Geekbrains

**Разработка мобильного приложения с возможностью отслеживания количества выполненных задач   
и учета не выполненных**

IT-специалист:

Зубарев О.В.

Волгоград

2023

СОДЕРЖАНИЕ

[Введение 3](#_Toc105883049)

[1 Теоретические аспекты создания мобильного приложения 6](#_Toc105883050)

[1.1 Мобильные приложения и их особенности 6](#_Toc105883051)

[1.2 Анализ и общая характеристика мобильных приложений 8](#_Toc105883052)

[1.3 Анализ средств разработки и обоснование выбора технологии проектирования 12](#_Toc105883053)

[1.4 Общий алгоритм реализации мобильного приложения 18](#_Toc105883054)

[2 Разработка мобильного приложения «Effektus» 19](#_Toc105883055)

[2.1 Будущие компоненты приложения 19](#_Toc105883056)

[2.2 Требования и жизненный цикл приложения 27](#_Toc105883057)

[2.3 Сохранение состояния приложения 36](#_Toc105883058)

[2.4 Разработка пользовательского интерфейса 38](#_Toc105883059)

[3 Результаты разработки мобильного приложения «Effektus» 47](#_Toc105883060)

[3.1 Хранение данных 47](#_Toc105883061)

[3.2 Выбор метода синхронизации 53](#_Toc105883062)

[3.3 Дальнейшее развитие и сопровождение приложения 54](#_Toc105883063)

[Заключение 56](#_Toc105883064)

[Список использованных источников 57](#_Toc105883065)

ВВЕДЕНИЕ

В современном мире стремительными темпами раскручиваются информационные и мобильные технологии, и до сих пор огромное влияние в нашу жизнедеятельность выражается мобильными приложениями и всемирная паутина. В настоящее время малограмотным человеком значится не тот, кто никак не может научиться считать математические задачи или сочинять, а тот, кто не может пользоваться смартфоном, компьютером или же интернетом, однако подобных людей с каждым годом остается все меньше и меньше.

С возникновением сети интернет на него моментально устремил внимание бизнес, так как это самое пространство могло стать превосходной нишей с целью реализации продукции и услуг. Точно так и произошло, обороты фирм возросли в десятки и сотни раз, и это очень повлияло на будущее развитие технологий и новаторскому созданию мобильных приложений.

С помощью мобильных приложений мы можем сейчас оплатить парковку буквально с одного клика телефона. Существующие карты в навигаторах дают возможность не потеряться в любом районе неизвестного города. Нам доступна возможность просмотра расписания автотранспорта. В конце концов, мы можем элементарно узнавать и изучать увлекательные площади, которые есть поблизости.

Кроме перемен в нашей социальной активности и легкодоступности информации, мобильные приложения в недалеком будущем смогут начать воздействовать на наше состояние здоровья. В нашу жизнедеятельность стабильно уместились мобильные технологии, которые кардинальным образом улучшают и ход производства, и общее направление использования информации. Внедрение мобильных технологий дает возможность быть в курсе всех событий мира, прилагая к этому минимальное количество усилий. Помимо этого, мобильные приложения позволяют понизить цену продукта для покупателей благодаря оптимизации процессов, уменьшения производственных потерь и непроизводственных расходов.

Используя мобильные устройства, мы совсем не привязаны к рабочему месту и без особого труда можем получить доступ к нужным данным в пути, в цеху, на совещании и где–либо еще, потратив на это минимальное количество денег и сил.

Всё это только незначительная часть того, как смартфоны модифицировали, улучшили и облегчили жизнедеятельность большого количества людей. И данный процесс улучшается в соответствии с развитием самих технологий. Нам очень посчастливилось жить в эру информационных технологий. Они за какие-то 10 лет стали не подменными помощниками. Кто из нас временами не чувствовал себя без смартфона как без рук? Однако новые возможности принесли с собой и свежеиспеченные проблемы. У любой технологии имеются свои плюсы и минусы, с коими нужно или мириться, или искать возможности их решения.

Актуальность выбранной темы формируется из учета выполненных задач, задач в работе и анализе эффективности, продуктивности сотрудников.

Объектом данного исследования является отдел сопровождения закупок.

Предмет исследования – мобильное приложение с возможностью отслеживания количества выполненных задач и учета не выполненных.

Целью выпускной квалификационной работы является разработка мобильного приложения, с помощью программного обеспечения Android Studio, для поддержания стабильности работы сотрудников, путем учета выполненных рабочих поручений.

Для достижения указанной цели поставлены следующие задачи:

* изучить теоретико-методологические основы разработки мобильных приложений;
* изучить особенности различных мобильных систем;
* проанализировать программное обеспечение для реализации данной работы и выбрать наиболее подходящее;
* проектирование разделов мобильного приложения;
* разработка разделов с различными видами результатов.

Выпускная квалификационная работа состоит из введения, трех глав, заключения и списка литературы.

В первой главе «Теоретические аспекты создания мобильного приложения» рассмотрены различные операционные системы, для различных мобильных телефонов.

Во второй главе «Разработка мобильного приложения «Effektus»» рассмотрены аспекты мобильного приложения, спроектированы разделы будущего проекта.

Третья глава «Результаты разработки мобильного приложения» содержит информацию про выбор типа хранения данных и про дальнейшее возможное развитие мобильного приложения.

При написании работы были использованы материалы научных статьей, семинаров, нормативно-правовых актов, а также интернет-сайтов.

Данная выпускная квалификационная работы содержит 7 рисунков. Объем работы составляет 60 страниц.

# 1 Теоретические аспекты создания мобильного приложения

## 1.1 Мобильные приложения и их особенности

Совершенствование информационных технологий (ИТ)и многочисленное увеличение смартфонов дали возможность организовать новый рынок мобильных приложений. Они считаются одним из основных трендов крайних лет в сфере ИТ. Увеличивается численность разработчиков, возрастает число приложений и юзеров, а еще больше и больше фирм осмысляют потребность создания своих приложений ради повышения конкурентоспособности.

Подобный активный рост мобильных пользователей и стабильное развитие технологий показывают, что российский рынок мобильных приложений располагается в фазе энергичного роста. Наверное, это один из немногих рынков, который показывает обширный потенциал для ИТ–фирм.

Вначале, нужно обусловить само понятие «мобильное приложение». Мобильное приложение — это преднамеренно разработанное приложение на определенную мобильную платформу (iOS, Android, WindowsPhone). Почти все мобильные приложения предустановлены в самом смартфоне, либо могут быть загружены из специальных магазинов приложений, по типу App Store, Google play, Windows Phone Store, Yandex.store — и других, скачать можно как бесплатно, так и платно. Распространяющиеся мобильные приложения должны упростить жизнедеятельность юзеров смартфонов и разнообразить ее.

В настоящий момент у компаний в разнообразных сферах деятельности наступает понимание того, что в не далеком будущем разработка мобильных приложений станет необходима. Мобильное приложение иметь в своем распоряжении ряд превосходств в сравнении с мобильной версией вебсайта:

* более удобный и понятный интерфейс;
* программы лояльности онлайн;
* постоянная коммуникация с пользователем;
* геолокация;
* наиболее точный сбор данных о целевой аудитории.

Перед тем как разбирать виды приложений нужно понимать, какие приложения являются самыми популярными в окружении пользователей.

Большая часть (по статистике около 59%) используют мобильные приложения, которые устанавливают. Другая часть (41 %) посещают веб-сайты через смартфон. Треть (примерно — 31 %) пользуется возможностью сидеть в социальной сети посредством мобильного телефона. Некоторые люди сражаются в мобильных играх (примерно 44 %). Ориентировочно 3/4 людей пользуются общением при помощи смартфона: это далеко не только SMS сообщения, звонки, а еще мессенджер и социальные сети.

Разработка мобильных приложений нацелена на образовывание приложений, кои смогут прогнозировать потребительские предпочтения; принимать решения, способ принятия которых заблаговременно неведом. Сегодня приложения умеют делать исследовательский анализ данных, приобретенной из нескольких источников, и содействовать человеку в принятии решения, контролировать ход процессов и решать остальные важные вопросы с меньшими временными и умозаключительными затратами для него самого. Что содействует оптимизации бизнес–процессов, росту производительности и действенности принимаемых заключений. Мобильное приложение — это своего рода адаптер, который помогает человеку взаимодействовать с всевозможной информацией. Поэтому отличают такие виды приложений как:

* приложения–события, для стриминга спортивных или подобных мероприятия;
* приложения службы, которые являются аналогами сайтов;
* игры - развивающие и обучающие;
* интернет-магазины, для покупок в онлайн режиме;
* промо–приложения, нужные для рекламы различных брендов;
* бизнес–приложения, которые оптимизируют процесс работы
* организации, предоставляя доступ к информации и интеграцию с базами данных;
* системные приложения, использующие дополнительные настройки и опции смартфона;
* навигационные сервисы, применяющие GPS–модуль, что делают из телефона полноценный навигатор;
* мультимедийные приложения, расширяющие возможности телефона при работе с видео– аудио информацией;
* социальные сети, представляющие собой онлайн–сервисы для общения, распространения информации;
* контентные приложения и др.

1.2 Анализ и общая характеристика мобильных приложений

Инновационные технологии все совершеннее попадают в нашу повседневную жизнь и сейчас уже трудно наткнуться на человека, у которого нет телефона или же иного многофункционального гаджета. Их работа недопустима при отсутствии операционной системы и на настоящий время ОС Android считается наиболее известной. В соответствии с данным южноамериканской компании Gartner, что специализируется в разборе рынка информационных технологий, часть Android в 4 квартале 2022 собрала 72,4 %. Во всем мире было реализовано 1,41 миллиард устройств на данной операционной системе. Android считается условно свежей операционной системой, но за краткое время она оказалась популярна на рынке смартфонов.

Мысль создания ОС принадлежала фирме Android Inc. В 2005 году фирму приобрела компания Google и дала требуемые ресурсы и содействие ради продолжения разработки платформы до состояния конечного продукта. Создатели операционной системы обязаны были придерживаться некоторых ключевых условий: новая платформа должна быть базирована на открытых эталонах, позволяющих понизить издержки на разработку и дать создателям мобильных приложений возможность доступа к большинству функций системы для сотворения приложений. В первый раз платформу презентовали в 2007 году, главный механизм и реализация коей в большинстве своем была похожа на архитектуру современной системы. [20]

Также в 2007 году под руководительством Google был организован бизнесальянс разработчиков открытых стандартов мобильных устройств Open Handset Alliance, какой на настоящие время состоит из 87 компаний (Google Inc., Intel, Asus, Motorola, HTC, Samsung, Huawei и др.). Объединение организовывалось с целью содействия и становления платформы Android путем введения инноваций в технологии, используемые при разработке устройств и с целью создания приложений для ОС.

