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

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

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Автоматизация разработки электронных обучающих программ

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

Выполнил: Жалгасов Ж.С. Группа: ВТ-15-2
Научный руководитель: профессор, доктор технических наук Казиев Г.З

Консультанты:
по экономической части: к.э.н., профессор _____ Ж.Г. Аренбаева
«13» мая 2019 г.
по безопасности
жизнедеятельности: д.т.н., ст. преп. _____ Ш.Ш. Бекбасаров
«7» мая 2019 г.
по применению
вычислительной техники: ст. преп. _____ М.Н. Майкотов
«16» мая 2019 г.

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

Рецензент: д.т.н., профессор _____ У.А. Тукеев
«____» _____ 2019 г.

Алматы, 2019 г.

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

Институт систем управления и информационных технологий

Кафедра IT-инжиниринг

Специальность 5В070400 – «Вычислительная техника и программное обеспечение»

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Жалгасову Жандосу Садировичу

Тема проекта: Автоматизация разработки электронных обучающих программ

Утверждена приказом по университету № 33 от «01» марта 2019 г.

Срок сдачи законченного проекта «24» мая 2019 г.

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

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

Перечень графического материала (с точным указанием обязательных чертежей): представлены 10 таблиц, 29 иллюстрации.

Основная рекомендуемая литература:
1 Ben Evans, «Мобильники Съедят Мир», 2013.
2 Dobie, Alex "Android Studio unveiled at Google I/O keynote". Android Central. Mobile Nations. Retrieved May 16, 2013.
3 Honig, Zach. "Google intros Android Studio, an IDE for building apps". Engadget. AOL. Retrieved May 16, 2013.
4 Olanoff, Drew "Google Launches Android Studio And New Features For Developer Console, Including Beta Releases And Staged Rollout". TechCrunch. AOL. Retrieved May 16, 2013.

Консультации по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант	Сроки	Подпись
Экономическая часть	Аренбаева Ж.Г.	04.03.2019 – 13.05.2019	
Безопасность жизнедеятельности	Бекбасаров Ш.Ш.	5.03.2019 – 7.05.2019	
Программное обеспечение	Майкотов М.Н.	1.04.2019 – 15.05.2019	
Нормоконтролер	Айтказина А.А.	02.04.19 – 15.05.19	

ГРАФИК
подготовки дипломной проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Аналитическая часть		
Проектная часть		
Экспериментальная часть		

Дата выдачи задания «28» октября 20__ г.

Заведующий кафедрой _____ Т.С. Картбаев

Научный руководитель проект _____ Г.З. Казиев

Задание принял к
исполнению студент _____ Ж.С. Жалгасов

Аңдатпа

Бұл дипломдық жобада Android операциялық жүйесіне арналған электрондық оқыту бағдарламаларын әзірлеуді автоматтандыру қарастырылған. Android Studio платформасында жобаларды әзірлеу туралы үлкен білімі жоқ және көп уақыт пен ресурстарды жұмсамай, өз оқу платформасын құру үшін пайдаланушыларға құралдарды ұсынады және курстар мазмұнына көңіл бөлуге мүмкіндік беретін Android Academy білім беру бағдарламасының үлгісі жасалды

Сонымен қатар, қарастырылған қосымшаларды әзірлеу үшін еңбек жағдайына талдау жасалды.

Сондай-ақ жобаның экономикалық орындылығын растайтын экономикалық негіздемесі жасалды.

Аннотация

В данном дипломном проекте рассмотрена автоматизация разработки электронных обучающих программ для операционной системы Android. Создан шаблон образовательного приложения Android Academy, который позволяет сосредоточиться на содержании курсов и предоставляет пользователям инструменты для создания своей учебной платформы, не тратя на них много времени и ресурсов и не имея больших знаний о разработке проектов на платформе Android Studio

Кроме того, сделан анализ условий труда для разработки рассматриваемого приложения.

Также составлено экономическое обоснование проекта, подтверждающее его экономическую целесообразность.

Annotation

In this diploma project, the automation of the development of electronic training programs for the Android operating system is considered. A template of the educational application Android Academy has been created, which allows you to focus on the content of the courses and provides users with tools to create their learning platform without spending a lot of time and resources on them and without having much knowledge about the development of projects on the Android Studio platform

In addition, the analysis of working conditions for the development of the application.

Also the economic justification of the project confirming its economic feasibility is made.

Содержание

Введение

Автоматизация любого рода направлена на минимизацию или устранение человеческого взаимодействия в процессе.

Автоматизация изменила почти каждую отрасль, а также то, как мы производим товары и поставляем многие услуги. На самом деле, мы принимаем многие автоматизированные системы как должное. Допустим, вы зарегистрировались на новый банковский счет в интернете: вы можете получить приветственное письмо и заявление о раскрытии продукта, а также пакет информации по почте. Или вы можете получить определенное количество очков вознаграждения в программе лояльности вашего любимого магазина, вызвав обновление до VIP-членства. Может, они и не такие броские, как роботы-манипуляторы, создающие высококлассные компьютеры, но все они примеры автоматизации.

В то время как преподаватели в классах приняли автоматизацию для административных задач, таких как маркировка экзаменов с множественным выбором или посещение класса, технология электронного обучения идеально подходит для более широкой автоматизации.

По мере того как электронное обучение становится все более популярным и более широко используется в реальном обучении, возможности интеграции автоматизации в технологии и программное обеспечение, стоящие за ним, также растут. И нетрудно понять, почему он взлетает, с таким количеством возможностей, доступных автоматизация в электронном обучении может сделать большую часть образовательного контента, чтобы вы могли более эффективно форматировать его и распространять его среди пользователей.

Автоматизация также может изменить процесс электронного обучения – например, с помощью программного обеспечения автоматически определить, как наилучшим образом адаптировать учебный контент для пользователей и групп. Прежде чем начать курс, сотрудники могут пройти быстрый тест, чтобы выяснить, какие области они уже знакомы, и где им нужно сосредоточиться немного больше. Затем платформа может отбрасывать и добавлять модули в соответствии с уровнем опыта человека и оптимизировать процесс обучения.

Образование наиболее эффективно, когда оно вовлекает, и убедившись, что ученик не покрывает знакомую почву, поможет им заинтересовать и быстро продвигаться по курсу.

Для максимального эффекта этот вид автоматизации должен быть интегрирован с этапа проектирования, чтобы обеспечить гибкость и эффективность всех курсов.

Автоматизация электронного обучения может обеспечить преимущества как для учителей, так и для студентов с точки зрения рабочего процесса, посредством автоматических уведомлений о новых модулях или оценках, автоматического построения курса на основе обратной связи с учащихся и

встроенных оценок. С платформой, автоматически запускающей ответы, основанные на поведении обучаемого – прохождение или провал викторины, отставание от материала или досрочное завершение модуля – никто не остается позади, и ничто не проскальзывает через трещины. Преподаватели имеют больше времени для преподавания, и студенты всегда в курсе.

Автоматизированные решения могут помочь преподавателям более эффективно создавать новые курсы или учебные материалы, экономя время и деньги. Например, экзамены, викторины и вопросы пересмотра могут быть автоматически сгенерированы на основе содержания курсовой работы.

Пользователи на определенном этапе своей карьеры - будь то новички в отрасли, продвигаемые по службе или даже участвующие в дисциплинарных процессах – могут быть автоматически зачислены в правильные учебные модули для них, экономя время для менеджеров и HR-отделов.

Преимущества автоматизации в электронном обучении не только экономические. Поскольку преподаватели переходят от традиционного сосредоточения на группах к индивидуальному подходу, автоматизация дает много преимуществ. Это может изменить способ обучения студентов-точно адаптировать материал к группам и пользователям.

Допустим, пользователь не прошел тест. Это может привести к тому, что платформа отправит им электронное письмо с некоторыми дополнительными ресурсами или зачислит их в догоняющее подразделение с мини-модулями, ориентированными на области, которые нуждаются в работе. Наблюдение за тем, сколько пользователей перенаправлено в эту систему, также может стать отличным инструментом отслеживания для преподавателей, которые могут видеть, какие части курса нуждаются в более подробной информации и какие методы обучения работают лучше всего.

Другими словами, автоматизация электронного обучения может работать в виде петли обратной связи для проверки и измерения знаний и, в свою очередь, создавать эффективный и релевантный новый контент для учащихся.

Нетрудно понять, как автоматизация может изменить электронное обучение и в процессе этого изменить способ создания и потребления всех видов образовательного контента. Более специализированный контент означает больше вовлеченных учащихся, что приводит к лучшим результатам, более высоким показателям завершения и более эффективным курсам.

Автоматизация становится все более важным инструментом для преподавателей в крупных организациях. Вот почему менеджерам по персоналу важно учитывать, как выбранная ими система управления обучением адаптирует автоматизацию, позволяя им адаптировать инновационные решения, которые упрощают процесс обучения и обогащают опыт учащихся.

Чтобы узнать больше о том, как технология электронного обучения может помочь подготовиться к грядущим тенденциям в образовании рабочей

силы в этом дипломном проекте создан шаблон приложения на Android Studio, который можно модифицировать под себя.

1 Принципы разработок на Android Studio

1.1 Операционная система Android

Операционная система Android − это многопользовательская система Linux, в которой каждое приложение является другим пользователем.

