б., к.ф.-м.н., доцент
(Фамилия и инициалы, ученая степень, звание)
_____ «31» 05 20 16 г.
(подпись)

Нормоконтролер: Зубкір С.Б., к.ф.-м.н., доцент
(Фамилия и инициалы, ученая степень, звание)
_____ «31» 05 20 16 г.
(подпись)

Рецензент: Сейлова И.А., к.п.н.,
(Фамилия и инициалы, ученая степень, звание)
_____ «_____» _____ 20__ г.
(подпись)

Алматы 2016 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Факультет Аэрокосмических и информационных технологий
Специальность Вычислительная техника и программное обеспечение
Кафедра Компьютерных технологий

ЗАДАНИЕ
на выполнение дипломного проекта

Студент Миназов Эльдар Рафаильевич
(фамилия, имя, отчество)

Тема проекта „Разработка мобильного приложения прогноза футбольных ставок"

утверждена приказом ректора № 21 от «10» марта 2016 г.
Срок сдачи законченной работы «___» _____ 20___ г.
Исходные данные к проекту требуемые параметры результатов проектирования (исследования) и исходные данные объекта
Разработка структуры мобильного приложения прогноза футбольных ставок

Перечень подлежащих разработке дипломного проекта вопросов или краткое содержание дипломного проекта:
1. Изучение предметной области.
2. Инструменты и технологии разработки
3. Разработка мобильного приложения
4. Технико-экономическое обоснование проекта
5. Безопасность жизнедеятельности.

Перечень графического материала (с точным указанием обязательных чертежей)

Рисунки структуры приложения
Рисунки инструментов интерфейса разработки
Рисунки описывающие требования к приложению
Рисунки интерфейса приложения
Таблицы экономического обоснования
Рисунки и описывающие безопасность жизнедеятельности

Рекомендуемая основная литература

Сайт http://developer.android.com
Книга „Android. Программирование приложений для планшетных компьютеров и смартфонов" Рето Майер, Эксмо, 2011
Саид Хашими, Сатия Команиени, Дэйв Маклин „Разработка приложений для Android"

Консультанты по проекту с указанием относящихся к ним разделов

Раздел	Консультант	Сроки	Подпись
БЗЖД	Приходько И.Г.	01.04.–21.04.16	
Экономич часть	Бекетнова А.И.	17.03 – 05.05.16	
Исслед. пред. области	Дубкін С.Б.	24.02.16	
Инстр. и техн. разработки	Дубкін С.Б.	08.03.16	
Разраб. мобильн. прилож.	Дубкін С.Б.	15.04.16	
Нормаконтроль	Дубкін С.Б.	31.05.16.	

Г Р А Ф И К
подготовки дипломного проекта

№ п/п	Наименование разделов, перечень разрабатываемых вопросов	Сроки представления руководителю	Примечание
1.	Исследование предметной области	24.02.16	
2.	Инструменты и технологии разработки	08.03.16	
3.	Разработка мобильного приложения	15.04.16	
4.	Технико-экономическое обоснование проекта	05.05.16	
5.	Безопасность жизнедеятельности	21.04.16.	

Дата выдачи задания «14» сентября 2015 г.

Заведующий кафедрой _____ Курмалбаев З.К.
(подпись) (Фамилия и инициалы)

Руководитель _____ Дубжіра С.Б.
(подпись) (Фамилия и инициалы)

Задание принял к исполнению
студент _____ Мингазов Э.Р.
(подпись) (Фамилия и инициалы)

Аңдатпа

Дипломдық жұмыста негізгі мақсаты Android ОЖ платформасындағы мультимедиялық құрылғыларға арналған мобильдік қосымша жобаланып, құрылды.

Аталған мақсатты орындау үшін әртүрлі дерек көздерден алынатын көптеген деректерді жобалайтын дерекқор мен білімді ұсыну үлгісі жасалды. Деректер қоры мен әдістерін зерттеу міндеттері мен білім үлгісін таңдау нәтижесінде оларды байланыстыратын интерфейске арналған алгоритм құрылды.

Ақпараттық технологиялады талдау нәтижесінде жобаны жасайтын құралдар есебінде Android Studio IDE, Java программалау тілі таңдалып алынды.

Аннотация

Данная дипломная работа создана и разработана для мультимедийных устройств под управлением операционной системы ОС Android.

Проектирование базы данных, базы знаний ориентировано на получение максимального количества информации с разных источников. Для связи базы данных и базы знаний был разработан алгоритм создания интерфейса.

На основе анализа информационных технологий были выбраны технологии и инструменты разработки, как Android Studio IDE, язык программирования Java.

Annotation

This thesis work created and developed for multimedia devices running Android system operations.

Database design knowledge base is aimed at obtaining the maximum amount of information from different sources. An algorithm of interface creation was developed for connecting the data and knowledge base.

The special research of subject area was conducted and selected the technologies and tools of development such as Android Studio, Java programming language and so on.

Содержание

Введение

В наши дни, разработка мобильных приложений является очень актуальным видом деятельности среди программистов, так как появляются все больше новых устройств с различными операционными системами и возможностями, а это значит большие перспективы для внедрения в разные виды бизнеса. Так же немаловажным фактором является то, что существует множество технологий разработки, которые упрощают создание программных продуктов ориентированных под популярные операционные системы Android, IOS. Эти инструменты постоянно развиваются и усовершенствуются за счет поддержки таких компаний-гигантов как Google, Apple и других.

Увеличение количество букмекерских контор и возрастание популярности ставок в области спорта также является актуальным. Сфера ставок у букмекеров растет и затрагивает большое количество людей в нашей стране. Аналитики и эксперты составляют прогнозы и делятся мнением и взглядом на матч, игроки ищут информацию, совещаются и делают свои ставки.

Высокий показатель пользования смартфонов и мобильных ПК говорит о том, что все чаще люди пользуются гаджетами и регулярно устанавливают приложения для них. Человек в любой момент может воспользоваться смартфоном, это очень удобно и вся необходимая информация всегда под рукой. На этом основании было принято решение разработать мобильное приложение.

После проведения анализа рынка в сфере ставок мы сделали вывод, что наш продукт заинтересует людей, занимающихся прогнозированием спортивных событий. Для качественного прогноза необходимо владеть всей необходимой информацией. Под необходимой информацией мы подразумеваем: мнение экспертов, последние новости в мире спорта, статистика. Так же при разработке мы учли различные стратегии игры в букмекерских конторах. На основе этих факторов мы разработали приложение, аналогов которого в нашей стране пока не существует.

Неотъемлемой и необходимой частью любого приложения является интерфейс пользователя. Он создается для графического вывода информации на экран. Благодаря ему пользователь может совершать те или иные действия, нажимая на нужные вкладки кнопки и другие графические элементы. Именно интерфейс отражает весь функционал разрабатываемого приложения. Интерфейс в нашем приложении связывает базу данных с базой знаний, а также создает и описывает каждый элемент приложения. Перед разработчиком интерфейса стоит цель создать легкий для восприятия дизайн. Удобное перемещение по разным фрагментам и, конечно же, постараться оптимизировать скорость реагирования приложения.

1. Исследование предметной области

1.1 Модели и методы прогнозирования

Ставки на спортивные события приобрели большую популярность в последнее время. Букмекерские конторы теперь на каждом шагу Один из самых популярных видов ставок это ставки на футбол, потому что сотни футбольных матчей играют каждую неделю. Метод всегда тот же, человек ставит какое-то количество денег на ставки по своему выбору. Если его предсказание оказывается верным, он выигрывают некоторую сумму денег, равную шансу на выигрыш. Если прогноз неправильный то ставка проигрывает.

Букмекеры выставляют коэффициенты, основываясь на вероятности выигрыша какого либо события. Очевидно, что букмекеры бы сделали немного прибыли, если бы высчитывали, правильную вероятность некоторого события. Из-за этих "недобросовестных" букмекеров, делать прибыль сделав ставку на футболе не так просто. Много исследований уже сделано по этой теме. В 2003 году, исследовали коэффициенты букмекеров во всех видах спорта, в том числе футбола. Они пришли к выводу, что, когда у вас есть достаточное количество данных для прогнозирования, или если даже у вас есть небольшой набор функций то будет достаточно, чтобы побить шансы букмекеров. С большим и более интеллектуальным набором функций, можно предсказать футбольные матчи даже более точно и, следовательно, получить более высокую прибыль. В 2004 компания Goddard, использовала гораздо больший набор функций для своего футбольного прогноза [1].

Этот набор функций включал в себя географическое расстояние между командами и их среднего зрителя, то есть посещаемость. Мы считаем, что основной причиной этого является то, что они просто ставили деньги на самый вероятный исход каждого матча. Разумнее хотя это может показаться, не так уж трудно, что, из-за несправедливости букмекеров, иногда может быть лучше не делать ставку на матч вообще. В нашем исследовании будет предпринята попытка точно выбрать, какие ставки взять на себя, чтобы максимизировать прибыль.

Прогнозы, которые мы рассматриваем, могут поступать из трех источников. Во-первых, существует рынок. Во-вторых, прогнозы могут быть получены из статистических моделей, которые строятся на основах спорта или основаны на переменных.

Для каждого вида спорта, наибольшее количество прогнозов приходят с рынка ставок. Прогноз рынка показывает окончательные шансы на то, что команда выиграет или проиграет. Выше было указано, эти прогнозы могут быть проанализированы несколькими различными способами, в зависимости от этого, букмекеры выставляют коэффициенты на форы, гандикапы и так далее.

Рисунок 1.1 – Методы и модели прогнозирования

Для того чтобы предсказать исход спортивных событий, могут быть построены различные типы моделей. На самом детализированном уровне, можно предсказать исход игры, моделирование последствий каждой игры. На более высоком уровне, используются производственные функции.
Эти функции сосредоточены на основных факторах, определяющих исход игры. Модель, которая используется для прогнозирования
исхода игры будет основан на различиях в основных характеристиках двух команд.

Альтернативная статистическая процедура является построением счета мощности или индекса, который является доверенным фактором этих фундаментальных характеристик или скрытых навыков и сильных команд. Такая модель просто использует разницу в командах - оценивается как предсказатель. Сейчас эти модели центре внимания исключительно от относительного числа (и полей) побед конкурирующих команды и время тенденции в этих отношениях. Например, Нью-Йорк Таймс создал группу
оценки для каждой команды НФЛ, обобщая относительную производительность каждой команды в предыдущих играх.

Она была основана на проценте выигрыша каждой команды, ее конечной победы, и высокому качеству продуктивности противника. Аналогичные меры, которые включают в себя графы, были построены для других видов спорта.

Наконец, у нас есть прогнозы, сделанные отдельными лицами (экспертами), можно выявить их методы. Некоторые из этих специалистов являются спортивными писателями, редакторами газет, или спортивными комментаторами.

Прогнозы на ставки профессиональных футбольных игр, как и в других видах спорта, приходят из ставок рынка, статистических систем, а также экспертов. В отличие от скачек и бейсбола, где коэффициенты используются при установлении выигрыша на ставку, которая включает в себя выбор выигрышной лошади или команды, футбольные ставки не связаны с выбором команды на победу. Скорее выбор находится между тем чтобы поставить на тотал или фору.

Что необходимо воспользоваться, прежде, чем создать прогноз на какое-либо событие какого-либо чемпионата:

В первую очередь детально узнать местоположение команд в турнирной таблице в этом сезоне. Найти цели, поставленные начальством каждого клуба, перед своими игроками. Посмотреть игровую активность, предшествующую начальному этапу чемпионата и соотнести ее с задачами команд (Если начальство клуба-середнячка говорит, например, что хочет в ЛЧ или ЛЕ, а само не купило ни одного, сколь значимого игрока, то это можно расценивать как неудачный пиар-ход).

Необходим мониторинг динамики команды в прошлых сезонах. Оценить стабильность и будущие риски (Если даже омский «ГазКвас» феррит в преддверии сезона, а в прошлых годах показывал плохую игру – это тоже опасный звоночек).

Такой «проработке» необходимо подвергнуть абсолютно все команды в турнире. Это позволит сформулировать, еще достаточно туманное, но все же прогноз о «расстановке сил» на футбольном поле отдельно взятой страны.

Необходимо составление прогнозов на высшие лиги (минимум – первая лига) государств, с хорошим футболом и инфраструктурой для этого вида спорта. Составлять прогнозы на матчи 5-ой лиги Восточного дивизиона Челябинского региона – дело неблагодарное и убыточное. Команды из последних лиг, как обычно, не имеют необходимой материальной и технической базы, квалифицированного персонала и повлиять на результативность игры может большое количество субъективных факторов (кому то зарплату не дали и он обиделся, а кто-то уехал к дяде на именины).

Итак, переходим к основному аспекту. Когда вроде бы понятно: какая команда что из себя представляет, какие имеет задачи и что в ней за игроки, можно приступать к прогнозированию. Хочется выделить, что составлять прогнозы можно на любые команды за исключением нескольких случаев:

Не стоит брать начальные и конечные туры (а лучше проигнорировать по 2-3 круга в начале и конце сезона). В старте сезона большинство команд входят в ритм, ну а в конце, что понятно, те клубы, которые выполнили тренерскую цель (выиграли досрочно чемпионство, завоевали трофей, гарантировали себе европейские лиги и т.д.) играют «на каникулы».

Не рекомендуется прогнозировать матчи любимых команд, так как субъективные суждения, в обязательном порядке, будут брать верх над объективными факторами.

Большинство могут высказаться, что это очень трудное и скучное занятие [2]. Сутки напролет следить, слушать «бубнеж» экспертов, документировать кучу цифр, запоминать статистику и анализировать. Проще сделать ставку на фаворита и «снять сливки». На все это есть одно объяснение: вы ставите личные деньги, и если вам хочется выигрывать, рекомендуется, как ни странно, приложить все свои усилия; прогноз на фаворита с маленьким коэффициентом неизбежно, рано или поздно, приводит к тотальному проигрышу. Лучше уделить время на прогнозирование, но быть в выигрыше за счет высокой проходимости и не больших коэффициентов. Для людей, которые в хороших отношениях с математикой, можно настоятельно рекомендовать использовать методы математической статистики и моделирования (например, формулу Байеса) для осуществления ставок. Статистика и ум, вкупе с аналитикой и интуицией дают большие результаты.

Метод предсказания спортивного события с помощью теории нечетких множеств. Теория нечётких множеств — это один из разделов всеми любимой математики, посвященный методологиям аналитики неограниченных данных, в которых описание этих неопределенностей некоторых явлений и процессов используется с помощью понятия о множествах, не имеющих конкретных границ.

Впервые эти разработки обнаружились в 1965 г, и с прогрессом вычислительной техники, начали использоваться с вопросами финансового аналитики и прогнозирования. Было создано объемное программное обеспечение, которое позволило решать тяжелые вопросы. Однако, ради справедливости, стоит отметить, что и по наши дни к работам этой теории эксперты относятся с большой осторожностью. Примером применения этой теории может служить система прогнозирования ставок футбольных матчей, созданная в Винницком государственном техническом университет доктором технических наук Шторбой С.Д. и студентом 5-го курса.

Для правильной настройки и тестирования теории использовались данные о событиях чемпионатов Украины по футболу 2001 -2003 годов.

Если погружаться в научные дебри, то можно потратить кучу времени, чтобы разобраться в терминах, сложение множеств, гауссовские модели и так далее.

1.2 Исследование букмекерских контор

Для открытия букмекерской фирмы либо тотализатора нужно привнести лицензионный сбор в объеме 639 МРП (примерно 1 257 475 тенге). Кроме данного, необходимо существование снабжения в виде непременных запасов: для букмекерских компаний – 20 211 МРП (38 612 000 тенге), для тотализаторов – 6 101 МРП (9 930 133 тенге). Ежемесячная оплата налога

составляет: для кассы букмекерской фирмы – 78 МРП (123 855 тенге), для кассы тотализатора – 134 МРП (223 567 тенге).
Годичный малый кругооборот работающих букмекерских компаний, сообразно информации официальных органов, 30 млрд. тенге. При этом, чистая прибыль равна 11-14%.Налоговые же поступления в бюджет от их деловитости с 2007 по 2013 год составили 3,2 млрд. тенге (к образцу, ежели в 2004-м – 43,8 миллиона тенге, то в 2013-м возросли до 1 500 млн. тенге). Для сопоставления: вместимость предоставленного рынка только в мире – $650 млрд., в USA – $240 млрд., в РФ – $500 млн.
Обзор букмекерских контор:

– Букмекерская фирма «Олимп» создана в 2004 году в Казахстане. С 2011 года ее конторы приема ставок функционируют в Российской Федерации согласно лицензии №26, выданной Федеральной налоговой службой 26 ноября 2011 года. Еще много пунктов приема ставок функционируют на отчизне букмекера, которые действуют согласно лицензии Министерства спорта и туризма Казахстана. Еще рассказывается о наличии лицензий Беларуси, Украины и Таджикистана.Интернет-букмекер olimp.com воспринимает ставки согласно лицензии Западно-Капской периферии ЮАР. Активность букмекерской фирмы «Олимп» нацелена на Казахстан, Российскую Федерацию и остальные державы постсоветского пространства. Интернет-сайт переведен на некоторое количество языков, так же на русский. Пополнить счет и заключать средства, разрешено хоть какими обычными платежными системами и чрез терминалы. Скидки отсутствуют. В Казахстане у фирмы выстроена широкая сеть, она обхватывает фактически все города. В Алматы – 123 «точек», Астане – 98, Шымкенте – 42 и т.д. Что интересно, «Олимп» деятельно сотрудничает с футбольной студией телеканала НТВ, в каком месте водящие специалисты канала водят передачу, спонсируемую «Олимпом». Есть веб-версия на казахском, русском, английском, китайском, итальянском и испанском языках. На веб-сайте кроме ставок есть сумматор, разные виды виртуального спорта, денежные ставки и т.д.

В Казахстане букмекерская контора действует согласно лицензии, она была выдана в 2007 году комитетом промышленности туризма министерства туризма и спорта РК. Срок ее кончается в 2017 году. В ЮАР она работает согласно лицензии комитета отвечающего за развлечения и азартные игры Западной Капской периферии.

Рисунок 1.2 - Официальные лицензии БК «Олимп»:

БК «Олимп» дает довольно широкую линию ставок. Пользователь имеет возможность поставить на 16 разных видов спорта. Особенный интерес уделен традиционному футболу и регби. Сказать, что коэффициенты занижаются невозможно. Самая маленькая ставка – 10 руб. Маржа фирмы ориентируется на обычный уровень. В целом, коэффициенты “Олимп” слабо различаются от коэффициентов, предлагаемых иными букмекерами из СНГ. Зрительно, линия “Олимпа” смотрится немного неполноценно и неинформативно, однако неопытных людей навряд ли заинтересует эта информация. Росписи традиционны и никак не различаются каким-то инноваторским дизайном. В вашем распоряжении классические ставки на тоталы, и форы [3].

Кроме обыденных букмекерских ставок, людям доступны и наиболее экзотичные ставки. Например, здесь работает таковая увлекательная игра как “Футбольная рулетка”. Вы сможете испытать себя на удачливость, какое явление в ходе матча случится быстрее всех: пенальти, гол либо что-то иное. Победить тут сложно, но, несмотря на все вышесказанное деньги, вы получаете мгновенно, спустя 5 мин. после сделанной ставки. Еще на веб-сайте разрешено сыграть в “Условный футбол”. Наверное это что-то вроде футбольного симулятора. Условия в симуляторе очень приближены к настоящим, с учетом всех вероятных характеристик. В виртуальном футболе исключена вероятность нечестного исхода события либо договорных матчей, потому победить возможно исходя из личного эксперимента и кропотливого разбора игровых критерий.

На веб-сайте существуют live-ставки. Олимп дает вам шанс поставить средства на конкретные действия во время матча. Вы можете воспользоваться компьютером прямо в букмекерской конторе , на котором в режиме настоящего времени показывается ход матча. Еще одной увлекательной особенностью данной фирмы считается то, что тут возможно совершать ставки на виртуальные деньги, элементарно для того, чтобы испытать собственную везучесть.