Наравне с увеличением продаж телефонов, реализованных на операционной системе Android, вырастала и популярность приложений, разрабатываемых ради предоставленной ОС. В 2008 году с выходом первых Android телефонов Google заявила об открытии маркета приложений, который назвали Android Market. Он давал возможность скачивать программы и игры юзерам новоиспеченной операционной системы. Потихоньку появлялись различные возможности: поддержка платных программ, скачивание музыки, книжек и кинофильмов, а уже 2012 году было принято решение весь этот набор сервисов соединить в один и поменять название на Google Play. [20]

Google Play считается далеко не только маркетом Android–приложений. В соответствии с своим назначением он считается каналом доставки программ на Android–смартфон, и также данная служба способствует тому, что скачиваемые приложения будут корректно работать на смартфоне, обеспечивая их совместность. В Google Play применяются последующие механизмы сопоставимости:

* смартфон должен соответствовать определению совместимости Compatibility Definition Document (CCD), которое содержит описание вариантов поведения системы Android;
* Google Play должен знать о всех дополнительных устройствах, которые нужны для работы приложения (например, наличие камеры, навигации и др.). Поэтому пользователям доступны те приложения, которые подходят под их устройство.

Публикация приложений в Google Play значительно легче в сравнении с подобной операцией в App Store. Вроде это доставляет побольше воли создателям и, как итог, обширный ассортимент приложений, но при этом — дефицит в Google Play сильной модерации новых программ приводит к тому, что в маркете стает очень много приложений сомнительного качества, в том числе может быть содержание в них вредоносное ПО, которое является опасностью для сохранности системы. Google всегда оказывает содействие создателям приложений. Они двукратно — в 2007 г. и в 2009 г. — организовывали побуждающее соперничество Android Developer Challenge между сторонними производителями, которым предполагалось сделать приложение различной направлености. При подборе фаворитов состязания, кои награждались большими денежными призами, особенное внимание уделялось современности и полезности разработанного продукта. [20]

Еще стоит заявить о ежегодной премии Google Play Awards, которую выдают создателям наилучших приложений и игр для Android. Еще, кроме помощи от Google, операционная система Android притягивает внимание создателей так что средства и инструменты, нужные в разработке приложений почти все находятся в свободном доступе на сайте developer.android.com (официальный интернет-сайт разработчиков Android). На веб-сайте презентована информация о основах разработки приложений, о сопоставимости устройств, о творении операций и реализации пользовательского интерфейса, также имеется информация, в которой доводятся практические руководства по разработке приложений с использованием разных API–интерфейсов. Android используется на различных устройствах. Это и телефоны, и планшеты, и тв, и часы и множество других гаджетов. Общая численность пользователей сматрфонов под управлением ОС Android оценивается в 2,2 миллиарда человек по целому миру. На сегодняшний день крайней версией считается Android 12.1, выпущенная 7 марта 2022 года.

Объемы продаж смартфонов на Android и сумма скачиваний Android приложений очень быстро увеличиваются. Мобильные телефоны Android 1-го поколения появились в продаже в октябре 2008 года. В соответствии с отчетом IDC, к концу 1 квартала 2015 года у Android было 69 % массового рынка телефонов, в сравнении с 21,3 % у Apple, 2,7 % у Microsoft и 0,3 % у Blackberry. Во время конференции Google I/O в 2015 годку компания Google огласила, что последние 12 месяцев в маркете Google Play™ — величина скачиваний приложений дошло до 53 миллиардов. Жесткое соперничество между создателями известных мобильных платформ и мобильных сервисов подвел к скорому введению инноваций и быстрому обвалу цен. Вследствие соперничества меж десятками разработчиков устройств Android убыстряется введение аппаратных и программных инноваций в обществе Android [20]

Одно из главных превосходств платформы Android — ее открытость. ОС Android сотворена на базе открытого исходного кода и располагаться в свободном доступе. Таким образом, это дает возможность создателям получить доступ к исходному коду и осознать, как выполнены свойства и функции программ. Каждый человек имеет возможность принять участие в совершенствовании предоставленной операционной системы. Для сего достаточно выслать отчет о найденных ошибках или принять участие в одной из дискуссий Open Source Project. В Сети интернет легкодоступны разные приложения Android с свободным исходным кодом, представляемые фирмой Google и другими производителями.

## 1.3 Анализ средств разработки

## и обоснование выбора технологии проектирования

На данный момент есть неплохой выбор языков программирования для создания мобильных приложений. Данный факт связан с тем, что для разных смартфонов нужно применять свои языки программирования. Это обусловлено тем, что мобильные устройства обладают разными ОС. Далее будут просмотрены такие языки, как Java, Qt (базирующийся на библиотеках C++), Windows Phone SDK (разработка на языке XAML), iPhone SDK (главный язык — Objective–C), Android SDK (фундаментальный язык — Java) и Symdian (основной язык — C++):

*Java 2 Micro Edition (J2ME).* В 1-ую очередь J2ME комплект спецификаций и технологий, специализированных ради разных видов портативных устройств. Направленность определяет вид конфигурации центральных библиотек Java, а еще характеристик виртуальной машины Java (где станут исполняться приложения). Разумно допустить, что устройства Connected Device Configuration (CDC) станут наиболее «развитыми», для образца можно привести коммуникаторы. К устройствам Connected Limited Device Configuration имеют отношении обычные смарфоны, аппаратно обладающие наиболее малыми потенциалами, ресурсами. Особые режимы дают возможность предопределять функциональность конфигураций для разных видов устройств. Кодирование на машинном Java–приложений на нынешний день занимает великую часть, так как большая часть телефонов в мире обладают уже предустановленной Java–машиной. [20]

*Qt* считается одной из наиболее успешных библиотек для С++. Наладка приложений, созданных для мобильных устройств, делается с помощью эмулятора, который вшит в саму среду разработки. Так, мы можем кодить трудные приложения для смартфонов с внедрением библиотек C++ и поддержкой кроссплатформенности. В нынешнее время крайняя версия — Qt 5 бета. Последний релиз 2012 год. Для работы Qt на телефоне нужна установка определенного Фреймворка.

*Windows Phone SDK.* Весь цикл разработки на Windows Phone проводится в программе Visual Studio. Среда считается комфортной для создания и отладки приложений. Для приложений на Windows Phone настройка делается в эмуляторе Windows Phone с через среду разработки.

*iPhone SDK*. Разработка для iPhone под ОС iOS возможна исключительно на Mac OS X. Однако в Сети интернет возможно отыскать статьи, как программировать и на Macintosh и в том числе на Virtual Machine.

Имеет смысл заметить, будто Apple выдает инструменты разработки безвозмездно, заплатить нужно за подписку создателя. Отладка приложения совершается с использованием среды XCode и эмулятора iPhone предустановленного в ней.

*Android SD*K. При кодировании под Android лучше применять среду Eclipse с установленным плагином Android Development Tools. Имееются разные версии SDK, коие употребляются для написания кодировки под разные версий Android. В сегодняшний день огромное распространение приобрели версии 24 и 26. Поддерживается практически абсолютная обратная совместность версий. [20]

*Symbian* и C++. Для написания кода под Symbian нужно использовать язык программирования C++. В большинстве своем данный подход используется для Symbian OS v6.1, 7.0, 7.0s и 8.0.

В соответствии с шагами проектирования и создания текущего приложения были проанализированы средства, которые решают поставленные задачи, будто проектирование визуального элемента проекта, этак и написание кода. В первую очередность нужно было запроектировать интерфейс приложения на базе актуальности его разработки, возможной аудитории и иных характеристик. Конструирование интерфейса исполнялось подручными средствами в виде очерков, чертежей и схем.

В следующем процессе разработки макетов и шаблонов приложения использованы графические редакторы. А если быть точнее — комбинация Adobe Photoshop и Adobe Illustrator.

*Adobe Photoshop* — это графический редактор, разработанный и поставляемый фирмой Adobe Systems. Он работает с растровыми изображениями, однако имеет и несколько векторных инструментов.

*Adobe Illustrator* — это уже векторный графический редактор, с помощью которого можно делать разнообразный контент: от маленьких значков до рекламных билбордов, от оформления сайтов до иллюстраций в книги. [20]

Выбор этих средств обусловлен их богатым функционалом, а также наличием опыта для работы с данным программами, полученный в процессе обучения в университете.

При разработке приложений Android применяется Java — один из самых популярных языков программирования, но для разработки нужны еще файлы XML. Язык Java — это самый современный язык, каковой действует на трех фронтах: портативность, быстрота и надежность. Потому он считается господствующим языком в круге программирования больше 18 лет. Язык программирования Java, разработанный фирмой Sun Microsystems под управлением светил интернета Джеймса Гослинга и Билла Джоя, был сотворен как машинно–независимый язычок программирования, который довольно безопасен при сетевых применениях и довольно мощный, чтоб сменить родной выполняемый код. [20]

Язык Java при разработке android–приложений употребляется далеко не в полнофункциональном виде, а лишь в незначительном подмножестве, в регистровой виртуальной машине Dalvik. В данном подмножестве не употребляются те классы Java, которые не смогут использоваться либо не обладают значением при разработке приложений на смартфоны. Dalvik оптимизирован ради невысокого потребления памяти, это необычная регистр–ориентированная виртуальная машина, отлично подходящая к выполнению в процессорах RISC–архитектур, нередко применяемых в мобильных и встраиваемых устройствах, таковых как коммуникаторы и планшетные пк. (Большая часть виртуальных машин, применяемых в настольных системах, считаются стек ориентированными, в том числе и виртуальная машина Java). Программы для Dalvik создаются на языке Java. Невзирая на это, шаблонный байт–код Java никак не употребляется, вместо него Dalvik выполняет байт–код собственного формата.

В версиях, стартуя с Android 4.4, есть возможность переключения с Dalvik на более быстрый Android Runtime (ART). В Android 5.0 уже полностью заменен ART. В отличие от Dalvik, который пользуется JIT компиляцию ART компилирует приложение во процессе его загрузки. Благодаря этому планируется увеличение скорости работы программ и одновременно повышение времени работы от аккумулятора. Недостатком является более продолжительная загрузка смартфона. Для обеспечения обратной совместимости ART пользуется тем же байт–кодом, что и Dalvik.

Скомпилированный программный код упаковывается в специальный файл–архив, Android Package. Данный файл имеет расширение \*.apk и упаковывается определенной утилитой aapt tool. В дальнейшем он распространяется как программа и устанавливается на телефон. Один такой файл может быть связан с кодом единичного приложения. Каждое приложение в ОС Android находится в своем собственном мире — в такой машине. По умолчанию, каждая программа выполняется в собственном процессе, а управлением занимается ядро Linux, которое паралельно осуществляет менеджмент памяти. Чаще всего код приложения компилируется в изоляции от всех остальных приложений. Android стартует процесс, когда нужно выполнить какой–либо программный код и завершает его, когда в нем больше нет нужды и системные ресурсы нужны для других приложений. Обычно каждому приложению назначается уникальный ID Linux–пользователя.

Проводя аналогию с Windows, приложение состоит из окон, которые называются Activity. В определенный момент времени отображается одно такое окно, и оно занимает целый экран, а приложение просто переключается между ними. Для примера рассмотрим приложение почты. Так, одно Activity — входящие письма, другое — просмотр определенного письма, третье — настройки. При работе мы переключаемся между ними.

Содержимое Activity составляется из разных компонентов, называемых View. Самые популярные из них это кнопка, поле ввода, чекбокс. View обычно распологается в ViewGroup. Пример ViewGroup — это Layout. Он же бывает разных типов и ответственный за то, как будут распологаться его дочерние View на экране (таблицей, строкой, столбцом). [20]

Для кодирования Android–приложений нужен пакет Java Development Kit (JDK) и Android Studio — это интегрированная среда разработки для работы с ОС Android. Одним из главных инструментов для разработки Android–приложений является многофункциональное средство разработки для операционной системы Android SDK.