По умолчанию система назначает каждому приложению уникальный идентификатор пользователя Linux (этот идентификатор используется только системой и неизвестен приложению). Система устанавливает разрешения для всех файлов в приложении, чтобы только идентификатор пользователя, назначенный этому приложению, мог получить к ним доступ.

Каждый процесс имеет свою собственную виртуальную машину (VM), поэтому код приложения выполняется отдельно от других приложений.

По умолчанию, каждое приложение работает в собственном Linux процесса. Система Android запускает процесс, когда любой из компонентов приложения должен быть выполнен, и затем завершает процесс, когда он больше не нужен или, когда система должна восстановить память для других приложений.

Система Android реализует принцип наименьших привилегий. То есть каждое приложение по умолчанию имеет доступ только к тем компонентам, которые необходимы для его работы и не более. Это создает очень безопасную среду, в которой приложение не может получить доступ к частям системы, для которых ему не дано разрешение. Однако есть способы для приложения обмениваться данными с другими приложениями и для приложения для доступа к системным службам:

− Можно организовать два приложения для совместного использования одного и того же идентификатора пользователя Linux, и в этом случае они могут получить доступ к файлам друг друга. Для экономии системных ресурсов приложения с одинаковым идентификатором пользователя могут работать в одном процессе Linux и совместно использовать одну виртуальную машину. Приложения также должны быть подписаны одним сертификатом.

− Приложение может запросить разрешение на доступ к данным устройства, таким как контакты пользователя, SMS-сообщения, монтируемое хранилище (SD-карта), камера и Bluetooth. Пользователь должен явно предоставить эти разрешения. Дополнительные сведения см. В разделе Работа с системными разрешениями.

Этот документ вводит следующие понятия:

− Основные компоненты, которые определяют приложением.

− Файл манифеста, в котором объявляются компоненты и необходимые функции устройства для приложения.

− Ресурсы, которые отделены от кода приложения и позволяют приложению изящно оптимизировать его поведение для различных конфигураций устройств.

Компоненты приложения являются основными строительными блоками приложения для Android. Каждый компонент является точкой входа, через которую система или пользователь может войти в приложение. Некоторые компоненты зависят от других.

Существует четыре различных типа компонентов приложения:

- Activities;
- Services;
- Broadcast receivers;
- Content providers;

Каждый тип служит определенной цели и имеет определенный жизненный цикл, который определяет, как компонент создается и уничтожается. В следующих разделах описаны четыре типа компонентов приложения.

Activity − это точка входа для взаимодействия с пользователем. Он представляет собой единый экран с пользовательским интерфейсом. Например, в приложении электронной почты может быть одно действие, отображающее список новых писем, другое действие для создания письма и другое действие для чтения писем. Хотя мероприятия работают вместе, чтобы сформировать сплоченный пользовательский опыт в приложении электронной почты, каждый из них независим от других. Как таковое, другое приложение может запустить одно из этих действий, если приложение электронной почты позволяет. Например, приложение камеры может запустить действие в приложении электронной почты, которое создает новую почту, чтобы пользователь мог поделиться изображением. Действие облегчает следующие ключевые взаимодействия между системой и приложением:

- отслеживание того, что пользователь в настоящее время заботится о (что на экране), чтобы убедиться, что система продолжает работать процесс, в котором размещается деятельность.
- зная, что ранее используемые процессы содержат вещи, к которым пользователь может вернуться (остановленные действия), и, таким образом, более высокий приоритет сохранения этих процессов.
- помогая приложению обрабатывать процесс, чтобы пользователь мог вернуться к действиям с восстановленным предыдущим состоянием.
- предоставление возможности приложениям реализовывать потоки пользователей между собой, а системе координировать эти потоки. (Самый классический пример здесь поделиться.)

Действие реализуется как подкласс класса Activity. Для получения дополнительной информации о классе деятельности, см. деятельность руководства разработчика.

Services − это точка входа общего назначения для сохранения приложения в фоновом режиме по всевозможным причинам. Это компонент, который работает в фоновом режиме для выполнения длительных операций или выполнения работы для удаленных процессов. Служба не предоставляет пользовательский интерфейс. Например, служба может воспроизводить

музыку в фоновом режиме, когда пользователь находится в другом приложении, или получать данные по сети, не блокируя взаимодействие пользователя с активностью. Другой компонент, например, activity, может запустить службу и позволить ей работать или привязываться к ней для взаимодействия с ней. На самом деле есть две очень разные службы семантики, которые рассказывают системе о том, как управлять приложением: запущенные службы говорят системе, чтобы они работали, пока их работа не будет завершена. Это может быть синхронизация некоторых данных в фоновом режиме или воспроизведение музыки даже после выхода пользователя из приложения. Синхронизация данных в фоновом режиме или воспроизведение музыки также представляют два различных типа запущенных служб, которые изменяют способ их обработки системой:

− воспроизведение музыки − это то, что пользователь непосредственно знает, поэтому приложение сообщает системе об этом, говоря, что оно хочет быть на переднем плане с уведомлением, чтобы сообщить об этом пользователю; в этом случае система знает, что она должна очень стараться, чтобы процесс этой службы работал, потому что пользователь будет недоволен, если он уйдет.

− обычная фоновая служба не является чем-то, что пользователь непосредственно знает, как работает, поэтому система имеет больше свободы в управлении своим процессом. Это может позволить ему быть убитым (а затем перезапустить службу некоторое время спустя), если ему нужна ОЗУ для вещей, которые представляют более непосредственный интерес для пользователя.

Связанные службы запускаются, потому что какое-то другое приложение (или система) заявило, что хочет использовать службу. Это в основном сервис, предоставляющий API другому процессу. Таким образом, система знает, что между этими процессами существует зависимость, поэтому, если процесс А привязан к службе в процессе В, она знает, что ей нужно поддерживать процесс В (и его службу) для А. далее, если процесс – А это то, о чем заботится пользователь, то он также знает, что процесс В также заботится о пользователе. Благодаря своей гибкости (к лучшему или худшему) сервисы оказались действительно полезным строительным блоком для всех видов системных концепций более высокого уровня. Живые обои, прослушиватели уведомлений, хранители экрана, методы ввода, службы доступности и многие другие основные функции системы построены как службы, которые реализуют приложения, и система привязывается к тому, когда они должны быть запущены.

Сервис реализуется как подкласс сервиса. Дополнительные сведения о классе служб см. В руководстве разработчика служб.

Широковещательный приемник (BroadcastReceiver) − это компонент, который позволяет системе доставлять события в приложение вне обычного потока пользователей, позволяя приложению реагировать на общесистемные широковещательные объявления. Поскольку широковещательные приемники

являются еще одной четко определенной записью в приложении, система может доставлять трансляции даже в приложения, которые в настоящее время не запущены. Так, например, приложение может запланировать оповещение для публикации уведомления о предстоящем событии... и поставляя этот сигнал тревоги в BroadcastReceiver приложения, нет необходимости в том, чтобы приложение оставалось запущенным до тех пор, пока сигнал тревоги не погаснет. Многие трансляции исходят из системы—например, трансляция, объявляющая, что экран выключен, батарея разряжена, или изображение было захвачено. Приложения также могут инициировать трансляции—например, чтобы другие приложения знали, что некоторые данные загружены на устройство и доступны для использования. Хотя широковещательные приемники не отображают пользовательский интерфейс, они могут создать уведомление строки состояния, чтобы предупредить пользователя о возникновении широковещательного события. Однако чаще всего широковещательный приемник является просто шлюзом для других компонентов и предназначен для выполнения очень минимального объема работы. Например, он может запланировать JobService для выполнения некоторых работ на основе события с JobScheduler

BroadcastReceiver реализован как подкласс BroadcastReceiver, и каждая трансляция доставляется как объект Intent. Дополнительные сведения см. В разделе класс BroadcastReceiver.

1.1.1 Встроенные возможности Android

Android (API level 28) представляет новые функции и возможности для пользователей и разработчиков.

Android добавляет поддержку платформы для протокола IEEE 802.11 mc Wi-Fi, также известного как Wi−Fi Round-Trip-Time (RTT), чтобы вы могли воспользоваться внутренним позиционированием в своих приложениях.

На устройствах под управлением Android 9 с аппаратной поддержкой ваши приложения могут использовать API RTT для измерения расстояния до близлежащих точек доступа Wi-Fi (APs) с поддержкой RTT. На устройстве должны быть включены службы определения местоположения и включено сканирование Wi-Fi (в разделе Настройки> расположение), а приложение должно иметь разрешение ACCESS_FINE_LOCATION. Для использования RTT устройству не нужно подключаться к точкам доступа. Чтобы сохранить конфиденциальность, только телефон может определить расстояние до точки доступа; точки доступа не имеют этой информации.

Если устройство измеряет расстояние до 3 или более точек доступа, можно использовать алгоритм мультилатерации для оценки положения устройства, которое наилучшим образом соответствует этим измерениям. Результат типично точен не познее 1 до 2 метра.

С этой точностью вы можете создавать новые впечатления, такие как встроенная навигация и мелкозернистые сервисы на основе местоположения,

такие как двусмысленное голосовое управление (например, "включите этот свет") и информация о местоположении (например, "есть ли специальные предложения для этого продукта?").