В Букмекерской Фирме «Олимп» имеется особые предписания, которые предоставляют вероятность совершать экспресс-ставки сообразно завышенным коэффициентам. Еще имеется бонус 5% ко всем экспрессам с коэффициентом более 2,5, ежели сумма ставки никак не превосходит 15 тыс руб.. Бонус данный прибавляется автоматически.

– «ProfitBet». Эта букмекерская контора считается молодым и многообещающим проектом. Фирма показывает игровую активность с июня 2011 года. Лицензия была выдана министерством спорта и туризма в Казахстане, номер лицензии №ТИН 0000065.

Эта букмекерская контора считается единым лицензированным букмекером в Казахстане. Профит передает спортивные события в Live - режиме. Контора предоставляет свои сервисы как в интернете, так и в офлайн режиме. В Казахстане имеется более 45 касс принимающих ставки. Кассы этой фирмы есть в Астане, Атырау, Усть-Каменогорске, Кызылорде,

Петропавловске, Таразе, Алматы. Однако, энтузиазм основной массы людей которые делают ставки прикован к интернет ставкам.

Букмекерская контора предоставляет возможность пользователям интернет-сайта, выбрать один из трех языков: русский, казахский, английский. Интернет-сайт выглядит в очень броском стиле, на сайте преобладают черный и красный цвета, будто это выделяет букмекера на фоне соперников, использующих дизайн приятнее. Невозможно никак не подметить перегруженность интернет-сайта www.profit.kz графикой, создается впечатление, что это отвлекает пользователя от принципиальной информации и воздействует на скорость загрузки страниц. Однако, не смотря ни на что, ощущение о ресурсе остаётся позитивным. Букмекерская контора Profitbet придумала интернет-сайт гораздо лучше, нежели остальные казахстанские букмекеры такого рода деятельности.

У ресурса лучшая навигация: кнопки, меню и остальные составляющие размещены, традиционным образом для букмекерского интернет-сайта. В следствии этого новые пользователи быстро осваиваются.

«ProfitBet» дает не самую широкую линию, однако она фактически никак не уступает соперникам. В линии есть место всем главным видам спорта, имеется “экзотика” вроде сёрфинга, сквоша. Также, разрешено совершать ставки на: компьютерные игры, лотереи. Однако, основной интерес уделяется спорту. Здесь никак не обделены и приверженцы тотализаторов. Тут разрешено играть на пятнадцати разновидностях. В целом, линия букмекера прилична, и полностью способна удовлетворить необходимости пользователя.

Сбросить выигранные средства разрешено чрез ту же систему, через которую средства были введены на счёт. Если счёт пополнялся разными методами, то начальство «ProfitBet» оставляет за собой преимущество выбора системы для перевода ваших средств. Пополнить счёт либо вывести прибыль разрешено с поддержкою следующих платёжных систем: Webmoney, Moneta, Moneybookers. На веб-сайте разрешено раскрыть счёт в руб., евро и гривнах.

– «Tennisi». До 2010 года букмекерская фирма «Tennisi», вступала в пятерку самых масштабных фирм РФ. После того как провели лицензирование, из ста мест приема ставок на нынешний день в стране действует меньше десяти. Что на это повлияло сказать трудно, однако невзирая ни на что, эта букмекерская контора, которая считается членом Государственной Ассоциации Букмекеров, владеет высочайшим престижем и удачно продолжает давать высококачественные сервисы приема ставок на спорт. Но сейчас в основном упор идет на веб.

Линия букмекерской фирмы Tennisi дает шанс делать ставки как на знаменитые виды спорта: футбол, баскетбол, гандбол, бокс, теннис, бейсбол, волейбол. Так и на наименее знамениты: гандбол, шахматы, бильярд, регби. Футбольная линия разнородна. Для всех матчей, вне зависимости от их значимости, раскрывается несколько линий росписи, но позиции матчей, на которые ставки никак не принимаются, отмечаются коэффициентом 0.0.

Размер коэффициентов не совсем высокий – 1.85%.
Очень многообразна и различна у БК Tennisi баскетбольная линия. Незначительно меньше теннисная роспись, однако и она охватывает наиболее пяти разновидностей для ставок. Габариты ставок находятся в зависимости не только лишь от видов спорта и значимости турниров, однако и от всякого отдельно взятого матча.

Пополнение счета производится через смс-платежи, терминалы «Элекснет», а также через специальные системы платежа QIWI(без процентов) и WebMoney(комиссия 0,9%) с их помощью можно осуществить и вывод денег.

1.3 Операционная система Android

1.3.1 История создания OS Android

Android Inc была основана в Пало-Альто, штат Калифорния в октябре 2003 Ранние намерения компании состояли из того, чтобы разработать современную операционную система для цифровых камер. Хотя, когда стало ясно, что рынок устройств не был достаточно большим, компания перенаправила свои усилия на производство операционной системы для смартфонов, которая будет конкурировать с Symbian и Microsoft Windows Mobile. Несмотря на прошлые достижения основателей и первых сотрудников , Android Inc. работали тайно, показывая только то, что они работают над программным обеспечением для мобильных телефонов[4].

В июле 2005 года Google приобрела Android Inc. за 50 миллионов долларов и ключевых сотрудников. Рубин, Минера и Уайт – ключевые сотрудники остались в компании после приобретения. Не так много было известно о Android Inc. на время, но многие предполагали, что Google планирует выйти на рынок мобильных телефонов. В Google, команда во главе с Рубином разработали платформу для мобильных устройств, которая много чего берет от ядра Linux. Google выпустила на рынок платформу для производителей телефонов и операторов, пообещав им гибкое программное обеспечение, обновляемой системы [511].

Google выпустила линию компонентов, аппаратного и программного обеспечения для партнеров. Слухи о намерении компании Google выйти на рынок мобильной связи продолжали строить до конца декабря 2006 года. Ранее прототип под кодовым названием "Sooner". Который имел более близкое сходство с телефоном BlackBerry, без сенсорного экрана, и физической, QWERTY клавиатурой, но позже модернизирована для поддержки сенсорного экрана, чтобы конкурировать с другими анонсированными устройствами, такими как 2006 LG Prada и в 2007 Apple iPhone.

В сентябре 2007 года InformationWeek раскрыл исследование Evalueserve отчетности, что Google подала несколько патентных заявок в области мобильной телефонии.

Эрик Шмидт, Энди Рубин и Уго Барра в 2012 на пресс-конференции объявили Nexus 7 планшетом от Google. Пятого ноября 2007 года Open Handset Alliance - консорциум технологических компаний, включая Google, производители устройств, таких как HTC, Sony и Samsung, представили себя, с целью разработки открытых стандартов для мобильных устройств. В тот же день, Android был представлен в качестве своего первого продукта. Мобильную платформу устройства была построенной на ядре Linux. первый коммерчески доступный смартфон под управлением ОС Android был HTC Dream, выпущенный 22 октября 2008 года.

С 2008 года Android показал многочисленные обновления, которые постепенно улучшили операционную систему, добавляя новые функции и исправляя ошибки в предыдущих версиях. Каждый крупный релиз назван в алфавитном порядке в виде десерта, или какого либо другого угощения. В 2010 году Google запустил серию Nexus устройств - линию смартфонов и планшетов, работающих под управлением операционной системы Android, и построенную за счет производственных партнеров.

HTC сотрудничало с Google, чтобы выпустить первый смартфон Nexus. Google с тех пор обновил серию новых устройств, таких как Nexus 5 (производства LG) и Nexus 7 таблетки (производства Asus). Google выпускает Nexus телефоны и планшеты, в качестве своих флагманских устройств Android, демонстрируя новейшие программные и аппаратные функции Android.

С 2013 до 2015 года, компания Google предложила несколько устройств Google Play Edition, для Google Play. С 2010 по 2013 год, Hugo Barra служил в качестве представителя продукта, представляя Android на пресс-конференциях Google I / O, годовой разработчиков ориентированной конференции Google. В 2013 году Барра покинул Android команду ради китайских смартфонов Xiaomi. В этом же году, объявили что Энди Рубин переехал из проекта Android, чтобы взять на себя новые проекты в Google. Он был заменен Сундаром Пичаи, который стал новым главой Android и Chrome OS. В 2014 году Google запустила Android One, линейку смартфонов в основном, для клиентов в развивающихся странах.

В мае 2015 года Google объявил Project BRILLO как версия урезанная Android, которая использует его более низкие уровни (за исключением пользовательского интерфейса), предназначенные для "Веб-технологий" и встраиваемых систем. Пользовательский интерфейс по умолчанию, в основном базируется на прямых манипуляциях, с помощью сенсорных входов, которые свободно соответствуют реальным действиям, как например если вы сильно ударите, или коснетесь.

Теперь если мы перейдем к интерфейсу то увидим: в верхней части экрана находится строка состояния, показывающая информацию об

устройстве и его подключениях. Это строка состояния, может быть «вытянутая» вниз, чтобы открыть экран уведомлений, где приложение отображает важную информацию или обновления. Например, если вы недавно получили по электронной почте или SMS. Уведомления остаются там, пока вы не прочтете их, нажав на письмо. Начиная с Android 4.1, "Выплывающие уведомления" могут отображать расширенные данные или дополнительные функциональные возможности; например, музыкальный проигрыватель может отображать элементы управления воспроизведением, а уведомление "пропущенный вызов" содержит кнопки для выполнения обратного вызова или отправки вызывающему абоненту сообщение SMS.

1.3.2 Material Design для OS Android

Material Design представляет собой комплексную концепцию создания визуальных, движущихся и интерактивных элементов для различных платформ и устройств. Теперь Android включает в себя поддержку приложений с элементами Material Design. Чтобы использовать элементы Material Design в своих приложениях под Android, руководствуйтесь инструкциями в спецификации Material Design, а также воспользуйтесь новыми компонентами и функциями, доступными в Android 5.0 (уровень API 21) и выше.

Рисунок 1.3 - Четыре кита, на которых строится Material Design

Material Design стоит на четырех главных принципах:

– Тактильные плоскости. В Material Design интерфейс состоит из осязаемых слоёв так называемой «цифровой поверхности». Эти поверхности находятся на разной высоте и отбрасывают тени друг на друга, что делает интерфейс более понятным для восприятия.

– Полиграфический дизайн. Если считать эти слои кусками «цифровой поверхности», то в том, что касается «цифровых текстов» (всего того, что показано на «цифровой бумаге»), используется подход из стандартного графического дизайна: например, журнального и альбомного.

– Понятная анимация. В нашем мире предметы не возникают из ниоткуда и не уходят в никуда — такое бывает только на экранах кино. Поэтому в Material Design мы должны всё время думать о том, как с помощью анимации в слоях и в «цифровых чернилах» давать пользователям как можно больше информации о работе интерфейса.

– Адаптационный дизайн. Речь идет о том, как можно применить предыдущие решения на разных мобильных устройствах с разными разрешениями и размерами экранов.

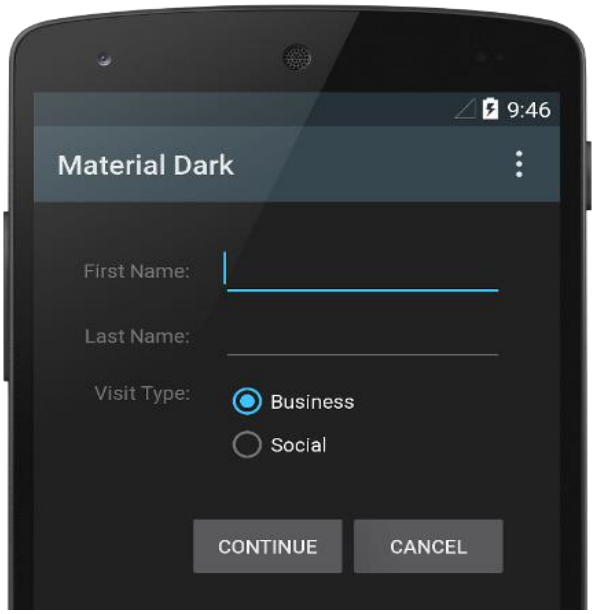

Рисунок 1.4 - Тема Material Design в темных тонах

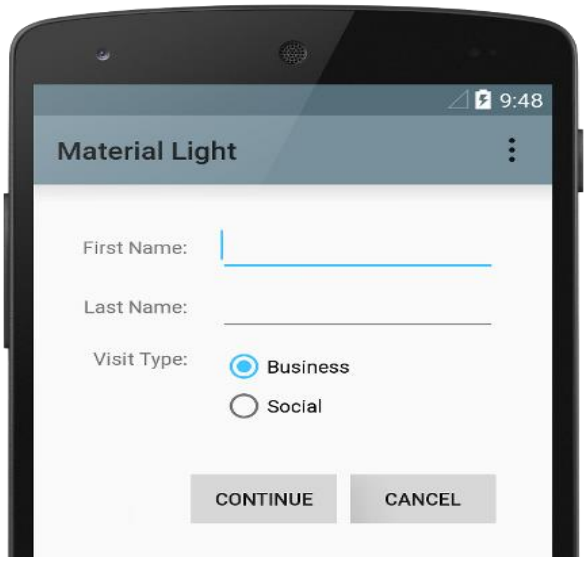

Рисунок 1.2 - Тема Material Design в светлых тонах

Android предоставляет следующие фишки для построения мобильных приложений в соответствии стиля Material Design.

Тема Material Design предоставляет новый стиль для ваших приложений, системные виджеты, для которых можно настраивать цветовую палитру, и анимации, выполняемые по умолчанию в качестве реакции на касание и при переходах между действиями.

Новые API-интерфейсы анимации позволяют реализовывать необычную анимацию для реакции на касание в элементах интерфейса, изменения состояния представления и переходов между событиями.

Эти API-интерфейсы позволяют:

– Чувствовать касание в представлениях, используя анимацию для реакции на касание;

– Прятать и показывать вновь представление с помощью анимации для кругового появления;

– Переходить между действиями с помощью установленной анимации для переходов между событиями;

– создавать более обыденные движения с помощью анимации для перемещения по разным траекториям;

– анимировать изменение одного или нескольких свойств представления с помощью анимации для изменения состояния представления;

– отображать анимацию в графических элементах списков состояний в промежутке между изменением состояний представления.

Анимация для реакции на касание встроена в некоторые стандартные функции, например списки. Новые API-интерфейсы дают возможность разработчику настраивать эти анимации и добавлять их в свои необычные представления.

Рассматриваемые утилиты по работе с графическими объектами облегчают создание приложений с элементами Material Design:

– У векторных объектов можно изменять масштаб без нанесения вреда четкости, и они очень хорошо подходят в качестве одноцветных значков мобильного приложения;

– Затемнение графических объектов позволяет показывать растровые изображения как альфа-маску и тонировать их нужным цветом палитры во время выполнения;

– Выбор цвета позволяет автоматически извлекать главные цветовые палитры из растровых изображений.

2. Инструменты и технологии разработки

2.1 Инструменты разработки интерфейса

Перед тем как начать создавать приложения под Android следует рассмотреть инструменты разработки, без которых разработка мобильного приложения была бы невозможна вот некоторые из них:

Рисунок 2.1 – Android Studio

Android Studio IDE – это среда разработки, предназначенная для создания мобильных приложений под операционную систему Android. Android Studio является самым быстрым способом создания высококачественных приложений для платформы Android, включая телефоны и планшеты, Android Auto, Android Wear и Android TV. Как сообщается на официальном IDE от Google , Android Studio включает в себя все, что нужно, чтобы создать приложение, в том числе редактор кода, инструменты анализа кода, эмуляторы и многое другое. Эта новая и стабильная версия Android Studio имеет высокую скорость сборки и быстрый эмулятор с поддержкой последней версии Android и Google Play. Android-Studio построена в сотрудничестве с Android platform и поддерживает все последние и самые большие API. Она доступна сегодня как файл загрузки или как файл обновления на стабильном канале выпуска. Android Studio 2.0 включает в себя следующие новые функции, которые Android-разработчик может использовать в своем рабочем процессе:

– Instant Run - для каждого разработчика, который любит строить быстрые приложения. Внесите необходимые изменения, и вы увидите, как

они появляются и запускаются в live-режиме в вашем приложении. Так же Instant Run поможет вам сэкономить время.

– Android Emulator - новый эмулятор работает в три раза быстрее, чем предыдущий эмулятор Android, и с расширениями ADB теперь вы можете обрабатывать больше информации, чем на физическом устройстве. Подобно физическому устройству, официальный Android эмулятор также включает в себя сервисы Google Play встроенные, так что вы можете проверить больше функциональных возможностей API.

– Cloud Test Lab Integration- написать один раз, работает везде. Улучшение качества ваших приложений, быстрое и легкое тестирование по широкому кругу физических устройств Android в лаборатории Test Cloud прямо из Android Studio.

– App Indexing Code Generation & Test – служит для того, чтобы помочь повысить видимость вашего приложения в Google Search для пользователей, добавив автоматически сгенерированные URLS с функцией App Indexing в Android Studio. При помощи нескольких щелчков мыши вы можете добавить индексируемые URL ссылки.

– GPU Debugger Preview - Для тех из вас, кто хочет заниматься разработкой в сфере игр, основанных на OpenGL ES, теперь вы можете видеть каждый кадр и состояние GL с новым GPU отладчик.

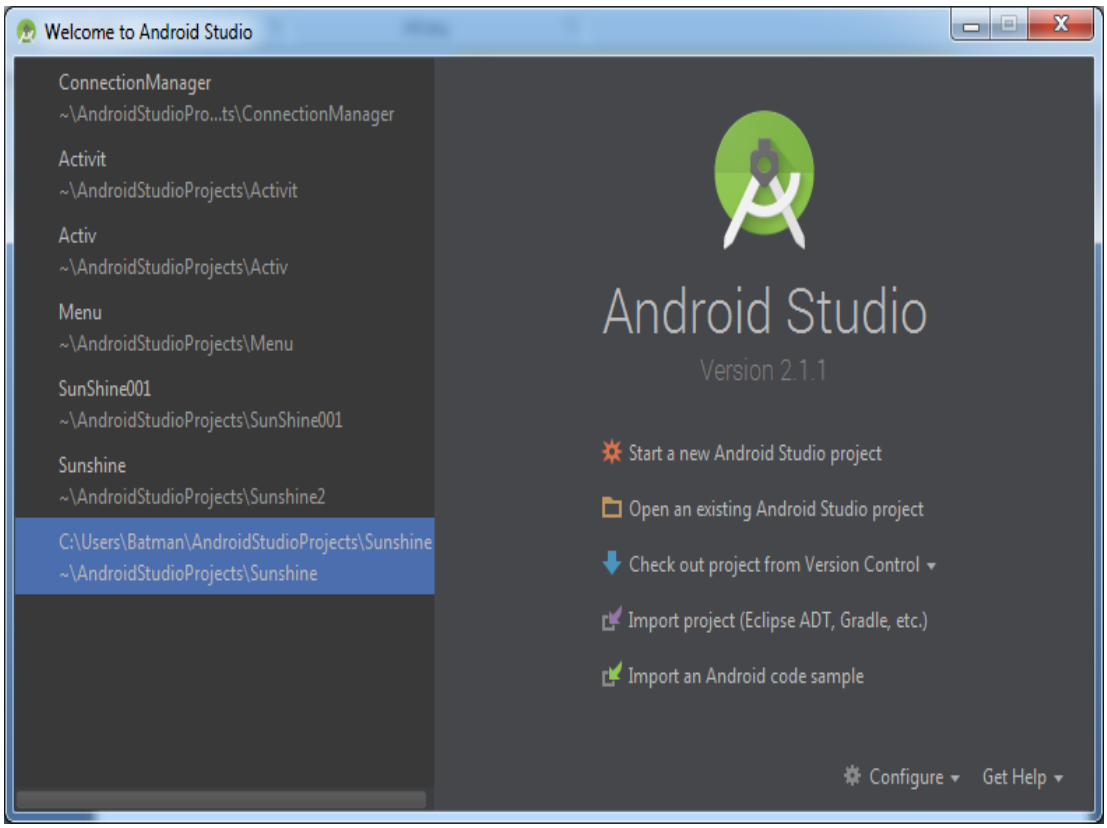

Рисунок 2.1 – Окно первоначальной настройки

Далее выбираем необходимый пункт меню и переходим к указанию имени проекта, пакета и так далее.