В современном мире создание ПО во многих случаях осуществляется с помощью среды интегрированной разработки. В IDE автоматизирован процесс компиляции, сборки и старта приложения, что облегчает работу и позволяет без особых усилий создать свое собственное первое приложение.

Существует две популярные среды разработки, которые рекомендованны Google: Android IDE и Android Studio.

*Android IDE* — среда разработки приложений под операционную систему Android, на интегрированной среде разработки Eclipse. Имеет вшитые инструменты для создания, компиляции, сборки и отладки приложений.

*Android Studio* — среда разработки мобильных приложений, основанная на интегрированной среде разработки ПО IntelliJ IDEA. Аналогично Android IDE, Android Studio имеет встроенные инструменты для разработки и отладки мобильных приложений. Эта среда является официальным средством разработки Android. Android Studio возможно установить на Windows, OS X и Linux. Язык Kotlin, который используется тут – официальный языком программирования для платформы Android в добавление к Java и С++.

В дополнее ко всем возможностям в Android Studio выполнены слудующие вещи:

* система автоматической сборки Gradle;
* система рефакторинга кода;
* инструменты для поиска и устранения разных проблем;
* окно предпросмотра, которое показывает запущенное приложение на нескольких устройствах в реальном времени;
* облачная платформа Google Cloud Platform.

На данный момент Google прекратила поддержку инструментов для разработки в среде Android IDE. [20]

Соответственно было принято решение создавать приложение в среде разработки Android Studio.

В процессе просмотра на практике данной среды разработки приложений и написания небольших программ, для применения полученных знаний, мной были выявлены некоторые *недостатки Android Studio:*

* необходимость обладать базовым уровенем программирования на языке Java и знание английского языка.
* из–за сопровождения разработчика системой автосборки (Gradle) замечается значительное замедление времени компиляции проекта.
* уверенные системные требования для пк разработчика.
* базовые эмуляторы используют большое количество системных ресурсов, долго запускаются и не соответствуют всем возможностями смартфонов.
* отсутствие русскоязычной справки.

Достоинства Android Studio:

* приятный конструктор интерфейсов, чтобы просматривать отображение экрана на различных устройствах. Элементы отображаются так, как они будут выглядеть на определенной версии ОС.
* удобная структура проекта.
* наличие логов для просмотра всех ошибок, процессов и потоков.
* обширное количество русскоязычной литературы и видеоуроков на просторах сети интернет в свободном доступе.

## 

## 1.4 Общий алгоритм реализации мобильного приложения

Наиболее главным шагом разработки ПО по праву именуется процесс выполнения многостороннего разбора требований к разрабатываемому программному обеспечиванию, чтобы найти первостепенные цели и задачи продукта. Итогом проделанного разбора стает создание технического задания, на рамки которого станет базироваться разработчик в собственной работе.

Техническое задание помогает полностью характеризовать установленные перед создателем приложения задачи и обозначить окончательную цель проекта в соображении клиента. Последующий ключевой шаг в разработке программы – это ступень проектирования, то есть создание теоретической базы разрабатываемого продукта.

Нынешние средства программирования дают возможность отчасти соединить этапы проектирования и написания кода, будучи основанными на объектно–ориентированном подходе, однако хорошее составление плана просит наиболее кропотливого и тщательного моделирования. Высококачественный анализ возможностей и способностей разрабатываемого продукта станет костяком для его настоящего функционирования и исполнения итогового комплекса возлагаемых на ПО задач. Одна из составных долей стадии проектирования, например, отбор средств разработки и ОС, каких сейчас находится бесконечно великое обилие.

В предоставленной работе выработан дальнейший алгоритм создания мобильного приложения, который состоит из трех главных шагов, подключающих в себе подэтапы.

* подготовительный этап:
  1. формирование идеи;
  2. изучение подобных разработок;
  3. проектирование интерфейса, разработка эскиза;
  4. формулировка названия;
  5. знакомство с технологией разработки приложений на ОС Android.
* главный этап:
  1. систематизация информации для заполнения приложения;
  2. форматирование данных для работы приложения;
  3. выбор оттенков цветов и стиля шрифта;
  4. разработка макетов экранов;
  5. разработка в Android Studio;
  6. формирование шаблонов базовых окон;
  7. интегрирование анимации;
  8. проверка работоспособности на смартфонах
* заключительный этап:
  1. исправление недостатков и пожеланий.

Получается, что процесс разработки мобильного приложения состоит из подготовительного шага, в котором продумывается полный цикл разработки приложения и его составляющего контента и соответственно этап разработки, на котором создается мобильное приложение и оно проверяется работоспособность.

2 РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ «EFFEKTUS»

## 2.1 Будущие компоненты приложения

Составляющие разных приложений представляют из себя блоки, из которых получается программа для Android. Составляющие компонент является особой точкой, благодаря которой ОС получает возможность запустить приложение. Однако некоторые составляющие не могут являться методами входа для юзера, а некоторые из них могут содержать общую зависимость друг от друга. При данном раскладе любая составляющая считается отдельной самостоятельной структурной единицей и играет конкретную роль – каждая из которых представляет из себя неповторимый элемент структуры, определяющий работу приложения в целом.

Элементы приложения можно разделить на несколько видов. Компоненты каждого вида используются для особой цели, все они работают через свой собственный жизненный цикл, который уже формирует метод старта и прекращения существования компонента.

Различают четыре типа компонентов приложения:

* активности;
* сервисы;
* получатели широковещательных сообщений;
* поставщики контента.

Каждый из них нужен для выполнения своей задачи и обладает уникальным циклом.

*Активность* – точка входа для работы с юзером. Она представляет из себя один экран графического интерфейса. К примеру, приложение почты имеет одну активность, для демонстрации списка новых писем, другая для написания письма и следующая для просмотра сообщения. Хоть активности и функционируют совместно, для обеспечения единого пользовательского интерфейса приложения, но они остаются независимыми друг от друга. Получается, что иное приложение сможет включать любую из используемых активностей параллельно, если это возможно со стороны почты. Как вариант, приложение которому нужна камера смартфона может быть запущено из электронной почты, например для формирования вложения для письма. В задачи активности включается обеспечение определенных ключевых элементов взаимоотношений между системой и программой, к примеру.

Отслеживание, важных для пользователя компонентов, чтобы убедиться, в том, что система продолжит поддерживать процесс, содержащий данную активность в рабочем состоянии, и не завершит его принудительно;

Сохранение списка последних включенных активностей, в которые юзер может возвратиться;

Помощь программе в обработке завершения его процесса, чтобы сохранить существующее состояние активности и суметь восстановить его при возврате в приложение;

Предоставление способа для программы, которым организуется взаимодействие между рабочими активностями, и для системы координации данных взаимодействий, как пример, работа кнопки "поделиться".

Активности представляют из себя наследников базового класса Activity.

В нашем спроектированном мобильном приложении используется три класса активности;

*StartActivity* представляет точкой входа в приложение. При запуске проверяется, осуществлен вход пользователя или нет, на основании чего управление предоставляется в следующие необходимые активности;

*LoginActivity* содержит графический интерфейс для вызова методов авторизации в приложении;

*MainActivity* представляет основную рабочую активность приложения содержит в себе основные фрагментов программы.

Сервисы имеют точку основного назначения для поддержки фоновых процессов. Данный компонент, существует в фоновом режиме для того, чтобы выполнять большие операций или вызов удаленных процессов.

Например, служба может воспроизводить фоновую музыку, пока пользователь находится в другом приложении, или получать данные по сети, не ограничивая взаимодействие пользователя с действием. Служба может быть запущена другой сущностью, которая взаимодействует с ней как с активностью. Службы могут принимать две разные формы: работающие службы и взаимосвязанные службы. Служба запускается, когда есть компонент приложения запускает ее вызовом startService(). После того, как служба запущена и запущена, она может бесконечно работать в фоновом режиме, даже если компонент, запустивший ее, удален. Обычно запущенная служба выполняет операцию и не отвечает вызывающему компоненту. Например, он может скачать или загрузить файл по сети. После выполнения операции служба должна автоматически остановиться. Доступны два разных типа служб, влияющих на фоновую синхронизацию данных или воспроизведение музыки, а также на производительность системы:

Сервис первого плана — это услуга, при которой пользователь получает активную информацию, поэтому нет необходимости удалять ее из системы при низком уровне цветовой системы. Служба предыдущего плана должна отображать предупреждение в строке состояния. Эта память может быть удалена, если служба не остановлена ​​или не удалена из предыдущего плана;

Фоновый сервис – это не то, с чем именно пользователь взаимодействует, поэтому ОС имеет в разы больше свободы в управлении процессом этого сервиса. Что позволяет операционной системе убивать данный процесс при нехватке оперативной памяти для процессов, с которыми пользователь в данный момент работает. При освобождении оперативной памяти ОС сможет сама перезапустить фоновый сервис.

Сервис считается привязанным, если компонент приложения вызовается bindService(). Привязанный сервис представляет из себя клиент–серверный интерфейс, позволяющий компонентам сотрудничать с сервисом, отправлять запросы, принимать результаты и даже делать это помимо разных процессов посредством взаимодействия (IPC). Привязанный сервис работает только если к нему обращается иной компонент приложения. К одному сервису можно привязать несколько компонентов одновременно, но, когда все они перестают обращаться к нему, сервис самоуничтожается. Он реализуется как подкласс Service. [20]

При наличии корректировок в локальной базе данных, сервис уведомляет о содержании краткой информации про дынные изменения и выгружает существующие отчеты с помощью метода upload API CRM системы.

Broadcast receiver – элемент, позволяющий получать широковещательные сообщения ОС. Данными сообщениями считаются, объявления когда, экран выключился, аккумулятор разрядился или сделано фото. Данные сообщения могут быть отправлены пользовательскими приложениями, вроде уведомления о изменении определенных данных и готовности к их загрузке и использованию. Сервисы Broadcast receiver не обладают графическим интерфейсом, но они могут создавать сообщения в консольной строке. Однако, чаще всего они используются как просто шлюз для работы отдельных элементов.

Особенностью ОС Android выделяют то, что приложения могут включать элементы других приложений независимо друг от друга. Например, если человек хочет сделать фото, то не обязательно для этого разрабатывать новое приложение, достаточно просто обратиться к существующему. После использования фото вернётся в нужное приложение, из которого было вызвано и будет возможность его использования. Для человека это будет казаться как одно единое приложение. [20]

Для старта отличных компонентов ОС запускает другой процесс и начинает инициализировать элементы, нужные для работы соответствующего элемента. Так, если включается Activity для работы с камерой, которая привязана к другому приложению, соответственно она включается в отдельном процессе, который соответственно принадлежит данному приложению. Из-за этого приложения на ОС Android не обладают одной единственной функцией входа, отсутствие функции main.

Система включает разные приложения в разных изолированных друг от друга процессах с соответствующими правами доступа, с помощью уже которых имеется возможность ограничить доступ других приложений, соответственно у используемого приложения не получится сразу вызвать компонент другого приложения. На помощь в таком случае приходит ОС Android, она использует для активации нужного элемента, отправку уведомления операционной системе о желании запустить определенный процесс, и уже ОС его запускает. В ПО реализовано три подкласса BroadcastReceiver.