Android предлагает поддержку новейших экранов edge-to-edge, которые содержат вырезы для камер и динамиков. Класс DisplayCutout позволяет узнать расположение и форму нефункциональных областей, в которых не должно отображаться содержимое. Чтобы определить наличие и размещение этих областей выреза, используйте метод getDisplayCutout ().

Новое окно макета атрибутов, layoutInDisplayCutoutMode, позволяет приложение, чтобы выкладывать его содержимое вокруг устройства вырезы. Для этого атрибута можно задать одно из следующих значений:

− layout_in_display_cutout_mode_default;

− layout_in_display_cutout_mode_short_edges;

− layout_in_display_cutout_mode_never.

Вы можете имитировать вырез экрана на любом устройстве или эмуляторе под управлением Android следующим образом:

− включить параметры разработчика;

− на экране Параметры разработчика прокрутите вниз до раздела чертеж и выберите имитация отображения с вырезом;

− выберите размер выреза.

Android 9 представляет несколько улучшений уведомлений, все из которых доступны разработчикам, ориентированным на уровень API 28 и выше.

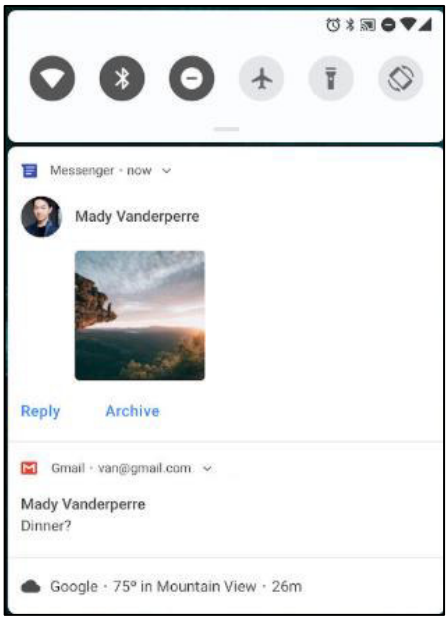


Рисунок 1.1.1 − Уведомления MessagingStyle с прикрепленной фотографией.

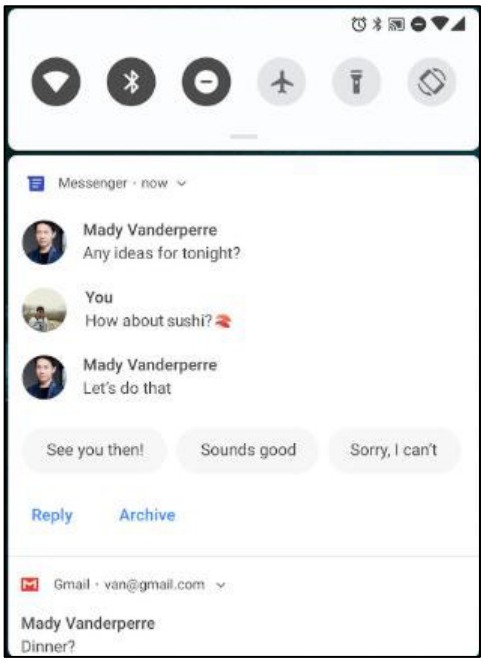


Рисунок 1.1.2 − Уведомление об обмене сообщениями MessagingStyle с ответами и разговорами.

Расширенные возможности обмена сообщениями. Начиная с Android 7.0 (уровень API 24), можно добавлять действие для ответа на сообщения или ввести другой текст непосредственно из уведомления. Android 9 улучшает эту функцию со следующими улучшениями:

− упрощенная поддержка участников разговора: класс Person используется для идентификации людей, участвующих в разговоре, включая их аватары и URI. Многие другие API-интерфейсы, такие как метод addmessage (), теперь использовать класс человека, а не объект CharSequence. Класс Person также поддерживает шаблон проектирования Builder.

− поддержка изображений: Android 9 теперь отображает изображения в уведомлениях обмена сообщениями на телефонах. Для отображения изображения в сообщении можно использовать setData (). В следующем фрагменте кода показано, как создать человека и сообщение, содержащее изображение.

```
// Create new Person.
Person sender = new Person()
        .setName(name)
        .setUri(uri)
        .setIcon(null)
        .build();
// Create image message.
Message message = new Message("Picture", time, sender)
        .setData("image/", imageUri);
Notification.MessagingStyle style = new
Notification.MessagingStyle(getUser())
```

```
        .addMessage("Check this out!", 0, sender)
        .addMessage(message);
```

− сохранить ответы как черновики: приложение может получить EXTRA_REMOTE_INPUT_DRAFT, отправленный системой, когда пользователь непреднамеренно закрывает уведомление обмена сообщениями. Можно использовать это дополнительно для предварительного заполнения текстовых полей в приложении, чтобы пользователи могли закончить свой ответ.

− определение, является ли разговор групповым: можно использовать setGroupConversation (), чтобы целенаправленно идентифицировать разговор как групповой или негрупповой.

− задать семантическое действие для намерения: метод setSemanticAction () позволяет придать семантическое значение действию, например, "отметить как прочитанное", "удалить", "ответить" и т. д.

− SmartReply: Android 9-поддерживает те же предлагаемые ответы, доступные в приложении для обмена сообщениями. Использование RemoteInput.setChoices (), чтобы предоставить массив стандартных ответов пользователю.

Android 8.0 представил каналы уведомлений, позволяющие создавать настраиваемые пользователем каналы для каждого типа уведомлений, которые можно отобразить. Android 9 упрощает настройки канала уведомлений с этими изменениями:

− блокировка группы каналов: теперь пользователи могут блокировать целые группы каналов в настройках уведомлений для приложения. Можно использовать метод isBlocked (), чтобы определить, когда группа заблокирована, и в результате не отправлять уведомления для каналов в этой группе. Кроме того, приложение может запросить текущие параметры группы каналов с помощью нового метода getNotificationChannelGroup ().

− новые типы широковещательных намерений: система Android теперь отправляет широковещательные намерения при изменении состояния блокировки каналов уведомлений и групп каналов. Приложение, которому принадлежит заблокированный канал или группа, может слушать эти намерения и реагировать соответствующим образом. Дополнительные сведения об этих действиях и дополнительных функциях см. В обновленном списке констант в ссылке NotificationManager.

Диспетчер уведомлений. Политика имеет три новые категории приоритетов " не беспокоить:

− priority_category_alarms приоритезирует сигналы тревоги;

− priority_category_media приоритезирует звуки из медиаисточников, таких как медиа и голосовая навигация;

− priority_category_system приоритезирует системные звуки.

Диспетчер уведомлений. Политика также содержит семь новых констант "не беспокоить", которые можно использовать для подавления визуального прерывания:

− suppressed_effect_full_screen_intent предотвращает запуск полноэкранного действия уведомления;

− suppressed_effect_lights блокирует индикаторы уведомлений;

− suppressed_effect_peek предотвращает кратковременное скольжение уведомлений в поле зрения ("подглядывание");

− suppressed_effect_status_bar предотвращает появление уведомлений в строке состояния на устройствах, поддерживающих строки состояния;

− suppressed_effect_badge блокирует значки на устройствах, поддерживающих маркировку;

− suppressed_effect_ambient блокирует уведомления на устройствах, поддерживающих внешние дисплеи;

− suppressed_effect_notification_list предотвращает появление уведомлений в представлении списка на устройствах, поддерживающих представление списка, таких как тень уведомлений или экран блокировки.

Android 9 представляет класс ImageDecoder, который обеспечивает модернизированный подход к декодированию изображений. Используйте этот класс вместо BitmapFactory и BitmapFactory.Options API.

ImageDecoder позволяет создавать Катры или растрового изображения из буфера байта, файла или URI. Чтобы декодировать изображение, сначала вызовите createSource () с источником закодированного изображения. Затем вызовите decodeDrawable () или decodeBitmap (), передав ImageDecoder.Исходный объект для создания Drawable или растрового изображения. Чтобы изменить настройки по умолчанию, передайте OnHeaderDecodedListener в decodeDrawable () или decodeBitmap (). ImageDecoder вызывает onHeaderDecoded () с шириной и высотой изображения по умолчанию, как только они известны. Если закодированное изображение является анимированным GIF или WebP, decodeDrawable () возвращает Drawable, который является экземпляром класса AnimatedImageDrawable.

Для задания свойств изображения можно использовать различные методы:

− чтобы масштабировать декодированное изображение до точного размера, передайте целевые размеры в setTargetSize (). Можно также масштабировать изображения с помощью размера выборки. Передайте размер выборки непосредственно в setTargetSampleSize ().

− чтобы обрезать изображение в пределах диапазона масштабируемого изображения, вызовите setCrop ().

− чтобы создать изменяемое растровое изображение, передайте true в setMutableRequired ().

ImageDecoder также позволяет добавлять индивидуальные и сложные эффекты к изображению, такие как закругленные углы или круговые маски. Использование setPostProcessor () с экземпляром класса PostProcessor для выполнения любых команд рисования.