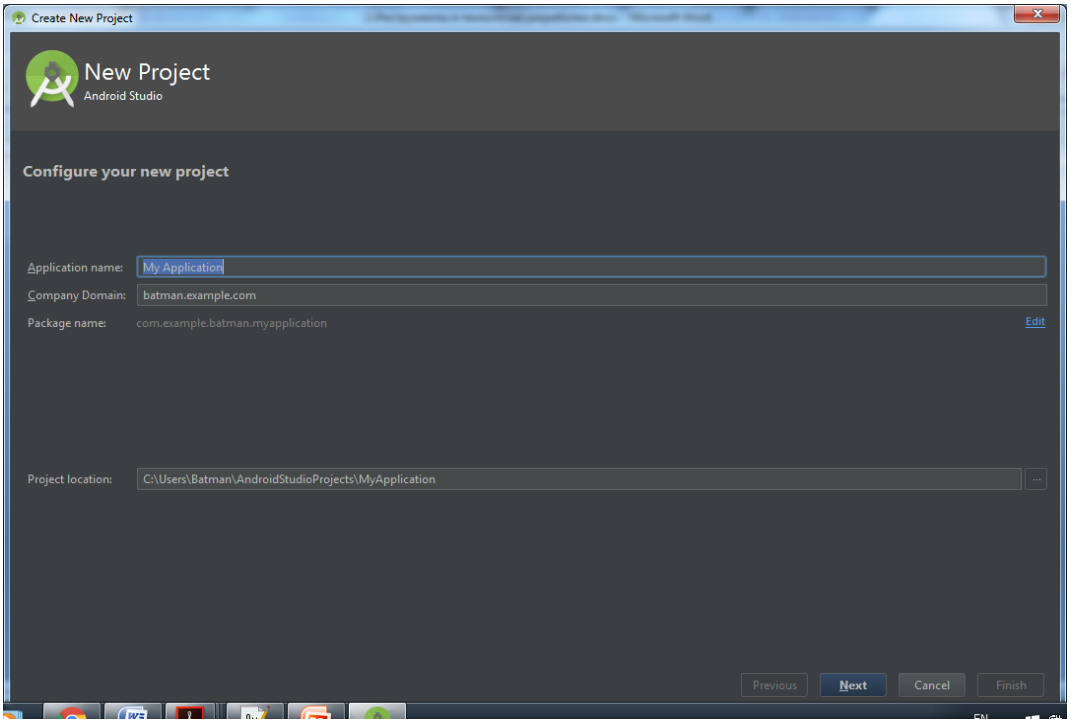

Рисунок 2.2 – Название проекта

На следующем рисунке мы выбираем нужную версию API .

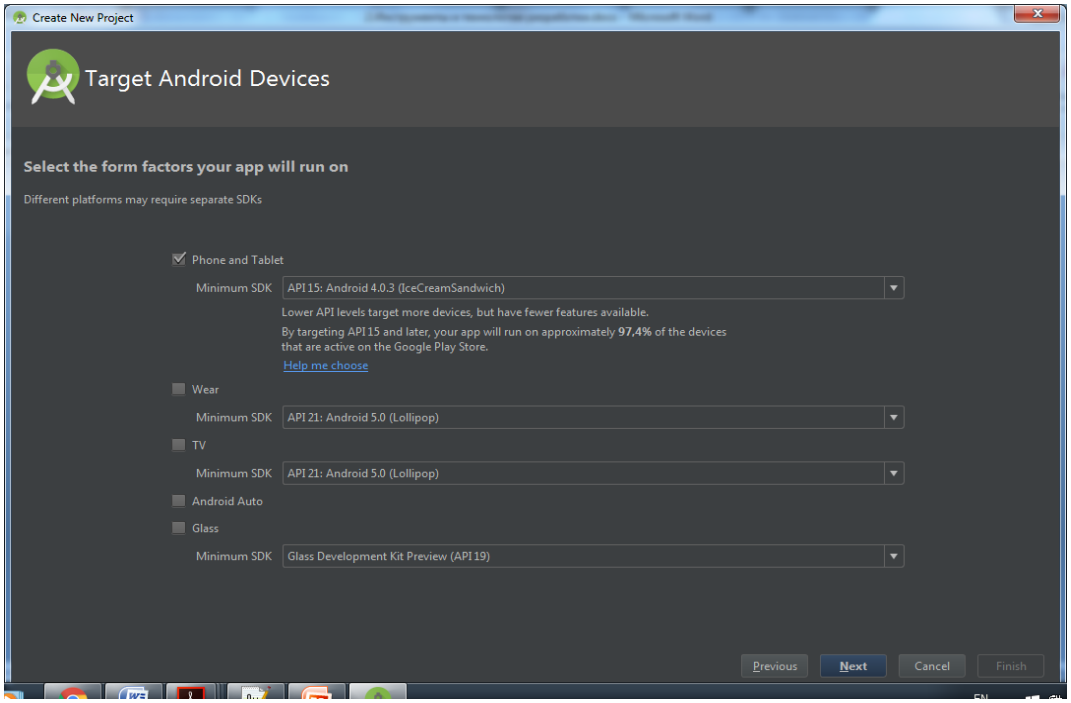

Рисунок 2.3 – Возможные версии API

Также на рисунке мы видим параметры выбора нужного нам устройства, например:

- Phone and Tablet
- Android Glass
- Android Wear
- TV

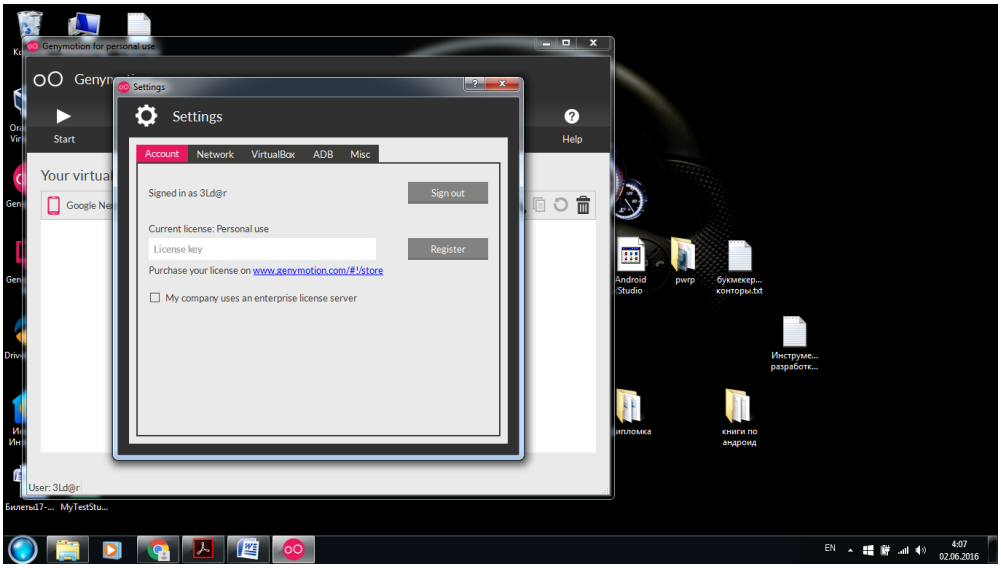

Рисунок 2.4 – Genymotion Emulator

Genymotion – это эмулятор ОС Android, внутри которого есть настроенные образы Android (x86-x64 с новым ускорением OpenGL), он идеален для эмуляции физического устройства. Этот потомок древнего Android-VM, и если их сравнить то Genymotion имеет современное оформление плеера, разные установщики и многое другое.

По словам разработчиков, цель этого эмулятора - заменить эмулятор Android от Google не только разработчикам Android, но и тем, кто делает демонстрации своих разработок под Android.

Genymotion можно установить на Linux, Windows и Mac OS X и так же для этого нужен VirtualBox. VirtualBox находится в открытом доступе, но программное обеспечение, которое работает на хосте, является бесплатным для использования, но его исходник закрыт. В ближайшем будущем этот эмулятор будет иметь бесплатную версию с большим списком функционала, но также будут существовать коммерческие версии, для больших компаний, которые хотят работать совместно над Genymotion.Эмулятор Genymotion имеет более трех тысяч Android-конфигураций на выбор. Разработка и тестирование приложений для Android-устройств является лучшим решением из всех возможных и проще, чем когда-либо прежде, и это дешевле и быстрее.

На этом рисунке мы выбираем устройство, которое хотим эмулировать.

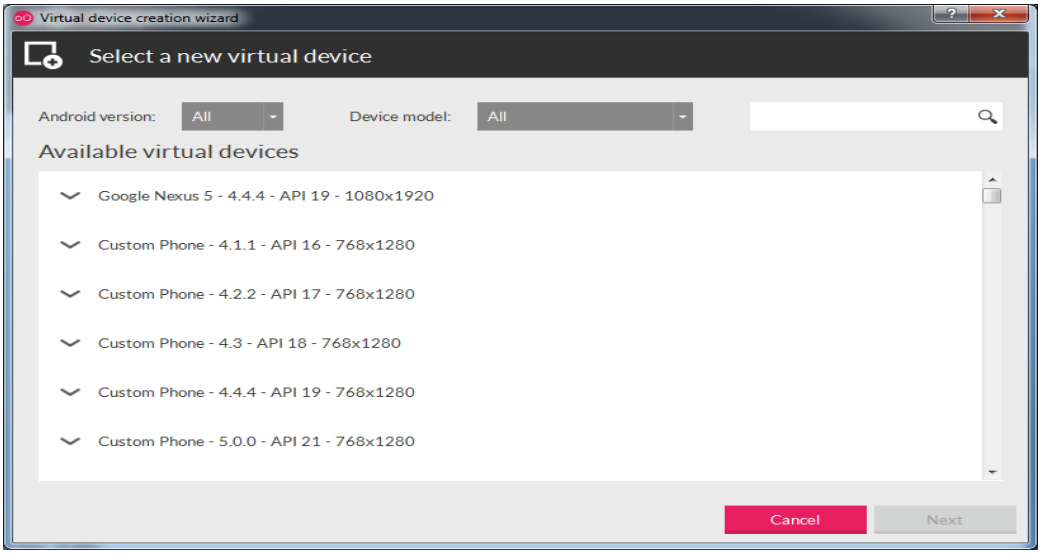

Рисунок 2.5 – выбор устройства

Демонстрация устройства выбранного в предыдущем окне.

Рисунок 2.6 – Выбранный эмулятор

DB Browser для SQLite – это высококачественный визуальный инструмент с открытым кодом для создания, разработки и редактирования базы данных файлов, совместимых с SQLite.

Он предназначен для пользователей и разработчиков, которые хотят создавать базы данных, осуществлять поиск и редактирование данных. Он использует знакомый с электронными таблицами - подобный интерфейс , и вам не нужно изучать сложные команды SQL .

Такие функции будут доступны для пользователей :

- Создание и компоновка файлов базы данных
- Создание, определение, изменение и удаление таблицы
- Пользователи могут редактировать, добавлять и удалять записи
- Поиск записей
- Импорт и экспорт записей в виде текста
- Импорт и экспорт таблиц из / в CSV- файлов
- Импорт и экспорт базы данных из / в файлы дампа SQL
- Вывод SQL запросов и проверка результатов
- Проверка журналов всех команд SQL , выданных приложением

На данном рисунке мы видим интерфейс данного приложения

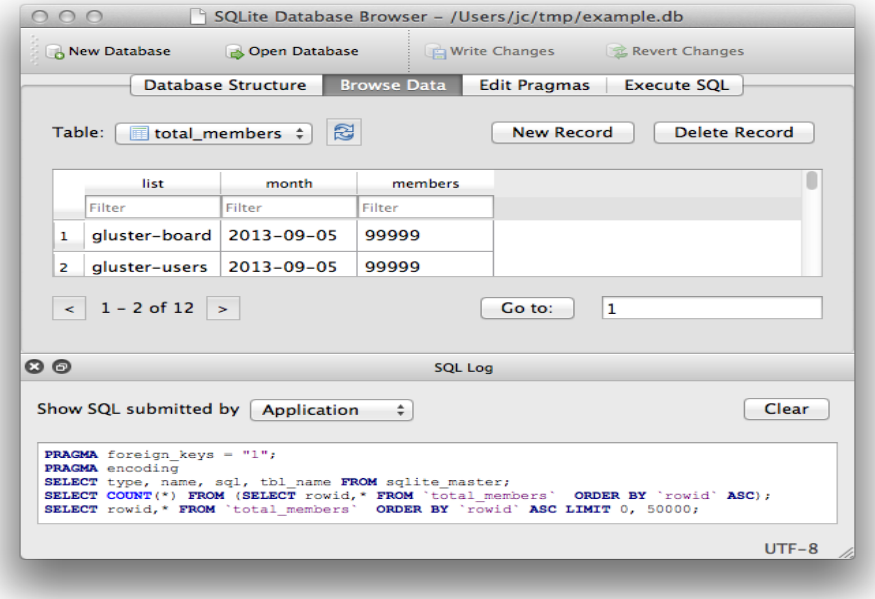

Рисунок 2.7 – DB Browser

2.2 Технологии разработки интерфейса

Создание мобильных приложений (телефонов, планшетов) развертывается очень быстро. Почти все владельцы бизнесов осмыслили, что пребывание на мобильных платформах считается неотъемлемой частью результативной рекламной стратегии. Чтоб удачно стартовать на рынке мобильных устройств, необходимо знать некоторое количество принципиальных решений. Одним из них станет отбор верной технологии разработки приложения. В этой дипломной работа осмотрены главные легкодоступные на данный момент технологии и их индивидуальности с

точки зрения полезности решения определенных бизнес - проектов. Создание данного программного продукта состоит из нескольких огромных шагов. Поначалу возникает главная мысль и создается перечень целей, которых этот проект обязан добиться. Задачи могут быть технические и рекламные. К примеру: притянуть новых людей и улучшить отношения среди имеющихся людей

Начать рассказ о языке Java стоит с того, что существуют и другие языки программирования, в названии которых есть слово «Java» — JavaScript или Enterprise Java Beans, однако все это разные системы. Из них троих конкретное отношение к мобильной разработке имеет Java. Это объектно-ориентированный язык, созданный в Sun Microsystems. Java, как вобщем, и остальная собственность Sun Microsystems, в это время принадлежит Oracle. По мнению множества экспертов, Java является одним из наиболее часто работающих языков программирования, навыки владения им пригодятся не только для личной разработки программ, однако также требуются Компаниям. Основная часть приложений для операционной системы Андроид создается на языке Java. Создатели этой операционной системы предлагают разработчикам специальные фреймворки, которые сильно облегчают процесс разработки программ на таких языках, как C/C++, Python и Java Script через библиотеку jQuery и PhoneGap. Если вы считаете себя начинающим разработчиком приложений для Андроид, мы советуем вам прочитать ресурс http://www.cyberforum.ru/android-dev/, где собрано большое количество интересных рекомендаций.

Java для операционной системы Android. Этот язык программирования является наиболее популярным среди всех имеющихся. Для написания приложений на данном языке можно использовать большое количество разных программных сред, но чаще всего разработчики используют Android studio.

В этой IDE компилятор специально заточен для режим написания приложений под мобильные устройства.

Такой же плагин можно встретить в таких популярных программных средах разработки, как NetBeans и Eclipse. Также для написания кода на этом языке можно воспользоваться пакетом Motodev Studio for Android, он был разработан на основе Eclipse.

C/C++.

Это второй по заинтересованности язык программирования. Главным его плюсом является то, что из-за подключения специальных библиотек у вас будет возможность разрабатывать программы с минимальным содержанием кода, что существенно разгоняет процесс написания. Однако, для написания приложений на данном языке необходимо использовать специальный инструмент для разработчиков. Он называется Android Native Development Kit. Для разработки нативных приложений можно использовать пакет Embaro RAD Studio XE5.

2.3 Виды интерфейсов

Технологии реализации интерфейсов. Интерфейс может быть ясным или непонятным, дружелюбным или нет. Инновационные виды интерфейсов:

– Командный интерфейс – пользователь передает команды компьютеру, который их обрабатывает и выдает результат обратно. Командный интерфейс создан в виде пакетной технологии и технологии командной строки;

– WIMP-интерфейс – общение пользователя с компьютером ведется при помощи графических элементов: меню, окон и других образов. Интерфейс создан на двух уровнях технологий: простой пользовательский интерфейс и WIMP-интерфейс;

– SILK-интерфейс – общение пользователя с компьютером. Интерфейс сильно приближен к стандартной, людской форме общения. При том, что компьютер определяет команды, анализируя людскую речь, и ищет в ней ключевые слова. В результате выполнения функций компьютер преобразует данные в понятную человеку форму. Этот вид пользовательского интерфейса максимально зависит от аппаратных ресурсов компьютера, поэтому его очень часто применяют для военных целей.

Основными технологиями реализации интерфейсов являются следующие:

– Пакетная технология. Изначально технология появилась первой и существовала уже на релейных машинах Зюса и Цюзе (Германия, 1937 г.). На вход ЭВМ передавалась последовательность символов, в которых по определенным стандартам указывалась последовательность отправленных на выполнение программ. После завершения очередной программы запускалась следующая программа и т. д. Вычислительная Машина по определенным стандартам, находит команды и данные. К примеру, в качестве такой линейки выступали: перфолента, стопка перфокарт, последовательность нажатия клавиш электрической пишущей машинки (типа CONSUL). Машина выводила свои сообщения на перфоратор, алфавитно-цифровое печатающее устройство (АЦПУ), ленту пишущей машинки. Такая машина выглядела как шкаф, в который постоянно подводилась информация, а он постоянно информировал о своих действиях. Пользователь имел небольшое влияние на работу машины. Он мог лишь остановить работу машины, сменить перфокарту и вновь запустить Машину.

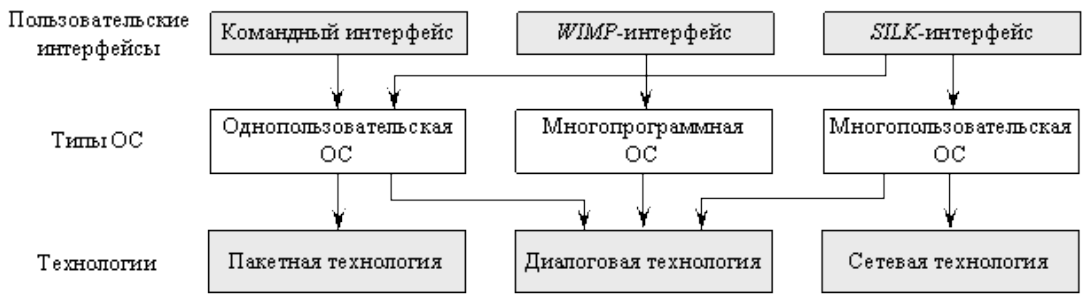

Рисунок 2.8 - Взаимодействие типов операционных систем, пользовательских интерфейсов и основных технологий их реализации

– Технология командной строки. Данные пользователя для компьютера передаются посредством клавиатуры. Компьютер выводит данные на алфавитно-цифровой дисплей (монитор). Связку «монитор + клавиатура» назвали терминалом или консолью. Команды печатаются в командной строке. Она представляет собой символ приглашения и мигающий прямоугольник – курсор. При нажатии клавиши в консоли вместо курсора появляются символы, и курсор смещается, неправильно набранный символ удаляется нажатием клавиши Delete (del). Команда завершается нажатием клавиши Enter (Return), после чего происходит переход к началу следующей строки, в позиции которой компьютер выдает на дисплей результаты этой работы. Затем процесс повторяется. Технология командной строки уже работала на монохромных алфавитно-цифровых мониторах.

Так как вводить позволялось только символы, цифры и знаки препинания, то технические характеристики монитора были не существенны. В качестве дисплея можно было использовать телевизор или трубку осциллографа. Преобладающим видом файлов при работе с командным интерфейсом были текстовые файлы, которые можно было создавать при помощи клавиатуры. Во время наиболее широкого использования интерфейса командной строки приходит появление операционной системы UNIX и первых восьмиразрядных персональных компьютеров с многоплатформенной операционной системой CP/M.