StartServicesReceiver использует состояние события "android.intent.action.BOOT\_COMPLETED" используемое при успешной загрузке ОС. Данный класс восстанавливает стертые после рестарта устройства процессы необходимые для включения сервисов синхронизации, также занимается обработкой событий необходимых для старта процесса UpdateMyTryService

UploadPhotosAlarmReceiver использует процесс запланированного включения сервиса UploadMyANDFriendsTryService

Поставщик контента занимается управлением наборами данных, которые хранятся в БД SQLite, в всемирной паутине или другом хранилище, доступ к соответствующим обладает приложение. Используя поставщик- контент, остальные программы смогут получать или редактировать данные, если права на это настроены внутри поставщика. ОС Android использует поставщика контента, занимающегося управлением персональной информацией человека. Стороннее приложение, которое обладает нужными правами, может получить определенный элемент этого поставщика для осуществления чтения и записи данных. Сама система использует поставщик контента для определения точки входа в приложение чтобы публиковать именованные компоненты информации, выделенные в схеме URI. Получается, программа умеет сама решать, как ей необходимо сопоставить данные, которыми оно обладает, используя пространство имен URI, отправляя данные URI следующим элементам, чтобы они в свою черед, использовали их для получения доступа к данным. Имеется парочка особенностей, которые позволяют системе использовать управление с помощью поставщиков контента; [20]

Регистрация URI не просит, чтобы приложение было включено, соответственно URI могут быть сохранены уже после окончания действия приложения. ОС может включить необходимую программу, когда образуется необходимый запрос на информацию по привязанному URI;

URI обладают собственной схемой безопасности. Программа может зарегистрировать URI для картинки, хранящейся в буфере обмена, но предоставить использование через поставщика контента блокированным, чтобы остальные приложения не имели права свободно использовать его. Если стороннее приложение захочет получить доступ к данному URI, то ОС может оформить временное право на доступ только к данному URI, не оформляя разрешения на использование остальных ресурсов приложения.

Поставщики контента разрабатываются как подкласс ContentProvider и обладают стандартным набором API.

Уникальным свойством графического дизайна ОС Android можно считать, приложение может стартовать различные элементы другого приложения. Для того, чтобы юзер смог сделать фото с использованием камеры смартфона, нет как таковой нужды в разработке способов захвата изображения, потому что для этого можно использовать сторонние приложения, которые уже имеют в себе функционал работы с камерой. Чтобы осуществить это нужно просто запустить активность, которая предназначена для работы с камерой и вернуть после использования готовое фото. Человеку даже может показаться, что он использовал камеру телефона как единую часть с приложением.

Вместе с запуском компонента, происходит старт процесса для приложения в котором он существует, конечно только если оно уже не включено. Объясняя понятным языком, если приложение включает активность в камере, данный процесс начинает выполняться, в принадлежащем камере приложении, а не в процессе приложения, в котором осуществляется работа.

В связи с тем, что система стартует каждое приложение в изолированном процессе с применением ограничения доступа к остальным приложениям, то получается, что приложение не сможет сразу вызвать нужный элемент из другого приложения. Данным процессом занимается непосредственно ОС. Чтобы вызвать элемент стороннего приложения, нужно уведомить операционную систему о желании включить нужный компонент, после чего ОС самостоятельно сделает эти нужные.

Еще одним видом компонентов выделяют – активности, широковещательные приемники и службы – они уже в свою очередь запускаются методом вызова асинхронных сообщений, которые также могут называться Intents – намерения. Они являются связывающим звеном между отдельными элементами друг с другом для их выполнения.

Намерение делается через вызов объекта Intent, в котором существует уведомление активации определенного компонента, либо нужного типа компонента, что еще называют неявным намерением.

Намерение определяет действие служб и активностей, чтобы выполнить, например, просмотр, отправку какой-либо информации, и может содержать в себе URI данных, которые обращаются к действиям. С использованием намерения можно, например, отправить запрос для запуска активности, которая показывает веб-сайт или видео.

Для получателей широковещательных сообщений намерение является просто передатчиком.

Сравнивая активности, сервисы и получатели сообщений, поставщики контента невозможно вызвать через намерения.

Существуют и другие способы вызова каждого вида компонента:

Активность вызывается через методы startActivity() или startActivityForResult(). Последний используется, когда от активность нужно получить обратный результат;

Начиная с Android 4.1 есть вариант использования класса JobScheduler для отложенного старта нужных действий;

Можно отправлять широковещательное сообщения через sendBroadcast(), sendOrderedBroadcast() или sendStickyBroadcast();

Можно обратиться к поставщику с запросом контента путем метода query() из класса ContentResolver.

В создающемся мобильном приложении намерения нужны для запуска определенных активностей в StartActivity, обращения к сервисам синхронизации из широковещательных получателей и для осуществления способа запуска приложения через уведомление об изменениях заказов.

Прежде чем ОС Android запустит определенный элемень приложения, система должна убедится, что данный компонент имеется в наличии. Приложение обычно регистрирует свои элементы в файле AndroidManifest.xml. Xamarin генерирует записи в этом файле опираясь на атрибуты классов, так активности нашего приложения зарегистрированы через атрибут Activity, а вот уже широковещательные получатели через атрибут BroadcastReceiver и IntentFilter. [20]

Помимо элементов приложения файл манифеста обладает списоком прав, которые необходимы приложению как для доступа в сеть, так и для чтения контактов из записной книги. Еще в данном файле регулируется минимальная версия ОС для работы разрабатываемого приложения, разные библиотеки к которым может обращаться приложение и требования к функциональным возможностям смартфона.

Активности, сервисы и поставщики контента, которые указаны в приложении, но не были записаны в файл манифеста, не отображаются ОС и, соответственно, не смогут быть вызваны. Но широковещательные получатели сообщений также могут быть указаны в период выполнения приложения через вызов метода registerReceiver().

## 2.2 Требования и жизненный цикл приложения

Есть очень большое количество разнообразных устройств, на управлении ОС Android, и все они обладают отличающимися возможностями. Чтобы предотвратить вариант установки приложения на смартфон, который не обладает необходимым функционалом для корректной работы приложения, необходимо конкретно выделить список устройств, которые смогут поддерживаться приложением. Чтобы это сделать нужно объявить минимальные требования к смартфонам и установленному на них ПО в файле манифеста. Множество из данных записей несут в себе информационный характер, и ОС их не читает, однако внешние процессы, например такие как Google Play, изучают их, чтобы оформить фильтр при поиске приложений для определенного устройства.

Например, если приложению нужен микрофон, и он взаимодействует с API, представленным в Android начиная с версии 5, то данную информацию нужно указать в файле манифеста. При этом устройства под управлением ОС Android ниже версии 5 или же смартфоны, с отсутствием микрофона, не смогут скачать данную программу из Google Play. Также можно уточнить, что приложение обращается к микрофону, но не требует ее обязательного присутствия, в данном случае приложение в процессе своего выполнения может проверить, есть ли у смартфона, на котором оно используется микрофон, и если он отсутствует, то просто выключить функции, в которых он может использоваться.

Приложение Android включает в себя не только код, но и ресурсы, которые изолированы от исходного кода, например это может быть как изображения, аудио так и все, что имеет отношение к визуальному представлению приложения. Нам нужно указывать информацию о анимациях, меню, стилях, цветах и макетах графических интерфейсов в файлах называемых XML. Путем использования возможностей приложения, мы можем без особых трудностей поменять его характеристики, не изменяя код, а, кроме того, используя предоставления альтернативных ресурсов мы можем оптимизировать приложение для корректной работоспособности с индивидуальными конфигурациями смартфонов.

Для всех ресурсов, используемых в проекте Android, SDK присваивает индивидуальный уникальный целочисленный идентификатор, который используется для того, чтобы сослаться на определенный элемент из кода приложения, определенного в файле XML. Так если у нас есть файл с именем logo.png, инструменты SDK определят уникальный идентификатор ресурса с именем R.drawable.logo, используя который, на изображение можно будет выйти и использовать его уже в графическом интерфейсе.

Самый важный аспект использования ресурсов изолированно от исходного кода состоит в возможности использовать ресурсы для разных конфигураций смартфонов. Выделив строки графического интерфейса в файле XML, нужно перевести их и зарегистрировать данные переводы в отдельных друг от друга файлах. Затем используемому квалификатору языка, который добавляется к имени каталога ресурса, например ru, для строк, описанных на русском языке, и указанному человеком языке ОС Android применит к интерфейсу текст на необходимом языке.

Квалификатор представляет из себя небольшую строку, которая включена в названия каталогов ресурсов преследуя цель определения типа устройства, для которого данные ресурсы будут нужны. Аналогичным примером можно выделить то, что для активностей необходимо делать уникальные макеты, которые в свою очередь будут подходить по размеру и ориентации экрана смартфона.

Так, когда экран смартфона находится в книжной ориентации, то есть расположен вертикально, кнопки в макете будут также применять вертикальную форму, а когда экран находится в горизонтальном положении, альбомная ориентация, кнопки примут вариант размещения по горизонтали. Для того чтобы это функционировало, нужно выделить два отличных макета и создать соответствующий им квалификатор для каталога каждого уникального макета. После этого ОС сможет сама автоматически использовать необходимый макет в зависимости от выбранной ориентации устройства. [20]

Активность – уникальный вариант программирования, который характерен для Android приложений. В классической разработке используется статический метод main, который включается во время старта программы. Но в величайшем Android все совсем по-другому. Приложения Android могут стартовать через различные указанные активности в приложении. На практике почти все программы имеют только одну единственную активность, которая, указана в качестве точки открытия приложения. Но, если программа неожиданно крашнулась или закрыто ОС принудительно, тогда операционная система будет пробовать перезапустить программу включив последнюю используемую активность. Помимо этого, ОС может тормозить активность, если вдруг юзер с ней перестал взаимодействовать, и восстанавливать ее при повторном обращении. Нужно принимать во внимание очень много различных нюансов для того, чтобы программа смогла корректно восстановить свое прежнее состояние после рестарта активности, особенно если эта активность зависит от других активностей, которые были запущенны перед ней.

Жизненный цикл активности создается как набор методов, которые вызывает ОС во время жизненного цикла активности. Данные методы способствуют реализации функциональных возможностей, которые нужны для согласования требований к управлению и состояниями приложений.

Очень важно рассматривать требования каждой отдельной активности, чтобы понять, какие именно методы, жизненного цикла, должны быть сформированы. Неверная разработка данных методов сможет подвести к плохой работе приложения, частым сбоям, потере информации и некорректной работе ОС.

Жизненный цикл активности Android содержит совокупность методов, которые определенны в основном классе Activity, предоставляющем структурированный метод управления ресурсами.

ОС Android управляет активностями опираясь на их состояние. Этот факт облегчает операционной системе определение активности, которые уже не используются и которые можно завершить, что позволит освободить память и ресурсы смартфона для следующих активностей. Диаграмма, изображенная на рисунке 1 показывает состояния, которые может принимать активность в течение жизненного цикла.