Android 9 представляет класс AnimatedImageDrawable для рисования и отображения анимированных изображений GIF и WebP. AnimatedImageDrawable работает аналогично AnimatedVectorDrawable в том, что поток визуализации управляет анимацией AnimatedImageDrawable. Поток рендеринга также использует рабочий поток для декодирования, чтобы декодирование не мешало другим операциям в потоке рендеринга. Эта реализация позволяет приложению отображать анимированное изображение без управления его обновлениями или вмешательства в другие события в потоке пользовательского интерфейса вашего приложения.

В AnimatedImageDrawable могут быть декодированы с использованием экземпляра ImageDecoder. В следующем фрагменте кода показано, как использовать ImageDecoder для декодирования AnimatedImageDrawable:

```
private void decodeImage() throws IOException {
  Drawable decodedAnimation = ImageDecoder.decodeDrawable(
    ImageDecoder.createSource(getResources(), R.drawable.my_drawable));
  if (decodedAnimation instanceof AnimatedImageDrawable) {
    // Prior to start(), the first frame is displayed.
    ((AnimatedImageDrawable) decodedAnimation).start ();
  }
```

ImageDecoder имеет несколько методов, позволяющих вам в дальнейшем модифицировать изображения. Например, можно использовать метод setPostProcessor() для изменения внешнего вида изображения, например применения маски окружности или закругленных углов.

Начиная с Android 9, JobScheduler может использовать сигналы состояния сети, предоставляемые операторами для улучшения обработки сетевых заданий.

Задания могут объявлять предполагаемый размер данных, предварительную выборку сигналов и указывать подробные требования к сети. Затем JobScheduler управляет работой в соответствии со статусом сети. Например, когда сеть сигнализирует, что она перегружена, JobScheduler может отложить большие сетевые запросы. В незамеченной сети JobScheduler может выполнять задания предварительной выборки для улучшения работы пользователя, например, путем предварительной выборки заголовков.

При добавлении заданий обязательно используйте setEstimatedNetworkBytes (), setPrefetch() и setRequiredNetwork (), когда это необходимо, чтобы помочь JobScheduler правильно обрабатывать работу. При выполнении задания обязательно используйте сетевой объект, возвращаемый JobParameters.getNetwork (). В противном случае неявно используется сеть устройства по умолчанию, которая может не соответствовать требованиям юзера, вызывая непреднамеренное использование данных.

1.1.2 Основные характеристики среды разработки для платформы Android

Основной особенностью Android Studio как платформы для разработо является API.

Программы в данной среде разработки становятся частью самого мобильного устройства вместе с компонента, которые были внедрены, это предоставляется благодаря Andoid приложению.

Список ниже демонстрирует основные характеристики Android:

− отсутствуют затраты на лицензию, популяризацию и разработку, а еще иных элементов сертификаии завершенных программных товаров;

– допуск к Wi−Fi аппарату;

– в сетях GSM, EDGE и 3G, специализированных с целью телефонии и передачи сведений, можно названивать либо осуществлять звонки и SMS, посылать и принимать сведения;

– всеобъятное API с целью использования навигационных услуг, такими как GPS;

– подробное наблюдение над мультимедийными механизмами, включая воспроизведение или запись информации с камеры и микрофона;

– API для работы с сенсорными устройствами, такими как акселерометр и компас;

– библиотеки для работы с Bluetooth с возможностью передачи данных по протоколу p2p;

– передача сообщений МПК;

– хранение общих данных;

– фоновые приложения и процессы;

– виджеты для рабочего стола, живые папки и живые обои;

– возможность интеграции результатов поиска приложения в системный поиск;

– встроенный браузер на основе WebKit с открытым исходным кодом и поддержкой HTML5;

– полная поддержка приложений, использующих функционал работы с картами в своем пользовательском интерфейсе;

– оптимизированная для мобильных устройств графическая система с аппаратным ускорением, включающая библиотеку для работы с векторной 2D графикой и поддержку 3D графики с использованием OpenGL ES 2.0;

– мультимедийных библиотек для воспроизведения и записи аудио, видео файлов и изображений;

– локализация инструментов для работы с динамическими ресурсами;

– набор программных компонентов для повторного использования компонентов и замены встроенных приложений.

1.1.3 Фреймворк разработчика

Язык программирования приложений для платформы Android − Java. Однако они выполняются не на классической виртуальной машине Java, а на специальной виртуальной машине Dalvik.

Каждое Android-приложение функционирует в отдельном процессе внутри собственной копии машины Dalvik. Вся ответственность за управление памятью и процессами возлагается на Android, который останавливает или убивает процессы, если вам нужно освободить ресурсы.

Dalvik и Android находятся в верхней части ядра Linux, которое занимается низкоуровневым взаимодействием с оборудованием, включая управление драйверами и памятью. При этом набор встроенных API позволяет получить доступ ко всем сервисам, функционалу и аппаратной начинке, что для всех совместимых устройств платформа и среда разработки остаются неизменными независимо от производителя или оператора. Пользовательский интерфейс может меняться, но программы будут работать точно так же на всех совместимых Android устройств.

1.1.4 Программный стек Android

В стек программного обеспечения Android состоит из элементов, показанных на рис. 1.1.3 Их подробное описание приводится ниже. Проще говоря, они могут быть представлены как комбинация ядра Linux и набора библиотек C / C++, доступных в рамках приложения. Последний обеспечивает управление и эксплуатацию рабочей среды и приложений.

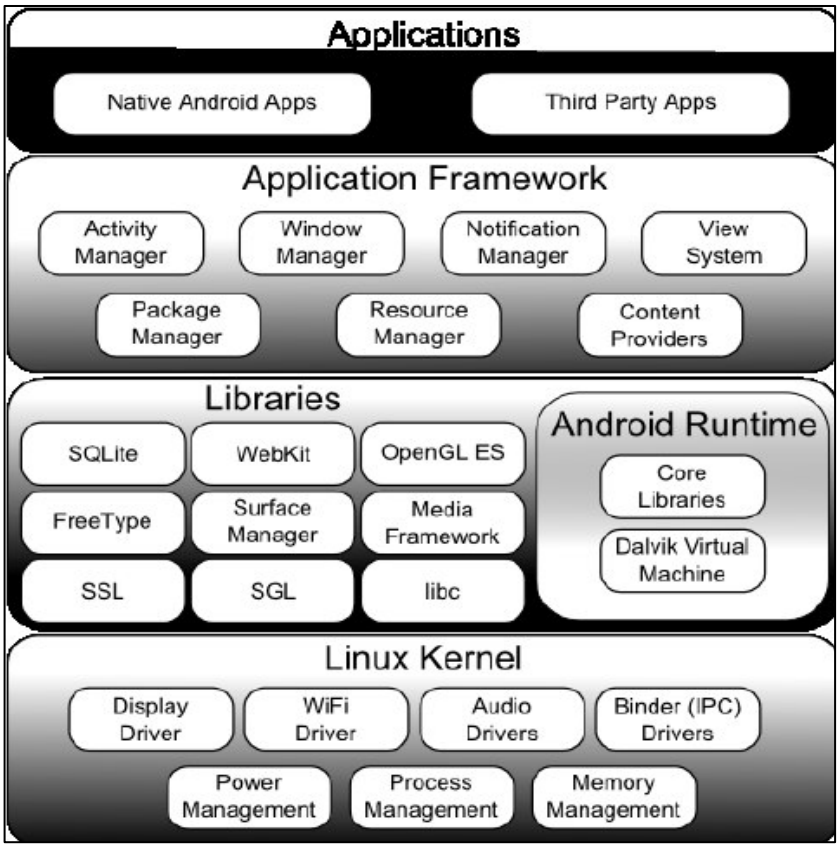


Рисунок 1.1.3 – Стек программного обеспечения Android.

The Linux Kernel. Работа системных служб (драйверы устройств, управление процессами и памятью, питание, безопасность, сетевые службы) обеспечивается ядром Linux версии 2.6. Он также отвечает за уровень абстракции между аппаратным обеспечением и остальной частью программного стека.

Libraries. Android включает в себя различные системные библиотеки C / C ++ (например, SSL и libc), которые работают поверх ядра. Среди них можно выделить:

– библиотека для работы с мультимедиа, обеспечивающая воспроизведение аудио-и видеофайлов;

– диспетчер интерфейс отвечает за управление дисплеем;

– графические библиотеки, такие как SQL и OpenGL, для 2D и 3D-графики;

– библиотека SQLite, которая обеспечивает работу встроенных баз данных;

– SSL и WebKit для встроенного веб-браузера и интернет-безопасности.

Android Working Environment. Специальный телефон на платформе Android сделан не столько мобильной версией ОС Linux, сколько операционной средой Android. Она включает в себя библиотеки ядра и виртуальную машину Dalvik и обеспечивает функционирование программ, а вместе с библиотеками формирует основу фреймворка приложения.

Kernel Libraries. Хотя Android-приложения разрабатываются на Java, Dalvik не является виртуальной Java-машиной. Библиотеки ядра Android обеспечивают основную функциональность библиотек ядра Java, а также уникальную функциональность Android.

The Dalvik Virtual Machine. Dalvik − это виртуальная машина на основе регистра, оптимизированная для одновременного запуска нескольких приложений на устройстве. Он основан на ядре Linux, которое обеспечивает рабочие процессы и низкоуровневое управление памятью.