Технология графического интерфейса. Идея пользовательского интерфейса возникла в середине 70-х гг., тогда в исследовательском центре Palo Alto Research Center (PARC) была создана концепция визуального интерфейса. Предпосылкой пользовательского интерфейса явилось уменьшение времени ответа компьютера на команду, увеличение оперативной памяти, развитие технической базы компьютеров. Основой аппаратной концепции явилось появление алфавитно-цифровых мониторов, которые начали поддерживать новые эффекты: мигание символов, инверсию цвета (смена рисования белых символов на черном фоне черными символами на белом фоне), выделение символов. Эффекты распространились не на весь монитор, а только на несколько символов. Следующим шагом является создание цветного дисплея, который мог выводить вместе с этими эффектами символы в 16 цветах на фоне с палитрой (т. е. цветовым набором) из восьми цветов.

Первая система с пользовательским интерфейсом 8114 Star Information System группы PARC была создана в начале 1981 г. Сначала интерфейс использовался только в программных продуктах. Не спеша он стал переходить и на операционные системы, используемые сначала на компьютерах Atari и Apple Macintosh, затем и на IBM-совместимых ЭВМ. Под воздействием новых концепций происходил процесс по унификации

использования клавиатуры и мыши программами. Пользовательский интерфейс за время своего создания с 1974 г. по настоящее время прошел две ступени.

Простой пользовательский интерфейс. На первом этапе пользовательский интерфейс был очень похож на технологию командной строки, но все же имел несколько отличий:

– При показе символы можно было выделить части символов каким-нибудь цветом, инверсным рисунком, подчеркиванием и мерцанием, благодаря чему повысилось качество изображения;

– в зависимости от конкретной цели пользовательского интерфейса курсор может представляться мерцающим прямоугольником или некоторой областью, охватывающей несколько букв, которая отличалась от других невыделенных частей;

– нажатие клавиши Enter не всегда приводило к вызову команды и переходу к следующей строке, так как реакция на нажатие какой-нибудь клавиши во сильно зависела от того, в какой части экрана находился курсор;

– кроме клавиши Enter, на клавиатуре стали использовать клавиши управления курсором и манипуляторы (мышь, трекбол и др.). Они позволяли быстро выделять нужную часть экрана и перемещать курсор.

– Биометрическая технология. Технология была создана в конце 92-х гг. XX в. Для управления ЭВМ используется мимика лица человека, ориентация его взгляда, параметры зрачка и другие признаки. Для распознавания пользователя применяется рисунок радужной оболочки, его глаз, отпечатки пальцев и другая личная информация. Изображения берутся с цифровой видеокамеры, а потом с помощью некоторых программ распознавания образов из этого изображения выделяются команды. Эта технология используется в программах и приложениях для определения пользователя компьютера.

– Технология семантического интерфейса (общественного интерфейса). Технология возникла в конце семидесятых годов двадцатого века. С прогрессом искусственного интеллекта, основанным на семантических сетях. Этот вид интерфейса включает в себя: интерфейс командной строки, графический, речевой и мимический интерфейсы. Важное отличие - отсутствие команд диалога с компьютером. Вопрос создается на естественном языке в виде связанного текста и представлений. По своей сути интерфейс это моделированное общение человека с ЭВМ.

3. Разработка мобильного приложения

3.1 Описание компонентов приложения

Activity — это компонент приложения, который выдает экран, и с которым пользователи могут взаимодействовать для выполнения каких-либо

действий, например набрать номер телефона, сделать фото, отправить письмо или просмотреть карту. Каждой операции присваивается окно для прорисовки соответствующего пользовательского интерфейса. Обычно окно отображается во весь экран, однако его размер может быть меньше, и оно может размещаться поверх других окон.

Когда операция останавливается по причине запуска новой операции, для уведомления об изменении ее состояния используются методы обратного вызова жизненного цикла операции. Существует несколько таких методов, которые может принимать операция вследствие изменения своего состояния — создание операции, ее остановка, возобновление или уничтожение системой; также каждый обратный вызов представляет возможность выполнить определенное действие, подходящее для соответствующего изменения состояния. Например, в случае остановки операция должна освободить любые крупные объекты, например, подключение к сети или базе данных. При возобновлении операции вы можете повторно получить необходимые ресурсы и возобновить выполнение прерванных действий. Такие изменения состояния являются частью жизненного цикла операции.

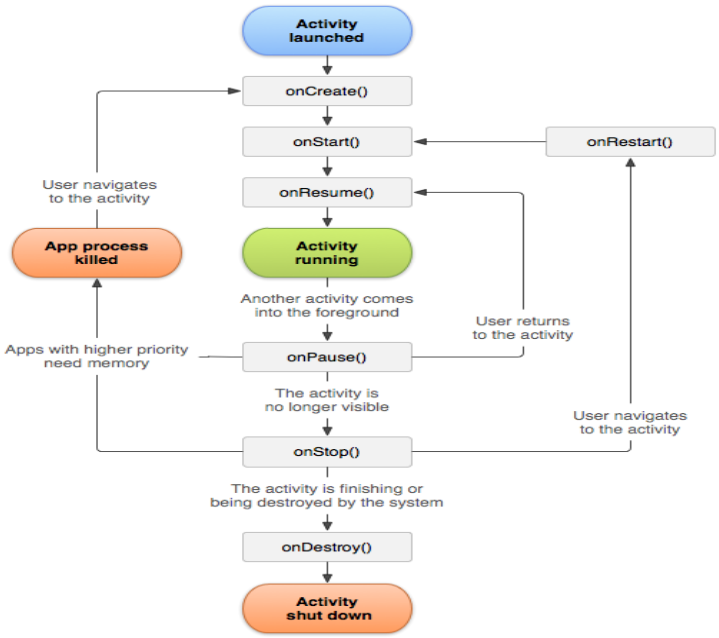

Рисунок 3.1- Жизненный цикл

Fragment - Фрагмент (класс Fragment) представляет поведение или часть пользовательского интерфейса в операции (класс Activity). Разработчик может объединить несколько фрагментов в одну операцию для построения многопанельного пользовательского интерфейса и повторного использования фрагмента в нескольких операциях. Фрагмент можно рассматривать как модульную часть операции. Такая часть имеет свой жизненный цикл и самостоятельно обрабатывает события ввода. Кроме того, ее можно добавить или удалить непосредственно во время выполнения операции. Это нечто

вроде вложенной операции, которую можно многократно использовать в различных операциях.

Фрагмент всегда должен быть встроен в операцию, и на его жизненный цикл напрямую влияет жизненный цикл операции. Например, когда операция приостановлена, в том же состоянии находятся и все фрагменты внутри нее, а когда операция уничтожается, уничтожаются и все фрагменты. Однако пока операция выполняется (это соответствует состоянию возобновлена жизненного цикла), можно манипулировать каждым фрагментом независимо, например добавлять или удалять их. Когда разработчик выполняет такие транзакции с фрагментами, он может также добавить их в стек переходов назад, которым управляет операция. Каждый элемент стека переходов назад в операции является записью выполненной транзакции с фрагментом. Стек переходов назад позволяет пользователю обратить транзакцию с фрагментом (выполнить навигацию в обратном направлении), нажимая кнопку Назад. Когда фрагмент добавлен как часть макета операции, он находится в объекте ViewGroup внутри иерархии представлений операции и определяет собственный макет представлений. Разработчик может вставить фрагмент в макет операции двумя способами. Для этого следует объявить фрагмент в файле макета операции как элемент <fragment> или добавить его в существующий объектViewGroup в коде приложения. Впрочем, фрагмент не обязан быть частью макета операции. Можно использовать фрагмент без интерфейса в качестве невидимого рабочего потока операции. Философия проектирования.

Фрагменты впервые появились в Android версии 3.0 (API уровня 11), главным образом, для обеспечения большей динамичности и гибкости пользовательских интерфейсов на больших экранах, например, у планшетов. Поскольку экраны планшетов гораздо больше, чем у смартфонов, они предоставляют больше возможностей для объединения и перестановки компонентов пользовательского интерфейса. Фрагменты позволяют делать это, избавляя разработчика от необходимости управлять сложными изменениями в иерархии представлений. Разбивая макет операции на фрагменты, разработчик получает возможность модифицировать внешний вид операции в ходе выполнения и сохранять эти изменения в стеке переходов назад, которым управляет операция.

Например, новостное приложение может использовать один фрагмент для показа списка статей слева, а другой—для отображения статьи справа. Оба фрагмента отображаются за одну операцию рядом друг с другом, и каждый имеет собственный набор методов обратного вызова жизненного цикла и управляет собственными событиями пользовательского ввода. Таким образом, вместо применения одной операции для выбора статьи, а другой — для чтения статей, пользователь может выбрать статью и читать ее в рамках одной операции, как на планшете, изображенном на рисунке 3.2.

Следует разрабатывать каждый фрагмент как модульный и повторно используемый компонент операции. Поскольку каждый фрагмент определяет

собственный макет и собственное поведение со своими обратными вызовами жизненного цикла, разработчик может включить один фрагмент в несколько операций. Поэтому он должен предусмотреть повторное использование фрагмента и не допускать, чтобы один фрагмент непосредственно манипулировал другим. Это особенно важно, потому что модульность фрагментов позволяет изменять их сочетания в соответствии с различными размерами экранов.

Если приложение должно работать и на планшетах, и на смартфонах, можно повторно использовать фрагменты в различных конфигурациях макета, чтобы оптимизировать взаимодействие с пользователем в зависимости от доступного размера экрана. Например, на смартфоне может возникнуть необходимость в разделении фрагментов для предоставления однопанельного пользовательского интерфейса, если разработчику не удается поместить более одного фрагмента в одну операцию. Для создания фрагмента необходимо создать подкласс класса Fragment (или его существующего подкласса). Класс Fragment имеет код, во многом схожий с кодом Activity. Он содержит методы обратного вызова, аналогичные методам операции, такие как onCreate(), onStart(), onPause() и onStop().На практике, если требуется преобразовать существующее приложение Android так, чтобы в нем использовались фрагменты, достаточно просто переместить код из методов обратного вызова операции в соответствующие методы обратного вызова фрагмента.

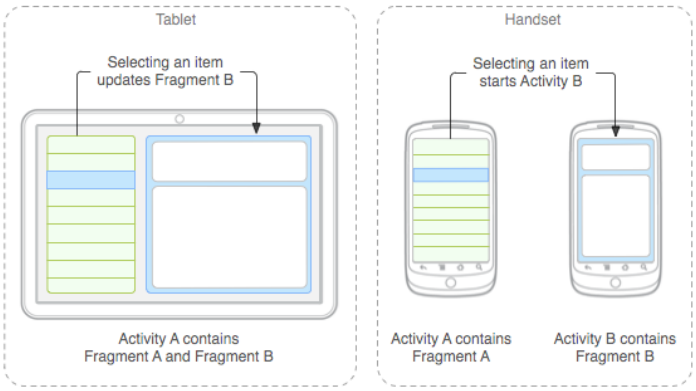

Рисунок 3.2 - Пример того, как два модуля пользовательского интерфейса, определенные фрагментами, могут быть объединены внутри одной операции для работы на планшетах, но разделены на смартфонах.

Как правило, необходимо реализовать следующие методы жизненного цикла:

– onCreate(). Система вызывает этот метод, когда создает фрагмент. В своей реализации разработчик должен инициализировать ключевые компоненты фрагмента, которые требуется сохранить, когда фрагмент находится в состоянии паузы или возобновлен после остановки.

– onCreateView(). Система вызывает этот метод при первом отображении пользовательского интерфейса фрагмента на дисплее. Для прорисовки пользовательского интерфейса фрагмента следует возвратить из этого метода объект View, который является корневым в макете фрагмента. Если фрагмент не имеет пользовательского интерфейса, можно возвратить null.

– onPause(). Система вызывает этот метод как первое указание того, что пользователь покидает фрагмент (это не всегда означает уничтожение фрагмента). Обычно именно в этот момент необходимо фиксировать все изменения, которые должны быть сохранены за рамками текущего сеанса работы пользователя (поскольку пользователь может не вернуться назад).

В большинстве приложений для каждого фрагмента должны быть реализованы, как минимум, эти три метода. Однако существуют и другие методы обратного вызова, которые следует использовать для управления различными этапами жизненного цикла фрагмента. Все методы обратного вызова жизненного цикла подробно обсуждаются в разделе Управление жизненным циклом фрагмента.

Существует также ряд подклассов, которые, возможно, потребуется расширить вместо использования базового класса Fragment:

– DialogFragment. Отображение перемещаемого диалогового окна. Использование этого класса для создания диалогового окна является хорошей альтернативой вспомогательным методам диалогового окна в классеActivity. Дело в том, что он дает возможность вставить диалоговое окно фрагмента в управляемый операцией стек переходов назад для фрагментов, что позволяет пользователю вернуться к закрытому фрагменту.

– ListFragment. Отображение списка элементов, управляемых адаптером (например, SimpleCursorAdapter), аналогично классу ListActivity. Этот класс предоставляет несколько методов для управления списком представлений, например, метод обратного вызова onListItemClick() для обработки нажатий.

– PreferenceFragment. Отображение иерархии объектов Preference в виде списка, аналогично классу PreferenceActivity. Этот класс полезен, когда в приложении создается операция «Настройки»

Макет определяет визуальную структуру пользовательского интерфейса, например, пользовательского интерфейса операции или виджета приложения.

Существует два способа объявить макет:

– Объявление элементов пользовательского интерфейса в XML. В Android имеется удобный справочник XML-элементов для классов View и их подклассов, например таких, которые используются для виджетов и макетов.

– Создание экземпляров элементов во время выполнения. Ваше приложение может программным образом создавать объекты View и ViewGroup (а также управлять их свойствами).

Платформа Android предоставляет вам гибкость при использовании любого из этих способов для объявления пользовательского интерфейса приложения и его управления. Например, вы можете объявить в XML макеты по умолчанию, включая элементы экрана, которые будут отображаться в макетах, и их свойства. Затем вы можете добавить в приложение код, который позволяет изменять состояние объектов на экране (включая объявленные в XML) во время выполнения.

В подключаемом модуле ADT для Eclipse предусмотрена функция предварительного просмотра созданного вами файла XML — достаточно открыть файл XML и выбрать вкладку Layout (Макет).

Для отладки макетов можно воспользоваться инструментом Hierarchy Viewer - с его помощью можно просмотреть значения свойств, рамки с индикаторами заполнения или полей, а также полностью обрисованные представления прямо во время отладки приложения в эмуляторе или на устройстве [5].

С помощью инструмента layoutopt можно быстро проанализировать макеты и их иерархии на предмет низкой эффективности или иных проблем.

Преимущество объявления пользовательского интерфейса в файле XML заключается в том, что таким образом вы можете более эффективно отделить представление своего приложения от кода, который управляет его поведением. Описания пользовательского интерфейса находятся за пределами кода вашего приложения. Это означает, что вы можете изменять или адаптировать интерфейс без необходимости вносить правки в исходный код и повторно компилировать его. Например, можно создать разные файлы XML макета для экранов разных размеров и разных ориентаций экрана, а также для различных языков. Кроме того, объявление макета в XML упрощает визуализацию структуры пользовательского интерфейса, благодаря чему отладка проблем также становится проще. В данной статье мы научим вас объявлять макет в XML. Если вы предпочитаете создавать экземпляры объектов View во время выполнения, обратитесь к справочной документации для классов ViewGroup и View.

Как правило, справочник [6] XML-элементов для объявления элементов пользовательского интерфейса точно следует структуре и правилам именования для классов и методов — названия элементов соответствуют названиям классов, а названия атрибутов соответствуют методам. Фактически, соответствие зачастую такое точное, что вы можете с легкостью догадаться, какой атрибут XML соответствует тому или иному методу класса, или какой класс соответствует заданному элементу XML. Однако следует отметить, что не все справочники являются идентичными. В некоторых случаях названия могут несколько отличаться. Например, у элемента EditText есть атрибут text, который соответствует методу EditText.setText().

ListView - отображает прокручиваемый список пунктов . Элементы списка автоматически добавляются в список с помощью адаптера, который

вытягивает содержимое из источника, такого как массив или база данных запроса и преобразует каждый элемент, приводит в представление то, что помещено в список

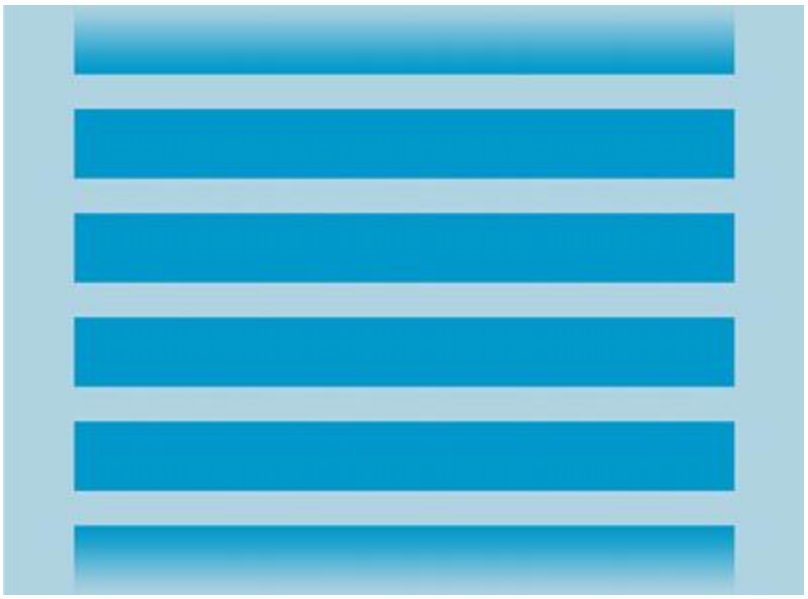

Рисунок 3.3 – Тривиальный вариант списка

Для заполнения списка рекомендуется использовать Cursor Loader Использование CursorLoader является стандартным способом для запроса курсора в качестве асинхронной задачи для того, чтобы избежать блокировки основного потока вашего приложения. Когда CursorLoader получает Cursor в качестве результата, LoaderCallbacks получает обратный вызов onLoadFinished (), в котором вы обновить адаптер с новым курсором и представление списка, затем отображает результаты.

3.2 Алгоритм и структура работы приложения

Разработка мобильных приложений — это очень непростой и долгий процесс, который представляет собой полную разработку данного приложения. Ниже представлен каждый этап нашей работы.

– Разработка технического задания. В первую очередь необходимо определить, какие потребности пользователей и клиента должно решать приложение, а также сформулировать его основные задачи. Этому этапу уделяется особое внимание: от задания зависят технические особенности будущего продукта. Упустив даже незначительную на первый взгляд деталь и не заложив ее в архитектуру приложения, мы можем столкнуться с необходимостью переделывать его практически с нуля.

На этом этапе мы:

– составляем подробное описание функционала приложения;

– определяем временные рамки и финансовые затраты на работу;
– оформляем договор с клиентом.
– Проектирование UI/UX

Чтобы понять, как покупатель будет пользоваться приложением, мы создаем графическую карту взаимодействия между экранами, также на данном этапе прорабатывается практически весь функционал продукта. Проектирование UI/UX является разработкой прототипа приложения: мы реализуем все описанные в техническом задании функции, определяем, как будет работать приложение и как будет работать с ним пользователь, продумываем, какие кнопки и какой функционал будет размещен на каждом экране

На этом этапе мы: оттачиваем функционал приложения и окончательно продумываем сценарий поведения пользователя; разрабатываем схемы всех экранов с указанием функционала на каждом из них; на схеме показываем связь всех экранов, то есть продумываем, как пользователь будет переходить на них.