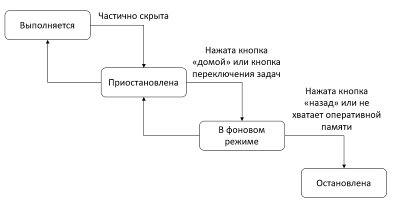


Рисунок 1 – Жизненный цикл активности [27]

Данные состояния можно разделить на пару основных групп определенным образом:

*Выполняется или работает*. Активности определяются работающими, если они находятся в самом верху стека активностей. Данное состояние имеет наивысший приоритет в ОС Android и, данная активность может быть выключена ОС только в экстренных ситуациях, например, таких как чрезмерное использование памяти, чем есть в свободном доступе на устройстве, что может привести к прекращению, реакции на запросы пользователя;

*Приостановлена*. Когда смартфон находится в состоянии сна или активность еще используется, но частично закрыта новой, не полноценной активностью, то она начинает считаться приостановленной. Приостановленные активности сохраняют информацию о состоянии и используемых элементах, также остаются закрепленными в оконном менеджере. Данное состояние имеет второй по счету приоритет в Android. Активности, пребывающие в данном состоянии, могут быть закрыты ОС только если это нужно для освобождения ресурсов для других активностей, пребывающих в рабочем состоянии;

*В фоне/остановлена*. Активности считаются остановленными или пребывающими в фоновом режиме если они перекрыты другими. Остановленные активности стараются сохранить данные о своем текущем состоянии и состоянии дочерних компонентов так долго как это возможно, но при этом они имеют самый низкий приоритет и могут быть завершены ОС в любое время для создания лучшей работоспособности активностей, имеющих более высокий приоритет;

*Перезапущена*. Активности, имеющие состояние от приостановлена до остановлена могут быть убраны из оперативной памяти ОС и если юзеру снова необходима данная активность, то она может быть рестартнута. Состояние данной активности будет восстановлено, и она снова будет показана пользователю.

Также имеет место быть усложнение управление жизненным циклом приложения из-за изменения конфигурации.

Изменения конфигурации – это моментальные циклы уничтожения или повторного создания активности, которые появляются при изменении конфигурации активности, когда смартфон повернут и нужно чтобы активность изменила режим отображения с ландшафтного или портретного режима.

Изменения конфигурации используют те же состояния активности, которые осуществляются при остановке и рестарте активности.

Но чтобы программа оставалась быстрореагирующей во время изменений конфигурации, важно, чтобы обработка осуществлялась как можно быстрее. Поэтому в ОС Android существует специальный API, который используется для заморозки состояния во момент изменений конфигурации.

Android SDK и Xamarin.Android предоставляют огромную модель управления состоянием активностей в мобильной программе. Когда состояние активности меняется, она получает сообщение от ОС через обращение к определенным для этого методам. Схема, показанная на рисунке 2 представляет эти методы в жизненном цикле активности.

Изменения состояния можно редактировать, изменяя данные методы внутри самой активности. Но также стоит отметить, что все методы жизненного цикла используются в потоке интерфейса и замораживают его работу.

Поэтому эти методы нужно делать максимально маленькими, насколько это возможно, чтобы программа была быстровыполнимой, а все большие задачи необходимо выполнять в фоновых потоках.

Нужно посмотреть работу всех методов жизненного цикла активности отдельно друг от друга.

Метод OnCreate – это метод, который выполняется при создании активности. Он всегда задается для исполнения инициализации. Во время его выполнения обычно происходит:

* создание представлений;
* загрузка переменных;
* соединение статических данных со списками.

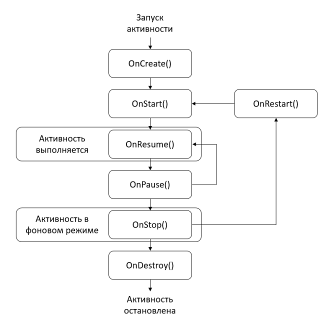


Рисунок 2 – Методы жизненного цикла активности [27]

Параметр Bundle, представляет из себя словарь для хранения и передачи данных состояния и объектов между активностями. Если bundle не равен null, то это указывает, что активность перезапускается и должна восстановить свое состояние из предыдущего варианта.

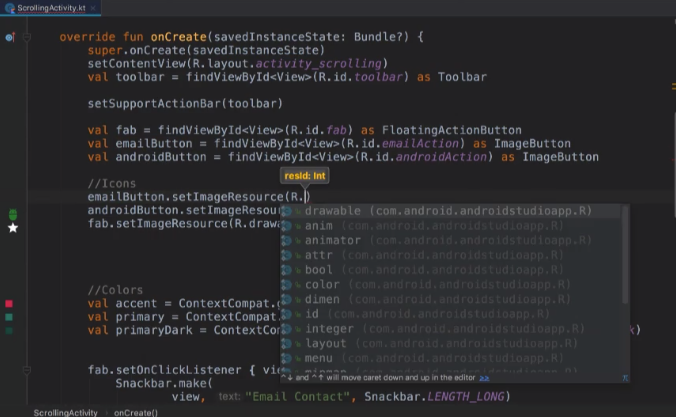
Как только OnCreate возвращает управление, операционная система, вызывает метод OnStart.  


Рисунок 3 – код onCreate [31]

Метод OnStart обычно переопределяют если необходимо выполнить какие–либо конкретные задачи непосредственно перед тем, как активность станет видимой, например, обновить текущее состояние элементов пользовательского интерфейса, принадлежащих активности.

Завершая данный метод, ОС сразу обращается к методу OnResume.

Система обращается к нему, когда активность уже может начать работу с юзером. Данный метод нужно сделать для таких задач, как:

* использование анимации;
* показ нужных предупреждений или диалоговых окон.

Метод OnResume является очень важным, потому что все операции, используемые в OnPause должны быть информированы в методе OnResume, потому что это один единственный метод жизненного цикла, который используется после OnPause для восстановления активности.

Метод OnPause используется, когда системе необходимо убрать активность в фоновый режим. В данном методе, нужно сделать выполнение следующих действий:

* сохранение несохраненных корректировок в постоянные данные;
* закрытие объектов, использующих ресурсы;
* сокращение частоты кадров и заморозка анимации;
* отмена регистраций обработчиков внешних событий или обработчиков сообщений, которые соответствуют какой–либо определенной службе. Это нужно чтобы обезопаситься от потери памяти.

При завершении OnPause следующими могут быть использованы методы:

OnResume. Используется, если активность необходимо вернуть на главное окно;

OnStop. Используется, при отправке активности в фоновый режим.

Метод OnStop используется, когда приложение больше не нуждается в отображении пользователю.

К методу OnStop обращение происходит не всегда. Он не вызывается при недостатке оперативной памяти, когда ОС не может справится с активностью. Именно поэтому лучше не возлагать больших надежд на метод OnStop, который работает для подготовки уничтожения активности.

Еще есть метод жизненного цикла, которые к которому можно обратиться – OnDestroy, если приложение закрыли, или же OnRestart при возвращении программы на первый экран.

OnDestroy – последний метод, используемый для уничтожения и полным удалением из памяти данных активности. В нестандартных ситуациях ОС может завершить процесс приложения, к которому привязана активность, что соответственно обрушит вызов данного метода. В жизни данный метод не редактируется, потому что все нужные операций прописываются в методах OnPause и OnStop. OnDestroy редактируют только для уничтожения сломанных фоновых процессов, которые могут вызвать потерю данных.

Метод OnRestart вызывается после того, как активность была заморожена и для ее очередного запуска.

Нет определенных стандартных правил того, какая логика может быть создана в методе OnRestart. Это обусловлено тем, что OnStart всегда запускается независимо от того, что делает активность, поэтому любые ресурсы, нужные для активности, необходимо указать в OnStart, а не в OnRestart.

Большинство Android смартфонов имеют две кнопки: кнопка Назад и кнопка Домой. Имеется небольшая разница в этих кнопках, хоть они и имеют схожий эффект, оказываемый на приложение делая его в фоновый режим. Если юзер использует кнопку Назад, то она сообщает ОС, что работа с активностью завершена и Android ее закрывает. Если пользователь использует кнопку Домой, то активность перемещается в фоновый режим и остается там.

## 

## 2.3 Сохранение состояния приложения

Когда активность замедляется или уничтожается, операционная система предоставляет возможность сохранить состояние активности для последующего его восстановления.

Известным методом поддержания состояния активности является использование словаря ключ/значения, известного как bundle. Этот объект передается методу оnCreate в качестве параметра, и его можно использовать для обновления состояния активности. Bundle рекомендуется применять для сохранения простых значений, таких как строки.

Активность предоставляет методы, позволяющие сохранить и восстановить состояние в Bundle.

Метод оnSaveInstanceState, вызывается операционной системой, когда активность уничтожается. Активности могут реализовывать этот метод, если необходимо сохранить любые элементы состояния в словаре bundle.

Метод оnRestoreInstanceState, вызывается после завершения метода OnCreate и предоставляет еще одну возможность для активности восстановить состояние после завершения инициализации.

Метод оnSaveInstanceState получает в виде параметра объект Bundle, в котором активность может сохранить свое состояние.

Переопределение OnSaveInstanceState является подходящим механизмом для сохранения временных данных активности, однако реализация OnSaveInstanceState по умолчанию позаботится о временных данных ,находящихся в элементах графического интерфейса, и об активности, если такие элементы имеют конкретный идентификатор, который может быть при помощи атрибута android:id в файле разметки XML. К примеру, если активность имеет элемент EditText определенный в XML, и он содержит идентификатор, то данные, которые ввел клиент, будут сохранены при изменении положения экрана.

Метод оnRestoreInstanceState получает в качестве параметра тот же объект Bundle что и метод оnCreate. Этот вариант предназначен для обеспечения некой гибкости при восстановлении состояния. Чаще всего нужно подождать полной инициализации активности прежде, чем восстанавливать состояние.

Хотя оnSaveInstanceState облегчает сохранение временных данных, он имеет свои ограничения:

Используется не во всех случаях. Например, нажатие кнопки "Домой" или "Назад" для выхода из действия не приведет к вызову метода OnSaveInstanceState;

Объект Bundle передаваемый в качестве параметра в OnSaveInstanceState предназначен для хранения больших объектов;

Данные, сохраненные с помощью пакета, сериализуются, что может привести к перебоям во время выполнения задач в приложении.

В активности LoginActivity разработанного приложения для сохранения состояния используется метод OnSaveInstanceState. Восстановление состояния происходит при вызове метода OnCreate, для того чтобы не нужно было обрабатывать вручную сохранение состояния текстового поля.

## 2.4 Разработка пользовательского интерфейса

При создании программ «распространена практика особого предания им наглядного вида, то есть визуализации. Эта практика играет не малую роль в проектировании и разработке пользовательских интерфейсов. Разработчики, заказчики и будущие пользователи программ могут уже на первых шагах разработки оценить направление развития приложения и внести нужные изменения. Прототип приложения на первых стадиях создания позволяет избежать дорогостоящих изменений в продукте позднее.

Визуализация проектных решений может принимать одни или несколько форм включающие в себя: наброски карандашом на бумаге, макеты, созданные в графических редакторах, имитационное моделирование, программные прототипы и другие.

Для прототипа графического интерфейса, разрабатываемого мобильного приложения, была задействована доступная версия онлайн сервиса NinjaMock.com

На основе выявленных требований к мобильному приложению было найдено необходимое количество экранов мобильного приложения. Для каждого экрана определены важные пункты пользовательского интерфейса.

Экран входа в приложение представлен на рисунке 4.



Рисунок 4 – Прототип экрана входа [41]

Экран входа в приложение будет показан при первом запуске приложения и при следующем входе, если клиент вышел из него.