The Application Framework. Фреймворк включает в себя набор классов, которые используются для разработки приложений. Он также предоставляет общие абстрактные классы для доступа к оборудованию и обеспечивает управление пользовательским интерфейсом и ресурсами приложений.

Application Layer. Все программы, как встроенные, так и сторонние, разрабатываются на уровне приложений с использованием одних и тех же библиотек API. Уровень приложения функционирует в рабочей среде Android, используя классы и службы, открытые для доступа на этом уровне.

1.1.5 Виртуальная машина Dalvik

Dalvik − это снятый с производства виртуальная машина (VM) в операционной системе Android Google, которая выполняет приложения,

написанные для Android. (Формат байт-кода Dalvik по-прежнему используется в качестве формата распространения, но больше не во время выполнения в новых версиях Android.) Dalvik был неотъемлемой частью стека программного обеспечения Android в (теперь неподдерживаемых) версиях Android 4.4 "KitKat" и ранее, которые обычно использовались на мобильных устройствах, таких как мобильные телефоны и планшетные компьютеры, а также на некоторых устройствах, таких как смарт-телевизоры и носимые устройства. Dalvik является открытым исходным кодом, изначально написанная Дэн Борнштейн, который назвал его в честь рыбацкой деревушки объектов: церковь в Дальвик, Исландия.

Программы для Android обычно написаны на Java и скомпилированы в байт-код для виртуальной машины Java, который затем переводится в байт-код Dalvik и хранится внутри .dex (Dalvik) и .odex (Optimized Dalvik EXecutable); связанные термины odex и de-odex связаны с соответствующими преобразованиями байт-кода. Компактный формат исполняемого файла Dalvik предназначен для систем, которые ограничены с точки зрения памяти и скорости процессора.

Преемником Dalvik является Android Runtime (ART), который использует тот же байт-код файлы dex (но odex files), с последовательностью, направленной на повышение производительности, прозрачной для конечных пользователей. Новая среда выполнения была впервые включена в Android 4.4 " KitKat "в качестве предварительного просмотра технологии и полностью заменила Dalvik в более поздних версиях; Android 5.0" Lollipop " - первая версия, в которой ART является единственной включенной средой выполнения.

1.2. Из чего состоят приложения для Android

Приложения в Android состоят из слабо связанных компонентов, которые собираются вместе с помощью манифеста программы. Манифест-это файл, описывающий все компоненты приложения и способы их взаимодействия, а также метаданные, включая требования к конфигурации платформы и оборудования.

Компоненты, перечисленные ниже, являются кирпичи, которые составляют приложения.

Activities. Уровень презентации. Каждый экран приложения является наследником класса Activity. Действия представления используются для формирования графического пользовательского интерфейса, отображающего информацию и взаимодействующего с пользователем. В плане разработки для настольных платформ активность - эквивалент формы (формы).

Services. Невидимые движки вашего приложения. Компоненты службы работают в фоновом режиме, запуская уведомления, обновляя источники данных и видимые действия. Используется для регулярных операций, которые

должны продолжаться, даже если активность вашего приложения не находится на переднем плане.

Data sources. Информационные магазины. Эти компоненты необходимы для управления базами данных в одном приложении и предоставления им доступа извне. Источники данных используются для обмена информацией между различными программами. Это означает, что вы можете настроить собственные объекты ContentProvider, открыть их для других приложений и использовать источники других людей для работы с данными, открытыми для вас внешними программами. Устройства под управлением Android содержат несколько стандартных источников, обеспечивающих доступ к полезным базам данных, включая хранение мультимедийных файлов и контактной информации.

Intentions. Система передачи сообщений между приложениями. С помощью Intention можно транслировать сообщения на системном уровне или для определенных действий или служб. Это диктует необходимость выполнения указанных действий. После этого Android сам определит компоненты, которые необходимо обработать входящий запрос.

Broadcast receivers. Компоненты, которые принимают широковещательные намерения. При создании и регистрации объекта BroadcastReceiver приложение сможет отслеживать трансляцию намерений, соответствующих указанным критериям. Широковещательные приемники автоматически запускают программу, чтобы она могла реагировать на полученное намерение. Благодаря этому данный механизм идеально подходит для создания приложений, использующих модель событий.

Widgets. Визуальные программные компоненты, которые можно добавить на главный экран. Этот специальный тип широковещательного приемника позволяет создавать динамические интерактивные компоненты, которые пользователи могут вставлять в свои домашние экраны. В главе 10 вы узнаете, как создавать собственные виджеты.

Notifications. Система уведомлений пользователей. Позволяет сигнализировать о чем-то, не обращая внимания на себя или не прерывая работу текущей активности. Механизм уведомления лучше всего подходит для служб и широковещательных приемников, когда необходимо привлечь внимание пользователя. Например, если вы получаете текстовое сообщение или входящий вызов, устройство уведомляет вас, мигая светодиодами, воспроизводя звуки, отображая значки или отображая сообщение.

1.2.1 AndroidManifest.XML

Каждое приложение для Android имеет AndroidManifest.XML-файл. AndroidManifest.xml-файл предоставляет информацию, необходимую устройству для запуска приложения. AndroidManifest.xml-файл в этом списке хранит некоторые параметры, которые вы выбираете при создании нового проекта Android. Например, список содержит имя пакета, минимально

необходимый SDK (атрибут android:minSdkVersion) и целевой SDK (атрибут android:targetSdkVersion).

Файл манифеста Android похож на подпись приложения Android для операционной системы Android. Папка манифесты включает в себя AndroidManifest.xml-файл, этот файл важен для приложения, поскольку он содержит все определения, такие как имя пакета, действия, услуги, минимальный уровень API Android, связанные библиотеки и разрешения, которые позволяют ОС Android запускать и запускать приложение.

Файл манифеста Android необходим в каждом приложении Android и должен быть назван AndroidManifest.xml точно. AndroidManifest использует XML для определения всего, что живет внутри него, что позволяет нам достаточно легко взглянуть на то, что внутри него. Следующий код

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3 package="com.brainbell.myfirstapp"
4 <application
5 android:allowBackup="true"
6 android:icon="@mipmap/ic_launcher"
7 android:label="@string/app_name"
8 android:roundIcon="@mipmap/ic_launcher_round"
9 android:supportsRtl="true"
10 android:theme="@style/AppTheme">
11 <activity android:name=".MainActivity">
12 <intent-filter>
13 <action android:name="android.intent.action.MAIN" />
14 <category android:name="android.intent.category.LAUNCHER" />
15 </intent-filter>
16 </activity>
17 </application>
18 </manifest>
```

Строка 2 и 3 < manifest > тег:

− родительский тег контейнера, содержащий ссылку на схему xmlns: android и имя пакета приложения.

Строка 4 < application> tag:

− файлы манифеста Android обычно включают один тег <manifest> с одним тегом <application>. Тег <application> содержит несколько параметров:

Строка 5 android: allowBackup = " true":

− значение этого атрибута по умолчанию-true. Этот атрибут определяет, могут ли пользователи создавать резервные копии и восстанавливать данные приложения.

Строка 6 android: icon= " @mipmap / ic_launcher":

− по умолчанию значок приложения хранятся в /РЭС/MIP-карты каталога (их может быть несколько версий для разных плотностью пикселей, таких как, /РЭС/MIP-карт-anydpi-версии 26, /РЭС/MIP-карт-ипчр, /РЭС/MIP-

карт-резкие /РЭС/MIP-карт-лоллипоп, /РЭС/MIP-карты-палитры, /РЭС/MIP-карт-сода).

Строка 7 android: label= " @string / app_name":

− удобное имя приложения. Должен быть установлен в качестве ссылки на ресурс string.

Строка 8 android: roundIcon= " @mipmap/ic_launcher_round":

− значок приложения по умолчанию в круглой форме. На некоторых устройствах Android на главном экране запуска могут отображаться круглые значки.

Строка 9 android:supportsRtl= "true":

− значение этого атрибута по умолчанию-false. Установите true, если ваше приложение на арабском, иврите, урду или любом другом языке, написанном справа налево.

Строка 10 android: theme= " @style / AppTheme":

− ссылка на ресурс стиля, определяющий тему по умолчанию для приложения.

Строка 11 тег <activity> :

− дочерний тег <application> имеет собственный дочерний тег, называемый тегом <activity>, который определяет имя MainActivity.java-файл, а также метка. Это приложение имеет одно действие MainActivity.

Строка 12 < intent-filter> tag:

− Тег <activity> также имеет дочерний тег, называемый тегом <intent-filter>, фильтр intent объявляет возможности действия, т. е. то, что может сделать действие.

Строка 13 <acton> tag:

− Тег < intent-filter> имеет дочерний тег <action>, который ссылается на основное действие. Тег <intent-filter> должен содержать один или несколько тегов <action>. андроид.намерение.действие.MAIN определяет основную точку входа для приложения.

Строка 14 <category>:

− Тег <category>, который ссылается на константу категории действия намерения запуска. Это намерение служит для запуска приложения Android, запустив подкласс активности верхнего уровня MainActivity на главный экран дисплея.

Приведенный ниже фрагмент кода показывает общую структуру файла манифеста и каждый элемент, который он может содержать.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest>
3 <uses-permission />
4 <permission />
5 <permission-tree />
6 <permission-group />
7 <instrumentation />
8 <uses-sdk />
```

```
9 <uses-configuration />
10 <uses-feature />
11 <supports-screens />
12 <compatible-screens />
13 <supports-gl-texture />
14 <application>
15 <activity>
16 <intent-filter>
17 <action />
18 <category />
19 <data />
20 </intent-filter>
21 <meta-data />
22 </activity>
23 <activity-alias>
24 <intent-filter> . . . </intent-filter>
25 <meta-data />
26 </activity-alias>
27 <service>
28 <intent-filter> . . . </intent-filter>
29 <meta-data/>
30 </service>
31 <receiver>
32 <intent-filter> . . . </intent-filter>
33 <meta-data />
34 </receiver
35 <provider>
36 <grant-uri-permission />
37 <meta-data />
38 <path-permission />
39 </provider>
40 <uses-library />
41 </application>
42 </manifest>
```

Строка 3. Разрешение-это способ для Android потребовать от разработчика приложения уведомить Пользователя о том, что приложение будет делать, что может вызвать проблемы с пользователем.