– Создание концепции дизайна. На примере основных экранов приложения (1–3) мы показываем его будущий дизайн, отталкиваясь в первую очередь от целей, аудитории и функционала[7]. На этом этапе мы:
– детально прорабатываем от 1 до 3 экранов будущего приложения;
– при необходимости создаем дизайн в нескольких разных стилях, чтобы выбрать наиболее подходящий.
– Обрисовка всех экранов. После утверждения концепции дизайна мы обрисовываем все остальные экраны, кнопки, иконки, экраны с помощью и подсказками и т. д. — то есть соединяем результат проектирования и создания концепции дизайна. Сроки работы зависят от сложности концепции и общего количества экранов приложения.

На этом этапе мы:
– детально прорабатываем все экраны будущего приложения.
– Разработка

На этом этапе мы верстаем все элементы приложения, т. е. из статичной картинки делаем интерактивную рабочую модель. Также мы соединяем серверную и клиентскую часть приложения, чтобы оно взаимодействовало с пользователем и полноценно работало.

На этом этапе мы:
– получаем первую версию работающего приложения;
– Создание иконки приложения

Иконка приложения является его неотъемлемой частью. Обычно это не просто уменьшенный логотип компании, а самостоятельный графический элемент. Как и при создании любой иллюстрации, при создании иконки сначала рисуется ее эскиз, затем он корректируется, прорисовывается и утверждается.

– Запуск в Play Market. Перед запуском в магазин Play Market компания Android проверяет соответствие приложения своим стандартам и техническим особенностям. Специалисты Android могут попросить внести в приложение некоторые изменения. Этот этап занимает от нескольких дней до двух недель.

Далее описываем алгоритм работы нашего приложения

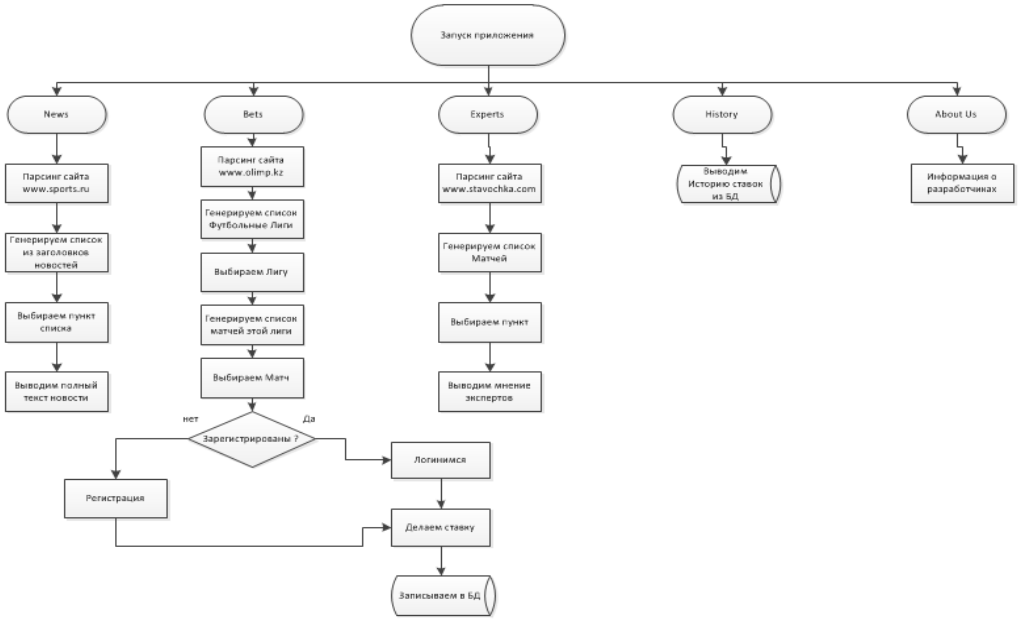

Рисунок 3.4 – Алгоритм работы приложения

Итак, если мы пройдем по ветке News

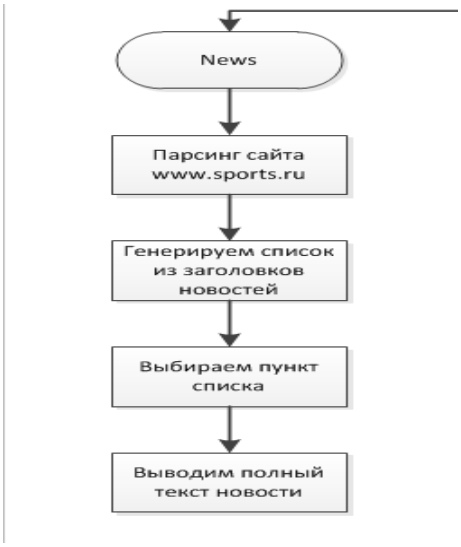

Рисунок 3.5 – Ветка News

Рисунок 3.6 – Проход по ветке News

При нажатии на пункт меню News, мы проходим по ссылке http://www.sports.ru/topnews/ , далее парсим все заголовки новостей и их ссылки перехода на полную новость. Генерируем HashMap <String,String> в нашем случае это hashMap<Заголовок новости, ссылка>.Из этого hashMap создаем список всех новостей и выводим на экран. После нажатия на пункт списка(заголовок новости), мы переходим непосредственно к самой новости.

```
listik = (ListView) v.findViewById(R.id.listik);

ParseTitle parseTitle = new ParseTitle();
parseTitle.execute();
try {
    final HashMap<String,String> hashMap = parseTitle.get();
    final ArrayList<String> arraylist = new ArrayList<>();
    for(Map.Entry entry : hashMap.entrySet()){
        arraylist.add(entry.getKey().toString());
    }
    ArrayAdapter<String> adapter = new ArrayAdapter<~>(getActivity(),android.R.layout.simp
    listik.setAdapter(adapter);
    listik.setOnItemClickListener((parent, view, position, id) -> {
            ParseText parseText = new ParseText();
            parseText.execute(hashMap.get(arraylist.get(position)));
            fragment_news_item = new ListItemOfNews();
            Bundle args11 = new Bundle();
            try {
                args11.putString("item_of_news",parseText.get());

            } catch (InterruptedException e) {
                e.printStackTrace();
            } catch (ExecutionException e) {
                e.printStackTrace();
            }
            fragment_news_item.setArguments(args11);

fragmentTransaction = getFragmentManager().beginTransaction();
fragmentTransaction.replace(R.id.fragment_holder, fragment_news_item);
fragmentTransaction.addToBackStack(null);
fragmentTransaction.commit();
```

Рисунок 3.7 – код реализации News

Рассмотрим следущую ветку – Bets

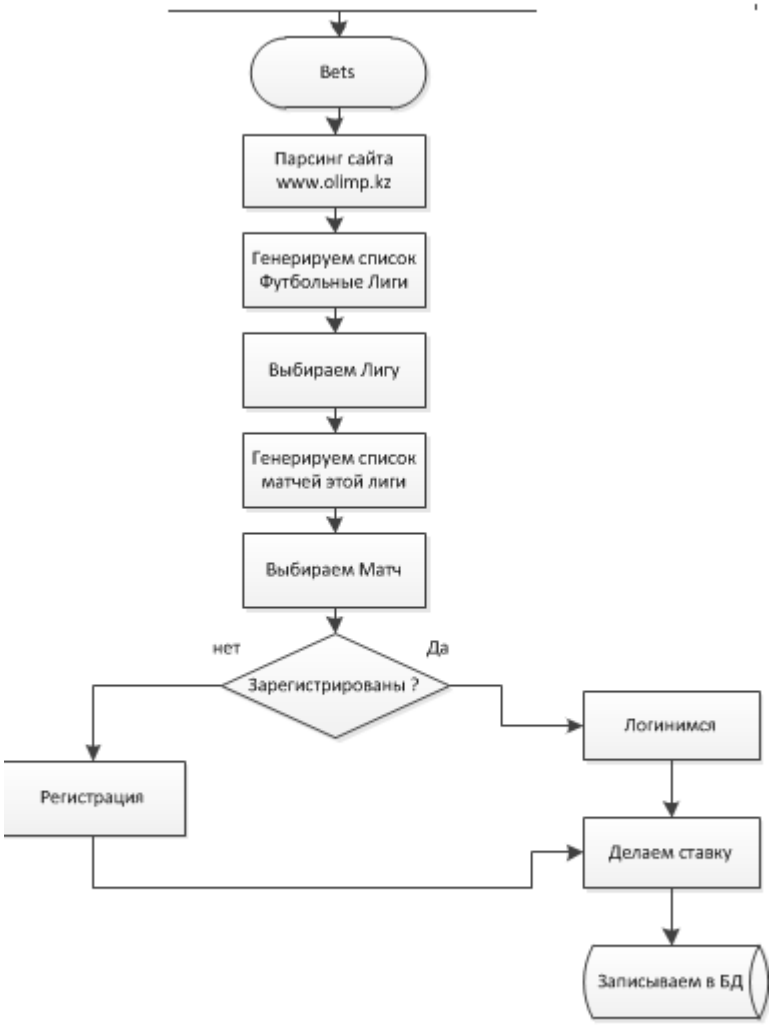

Рисунок 3.8 – Ветка Bets

При нажатии на пункт меню Bets, мы проходим по ссылке https://www.olimpkz.com/mobile/index.php?page=line&action=1&time=0&line_nums=0&sel[]=1.Далее парсим все футбольные лиги, по аналогии News создаем hashMap<Футбольная Лига, ссылка на матчи этой лиги>.
Создаем список всех футбольных лиг и выводим на экран. По нажатию на один из пунктов (Футбольная Лига) мы переходим по ссылке и парсим список матчей этой лиги и выводим на экран. Далее если мы хотим посмотреть ставки на этот матч нам необходимо зарегистрироваться в системе или если мы уже зарегистрированы то залогиниться. После аутентификации в системе мы нажимаем на нужный матч и видим диалоговое окно в котором выбираем ставку которую хотим сохранить в истории. Сохраняем в базе данных в таблице History

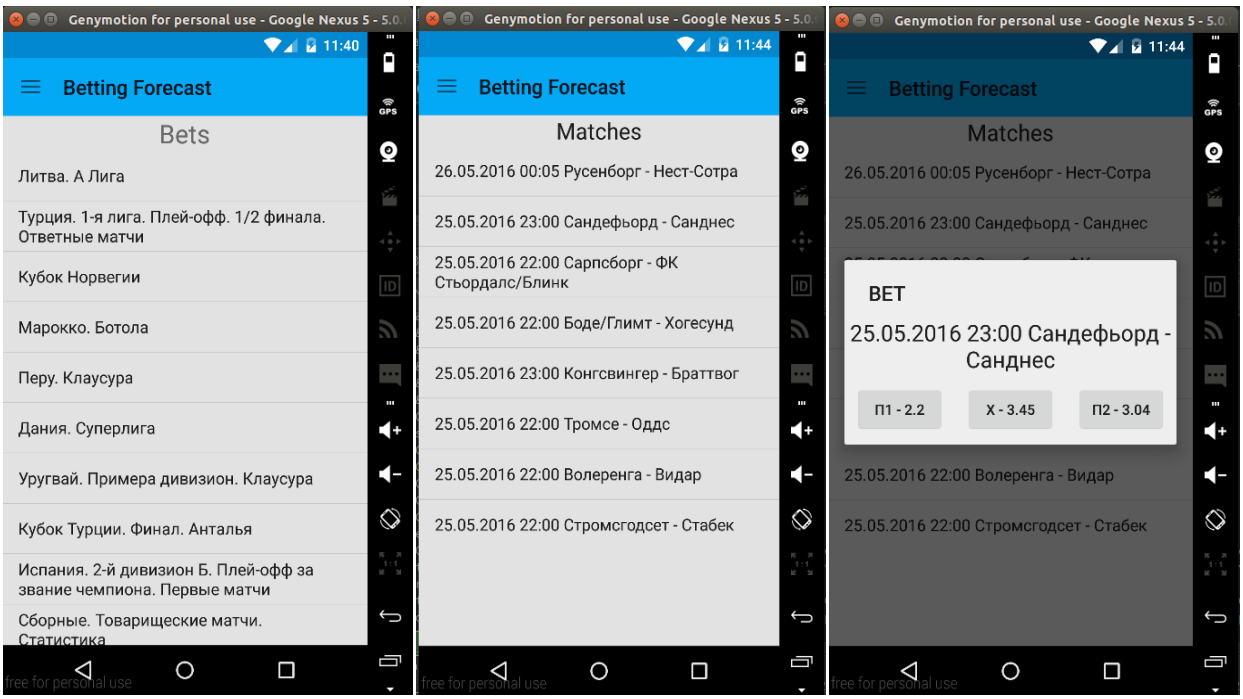

Рисунок 3.9 - Проход по ветке Bets

Следующий рисунок демонстрирует раеализацию кодовой части фрагмента Bets

```
class ParseTitle_bets extends AsyncTask<String,Void,HashMap<String,String>>{

    @Override
    protected HashMap<String, String> doInBackground(String... params) {
        HashMap<String, String> hashMap_bets = new HashMap<>();
        try {
            Document document = Jsoup.connect(params[0]).get();
            Elements elements = document.select(".row");
            for(Element element: elements){
                Element element1 = element.select("a[href]").first();
                hashMap_bets.put(element.text(),element1.attr("abs:href"));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return hashMap_bets;
    }
```

Рисунок 3.10 – Парсинг футбольных лиг

```
class ParseText_bets extends AsyncTask<String,Void,HashMap<String,ArrayList<String>>>{

    @Override
    protected HashMap<String,ArrayList<String>> doInBackground(String... params) {
        String str = " ";
        HashMap<String,ArrayList<String>> hashMap = new HashMap<>();
        ArrayList<String> array = new ArrayList<>();
        array.clear();
        int i;
        try {
            Document document = Jsoup.connect(params[0]).get();
            Elements elements = document.select(".row");
            for(Element element:elements){
                Element element11 = element.select("a[href]").first();
                Document docum1 = Jsoup.connect(element11.attr("abs:href")).get();
                Elements elem1 = docum1.select(".coefficient");
                i =0;
                for(Element kef: elem1){
                    array.add(kef.text());
                    i++;
                    if(i==3) break;
                }
                hashMap.put(element.text(),new ArrayList<String>(array));
                array.clear();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
```

Рисунок 3.11 – Парсинг футбольных матчей

Рассмотрим следущую ветку – Experts

Рисунок 3.12 – Ветка Experts

При нажатии на пункт меню Experts, мы проходим по ссылке http://stavochka.com/forecasts/ , далее парсим все матчb на которые дают прогноз и их ссылки перехода на полную новость. Генерируем HashMap <String,String> в нашем случае это hashMap<Матч, ссылка>.Из этого hashMap создаем список всех матчей и выводим на экран. После нажатия на пункт списка(матч), мы переходим непосредственно к самому матчу.

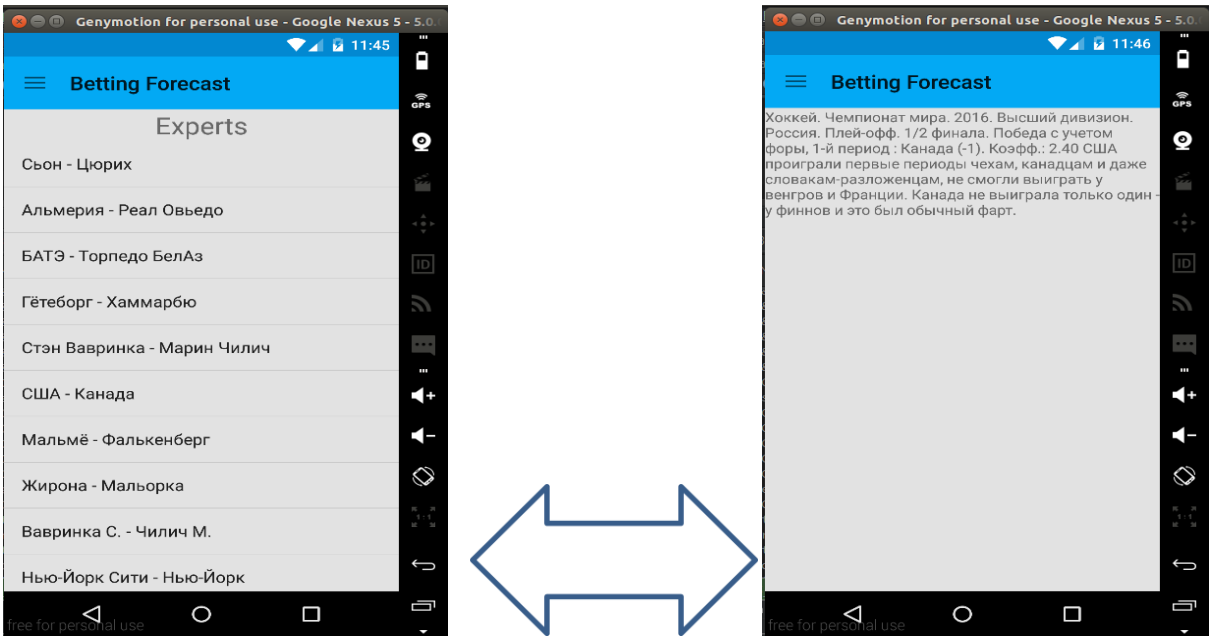

Рисунок 3.13 – Проход по ветке Experts

Следующий рисунок демонстрирует раеализацию кодовой части фрагмента Experts

```
class ParseTitle_experts extends AsyncTask<Void,Void,HashMap<String,String>>{

    @Override
    protected HashMap<String, String> doInBackground(Void... params) {
        HashMap<String,String> hashMap = new HashMap<>();
        try {
            Document document = Jsoup.connect("http://stavochka.com/forecasts/").get();
            Elements elements = document.select(".b-forecasts__th a");
            for(Element element :elements){
                Element element1 = element.select("a[href]").first();
                hashMap.put(element.text(),element1.attr("abs:href"));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return hashMap;
    }
}
```

Рисунок 3.14 – Создание бэк-граунд процесса

Описание вкладок History, About us

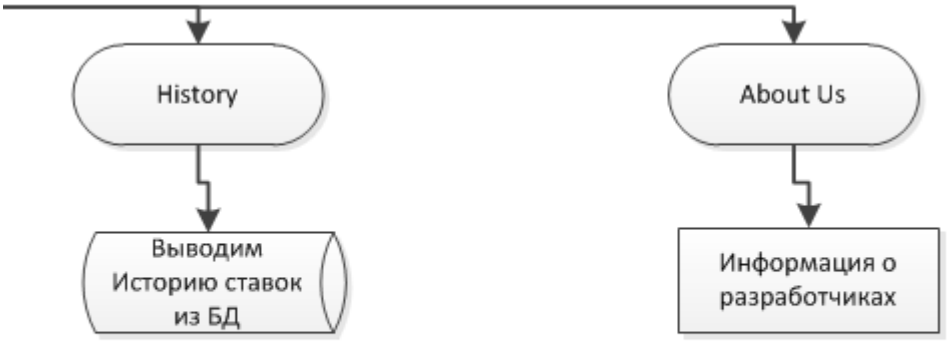

Рисунок 3.10 – History, About us

При нажатии на вкладку History мы создаем запрос для базы данных, вытаскиваем нужные нам данные и выводим на экран. About us, содержит информацию о разработчиках.