Заголовок экрана входа содержит название приложения.

В середине находится элемент, позволяющий пользователю выбрать свою запись из предоставленного списка, не вводя данные самому, а также поле для введения пароля. Данные, введенные в поле для пароля, скрываются точками.

После входа в приложение и при следующих запусках с ранее введенными логином и паролем будет открываться экран последней активности для текущего пользователя, образец которого представлен на рисунке 5.

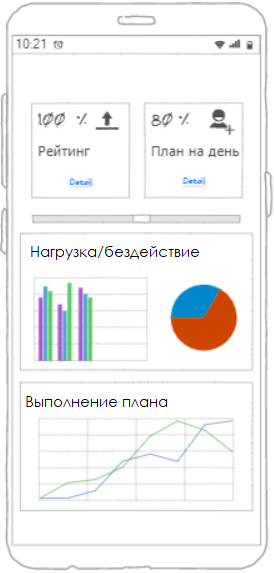


Рисунок 5 – Прототип экрана, содержащего активность [41]

Заголовок экрана со списком активностей содержит название "Тренировки".

В подзаголовке отображается имя текущего пользователя и его показатели в системе.

Также в подзаголовке можно посмотреть реализовано ли расписание активности на день.

В графике тренировок находятся данные о проведенных занятиях в зависимости от дней недели.

При нажатии кнопки "Detail" открывается диалоговое окно с функциями для дополнительных фильтраций.

Все пункты графического интерфейса в приложении для Android созданы с использованием объектов View и ViewGroup. View – это объект, который иллюстрирует что–то на экране, с которым клиент может взаимодействовать. ViewGroup – это объект, который содержит другие объекты View (и ViewGroup), чтобы определить модель интерфейса. Он предоставляет серию подклассов View и ViewGroup которые содержат постоянно используемые элементы, такие как кнопки и поля для ввода текста, и различные образцы (Layouts) для группировки элементов: линейный (Linear Layout), относительный (Relative Layout), табличный (Table Layout) и другие.

Каждый подкласс класса ViewGroup предоставляет особенный вариант расположения View, которые вы в нем размещаете. В созданном мобильном приложении применяются два наиболее многократно используемых макета: Linear Layout и Relative Layout.

Модель Linear Layout представляет собой подкласс ViewGroup который выравнивает все дочерние элементы в одном из направлений: вертикально или горизонтально. Направление выравнивания обозначается с помощью атрибута android: orientation. Все дочерние элементы Linear Layout располагаются друг за другом, поэтому в вертикальном положении будет всего один дочерний элемент на строку, независимо от того, насколько они широки, а в горизонтальном списке будет только один дочерний элемент на колонку.

Relative Layout – это подкласс ViewGroup, который показывает дочерние элементы в относительных положениях. Расположение каждого элемента может быть указано относительно ближайших в иерархии элементов (например, слева или ниже другого элемента) или в относительно родительской области Relative Layout (например, элемент прижат к низу родительской области, к левому краю, выравнен по центру). Relative Layout – очень мощный макет, так как он позволяет избежать многочисленных вложений макетов и помогает сохранить иерархию элементов графического интерфейса максимально плоской, что увеличивает производительность мобильного приложения.

Макет пользовательского интерфейса можно задать двумя способами:

Объявить элементы пользовательского интерфейса в XML файле. Android предлагает простой синтаксис XML, который содержит элементы, соответствующие классам и подклассам View и ViewGroup определенным в Android SDK;

Создавать элементы макета во время выполнения. Приложение может создавать объекты View и ViewGroup, и фиксировать их свойства самостоятельно с помощью программ. Это дает возможность применять один или оба этих метода для объявления и управления графическим интерфейсом мобильного приложения. К примеру, вы можете сделать макеты по умолчанию вашего приложения в XML, включая элементы экрана, которые будут отображаться в них,а также их свойства. Затем вы можете добавить код в приложение, который поменяет состояние объектов экрана, в том числе объявленных в XML, во время выполнения.

Достоинство объявления элементов графического интерфейса в XML состоит в том, что это помогает лучше отделить представление пользовательского интерфейса приложения от кода, который управляет его поведением. Описание пользовательского интерфейса является внешними по отношению к коду приложения, что дает возможность изменять или настраивать его, не изменяя исходный код и таким образом избавиться от перекомпиляции приложения. К примеру, можно создать макеты XML для разных вариаций экрана, разных размеров экрана мобильного устройства и разных языков. Кроме того, объявление макета в XML упрощает визуализацию структуры графического интерфейса во время его создания, что позволяет избежать многих неточностей проектирования.

Для управления внешним видом экранов, используются стили и темы приложения.

Для создания темы приложения, соответствующей цветам организации, нужно создать файл styles.xml в папке Resources/values–v21 приложения.

Это расположение файла специфично для версий Android 5 и выше. В файле стилей приложения возможно задать следующие свойства встроенной темы Material Theme:

colorPrimary – основной цвет приложения. Цвет заголовка приложения;

colorPrimaryDark – цвет строки состояния и контекстных заголовков приложения;

colorAccent – цвет таких элементов как флажки, радио-кнопки и границы полей для ввода текста;

windowBackground – цвет фона экрана;

textColorPrimary – цвет текста основного текста приложения;

statusBarColor – цвет панели состояния;

navigationBarColor – цвет панели навигации.

Файлы с описанием пользовательского интерфейса находятся в папке Resources/layout разрабатываемого приложения. Список файлов показан на рисунке 6.

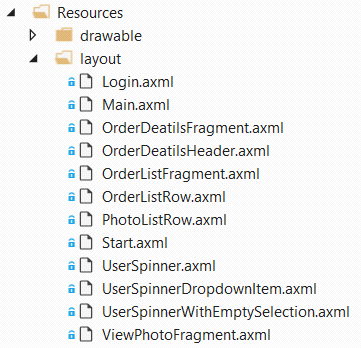


Рисунок 6 – Список файлов, содержащих описание пользовательского интерфейса [41]

Android поддерживает декларативный стиль дизайна пользовательского интерфейса, основанный на XML–файлах, а также создание интерфейса в коде приложения. При использовании декларативного подхода файлы XML могут быть либо отредактированы самостоятельно, либо изменены визуально при помощи визуального редактора графического интерфейса Xamarin.Android. Использование дизайнера позволяет сразу получать обратную связь при создании пользовательского интерфейса, ускоряет разработку и делает процесс создания пользовательского интерфейса более легким.

Визуальный редактор графического интерфейса запускается автоматически при создании макета или его открытии. Для того чтобы отобразить xml содержимое файла, в среде разработки VisualStudio нужно перейти на вкладку Source редактора файла.

Файл Login.axml содержит в себе описание пользовательского интерфейса для экрана входа в приложение. Макет экрана входа в приложение создается по принципу макета Relative Layout. Макет содержит такие элементы как:

ImageView – для отображения логотипа организации;

EditText – для отображения поля для ввода пароля. Элемент EditText содержит атрибут android:inputType установленный в значение "textPassword", что делает введенные в эту строку символы замаскированными;

Spinner – для отображения списка пользователей. Для того чтобы показать список пользователей, объекту Spinner следует передать объект SpinnerAdapter, который обеспечивает доступ к данным. В разработанном мобильном приложении реализован собственный SpinnerAdapter для отображения выбранным по умолчанию пустого значения.

Файлы UserSpinner.axml, UserSpinnerDropdownItem.axml и UserSpinnerWithEmptySelection.axml являются вспомогательными и описывают отображение выбранного элемента списка пользователей, элемента при открытом выпадающем списке и отсутствии выбранного значения соответственно.

Файл Main.axml cодержит один элемент Linear Layout который является контейнером для главных пунктов приложения.

Фрагменты (Fragments) представляют собой активность или элемент пользовательского интерфейса активности. Можно группировать несколько пунктов одной активности для создания многоуровневого графического интерфейса и снова пользоваться фрагментами в различных действиях. Фрагмент можно рассматривать как модуль активности, который имеет личный жизненный цикл, получает собственные входные события, которые можно как добавлять, так и убирать во время работы.

Фрагмент всегда должен быть встроен в активность, и жизненный цикл фрагмента напрямую зависит от жизненного цикла содержащей его активности. К примеру, когда активность замедляется, то приостанавливаются и все фрагменты, содержащиеся в ней, а когда активность уничтожается, то также удаляются и все её фрагменты. Однако, пока активность выполняется, каждым фрагментом можно управлять по отдельности. Например, добавлять или стирать их. Фрагменты при переключении можно добавлять в стек, который обрабатывается активностью, чтобы при нажатии кнопки "Назад" можно было получить прошлый фрагмент.

Фрагменты расположены внутри иерархии представлений активности и имеют свою собственную модель. Фрагмент может быть добавлен в активность благодаря явному объявлению в файле макета активности, используя элемент <fragment> или динамически из кода приложения, при помощи добавления его в существующую ViewGroup активности. Однако фрагмент не должен быть элементом объекта активности. Фрагмент может использоваться без графеческого интерфейса в роли рабочего потока активности.

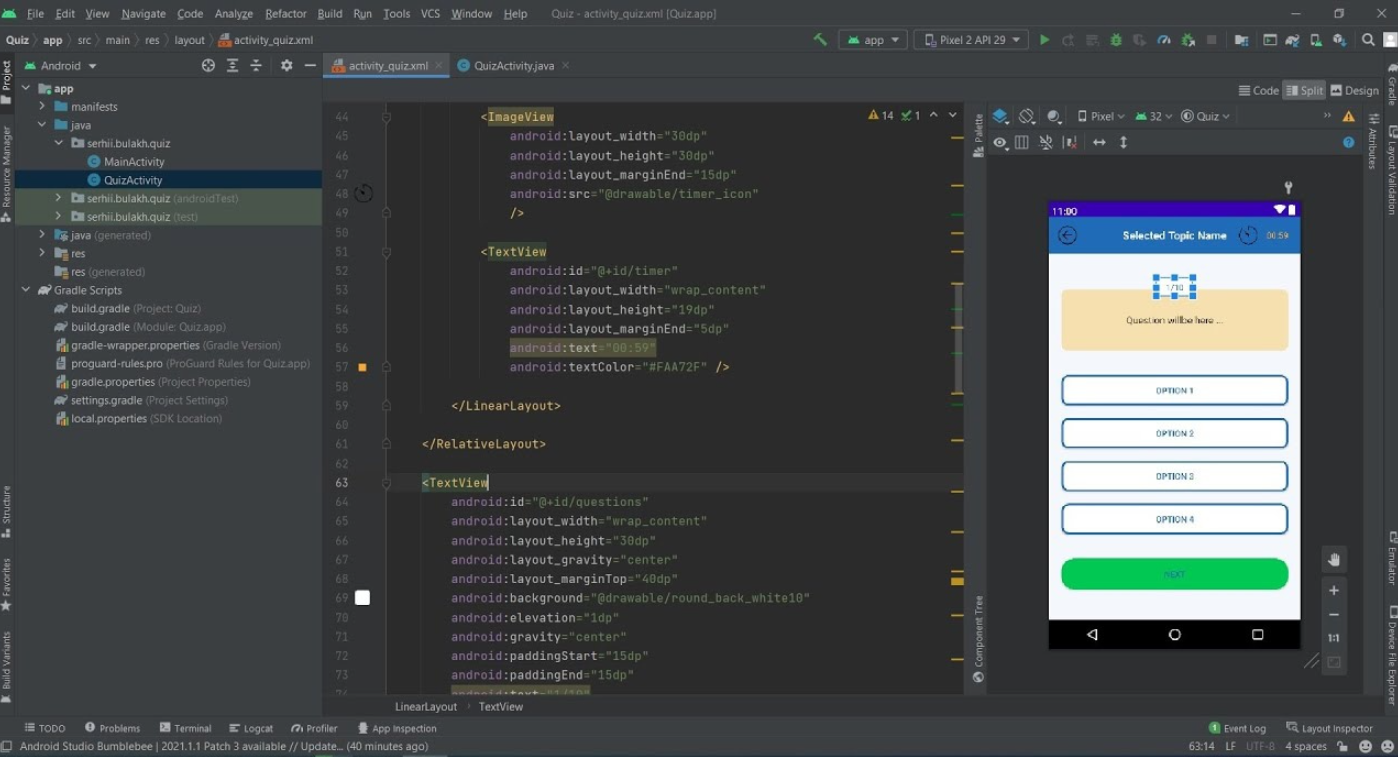


Рисунок 7 – код textView

Файл OrderListFragmet.axml содержит TextView для отображения текущего пользователя и ListView для отображения списка заказов. Для отображения отдельного объекта заказа в виде сложного элемента CardView с множеством TextView реализован адаптер данных OrderListAdapter который возвращает View для каждого элемента из списка на основе данных из объекта и модели, содержащихся в файле OrderListRow.axml.