Строка 4. Это разрешение уровня подписи для защиты вашего собственного приложения, например, при создании приложения, к которому другие приложения могут получить доступ, ограничение доступа является обязательным.

Строка 5. Объявляет базовое имя дерева разрешений, которое можно добавить программным способом с помощью метода addPermission () класса android.content.pm.PackageManager

Строка 6. Разрешения (<разрешение>) собираются в группы разрешений.

аппаратура

Строка 7. Позволяет настроить функцию модульного тестирования для мониторинга приложения.

Строка 8. Разрешить разработчикам указывать, какие версии платформы Android поддерживает приложение. Этот тег имеет три важных атрибута: minSdkVersion, targetSdkVersion и maxSdkVersion, которые определяют самый низкий, оптимальный и самый высокий уровень API соответственно.

Строка 9. Разрешить разработчикам указывать, какие аппаратные и программные методы ввода поддерживает приложение. Несколько тегов конфигурации можно использовать для перечисления различных комбинаций системных функций, которые может поддерживать приложение.

Строка 10. Разрешить разработчикам ограничивать приложение устройствами, имеющими определенные аппаратные или программные функции.

Строка 11. Размеры экрана и плотность пикселей сильно различаются в широком диапазоне устройств Android. Этот тег позволяет разработчикам указывать, какие" типы Android " (малый или большой и HDPI или extra-high-HDPI) экранов поддерживает приложение.

Строка 12. Элемент совместимых экранов не используется Android или вашим приложением напрямую. Он используется только в Google Play Store, чтобы определить, какие устройства — это приложение совместимо С.

Строка 13. Укажите, использует ли приложение графические библиотеки, и вы хотите разрешить только устройства, поддерживающие определенный формат сжатия GL.

Строка 14. Этот тег содержит параметры для всего приложения, включая метку и значок приложения, а также сведения о компонентах приложения, таких как действия и другие компоненты приложения, включая конфигурацию служб, фильтры намерений и поставщиков контента.

Строка 15. Объявляет действие, реализующее часть визуального пользовательского интерфейса приложения.

Срока 16. Фильтры намерений используются для объявления исправных намерений в Android. Эта информация затем используется Android для поиска правильного компонента обслуживания для неявных намерений.

Строка 17. Тег <intent-filter> должен содержать один или несколько тегов <action>.

Строка 18. Разрешить разработчикам указывать дополнительную информацию, описывающую поддерживаемые категории намерения.

Линия 19. Разрешить разработчикам описывать URL и / или тип MIME, на который они хотят ответить.

Линия 21. Этот тег может быть дочерним для тегов <activity>, <activity-alias>, <receiver> или <service>. Поместите больше данных для использования

с библиотеками или моделями распространения плагинов для совместного использования некоторой конфигурации.

Линия 23. Псевдонимы предоставляют альтернативный набор фильтров намерений или другие параметры компонента для уже определенного действия.

Строка 27. Служба-это сущность, которая работает без пользовательского интерфейса, имеет методы обратного вызова жизненного цикла, аналогичные методам Activity.

Строка 31. Приемник вещания прислушивается к входящим намерениям.

Строка 35. Поставщик контента позволяет приложению обмениваться данными (например, базой данных SQLite) с другими приложениями.

Строка 36. Указывает, для каких подмножеств данных может быть предоставлено разрешение родительского поставщика контента. Определяет URI-адрес компонента ContentProvider, к которому может быть предоставлен одноразовый доступ для сторонних приложений.

Строка 38. Разрешения пути позволяют поставщику контента назначать отдельные разрешения для различных путей в его URI, позволяя поставщику контента разрешать доступ к различным разделам данных, которые он предоставляет.

Строка 40. Разрешить разработчикам ссылаться на дополнительные внешние библиотеки.

1.2.2 Создание простых значений

Простые значения-строки, цвета, размеры и массивы (строка и целое число) поддерживаются, эти данные хранятся в формате XML внутри каталога res / values.

С помощью тегов указываются типы хранимых значений.

```
 ?xml version=" 1.0 "encoding=" utf-8"?>
<ресурсы>
<string name= "app_name">список дел< / string>
<color name= "app_background" > #FF0000FF< / color>
<dimen name= "default_border">5px< / dimen>
<array name= "string_array">
< item>пункт 1< / item>
< item>пункт 2< / item>
< item>пункт 3< / item>
</матрица>
<array name= "integer_array">
 3 < / item>
< item > 2< / item>
< item > 1< / item>
</матрица>
</ресурсы>
```

Данный пример содержит все доступные типы простых значений. Каждый тип ресурса хранится в отдельном файле, например, файл res / values / strings.xml включает только строковые константы.

1.2.3 Ресурсы

Класс для доступа к ресурсам приложения. Это находится поверх менеджера активов приложения (доступно через getAssets ()) и предоставляет API высокого уровня для получения типизированных данных из активов.

Система ресурсов Android отслеживает все некодовые активы, связанные с приложением. Этот класс можно использовать для доступа к ресурсам приложения. Как правило, экземпляр ресурсов, связанный с вашим приложением, можно получить с помощью getResources ().

Инструменты Android SDK компилируют ресурсы вашего приложения в двоичный файл приложения во время сборки. Чтобы использовать ресурс, необходимо правильно установить его в исходное дерево (в каталоге res/ проекта) и создать приложение. В процессе сборки средства SDK генерируют символы для каждого ресурса, которые можно использовать в коде приложения для доступа к ресурсам.

Использование ресурсов приложения позволяет легко обновлять различные характеристики приложения без изменения кода и—путем предоставления наборов альтернативных ресурсов-позволяет оптимизировать приложение для различных конфигураций устройств (например, для разных языков и размеров экрана). Это важный аспект разработки приложений для Android, которые совместимы на различных типах устройств.

Дополнительные сведения об использовании ресурсов см. В документации по ресурсам приложения.

1.2.4 Пользовательские интерфейсы для Android

Пользовательский интерфейс приложения для Android построен с использованием иерархии макетов (объектов ViewGroup) и виджетов (объектов View). Макеты − это контейнеры, которые управляют расположением дочерних представлений на экране. Виджеты − это компоненты пользовательского интерфейса, такие как кнопки и текстовые поля.

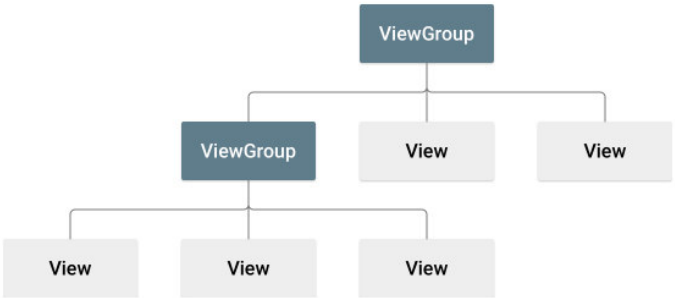


Рисунок 1.2.1 – Иерархия макетов ViewGroup

Android предоставляет словарь XML для классов ViewGroup и View, поэтому большая часть пользовательского интерфейса определяется в XML-файлах.