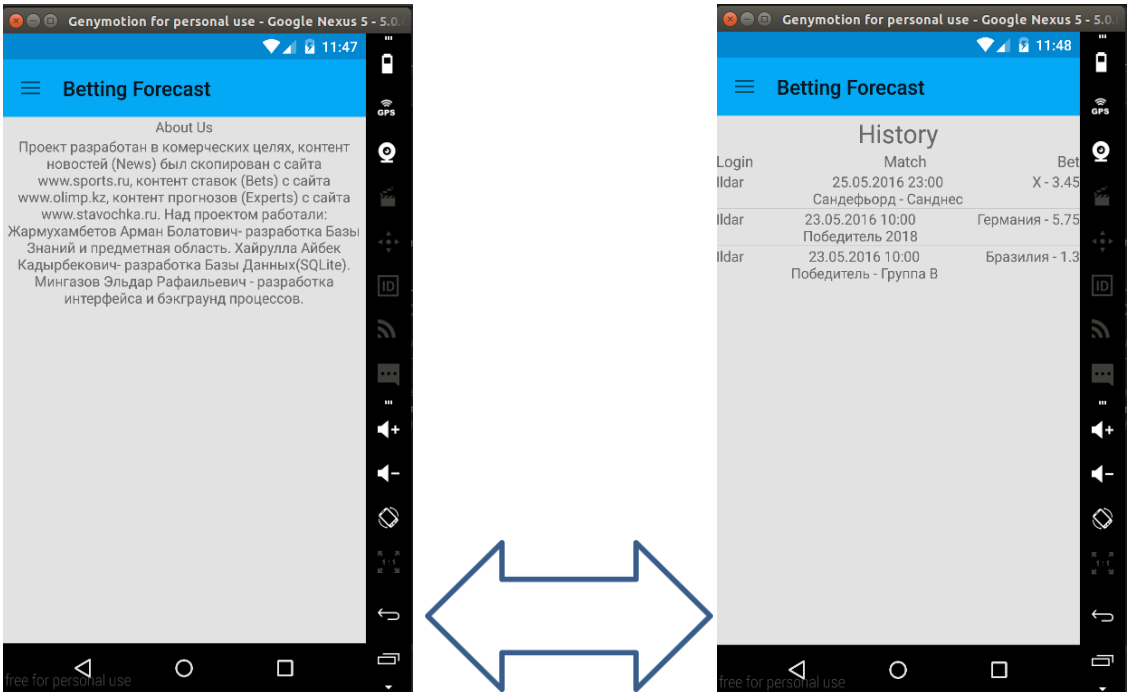

Рисунок 3.11 - Проход по ветке History, About us

На следущих рисунках демонстрация работы Аутентификации и бокового меню

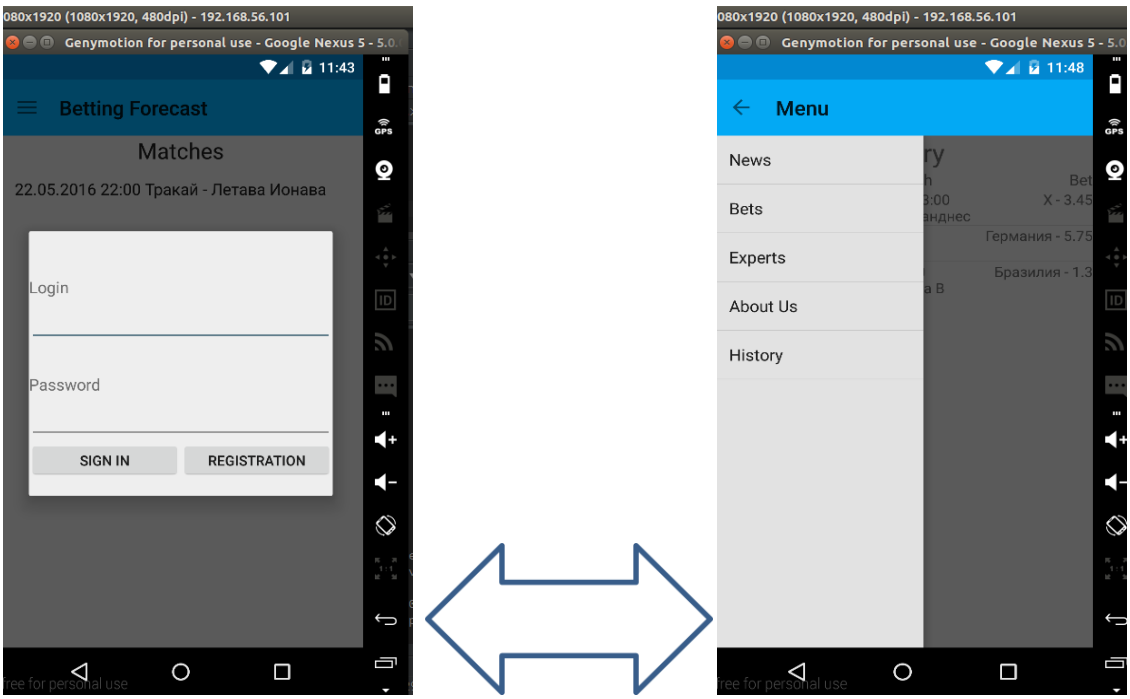

Рисунок 3.12 – Login, Menu

3.3 Связь интерфейса с базой данных

SQLite [8] является встроенным движком базы данных SQL. В отличие от большинства других баз данных SQL, SQLite не имеет отдельного процесса серверного процесса. SQLite считывает и записывает непосредственно на обычные дисковые файлы. Полная база данных SQL с несколькими таблицами, индексами, триггеры и представления, содержится в одном файле на диске. Формат файла базы данных является кросс-платформенным - вы можете свободно копировать базу данных из 32-битной в 64-битную систему. Эти особенности делают SQLite популярным выбором в качестве формата файла приложения. Подумайте о SQLite не в качестве замены для Oracle, но в качестве замены для FOPEN ()

SQLite является компактной библиотекой. При включении всех функций, размер библиотеки может быть меньше, чем 500KiB, в зависимости от настроек целевой платформы и оптимизации компилятора. Если дополнительные функции опущены, размер SQLite библиотеки может быть уменьшен ниже 300KiB. SQLite также может работать в минимальном пространстве стека (по 4Kb), что делает SQLite популярным выбором движка ограниченных в памяти устройств: смартфонов и MP3-плееров. Существует компромисс между использованием памяти и скорости. SQLite в целом работает быстрее тогда, когда осталось достаточно памяти. Тем не менее, производительность, как правило, достаточно высока, даже в условиях низкой памяти.

SQLite очень тщательно тестируются перед каждым выпуском и имеет репутацию очень надежной базы данных. Большая часть исходного кода

SQLite посвящена исключительно для тестирования и верификации. Автоматизированный набор тестов обрабатывает миллионы и миллионы тестовых случаев с участием сотен миллионов отдельных операторов SQL и достигает 100% тестового покрытия филиала. SQLite реагирует очень грациозно на ошибки выделения памяти в дисковых операциях ввода / вывода ошибок. Все это проверяется с помощью автоматизированных тестов с использованием специальных тестовых жгутов, которые имитируют сбои системы. Конечно, даже при всем этом тестировании, есть еще ошибки. Но в отличие от некоторых аналогичных проектов (особенно коммерческих конкурентов) SQLite открыто и честно обо всех ошибках и предоставляет списки багов .

Разработчики продолжают расширять возможности SQLite и повышать его надежность и производительность при сохранении обратной совместимости с опубликованным интерфейсом спецификации, синтаксиса SQL, и формата файла базы данных. Исходный код является абсолютно бесплатным для всех.

Далее проиллюстрируем программную часть того, как мы реализовали связь базы данных с интерфейсом. А именно описание класса DbHelper в котором мы реализуем создание таблиц

```
public class DbHelper extends SQLiteOpenHelper {


    private static final String DataBase_Name = "myDataBase.db";
    private static final int DataBase_Version = 1;



    private static final String DataBase_Table_BETS = "Bets";

    public static final String Bets_match = "Match";
    public static final String Bets_P1 = "Win1";
    public static final String Bets_X = "X";
    public static final String Bets_P2 = "Win2";

    private static final String DataBase_CreateTable_Bets= "create table "
            + DataBase_Table_BETS + "( Bets_ID integer primary key autoincrement, " + Bets_match
            + " text, " + Bets_P1 + " text, " + Bets_X
            + " text, " + Bets_P2 + " text);";

    private static final String DataBase_Table_User = "User";

    public static final String User_Login = "Login";
    public static final String User_Password = "Password";
    public static final String User_Email = "Email";

    private static final String DataBase_CreateTable_User= "create table "
            + DataBase_Table_User + "( _id integer primary key autoincrement, " + User_Login
            + " text, " + User_Password + " text, " + User_Email
            + " text);";
```

Рисунок 3.13 – класс DbHelper

Также после парсинга[9] выполняемом в вкладке Bets нам необходимо записать данные в талблицу Bets. На данном рисунке демонстрируется запись данных в таблицу Bets.

```
DbHelper mDbHelper = new DbHelper(getActivity(),"myDataBase.db",null,1);
SQLiteDatabase sdb = mDbHelper.getWritableDatabase();
sdb.delete("Bets", null, null);
ContentValues cv = new ContentValues();
for(String key :hashMap.keySet()){
      cv.put(DbHelper.Bets_match,key);
        ArrayList<String> values = hashMap.get(key);
        for(String val:values){Log.d("#######",val);}
            cv.put(DbHelper.Bets_P1, values.get(0));
            cv.put(DbHelper.Bets_X, values.get(1));
        if(values.size() == 3) {
            cv.put(DbHelper.Bets_P2, values.get(2));
        }
        sdb.insert("Bets", null, cv);
    }
    sdb.close();
```

Рисунок 3.14 – Запись в базу данных.

4. Технико-экономическое обоснование проекта

4.1 Расчет затрат на разработку интерфейса

Расчет полных затрат на разработку проектного решения в виде информационных технологий ($С_{пі}$) осуществляется по формуле:

$$C_{пі} = З_{фот} + З_{сзі} + M_i + Р_{сі} + Р_{мі} + П_{зі} + Р_{ні} \quad (4.1)$$

где $З_{фот}$ – общий фонд оплаты труда разработчиков, тенге;

$З_{сзі}$ – отчисления по социальному налогу, тенге;

M_i – затраты на материалы, тенге;

$Р_{сі}$ – затраты на специальные программные средства, необходимые для разработки проектного решения, тенге;

$Р_{мі}$ – затраты, связанные с эксплуатацией техники, тенге;

$П_{зі}$ – прочие затраты, тенге;

$Р_{ні}$ – накладные расходы, тенге.

Размер фонда оплаты труда разработчиков ($З_{фот}$) рассчитывается по формуле:

$$З_{фот} = З_{оі} + З_{ді} \quad (4.2)$$

где $З_{оі}$ – основная заработная плата, тенге;

$З_{ді}$ – дополнительная заработная плата, тенге.

Затраты на оплату труда зависят от объема и трудоемкости разработки программного обеспечения. Общий объем (V_0) программного продукта определяется исходя из количества и объема функции, реализуемых программой:

$$V_0 = \sum_{i=1}^{n} Vi \quad (4.3)$$

где V_i - объем отдельной функции ПО;

n − общее число функций.

Из приложения В устанавливаем объем ПО (строки исходного кода, LOC) составляют 6500. Таким образом:

V_0 = 6500 строк кода

Общая трудоемкость проекта рассчитывается по формуле:

$$T_о = T_н \cdot K_с \cdot K_т \cdot K_н \quad (4.4)$$

где $K_с$ – коэффициент, учитывающий сложность ПО;

$K_т$ – поправочный коэффициент, учитывающий степень использования при разработке стандартных модулей;

$K_н$ – коэффициент, учитывающий степень новизны ПО;

$T_н$ – нормативная трудоемкость.

Коэффициент сложности определяется на основе данных представленных в таблице 4.1 и составляет $K_с$ = 0,18, т.к. в проекте присутствует более 3-х характеристик – функционирование системы, интерактивный доступ, обеспечение хранения, ведения и поиска данных в сложных структурах.

Таблица 4.1 – Дополнительные коэффициенты сложности ПО

Характеристика ПО	Значения $K_с$
1 Функционирование ПО в расширенной операционной среде (связь с другими ПО)	0,08
2 Интерактивный доступ	0,06
3 Обеспечение хранения, ведения и поиска данных в сложных структурах	0,07
4 Наличие у ПО одновременно нескольких характеристик по табл.Г4.1, приложение Г	
4.1 2 характеристики	0,12
4.2 3 характеристики	0,18
4.3 Свыше 3-х характеристик	0,26

Поправочный коэффициент, учитывающий степень использования при разработке проекта стандартных модулей ($K_т$), определяется на основе данных представленных в таблице 4.2 и составляет 0,7.

Таблица 4.2 – Значения поправочного коэффициента, учитывающего использование стандартных модулей типовых программ и ПО ($K_т$)

Степень охвата реализуемых функций разрабатываемого ПО стандартными модулями, типовыми программами и ПО	Значения $K_т$
1 От 60 % и выше	0,6
2 От 40 % до 60	0,7
3 От 20 % до 40 %	0,8
4 До 20 %	0,9
5 Типовые программы и ПО не используемые для реализации	1,0

функций разрабатываемого ПО	

Поправочный коэффициент, учитывающий новизну разрабатываемого проекта ($К_н$) определяется на основе данных, представленных в таблице 4.3 и составляет 0,9.

Таблица 4.3 – Поправочные коэффициенты, учитывающие новизну ПО

Категория новизны	Степень новизны	Использование		Значение $К_н$
		На основе нового типа ПК	В среде новой ОС	
А	Принципиально новые ПО, не имеющие доступных аналогов	+	+	1,75
		–	+	1,6
		+	–	1,2
		–	–	1,0
Б	ПО, являющиеся развитием определенного параметрического ряда ПО	+	+	1,0
		–	–	0,9
		+	–	0,8
В	ПО, являющиеся развитием определенного параметрического ряда ПО, разработанных для ранее освоенных типов конфигурации ПК и ОС	–	–	0,7

Базой для определения нормативной трудоемкости являются, укрупненные нормы времени на разработку проекта в зависимости от уточненного объема ПО и группы сложности.

Нормативная трудоемкость проекта ($Т_н$) определяется на основе принятого в расчет объема ПП и категории сложности, которая уточняется с учетом сложности и новизны проекта и степени использования стандартных модулей при разработке.

Учитывая данные из Приложения В: для 1–ой категории сложности ПО

$$Т_н = 206$$

Общий объем трудоемкости исходя из формулы (4.4)

$$Т_о = 206 \cdot 0,18 \cdot 0,7 \cdot 0,9 = 23,36 \text{ чел./час}$$

На основе трудоемкости определяются плановое число разработчиков ($Ч_р$) и плановые сроки, необходимые для реализации проекта в целом ($Т_р$). При этом могут решаться следующие задачи:

– расчет числа исполнителей при заданных сроках разработки проекта;
– определение сроков разработки проекта при заданной численности исполнителей.

Численность исполнителей проекта ($Ч_р$) рассчитывается по формуле:

$$Ч = Т_о / (Т_р \cdot Ф_{эф)} \quad (4.5)$$

где $Ф_{эф}$ − эффективный фонд времени[10] работы одного работника в течение года (дн.);

$Т_о$ − общая трудоемкость разработки проекта (чел./дн.);

$Т_р$ − срок разработки проекта (лет).

Срок разработки проекта ($Т_р$) определяется по формуле

$$Т_р = Т_о / (Ч_р \cdot Ф_{эф}) \quad (4.6)$$

где $Ч_р$ – плановое число разработчиков.

Эффективный фонд времени работы одного работника ($Ф_{эф}$) рассчитывается по формуле:

$$Ф_{эф} = Д_г – Д_п – Д_в - Д_о \quad (4.7)$$

где $Д_г$ − количество дней в году;

$Д_п$ – количество праздничных дней в году;

$Д_в$ − количество выходных дней в году;

$Д_о$ − количество дней отпуска.

Т.к., в соответствии с производственным календарем на 2016 год:

$Д_г = 366$;

$Д_п = 15$;

$Д_в = 112$;

$Д_о = 14$, эффективный фонд времени одного работника составит:

$$Ф_{эв} = 366 - 15 - 112 – 14 = 225 \text{ дней}$$

Плановое число разработчиков $Ч_р = 1$, следовательно, по формуле (4.6)

$$Т_р = 23{,}36 / (1*225) = 0{,}1 \text{ лет} = 37 \text{ дней}$$

Таким образом, согласно произведенным расчетам и в соответствие с формулой (4.5)

$$Ч = 23,36 / (0,1*225) = 1 \text{ чел.}$$

Основная заработная плата исполнителей на конкретное ПО рассчитывается по формуле

$$З_{оі} = \sum_{i=1}^{n} Т_{чi} \cdot Т_ч \cdot К \quad (4.8)$$

где n − количество исполнителей, занятых разработкой конкретного ПО;

$Т_{чі}$ − часовая тарифная ставка і–го исполнителя (тыс. тенге);

$Ф_п$ − плановый фонд рабочего времени і–го исполнителя (дней), число рабочих дней в месяц;

$Т_ч$ − количество часов работы в день (час), 8 часов; К − коэффициент премирования, составляет 1,38.

Так как на разработку уйдет 2 месяца, а число рабочих дней в каждом месяце разное, возьмем среднее значение

$Ф_п$ = 22 раб. дней

По данным о специфике и сложности выполняемых функций составляется штатное расписание группы специалистов–исполнителей, участвующих в разработке ПО, с определением образования, специальности, квалификации и должности (таблица 4.4).

Таблица 4.4 - Сведения по работникам, задействованным в проекте

Специалист-Исполнитель	Количество, чел-к	Заработная плата в месяц, тенге
Разработчик	1	140 000,00
Итого	1	140 000,00

Часовая тарифная ставка рассчитывается путем деления месячной тарифной ставки, установленную при 40–часовой недельной норме рабочего времени и общего фонда времени (Фр)

$$Т_ч = \frac{Т_м}{Ф_р} \quad (4.9)$$

где $Т_ч$ − часовая тарифная ставка (тыс.тенге);

$Т_м$ − месячная тарифная ставка (тыс.тенге).

Общий фонд времени

$$Ф_р = Т_ч \cdot Ф_п \quad (4.10)$$

Таким образом

$$Ф_р = 8 \cdot 22 = 176 \text{ часов}$$

Тарифная ставка разработчика проекта

$$Т_ч = 140000 / 176 = 795 \text{ тенге в час}$$

В соответствие с формулой (4.8) основная заработная плата разработчика составит

$$З_{оі} = 795 \cdot 352 \cdot 1{,}38 = 386180 \text{ тенге}$$

Результаты расчета основной заработной платы представлены в виде таблице 4.5.

Таблица 4.5 – Сводные результаты расчета затрат основной заработной платы

Наименование содержания работ	Исполнитель	Трудоёмкость норма–час	Заработная плата за час работы, т/час	Сумма заработной платы, тенге
ТЗ	Разработчик	80	795	63600
Моделирование	Разработчик	80	795	63600
Программирование	Разработчик	100	795	79500
Тестирование	Разработчик	50	795	39750
Внедрение	Разработчик	42	795	33390
Итого		352		279840

Дополнительная заработная плата составляет 23% от основной заработной платы и рассчитывается по формуле

$$З_{ді} = З_{оі} \cdot Н_д / 100 \quad (4.11)$$

где $Н_д$ – коэффициент дополнительной заработной платы разработчика

$$З_{ді} = 386180 \cdot 23 / 100 = 88821 \text{ тенге}$$

Итоговый доход разработчика равен

$$З_{фот} = 386180 + 88821 = 475001 \text{ тенге}$$

Социальный налог составляет 11% (ст. 358 п. 1 НК РК) от дохода работника, и рассчитывается по формуле:

$$З_{сзі} = (ФОТ – ПО) \cdot 11\% \quad (4.12)$$

где ПО – пенсионные отчисления, которые составляют 10% от ФОТ и социальным налогом не облагаются

$$ПО = ФОТ \cdot 10\ \% \quad (4.13)$$

Таким образом

$$ПО = 475001 \cdot 0{,}1 = 47500 \text{ тенге}$$

$$З_{сзі} = (475001 – 47500) \cdot 0{,}11 = 47025{,}11 \text{ тенге}$$

Затраты на материалы определяются по формуле

$$М_і = (З_{оі} \cdot Н_{мз}) / 100\% \quad (4.14)$$

где $Н_{мз}$ − норма расхода материалов от основной заработной платы (3–5%)

$$М_і = 475001 \cdot 0{,}03 = 14250{,}03 \text{ тенге}$$

Расходы по статье «Машинное время» ($Р_{мі}$) включают оплату машинного времени, необходимого для разработки и отладки ПО, которое определяется по нормативам (в машино – часах) на 100 строк исходного кода (Нмв) машинного времени в зависимости от характера решаемых задач и типа ПК

$$Р_{мі} = Ц_{мі} \cdot (V_{оі} / 100) \cdot Н_{мв} \quad (4.15)$$

где $Ц_{мі}$ − цена одного машино–часа (тыс.тенге);

$V_{оі}$ − общий объем ПО (строк исходного кода);

$Н_{мв}$ − норматив расхода машинного времени на отладку 100 строк исходного кода (машино–часов).