Файл OrderDeatilsFragment.axml содержит ListView в котором в качестве заголовка используется развернутая информация по заказу,отображаемая с помощью модели OrderDeatilsHeader.axml, а элементы списка сгруппированы при помощи объектов данных об изображениях, прикрепленных к этому заказу и макета PhotoListRow.axml.

Файл ViewPhotoFragment.axml содержит элемент ImageView для транслирования изображения транспортного средства.

# 3 Результаты разработки мобильного приложения Effektus

## 3.1 Хранение данных

В операционной системе Android хранение данных приложения происходит путем использования баз данных SQLite. Чтобы работать с текущей базой для платформы Xamarin имеется способ через использование библиотеки SQLite.NET, которая в свою очередь взаимодействует с технологией Object–Relational Mapping.

При помощи ORM включенной в библиотеку SQLite.NET, нет такой нужды, как разработка схем баз данных для простых объектов, благодаря чему значительно ускоряется разработка приложения.

Чтобы хранить объекты в базе данных, можно просто описать их параметры используя пару атрибутов, описанных в самой библиотеке SQLite.NET.

Установка соединения с базой данных происходит через создание объекта SQLiteConnection и передачи ему в виде параметра путь к файлу базы данных. Если файл не найден, тогда он автоматически создается с указанным именем.

Во время обращения к таблицам, нужно обязательно проверять их существование через вызов обобщенного метода CreateTable класса SQLiteConnection. Таблица создается при условии, что ее нет в присутствии в базе данных.

Работа с данными осуществляется следующиит методами SQLiteConnection:

Insert – создает объект в необходимой таблице базы данных. Таблица сама определяется из значения, указанного в запросе;

Get<T> – предоставляет объект с определенным первичным ключом

Table<T> – показывает все существующие записи из таблицы;

Delete – стирает объект с нужным первичным ключом;

Query<T> – делает SQL запрос и в результате показывает список записей;

*Execute* – нужен для осуществления SQL запросов, которые не предполагают возврата значения, к примеру insert, update, delete.

С целью хранения маленькой коллекции информации переданных в виде ключ и значение, разрешено применять API групповых настроек SharedPreferences. Объект класса SharedPreferences дает обыкновенные технологии чтения или записи файла настроек программы, сохраняющего коллекцию пар ключ и значение.

При создании нового или получении доступа к существующему файлу настроек можно выбрать один из двух различных методов:

getSharedPreferences() – нужен, если необходимо производить работу с парой файлов настроек. Имя файла отправляется в качестве параметра;

getPreferences() – берется если создается или читается один файл совмеестных настроек для определенной активности.

Для указания уникального имени файлов настроек нужно использовать уникальное имя приложения, к примеру, "com.shablon.myapp.preference\_file\_key".

Запись настроек в файл происходит созданием объекта класса SharedPreferences.Editor, который вызвавается методом edit() объекта SharedPreferences.

Данные можно записать с использованием последовательного вызова методов putInt() или putString() при записи целочисленных и строчных значений и метода commit() нужного для сохранения изменений объекта SharedPreferences.

Для получения значения из файла настроек беруться такие методы, как getInt() и getString().

В разрабатываемом приложении используется шаблон Синглетон для получения доступа к общим настройкам. Синглетон – это порождающий шаблон проектирования, который обеспечивает нахождение в однопроцессном приложении единственного экземпляра определенного класса. Класс ApplicationPreferences реализует данный шаблон и предоставляет доступ к настройкам приложения через свойства. Для того, чтобы сохранить текущего активного пользователя коду достаточно привязать значение свойству CurrentUser класс, а для чтения настроек данных можно разумеется получить свойства CurrentUser.

Имеется пара API существующих в операционной системе Android, которые возможно применять для синхронизации данных с сервером.

Например это:

* AlarmManager;
* JobScheduler;
* GCM Network Manager;
* SyncAdapter.

Прежде чем применять какой–то из данных инструментов необходимо ознакомится с плюсами и минусами каждого их них, а также убедиться в их соответствии требованиям к работе приложения.

Класс AlarmManager дает возможность приобрести доступ к менеджеру сигнализаций значения ОС. AlarmManager разрешает приложению составлять план задач, включение коих возможно понадобиться за пределами жизненного цикла приложения. Благодаря этому приложение исполняет задуманные задачи даже если оно было выключено. Отлично действует, если нужно включить какую-нибудь службу с определенным промежутком времени. Операторная система Android пробует устроить пуск сигнализаций с схожим установленным интервалом времени повторения одновременно, для экономии энергии.

Одна из проблем, связанная с работой AlarmManager это то, что все запланированные сигналы стираются при перезагрузке телефона, чтобы этого избежать приложение должно иметь права receive\_boot\_complete для планирования запуска сервисов по новой после запуска операционной системы.

Класс JobScheduler подсобляет отлично действовать в низкоприоритетном режиме. Сервисы JobService стартуют, соответствуя критериям, поставленным в JobInfo с поддержкою метода JobInfo.Builder(). Данные критерии задают нужные обстоятельства для исполнения JobService, к примеру, проделываться лишь в системе ожидания, лишь при подсоединении к сети интернет либо исключительно при отсутствии лимитного подключения JobInfo еще имеет возможность устанавливать малые задержки среди исполненьями JobService и наивысшие промежутки ожидания исполнения JobService. Задачи хранятся в очередности операционной системы в случае если критерии не исполнены, для того чтоб исполниться позже. ОС еще попробует соединить данные задачи вместе таким образом, чтоб уменьшить время работы при включении к сети.

Чтобы создать подкласс обычного класса JobService нужно переделать методы onStartJob и onStopJob.

Метод onStartJob() – включается при работе планировщика задач. Этот метод нужен в главном потоке и может остановить активность приложения. Если нужно сделать длительную работу, то необходимо создать другой поток для ее исполнения и возвратить значение true в качестве результата метода либо false при ненужности создания второстепенного потока. Если вдруг потребовалось создание еще одного потока нужно обратиться к методу jobFinished() при завершении задания.

Метод onStopJob() вызвается ОС при завершения работы сервиса или если состояние смартфона перестает удовлетворять критериям, которые заданны в JobInfo. Имеет гораздо большую гибкостью, нежели чем AlarmManager. Запланированные так задачи не пропадают даже после перезагрузки смартфона.

GCM (Google Cloud Messaging) Network Manager просто поьзуется JobService для работы с ОС Android 5 и больше, а взаимодействие с предыдущими версиями Android происходит с помощью сервисов Google Play. GCMNetworkManager также имеет те же преимущества экономии заряда, как и JobScheduler, но при всем этом может похвастаться лучшей обратной совместимостью и более понятным API.

Задачи делаются с помощью методов OneoffTask.Builder() PeriodicTask.Builder(). Шаблон Builder делает проще добавление критериев планирования к заданию. Данные критерии можно сказать идентичны критериям JobInfo, и поэтому система старается оптимизировать и продлить время автономной работы путем объединения вызова заданий, спланированных через GCMNetworkManager.

Ради обрабатывания задуманных задач потребуется подкласс базисного класса GcmTaskService. GcmTaskService просит переопределения метода onRunTask(), а конкретно в нем обязана происходить работа. Способ onRunTask() стартует в фоновом режиме, благодаря чему его внедрение незначительно комфортней, нежели внедрение способа onStartJob() класса JobService.

В различие с JobScheduler задачи, поставленные через GCMNetworkManager, имеют все шансы быть затеряны при обновлении программы или же сервисов Google Play. Способ onInitializeTask() класса GcmTaskService способен находиться переопределен чтоб получить извещение о данном событии и подходящим образом откликаться на него при вторичном планировании задач. JobScheduler никак не сотворены для исполнения заданий сходу либо в конкретное момент. Они предусмотрены ради исполнения циклических либо однократных, низкоприоритетных служб чтобы сберечь временя работы батареи. Приложение, которому нужна какая-нибудь точности во времени призыва сервиса, никак не должно применять ни один из перечисленных способов планирования. Особенно важно это стало после вступления режима Doze в Android 5.

Адаптеры синхронизации сделаны с целью обеспечения решения задач синхронизации Android смартфонов и сервов. Пуск синхронизации инициируется преобразованиями информации на сервере либо устройстве, либо при истечении конкретного времени. ОС станет пробовать делать синхронизацию пакетно, чтоб сберечь продолжительность автономной работы смартфона, и транзакции станут копиться в очереди чтобы передаться позднее. Система будет запускать синхронизацию лишь при подключении смартфона к сети.

Большим преимуществом адаптеров синхронизации является возможность пользователю смотреть информацию про состояние синхронизации и наличие способа отключать ее.

Для применения синхронизации программам потребуется элемент называемый аутентификатор и заготовитель контента. Предполагается, будто приложение начнет обладать учетной записью и базой данных, ежели потребуется синхронизирование меж гаджетом и сервером. Не всегда, но для работы адаптера может потребоваться подклассы базисных классов AbstractAccountAuthenticator и ContentProvider. Программам, коие никак не нуждаются в данных классах, однако в каких необходимо применять адаптер, позволительно элементарно сменить аутентификатор и поставщик классами–заглушками.

Адаптер синхронизации создается в XML. Там определяется поставщик контента и аутентификатор.

Есть несколько вариантов запуска адаптера:

В связи с изменениями данных сервера. Пуск синхронизации в ответ на уведомление сервера, предписывающее, будто какие-либо данные на сервере поменялись. Данная функция дозволяет обновлять информацию с сервера на смартфоне без понижения работоспособности или же энергопотребления;

Из-за изменения информации на смартфоне. Пуск синхронизации после изменения информации на телефоне. Данная функция дозволяет посылать модифицированные данные с гаджета на сервер, полезно в тех случаях, когда серверу нужно обладать самыми последними данными смартфона;

При отправке сетевого уведомления. Пуск синхронизации, если ОС Android отправляет в сети известие, с информацией, которая удерживает TCP/IP соединение открытым;

Спустя определенные промежутки времени. Пуск адаптера при истечении определенного интервала времени, или же старт в конкретно заданное время дня;

По требованию. Запуск адаптера действием пользователя.