Чтобы приступить к настройке рабочего пространства следующим образом:

− в окне проекта Android Studio откройте app> res> layout> activity_main.XML;

− чтобы освободить место для редактора макетов, надо скрыть окно проекта View> Tool Windows> Project (или щелкнуть проект в левой части Android Studio);

− если в редакторе отображается источник XML, переход на вкладку Design в нижней части окна;

− щелкнуть Select Design Surface и выбрать Blueprint;

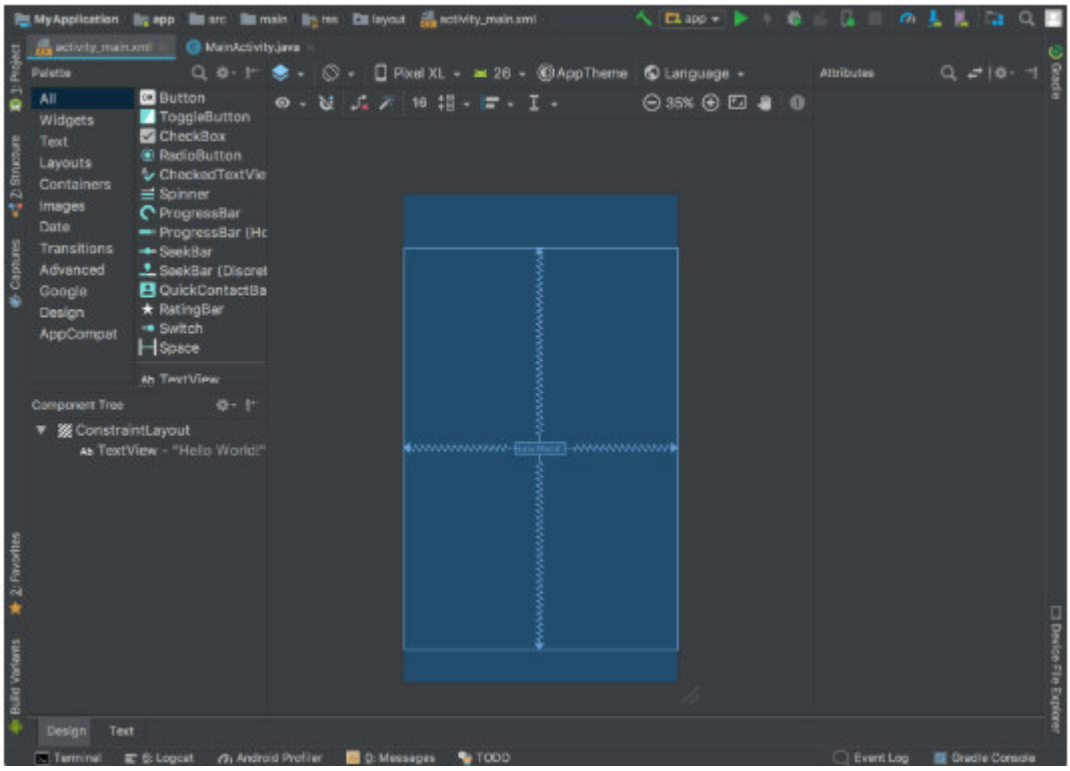
− нажмать кнопку Show на панели инструментов редактора макетов и убедитесь, что установлен флажок Show;

− убедитится, что Autoconnect выключено. Всплывающая подсказка на панели инструментов должна читать Turn on Autoconnect (потому что теперь оно выключено);

− нажать Default Margins на панели инструментов и выберите 16 (вы все еще можете настроить поля для каждого представления позже).

− щелкнуть Device for Preview на панели инструментов и выбрать 5.5, 1440 × 2560, 560dpi (Pixel XL).

Теперь редактор должен выглядеть так, как показано на рисунке 1.2.2



Рисуонк 1.2.2 – Редактор макета Activity_main.xml.

В окне дерева компонентов в нижней левой части отображается иерархия представлений макета. В этом случае корневое представление является ConstraintLayout, содержащим только один объект TextView.

ConstraintLayout − это макет, который определяет позицию для каждого представления на основе ограничений на родственные представления и родительский макет. Таким образом можно создавать как простые, так и сложные макеты с иерархией плоского вида. То есть, это позволяет избежать необходимости во вложенных макетах (макет внутри макета, как показано на рисунке 2), что может увеличить время, необходимое для рисования пользовательского интерфейса.

Например, можно объявить следующий макет (на рисунке 1.2.3):

− Вид а отображается 16dp в верхней части родительского макета;
− Вид а отображается 16dp слева от родительского макета;
− Вид В появляется 16dp справа от вида А;
− Вид В выровнен по верхней части вида А;
− В следующих разделах вы создадите макет, подобный этому;

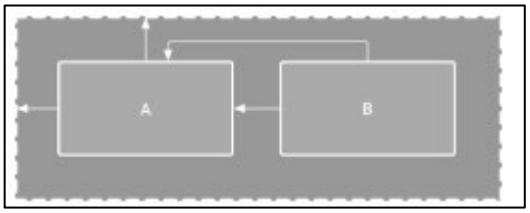


Рисунок 1.2.3 − Иллюстрация двух видов, расположенных внутри ConstraintLayout.

2 Определине средств для разработки программы

2.1 Средства разработки кода приложения

Для написания кода в этом проекте будет использоваться среда для разработки Android Studio, а Java будет служить языком. Для работы в Android Studio в используемой системе должен быть установлен Java SE Development Kit (JDK). Рекомендуемая версия JDK-не менее 7. Вы можете установить его, перейдя по ссылке. http:// www.oracle.com/ technetwork/ java/ javase/ downloads/ jdk7-downloads1880260.html

Java SE Development Kit-это бесплатный набор инструментов для разработки, который распространяется бесплатно. Итак, пройдя по приведенной выше ссылке, вам нужно выбрать версию, соответствующую вашей ОС, Windows, Linux или Mac OS X. После загрузки исполняемого файла, вы должны запустить его и следовать инструкциям по установке этого инструмента. Создание программного кода, предназначенного для доступа других разработчиков, создает ряд проблем. Одна из проблем заключается в том, как легко обучить разработчиков тому, как использать интерфейс этого программного кода. Это обычно делается путем предоставления программного обеспечения.

SDK − это программный инструментарий, который может отличаться по форме и, следовательно, определени также варьироваться. Это может быть все от простого интерфейса прикладного программирования (API) к обширной библиотеке полезной документации, примеров кода, учебных пособий и программного обеспечения. Короче говоря, SDK − это ряд частей, которые помогут разработчику понять, как использовать часть программного обеспечения. Это комплект с более или менее продвинутыми утилитами и документацией. Как пример, для того, чтобы разработать код на языке программирования Java, рекомендуется для использования Java SDK. Java SDK состоит из документации Java API, утилит для компиляции и отладка созданного программного обеспечения, а также файлов времени выполнения, чтобы разработчик мог выполнять программы, которые он создал. Подход с этими частями, API документации, утилиты и файлы выполнения довольно распространены при описании SDK

Настройка Android Studio довольно проста и проще, чем когда-либо, благодаря почти всему, что поставляется в одном установщике. Загрузите его здесь, и вы получите не только Android Studio, но и Android SDK, SDK manager и многое другое. Единственное, что еще понадобится, это Java Development Kit. Важно помнить, что Android Studio - это только ваше окно в Java! Примечание: Android Studio и SDK довольно большие, поэтому надо убедиться, что есть свободное место на диске C:\, прежде чем начать.

Чтобы установить Android Studio достаточо следовать простым инструкциям во время установки, и он также должен настроить вас с платформой Android, что вы сможете разработать С, а также надо обязательно

установить флажок, чтобы сообщить установщику, что нужно Android SDK, а также обратить внимание, где Android Studio и SDK устанавливаются.

Выбрать каталог для SDK, в котором нет пробелов и обратить внимание, что папка AppData, выбранная Android Studio, является скрытой папкой в Windows. Это означает, что нужно будет выбрать "Показать скрытые папки", если имеется желани перейти к нему с помощью проводника. Это значения по умолчанию, выбранные для моей установки на рисунке 2.1.1

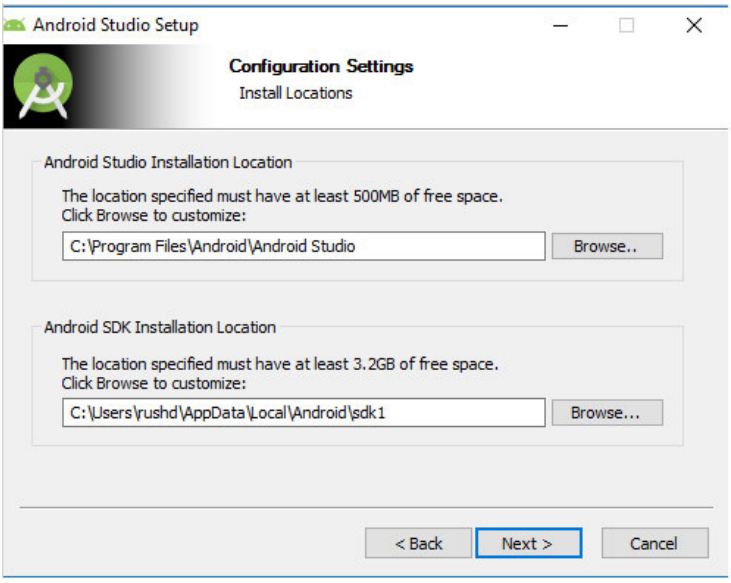


Рисунок 2.1.1 − экран установки Android Studio.

Как только Android Studio будет запущен можно будет создать новый проект. Это можно сделать, запустив Android Studio, а затем выбрав новый проект, или вы можете выбрать File> Creat> New Project в любое время из самой среды IDE.