Норматив расхода машинного времени на отладку 100 строк исходного кода определяется на основе таблицы Д.1 (Приложения Д) и составляет 12 ч/100 строк кода

$$Р_{мі} = 795 \cdot (6500 / 100) \cdot 12 = 620100 \text{ тенге}$$

Расходы по статье «Прочие затраты» ($П_{зі}$) на конкретное ПО включают затраты на приобретение и подготовку специальной научно–технической информации и специальной литературы. Определяются по нормативу, разрабатываемому в целом по организации, в процентах к основной заработной плате

$$П_{зі} = З_{оі} \cdot Н_{пз} / 100 \quad (4.16)$$

где $Н_{пз}$ − норматив прочих затрат в целом по организации в (20%)
Таким образом

$$П_{зі} = 475001 \cdot 0,2 = 95000,2 \text{ тенге}$$

Затраты по статье «Накладные расходы» ($Р_{ні}$), рассчитывается по нормативу ($Н_{рн}$) в процентном отношении к основной заработной плате исполнителей. Норматив устанавливается в целом по организации

$$Р_{ні} = З_{оі} \cdot Н_{рн}/100\% \quad (4.17)$$

где $Р_{ні}$ – накладные расходы на конкретную ПО (тыс.тенге);
$Н_{рн}$ – норматив накладных расходов в целом по организации 70%.
Таким образом, накладные расходы составят

$$Р_{ні} = 475001 \cdot 0,7 = 332500,7 \text{ тенге}$$

Полные затраты на разработку мобильного приложения, формула (4.1)

$С_{пі}$ = 475001 + 47025,11 + 14250,03 + 620100 + 95000,2+ 332500,7 = 1583877 тенге

Сводные результаты расчета затрат на разработку ПО и их структура представлены в таблице 4.6 и на рисунке 4.1.

Таблица 4.6 – Сводные результаты расчета затрат на разработку ПО

Затраты на разработку	Условное обозначение	Значение, тенге	В процентах от общей суммы
Фонд оплаты труда	$З_{фот}$	475001	29
Социальный налог	$З_{сзі}$	47025,11	2
Материалы	$М_і$	14250,03	1
Машинное время	$Р_{мі}$	620100	39
Прочие затраты	$П_{зі}$	95000,2	5

Накладные расходы	$Р_{ні}$	332500,7	17
Итого		1583877	100

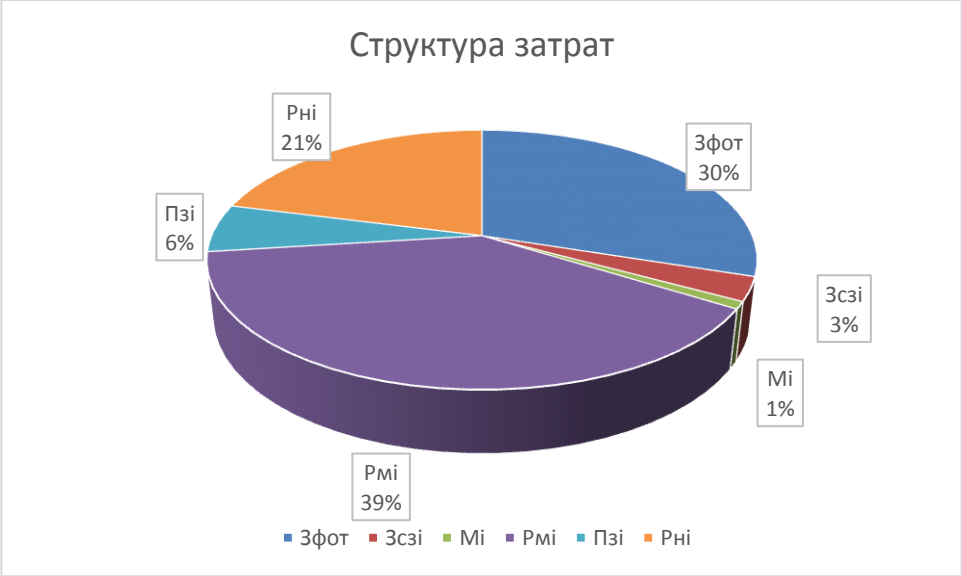

Рисунок 4.1 – Структура затрат на разработку мобильного приложения

4.2 Расчет цены программного продукта

Расчет цены ПП, который разработан [11] одной организаций по заказу другой и не предназначен для тиражирования, осуществляется по формуле

$$Ц_{пп} = З_{рпр} + П_{п} + НДС \quad (4.18)$$

где $Ц_{пп}$ – цена программного продукта, тенге;

$З_{рпр}$ – затраты на разработку проектного решения, в данном случае программного продукта, тенге;

$П_{п}$ – планируемая прибыль, тенге;

НДС – налог на добавленную стоимость, тенге.

Планируемая прибыль составляет (20%) от себестоимости разработки

$$П_{п} = 1583877 \cdot 0,2 = 316775,4 \text{ тенге}$$

НДС, начисленный на ПП, определяется следующим образом

$$НДС = (З_{рпр} + П_{п}) \cdot k_{НДС} \quad (4.19)$$

где $k_{НДС}$ – ставка налога на добавленную стоимость.
Подставив данные в формуле (4.19) получаем

$$НДС = (1583877 + 316775{,}4) \cdot 0{,}12 = 228078{,}29 \text{ тенге}$$

Подставив данные в формуле (4.18) получаем

$$Ц_{пп} = 1583877 + 316775{,}4 + 228078{,}29 = 2128731{,}69 \text{ тенге}$$

4.3 Вывод экономической части

Разработка мобильного приложения является дорогостоящим проектом, требующего значимых интеллектуальных и денежных затрат, а также обязательного использования компьютерной техники и программного обеспечения. В экономической части дипломной работы был произведен расчет затрат на разработку мобильного приложения и расчет цены программного продукта. Итоговая стоимость разработки базы знаний для прогнозирования спортивных событий составила 884900 тенге, которая состоит из всех возможных затрат при разработке проекта.

Программный продукт создан для пользования на платной основе. Проект является целесообразным, так как в настоящее время область азартных игр и ставок имеет очень высокую популярность. Огромное количество людей рассчитывают получить прибыль на ставках, данный программный продукт даст им больше возможностей. Разработка данного программного продукта является социально эффективной, так как он поможет людям в достижении успеха, предоставляя всю возможную информацию и статистику. Мобильное приложение предоставляет удобный интерфейс, помогает человеку рассчитать риски и выбрать оптимальную ставку на спортивное событие.

5. Безопасность жизнедеятельности

5.1 Анализ рабочего места

Аналитика необходимых условий труда в дипломной работе «Разработка мобильного приложения» даст нам возможность рассчитать рациональные и удобные условия труда, главные потенциально опасные источники риска жизнедеятельности. Усталость работников подразделена на два раздела. Один из этих разделов обосновывается характером данного трудового процесса, а следующий - результатом каких - либо ошибок организации труда, то есть недостаточно культурным уровнем труда.

В ходе проведенной аналитики и оценки имеющихся условий трудовой деятельности разработчиком было выявлено, что разработчик программного приложения на своем рабочем месте подвергается некоторым вредным факторам:

– Не всегда благоприятные микроклиматические условия. Они определяются температурой воздушных масс, их составом и атмосферным давлением, относительной влажностью, быстротой движения воздушных масс;

– Электромагнитное облучение, главным источником является Дисплей компьютера;

– Психофизиологические причины, то есть умственное перенапряжение визуальных и слуховых анализаторов, монотонность трудовых заданий, душевные тяготы и перегрузки и другие.

– Плохая освещенность рабочих мест, вследствие не правильного месторасположения и использования осветительных приборов;

Разработчик создал нужные условия для эффективных экономных и обычных рабочих движений. Беря во внимание изложенные причины, плохо влияющие на трудовые условия инженера-программиста, а также те необходимые требования, которые предъявляются при организации рабочего места разработчиком выяснено, что долгое пребывание человека в месте комбинированного действия различных не очень благоприятных факторов может привести к профессиональной болезни. Правильно устроенное рабочее место позволяет несколько повысить производительность труда и уменьшить вредное воздействие компьютера на здоровье. Развитию усталости на производстве способствует ошибочное эргономическое устройство рабочего места, ошибочные зоны размещения оборудования по высоте от пола, по фронту от оси симметрии и т.д. Правильное решение и обеспечение выполнения условий, влияющих на условия труда, делают труд более производительным, снижается утомляемость, уменьшается травматизм и профессиональные болезни. На месте работы обеспечиваются оптимальные параметры микроклимата. На работах, где люди чаще всего выполняют свою работу сидя и не требующих физического напряжения, температура

воздушных масс в холодный период года составляет от 20 до 23^0С, в теплый период года — от 21 до 26^0С. Относительная влажность воздушных потоков на постоянных рабочих местах обычно составляет 42-63%, быстрота движения воздушных потоков приблизительно равна 0,2 м/с.

Далее необходимо продемонстрировать правильно организованное рабочее место разработчика:

– размеры помещения: 4,2 х 2,4 х 2,9 м;
– Два окна, 1х1 м, окно закрывается жалюзями;
– Не натуральное освещение – две люминесцентных лампы;
– 1 рабочее место;
– вид работы – работа за компьютером, сидя;
– внутренняя отделка стен - темная;
– один диван, одно кресло, один журнальный столик, рабочее кресло и стол.

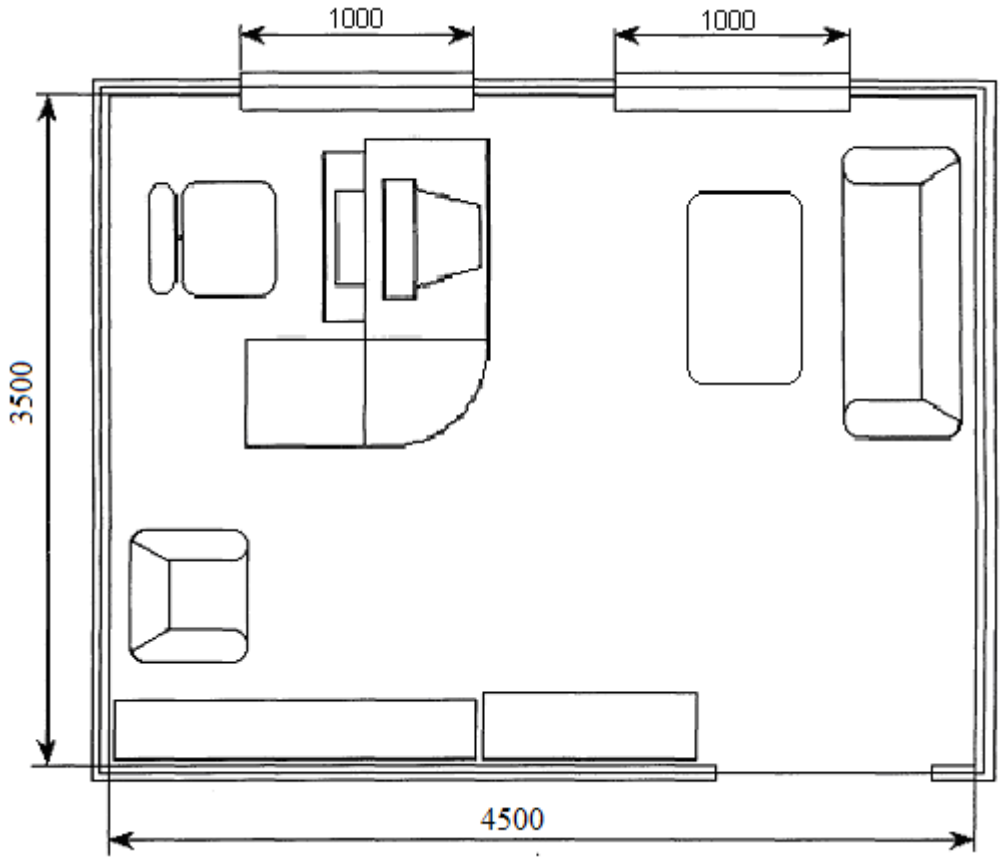

Рисунок 5.1 - План рабочего места.

5.2 Определение освещенности рабочих мест

Неблагоприятными факторами для профессиональной деятельности

разработчика, негативно влияющими на зрение, являются:

– пониженный уровень освещенности, приводящий к перенапряжению глаз и к быстрому утомлению;

– чрезмерно высокая освещенность также является причиной быстрой утомляемости, приводя к раздражению и рези в глазах;

– неправильное направление света способствует появлению резких теней или сильных бликов, заметно быстрее утомляющих глаза.

Для освещения помещений и рабочих мест с ПК должно применяться естественное, искусственное и совмещенное освещение. В преимущественном варианте желательно применение естественного освещения, так как за счет естественного освещения стимулируется обмен веществ, кровообращение, дыхание, деятельность центральной нервной системы, что в свою очередь, обеспечивает высокую производительность труда, также позволяет экономить электроэнергию. Однако освещенность на поверхности стола в зоне размещения рабочего документа, оборудования разработчика должна быть 300-500 лк и в случае ее не соблюдения, разработчик вынужден прибегать к использованию искусственного освещения. Искусственное освещение в помещениях, в которых эксплуатируются ПК, должно осуществляться системой общего равномерного освещения. Рабочее помещение имеет естественное освещение в виде двух окон размером 1000х1000 мм, данная степень освещенности указывает о необходимости использования системы общего освещения (искусственное освещение) с помощью люминесцентных ламп.

Для расчета искусственного освещения разработчиком было найдено и определено необходимое число ламп при помощи метода коэффициента использования. Расчёт системы общего освещения производился методом коэффициента использования светового потока, который выражается отношением светового потока, падающего на расчётную поверхность, к суммарному потоку всех ламп. Его величина зависит от характеристик светильника, размеров помещения, окраски стен и потолка, характеризуемой коэффициентами отражения стен и потолка.

Необходимый световой поток лампы в каждом светильнике рассчитывается по формуле:

$$F = \frac{E \cdot S \cdot K \cdot z}{N \cdot \eta}, \text{Лм} \qquad (5,1)$$

где E – заданная минимальная освещенность, лк (E = 500);

K – коэффициент запаса (K = 1,5);

S – освещаемая площадь, м^2 (S = 20);

Z – коэффициент неравномерности освещения (обычно принимается равным 1.1-1.2, для люмин. ламп - 1,1);

N – число светильников, шт;

η – коэффициент использования светового потока, т.е. отношение

потока, падающего на расчетную поверхность, к суммарному потоку всех ламп; находится в зависимости от величины индекса помещения (i) и коэффициента отражения потолка и стен. Индекс помещения определяется по выражению:

$$i = \frac{S}{H \cdot (A + B)}, \qquad (5,2)$$

где h – высота подвеса светильника над рабочей поверхностью, м;

A, B - длина и ширина помещения, м.

Коэффициенты отражения выбираются в зависимости от состояния производственных помещений.

Коэффициенты отражения от потолка стен и пола соответственно равны: $\rho_п = 70\%$; $\rho_с = 50\%$; $\rho_р = 30\%$.

Найдем h по формуле:

$$h = H - h_р - h_с = 3,2 - 0,75 - 0,05 = 2,4 \text{ (м)}$$

где H – высота помещения, м (H = 3,2);

$h_р$ – высота рабочей поверхности от пола, м ($h_р = 0,75$);

$h_с$ – высота свеса светильника от основного потолка, м ($h_с = 0,05$).

Наивыгоднейшее расстояние между светильниками определяется как

$$L = \lambda \cdot H, \qquad (5,3)$$

где $\lambda = 1,2 \div 1,4$.

$L = 1,2 \cdot 2,4 = 2,88$ м

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

$$l1 = (0,4 \div 0,5) \cdot L \qquad (5,4)$$

$$l_1 = 0,4 \cdot 2,88 = 1,152 \text{ м}$$

Определяем индекс помещения:

$$i = \frac{5 \cdot 4}{2,4 \cdot (5 + 4)} = 0,95$$

Коэффициент использования в данном случае равен $\eta = 65\%$, коэффициент запаса равен K = 1,5.

Для освещения выбираем люминесцентные лампы типа Е-27, световой поток которых $F_л = 4400$ Лм.

Исходя из формулы 2.1 определим количество люминесцентных ламп по формуле:

$$N = \frac{E \cdot S \cdot K \cdot z}{N\text{л. св} \cdot F\text{л} \cdot \eta} \qquad (5,5)$$

где $F_л$ – световой поток одной лампы;

$F_о$ – общий световой поток;

$N_{л.св}$ - количество ламп в светильнике;

N – число ламп;

η – коэффициент использования светового потока, η = 65%.

$$N = \frac{500 \cdot 1{,}5 \cdot 20 \cdot 1{,}1}{0{,}65 \cdot 2 \cdot 4400} = 2{,}88 \sim 3$$

Число светильников выбирается в зависимости от размеров освещаемого помещения, при этом количество светильников должно быть таким, чтобы отношение расстояния между ними к высоте их подвеса над поверхностью было равно 1,5 ÷ 2.

При выборе осветительных приборов используем светильники типа ЛСПО 2. Каждый светильник комплектуется двумя лампами. Размещаются светильники тремя рядами, по два в каждом ряду.

Допускается отклонение (ε) светового потока выбранной лампы от расчётного от – 10 % до + 10 %.

$$E\text{факт} = \frac{F\text{л} \cdot N \cdot N\text{л.св} \cdot \eta}{S \cdot K \cdot z} \qquad (5{,}6)$$

$$E\text{факт} = \frac{4400 \cdot 3 \cdot 2 \cdot 0{,}65}{20 \cdot 1{,}1 \cdot 1{,}5} = 520 \text{ лк}$$

Отличие от нормированного уровня

$$E\text{факт} = \frac{\text{Ефакт} - \text{Енорм}}{\text{Енорм}} \qquad (5{,}7)$$

$$E\text{факт} = \frac{520 - 500}{500} \cdot 100 = 4\% \text{ лк}$$

Для исключения засветки экранов дисплеев прямыми световыми потоками светильники общего освещения располагают сбоку от рабочего места, параллельно линии зрения оператора и стене с окнами. Такое размещение светильников позволяет производить их последовательное включение в зависимости от величины естественной освещённости и исключает раздражение глаз чередующимися полосами света и тени, возникающее при поперечном расположении светильников.

Электрическая мощность всей осветительной системы вычисляется по формуле:

$$P_{общ} = P_1 \cdot N \qquad (5,8)$$

где P_1 – мощность одной лампы, P_1= 65 Вт;

N – число ламп, N = 3.

$$P_{общ} = 65 \cdot 3 = 195 \text{ Вт}$$

Схему размещения и освещенности светильников в помещении можно рассмотреть на рисунке 5.2.

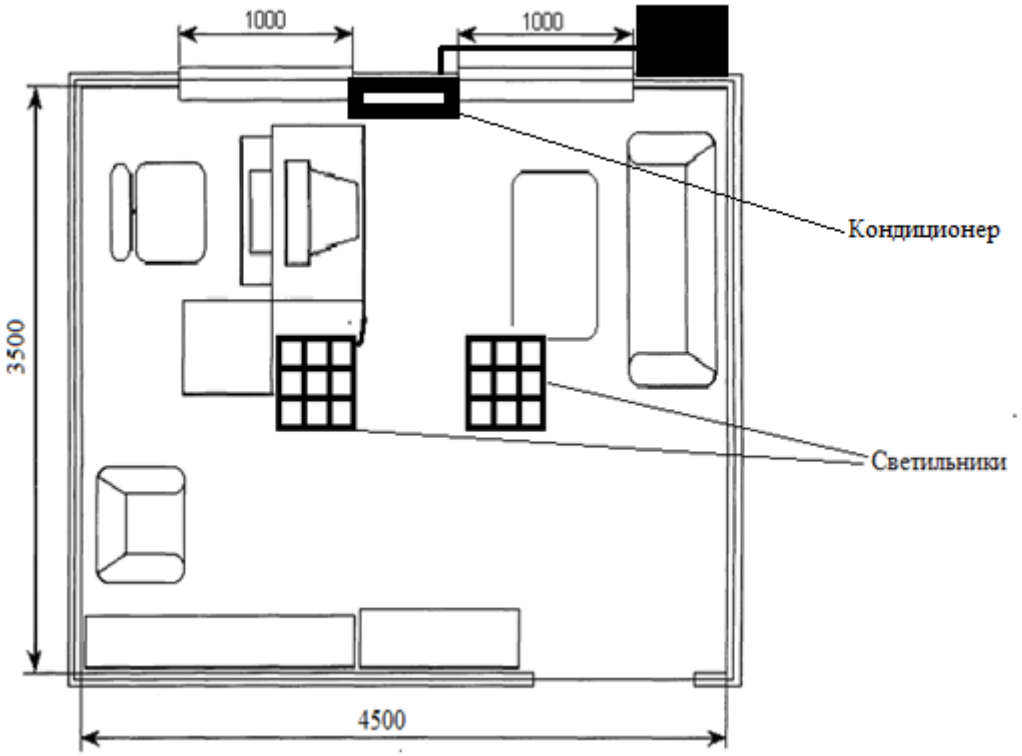

Рисунок 5.2 – Расположение светильников и кондиционера.