Про применения адаптеров потребуется незначительно больше затратить времени для настройки, нежели остальных методов, но, в случае если программа уже владеет аутентификатором и пользуется поставщиком контента, поздравляю, часть работы уже сделана. Самым сильным превосходством адаптеров синхронизации считается, что они смогут действовать с большим количеством поддерживаемых гаджетов и никак не требуют для своего функционирования добавочных сервисов.

## 

## 3.2 Выбор метода синхронизации

Есть много аспектов, которые следует учитывать при попытке выбрать правильный способ для планирования синхронизации данных в приложениях Android:

*AlarmManager*, позволяет строго держать частоту срабатывания сигнала и пробуждения устройства, но негативно влияет на автономную работу устройства;

*JobScheduler* обеспечивает эффективное планирование фоновых задач, но работает с версией Android выше 5;

*GCM Network Manager* обеспечивает эффективное планирование фоновых задач, если пользователь установил сервисы Google Play и проще в управлении, чем JobService.

Адаптеры синхронизации – хорошее решение для синхронизации локальных данных с данными сервера, если в приложении уже реализован аутентификатор и провайдеры контента.

Так как для разрабатываемого приложения наибольшим приоритетом обладает регулярность обновления информации, а время жизни устройства на батарее не играет никакой роли (мобильные устройство большую часть времени находятся на зарядке). Для планирования запуска синхронизации данных был выбран класс AlarmManager обладающий самым предсказуемым временем запуска и не требующий для своей работы наличия дополнительных сервисов.

Для осуществления процесса синхронизации в приложении реализованы два сервиса:

*UpdateMyTryService*, осуществляет синхронизацию списка существующих тренировок;

*UploadMyANDFriendsTryService*; осуществляет синхронизацию общих тренировок.

Оба этих сервиса являются наследниками базового класса IntentService.

Класс IntentService автоматически создает очередь намерений, через которые был вызван сервис, что упрощает работу с многопоточностью.

## 

## 3.3 Дальнейшее развитие и сопровождение приложения

Мобильное прибавление Effektus, готово к применению, однако на данном этапе его создание никак не заканчилось. Ежегодно, а то и чаще, для ОС android появляется больше, увлекательных возможностей. Время не стоит на одном месте. Этак и с мобильным ПО Effektus, со временем нужно проводить некие доработки и переделки возможностей, которые уже существуют.

Также, в ближайших планах на будущее, есть мысли о том, чтобы изменить полностью графический интерфейс приложения, используя не стандартные методы, предлагаемые Android Studio, а уже свои решения. Так же создать парочку возможностей, которые касаются безопасности. К примеру, возможность корректировать данные в созданных аккаунтах. Это может быть имя профиля или же изменение данных о пин-коде, пароле. Получится, что при потребности внести иные данные, пользователю не придется удалять и по новой регистрировать аккаунт, что в свою очередь является не самым удобным решением. В мыслях разработать возможность использования пароля не только вводом просто цифр и цифр с буквами, но и применение графического ключа, отпечатка пальца или faceid. Ни для кого не секрет, в сегодняшнем дне почти каждый второй смартфон обладает сканером отпечатка пальцев, а каждый 4 сканером faceid. Это и понятно, ведь такая функция дополняет возможности, многих приложений и самого смартфона. Все эти возможности позволяют приложению обрести новый уровень, новую уникальность, а для пользователя предоставят более совершенную безопасность персональных данных. Перечисленные и многие другие доработки будут разработаны в будущем.

# Заключение

В последние десятилетия работа за компьютером стала одной из самых популярных областей в трудовой деятельности. Сотрудники отдела сопровождения закупок играют важную роль в работе компании. В отличие от качественных показателей, эффективность тех или иных маркетинговых действий не может быть замерена в лаборатории, однако вполне можно сопоставить интерес потребителей и, как следствие, уровень прибыли с определённой маркетинговой активностью. Крупные предприятия интенсивнее проявляют заинтересованность в решениях, которые позволяют так или иначе снизить трудоёмкость рабочих процессов и заботы и сотрудников. Очевидно, что пара небольших функциональностей в компактном приложении для отслеживания количества и качества выполненных и будущих рабочих поручения –­ это лучшее что может быть для сотрудника. Разрабатываемый проект обладает лишь базовым функционалом, но при этом является вполне достаточным средством для помощи сотруднику в его деятельности. В качестве дальнейших этапов улучшения приложения можно рассматривать его портирование на другие мобильные операционные платформы, дальнейшее увеличение числа доступных функций, связанное с постепенным эволюционным развитием всей системы в целом, а также постепенную, более тесную интеграцию с другими приложениями, которые позволяют, например создавать или отправлять заявки без обращения в другие программы.

# Список использованных источников

1. Альфред В. Ахо Структуры данных и алгоритмы / Альфред В. Ахо, Джон Э. Хопкрофт, Джеффри Д. Ульман – Вильямс, 2010. – 400 с.
2. Веллинг Люк Разработка приложений с помощью MySQL / Веллинг Люк, Томсон Лора – Вильямс, 2012.
3. Гвоздева Т. В. Проектирование информационных систем / Гвоздева Т. В., Баллод Б. А. – Феникс, 2011.
4. Граннеман Скотт Java. Необходимый код и команды / Карманный справочник – Вильямс, 2011.
5. Гудман Д. Java и Android. Сборник рецептов. Для профессионалов / Питер, 2014. – 528 с.
6. Общая классификация CMS. - Текст: электронный. - URL: http://www.solus.ru/articles\_9.html (дата обращения 21.04.2022). - Режим доступа: свободный.
7. Гудман Дэнни Android. Библия пользователя / Гудман Дэнни, Моррисон Майкл – Диалектика, 2016.
8. Jay Kreibich. Using SQLite / O’Reilly Media.: 2011.
9. Нейгел, Кристиан, Ивьен, Билл, Глинн, Джей, и др. Java для профессионалов / Перев. с англ. – М.: Издательский дом «Вильямс», 2012. – 1376 с
10. Нейгел, Кристиан, Ивьен, Билл, Глинн, Джей, Карли Уотсон, Морган Скиннер. Java и платформа Android для профессионалов / М.: Издательский дом «Вильямс», 2011. – 1376 с.
11. Троелсен Э. Ядро Linux. Библиотека программиста / СПб.: Питер, 2014.
12. Троелсен Э. Язык программирования Java и платформа Android/ М.: Издательский дом «Вильямс» 5-е издание, 2015. – 1221 с
13. Макконнелл Стив Совершенный код. Мастер-класс / пер. с англ. – Питер, 2013.
14. MSDN – сеть разработчиков Майкрософт: - URL: https://msdn.microsoft.com (дата обращения 16.08.2023). - Режим доступа: свободный.
15. КиберФорум – форум программистов и сисадминов - URL: http://www.cyberforum.ru (дата обращения 19.08.2023). - Режим доступа: свободный.
16. Скиллкодинг - Текст: электронный. - URL: http://skillcoding.com (дата обращения 22.08.2023). - Режим доступа: свободный.
17. Введение в разработку приложений для ОС Android. - URL: http://seokleo.ru (дата обращения 29.08.2023). - Режим доступа: свободный.
18. Информационный портал по разработке и продвижению сайтов. - URL: http://www.intuit.ru/studies/courses/12643/1191/lecture/22004 (дата обращения 05.09.2023). - Режим доступа: свободный.
19. API как средство интеграции приложений - URL: http://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:Application\_Programming\_Interface\_(API) (дата обращения 14.09.2023). - Режим доступа: свободный.
20. Блог о компьютерах - URL: http://worldreferat.ru (дата обращения 08.09.2023). - Режим доступа: свободный.
21. [Словари и энциклопедии](http://dic.academic.ru/) [Электронный ресурс]. – Режим доступа: http://dic.academic.ru (дата обращения 19.09.2023). - Режим доступа: свободный.
22. Обучающие курсы по программированию: электронный. - URL: https://mva.microsoft.com (дата обращения 10.09.2023). - Режим доступа: свободный.
23. Обучающий портал для программистов. - URL: https://geekbrains.ru (дата обращения 22.08.2023). - Режим доступа: свободный.
24. Руководство по языку JavaScript - URL: https://learn.javascript.ru (дата обращения 29.08.2023). - Режим доступа: свободный.
25. Руководство по языку C# - URL: https://docs.microsoft.com (дата обращения 29.08.2023). - Режим доступа: свободный.
26. Болезни от продолжительной работы за компьютером - URL: http://www.medicus.ru/wellness/patient/bolezni-otprodolzhitelnoj-raboty-za-kompyuterom-162106.phtml (дата обращения 20.08.2023). - Режим доступа: свободный.
27. Сколько дней в году люди тратят на смартфоны - URL: http://mir24.tv/news/hi-tech/13588422 (дата обращения 27.08.2023). - Режим доступа: свободный.
28. 51% трафика в Рунете приходится на мобильные устройства. - URL: https://geektimes.ru/post/242645/ (дата обращения 01.08.2023). - Режим доступа: свободный.
29. Лучшие операционные системы для смартфонов - URL: http://topmira.com/tehnika/item/176-os-smartfony (дата обращения 10.08.2023). - Режим доступа: свободный.
30. Официальный сайт компании «Вконтакте»: официальный сайт. - URL: www.vk.ru (дата обращения 10.08.2023). - Режим доступа: свободный.
31. Visual Studio и Xamarin - URL: https://msdn.microsoft.com/ru-ru/library/mt299001.aspx (дата обращения 09.08.2023). - Режим доступа: свободный.
32. Обзор платформы Eclipse - URL: http://hightech.in.ua/content/art-eclipse-platform (дата обращения 09.08.2023). - Режим доступа: свободный.
33. Android от А до Я: Что такое Dalvik - URL: http://droidtune.com/2056/android-ot-a-do-ya-chto-takoe-dalvik.html (дата обращения 09.08.2023). - Режим доступа: свободный.
34. Documentation Android Developers. - URL: https://developer.android.com/docs (дата обращения 10.09.2023). - Режим доступа: свободный.
35. Тестирование Android приложений - URL: http://getbug.ru/testirovaniya-android-prilozheniy/ (дата обращения 10.09.2023). - Режим доступа: свободный.
36. Одним словом. Нейминг для мобильных приложений - URL: https://geekbrains.ru/posts/mob\_naming (дата обращения 19.09.2023). - Режим доступа: свободный.
37. Ахметов А. К. Операционная система Android: история создания и развития. Разработка приложений для платформы Android - URL: https://cyberleninka.ru/article/n/operatsionnayasistema-android-istoriya-sozdaniya-i-razvitiya-razrabotka-prilozheniy-dlya-platformy-android (дата обращения 10.08.2023). - Режим доступа: свободный.
38. Martin Fowler. UML Distilled: A Brief Guide to the Standard Object Modeling Language (Object Technology Series). Addison Wesley, 2003
39. Bill Phillips, Brian Hardy. Android Programming: The Big Nerd Ranch Big. Nerd Ranch Guides, 2013
40. Голощапов А.Л. Google Android. Создание приложений для смартфонов и планшетных ПК. Издательство Питер 2012.