Всего для создания нормируемой освещенности 500 лк необходимо 3 люминесцентных ламп в 2 светильниках серии Е-27, мощность каждой лампы должна быть не меньше 65 Вт, что соответствует действительности, а значит имеющегося в наличии освещения достаточно для соответствия санитарным нормам.

Используется система общего освещения с люминесцентными лампами Е-27 с параметрами, указанными выше. Следовательно, для освещения рабочего места вполне достаточно общего освещения, при котором рабочее место освещают два светильника, в каждом светильнике по три лампы со световым потоком излучения 4400 лм каждая [12].

5.3 Расчет системы кондиционирования

Определим необходимое количество кондиционеров[13] для создания комфортных условий труда в помещении. В помещении за счёт тепловыделений производственного оборудования могут иметь место значительные избытки тепла (разность между тепловыделениями в помещении и теплоотдачей через стены, окна, двери и т.д.), удаление которых, прежде всего, должна обеспечить система вентиляции.

Избыточное тепло

$$Q_{изб} = Q_{ОБ} + Q_{ОС} + Q_{Л} + Q_{Р} + Q_{ОТД} \quad (5,9)$$

где $Q_{ОБ}$, $Q_{ОСВ}$, $Q_{Л}$ – тепло, выделяемое производственным оборудованием, системой искусственного освещения помещения и работающим персоналом (людьми) соответственно, ккал/ч;

$Q_{Р}$ – тепло, вносимое в помещение солнцем (солнечная радиация), ккал/ч;

$Q_{отд}$ – теплоотдача естественным путём, ккал/ч. Тепло, выделяемое производственным оборудованием:

$$Q_{об} = 860 \cdot P_{ОБ} \cdot \eta \quad (5,10)$$

где 860 – тепловой эквивалент 1 кВт/ч;

$P_{об}$– мощность, потребляемая оборудованием, кВт/ч;

η – коэффициент перехода тепла в помещение. Для 3 компьютеров имеем

$$Q_{об} = 860 \cdot (3 \cdot 0,250) \cdot 0,95 = 612,75 \text{ ккал/ч}$$

Значение $\eta = 0,95$ – норма потерь потребляемой мощности на тепловыделения компьютерного оборудования.

Тепло, выделяемое осветительными установками

$$Q_{осв} = 860 \cdot N \cdot \eta, \quad (5,11)$$

где N – расходуемая мощность светильников, кВт;

$\eta = 0,55$ – норма потерь потребляемой мощности на тепловыделения люминесцентных ламп.

$$Q_{осв} = 860 \cdot 0,55 \cdot 0,52 = 246 \text{ ккал/ч}$$

Тепло, выделяемое людьми

$$Q_Л = K_Л \cdot (q \cdot q_{исп}) \qquad (5,12)$$

где Кл – количество работающих;

(q · $q_{исп}$) – явное тепло, ккал/ч;

q – тепловыделения одного человека при данной категории работ I-III, ккал/ч.

Работа, производимая в помещении, относится к I категории работ. q = 100 Вт, или 0,1 кВт для офисных помещений

$$Q_л = 4 \cdot (860 \cdot 0,1) = 344 \text{ ккал/ч}$$

Тепло, вносимое солнечной радиацией:

$$Q_Р = m \cdot F \cdot q_{ост}, \qquad (5,13)$$

где m – количество окон в помещении;

F –площадь одного окна, $м^2$;

$q_{ост}$ – солнечная радиация через остеклённую поверхность, т.е. количество тепла, вносимое за один час через остеклённую поверхность площадью 1 $м^2$.

Для окна с двойным остеклением с деревянными переплетами $q_{ОСТ}$ =105 (окна выходят на север, Алматы находиться на широте 43° сев.широты). Количество окон равно 2. Площадь окна равна F = 2,4 · 2 = 4,8

Qp = 2 · 4,8 · 105 = 1008 ккал/ч

Для тёплого периода года при расчётах можно принять $Q_{отд}$=0.

$Q_{изб}$= 612,75 + 246 + 344 + 1008 = 2210,75 ккал/ч

При наличии теплоизбытков количество воздуха, которое необходимо удалить из помещения.

$$Lb = \frac{Q_{ИЗБ}}{Cb \cdot \Delta t \cdot \gamma b} \qquad (5,14)$$

где $Q_{изб}$ – избыточное тепло, ккал/ч;

Cb – теплоёмкость воздуха (0,24 ккал/кг°С)

Согласно формуле,

$\Delta t = t_{вых} - t_{вх}$

где $t_{вых}$ - температура воздуха выходящего из помещения, °С; $t_{ВХ}$ – температура воздуха поступающего в помещение, °С;

γ_b =1,206 кг/ м3 - удельная масса приточного воздуха.

Величина Δt при расчётах выбирается в зависимости от теплонапряжённости воздуха:

$$Q_{\text{н}} = \frac{Q_{\text{изб}}}{V_{\text{п}}} \qquad (5{,}15)$$

$$Q_{\text{н}} = \frac{2210{,}75}{108} = 20{,}47 \text{ ккал/м}$$

Если теплонапряжённость воздуха Qн < 20 ккал/м^3, то принимают значения Δt = 6°С, а при Q_H > 20 ккал/м^3, Δt =8°С.

$$Lb = \frac{2210{,}75}{0{,}24 \cdot 8 \cdot 1{,}206} = 955 \text{м}^3/\text{ч} \qquad (5{,}16)$$

Существующий кондиционер имеет расход воздуха 955 м^3/ч., что соответствует действительности и является достаточным для обеспечения комфортного микроклимата. Параметры кондиционера:

- производительность по холоду – 6155 (W);
- производительность по теплу – 6700 (W);
- потребляемая мощность в режиме охлаждения – 2250 (W);
- потребляемая мощность в режиме обогрева – 2250 (W);
- уровень шума внутреннего блока – 45 (dB (A));
- напряжение питания – 1 / 220/ 50 (Ph/V/Hz);
- вес внутреннего блока – 13 (kg).

Также кондиционер обладает небольшими габаритами, что важно при малом размере помещения.

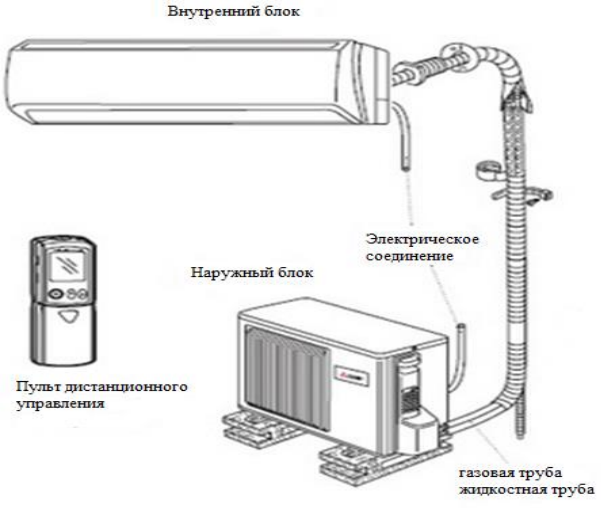

Рисунок 3 – Схема кондиционера.

Заключение

Разработка данного информационного продукта – очень увлекательная и непростая задача. При выполнении которой, необходимо было изучить и обработать большое количество информации в области ставок. Именно поиск и анализ этой информации позволил нам справиться с поставленными задачами. Были задействованы современные средства разработки: разные Фреймворки, утилиты, технологии. Особенно хотелось бы выделить среду разработки Android Studio IDE, которая поражает своим многообразием и многофункциональностью.

При реализации данного приложения для мобильных устройств было затрачено существенное количество времени на разработку интерактивного и понятного интерфейса пользователя. В разработке интерфейса значительную роль сыграл сайт поддержки дизайна, мобильных приложений Material Design.

В результате, получается, что исследуемая область является максимально востребованной в условиях современного рынка. Такие условия предполагают необходимость интерактивного двустороннего взаимодействия клиент-бизнес. Кроме того, в дополнение к удобному списку услуг, предоставляемых клиентским приложением. Этот программный продукт расширяет возможности мобильных приложений благодаря интеграции с веб-сайтами, которые позволяют загружать все последние обновления, получать информацию и новости в фоновом режиме. Кроме того, это приложение отвечает всем стандартам и требованиям платформ - носителей.

В технико-экономической части был произведен расчет затрат на разработку пользовательского интерфейса, а так же расчет реальной стоимости рассматриваемого приложения, с целью экономической выгоды.

Так же в части которая описывает безопасность жизнедеятельности были рассчитаны параметры искусственного освещения и системами циркуляции воздушных потоков в рабочем помещении.

Список литературы

1. Bert Altenburg, Alex Clarke и Philippe Mougin, «Become an Xcoder» – Creative Commons, 2008
2. Петцольд Ч. Программирование для Microsoft Windows 8. - 6-е изд. - П.: Питер, 2013. - 152 с.
3. Ликнесс Д. Приложения для Windows 8 на C# и XAML. - СПб.: Питер, 2013. - 268 с.
4. Хейлсберг А., Торгерсен М., Вилтамут С., Голд П. Язык программирования C#. Классика Computers Science. 4-е изд. - СПб.: Питер, 2011. - 189 с.
5. Стилмен Э., Грин Д. Изучаем C#. 3-е изд. - СПб.: Питер, 2014. – 12 с.
6. РихтерД. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. - СПб.: Питер, 2015. - 89 с.
7. 6 Петцольд Ч. Программирование для Microsoft Windows 8. - 6-е изд. - П.: Питер, 2013. - 152 с
8. ГОСТ 12.1.005-88 "Общие санитарно-гигиенические требования к воздуху рабочей зоны".
9. Аманбаев У.А. Экономика предприятия. - Алматы: «Бастау», 2012.
10. Буров В.П. Бизнес план фирмы. - М.: «Инфра М»,2011.
11. Куатова Д.Я. Экономика предприятия. - Аламаты: «Экономика», 2011.
12. Требования охраны труда при работе на персональных компьютерах. [Электронный ресурс] – Режим доступа: http://lib.rushkolnik.ru
13. Основы безопасности жизнедеятельности: Учебное пособие / сост.: М.К.Дюсебаев, Ж.С.Абдимуратов. - Алматы: АУЭС, 2013.- 79 с.

Приложение А

Листинг программной части

```
package com.example.batman.toolbarnavigationversion100500;

import android.os.Bundle;

import android.support.v4.app.Fragment;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.TextView;

/**

 * A simple {@link Fragment} subclass.

 */

public class ListItemOfExperts extends Fragment {

    String text_experts;

    TextView txt;

    public ListItemOfExperts() {

        // Required empty public constructor

    }

    @Override

    public View onCreateView(LayoutInflater inflater, ViewGroup container,

                             Bundle savedInstanceState) {

        View v =inflater.inflate(R.layout.list_item_of_experts, container, false);

        text_experts = this.getArguments().getString("expert");

        txt = (TextView) v.findViewById(R.id.expert_text);

        txt.setText(text_experts);

        return v;
```

```
    }
}
package com.example.batman.toolbarnavigationversion100500;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
/**
 * A simple {@link Fragment} subclass.
 */
public class ListItemOfNews extends Fragment {
    public ListItemOfNews() {
        // Required empty public constructor
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.list_item_of_news, container, false);
        String item_of_news = this.getArguments().getString("item_of_news");
        Log.d("@@@@@@@@@@@@@@@@", item_of_news);
        TextView item_txt = (TextView)v.findViewById(R.id.item_txt);
        item_txt.setText(item_of_news);
```

```
        return  v;
    }
}
 // ArrayList<String> mat4i2=new ArrayList<String>();
    ArrayAdapter<String> adapter;
    ArrayList<String> isxod = new ArrayList<String>();
    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        mat4i = this.getArguments().getStringArrayList("mat4i");
       // mat4i2 = this.getArguments().getStringArrayList("ttt");
        for(String s:mat4i){
            Log.d("%%%%%%%%%%%%%",s);
        }
//        for(String s:mat4i2){
//            Log.d("-----------",s);
//        }
        adapter = new ArrayAdapter<String>(getActivity(),android.R.layout.simple_list_item_1,mat4i);
        setListAdapter(adapter);
    }
    public ListOfMatches() {
        // Required empty public constructor
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
```

```
                    Bundle savedInstanceState) {
    View v = inflater.inflate(R.layout.fragment_list_of_matches, container, false);
    return v;
}
@Override
public void onListItemClick(ListView l, View v, int position, long id) {
    super.onListItemClick(l, v, position, id);
    TextView txt = (TextView)v;
    String item_list=txt.getText().toString();
    //Log.d("++++++++++++",item_list);
    DbHelper mDbHelper = new DbHelper(getActivity(),"myDataBase.db",null,1);
    SQLiteDatabase sdb = mDbHelper.getWritableDatabase();
    String selection ="Match LIKE ?";
    String[] selectionArgs = new String[]{item_list};
    Cursor cursor = sdb.query("Bets", null, selection, selectionArgs, null, null, null);
    cursor.moveToFirst();
    String Match = cursor.getString(cursor.getColumnIndex(DbHelper.Bets_match));
    String p1 = cursor.getString(cursor.getColumnIndex(DbHelper.Bets_P1));
    String x = cursor.getString(cursor.getColumnIndex(DbHelper.Bets_X));
    String p2 = cursor.getString(cursor.getColumnIndex(DbHelper.Bets_P2));
    Log.d("DAAAAAAVAAAAAIII JEEEEE", Match + " " + p1 + " " + x + " " + p2);
    cursor.close();
    sdb.close();
    isxod.add(Match);
    isxod.add(p1);
    isxod.add(x);
```

```
        isxod.add(p2);
        DialogFragment dlg = new Dialog_ListOfMatches();
        Bundle bundle = new Bundle();
        bundle.putStringArrayList("asd", new ArrayList<String>(isxod));
        dlg.setArguments(bundle);
        dlg.show(getFragmentManager(), "dlg");
        isxod.clear();
    }
}
package com.example.batman.toolbarnavigationversion100500;
import android.content.res.Configuration;
import android.database.sqlite.SQLiteDatabase;
import android.support.v4.app.DialogFragment;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentTransaction;
import android.support.v4.app.FragmentManager;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
```

```
import android.widget.ArrayAdapter;
import android.widget.ListView;
public class MainActivity extends AppCompatActivity {
    private Toolbar toolbar;
    public static String USER_LOGIN = "";
    public static String USER_PASSWORD = "";
    Fragment fragment = null;
    private FragmentTransaction fragmentTransaction;
    private DrawerLayout drawerLayout;
    private ActionBarDrawerToggle toggle;
    private String mActivityTitle;
    private ListView listView_navigation_drawer;
    private String[] items = {"News","Bets","Experts","About Us","History"};
    private ArrayAdapter<String> adapter;
    //
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        fragment = new FragmentNews();
        fragmentTransaction = getSupportFragmentManager().beginTransaction();
        fragmentTransaction.replace(R.id.fragment_holder, fragment);
        fragmentTransaction.addToBackStack(null);
        fragmentTransaction.commit();
        toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
```

```
        drawerLayout = (DrawerLayout) findViewById(R.id.drawerlayout);
        mActivityTitle = getTitle().toString();
        listView_navigation_drawer = (ListView) findViewById(R.id.navigationList);
        adapter = new ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,items);
        listView_navigation_drawer.setAdapter(adapter);
        listView_navigation_drawer.setOnItemClickListener(new DrawerItemClickListener());
        setupDrawer();
    }
    private void setupDrawer(){
        toggle = new ActionBarDrawerToggle(this,drawerLayout,R.string.drawer_open,R.string.drawer_close){
          public void onDrawerOpened(View drawerView){
              super.onDrawerOpened(drawerView);
              getSupportActionBar().setTitle("Menu");
              invalidateOptionsMenu();
          }
          public void onDrawerClosed(View drawerView){
              super.onDrawerClosed(drawerView);
              getSupportActionBar().setTitle("Betting Forecast");
              invalidateOptionsMenu();
          }
        };
        toggle.setDrawerIndicatorEnabled(true);
        drawerLayout.setDrawerListener(toggle);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setHomeButtonEnabled(true);
```

```
}
@Override
protected void onPostCreate(Bundle savedInstanceState) {
    super.onPostCreate(savedInstanceState);
    toggle.syncState();
}
@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
    toggle.onConfigurationChanged(newConfig);
}
@Override
public void onBackPressed() {
    FragmentManager fm = getSupportFragmentManager();
    if (fm.getBackStackEntryCount() > 0) {
        fm.popBackStack();
    } else {
        super.onBackPressed();
    }
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if(toggle.onOptionsItemSelected(item)){
        return true;
    }
```

```
        return super.onOptionsItemSelected(item);
    }
    private class DrawerItemClickListener implements ListView.OnItemClickListener {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            selectItem(position);
        }
    }
    private void selectItem(int position){
        switch (position){
            case 0 :
                fragment = new FragmentNews();
//                fragmentTransaction = getFragmentManager().beginTransaction();
//                fragmentTransaction.replace(R.id.fragment_holder, fragment);
//                fragmentTransaction.addToBackStack(null);
//                fragmentTransaction.commit();
                break;
            case 1 :
                fragment = new FragmentBets();
//                fragmentTransaction = getFragmentManager().beginTransaction();
//                fragmentTransaction.replace(R.id.fragment_holder, fragment);
//                //fragmentTransaction.addToBackStack(null);
//                fragmentTransaction.commit();
                break;
            case 2 :
                fragment = new FragmentExperts();
```

```
//              fragmentTransaction = getFragmentManager().beginTransaction();
//              fragmentTransaction.replace(R.id.fragment_holder, fragment);
//             //  fragmentTransaction.addToBackStack(null);
//              fragmentTransaction.commit();
                break;
            case 3 :
                fragment = new FragmentAbout();
//              fragmentTransaction = getFragmentManager().beginTransaction();
//              fragmentTransaction.replace(R.id.fragment_holder, fragment);
//              // fragmentTransaction.addToBackStack(null);
//              fragmentTransaction.commit();
                break;
            case 4 :
                fragment = new Fragment_History();
                break;
            default:
                break;
        }
        if(fragment != null) {
            fragmentTransaction = getSupportFragmentManager().beginTransaction();
            fragmentTransaction.replace(R.id.fragment_holder, fragment);
            fragmentTransaction.addToBackStack(null);
            fragmentTransaction.commit();
        }else {
            // Error
```

```
            Log.e(this.getClass().getName(), "Error. Fragment is not created");
        }
        listView_navigation_drawer.setItemChecked(position, true);
        drawerLayout.closeDrawer(listView_navigation_drawer);
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
//        DbHelper mDbHelper = new DbHelper(this,"myDataBase.db",null,1);
//        SQLiteDatabase sdb = mDbHelper.getWritableDatabase();
//        sdb.delete("Bets", null, null);
//        MainActivity.USER_LOGIN = "";
//        if(this.deleteDatabase("myDataBase.db")){
//            Log.d("On DESTROY", "DATABASE DELETED");
//        }
    }
}
```