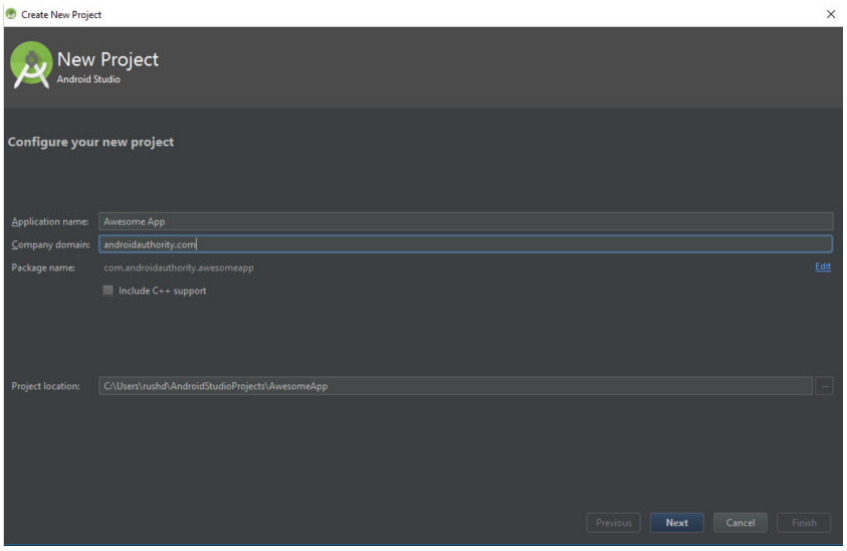


Рисунок 2.1.2 − Окно создания проекта Android Studio

Затем у вас будет возможность выбрать из нескольких различных видов activity. Activity в действительности являются ‘экранами’ в приложении. В некоторых случаях это будет все приложение или в других ваше приложение может переходить с одного экрана на другой. Вы можете начать новый проект без activity (в этом случае вы выберете "Add no activity"), но вы почти всегда захотите его, поэтому проще позволить Android Studio настроить вас с чем-то, напоминающим пустой шаблон приложения.

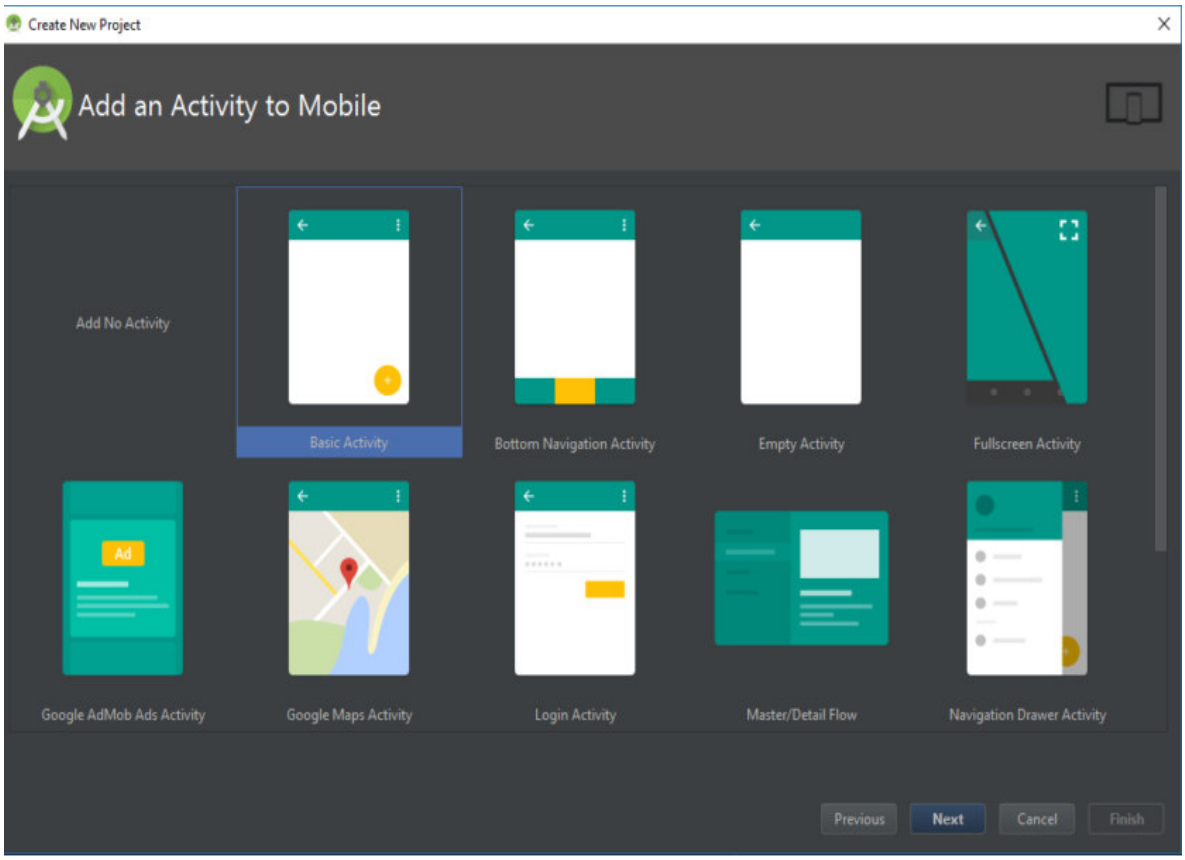


Рис. 2.1.3 − Главное окно выбора вида Activity.

Часто обычно выбирается "Basic Activity", которая является внешним видом по умолчанию для нового приложения для Android. Это будет включать в себя меню в правом верхнем углу, а также кнопку FAB – плавающая кнопка действия – это выбор дизайна, который Google пытается поощрять. Empty Activity − это то же самое, но без добавления хрома.

Выберите вариант, который лучше всего подходит для приложения, которое вы имеете в виду, чтобы построить, и это повлияет на вид файлов, которые вам представлены при первом запуске. На этом этапе вы также сможете выбрать имя своего приложения, минимальный Android SDK, который вы хотите поддерживать, и имя пакета. Имя пакета − это окончательное имя файла, которое приложение будет иметь при загрузке в Play Store-сочетание имени приложения вместе с именем разработчика.

Если в окне проекта был выбран значок "создать пользовательский лаунчер", то после нажатия кнопки" Далее " вы увидите окно создания и редактирования значка приложения, которое также показано на рисунке 2.6. Этот редактор удовлетворит потребности практически всех разработчиков, ведь он позволяет быстро и качественно создавать иконки приложений для разной плотности экрана и сразу видеть результат.

Основным "code" будет файл Java, который имеет то же имя, что и ваша activity. По умолчанию это MainActivity.Java, но, возможно, изменить это при первой настройке проекта. Здесь вводится Java-скрипт и определение поведения своих приложений.

Однако фактический макет вашего приложения полностью обрабатывается в другом фрагменте кода. Этот код в файл activity_main.XML. XML-это язык разметки, который определяет структуру документа – так же, как HTML, который используется для создания веб-сайтов. Это не совсем "программирование", но это своего рода код. На рисунке 2.1.4 показано окно activity и xml−файлов, то есть окно работы над кодом.

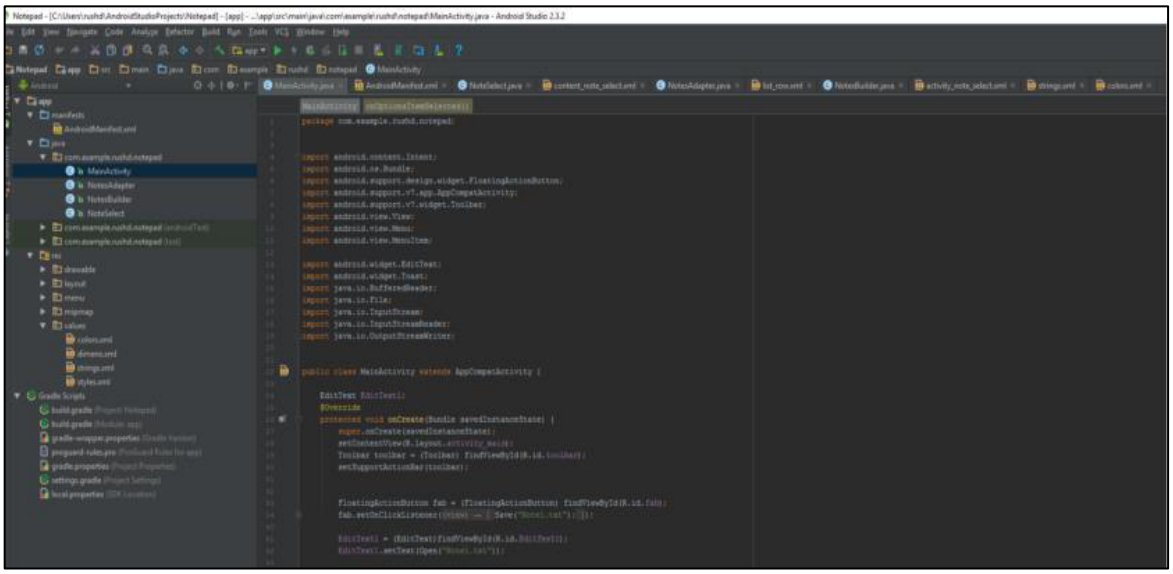


Рисунок 2.1.4 – Окно activity и xml−файлов.

Итак, если надо создать new button, для этого надо отредактировать activity_main.xml и если описать, что происходит, когда кто-то нажимает на эту кнопку, вы, вероятно, поместите это в MainActivity.Ява. Однако, чтобы сделать вещи немного сложнее, вы можете использовать любой XML-файл для определения макета любого Java-скрипта (называемого классом). Это находится в верхней части вашего кода Java, с линией: setContentView(R.layout.activity_main).

Это говорит о том, что Android Studio будет иметь свой макет, установленный activity_main.xml. Это также означает, что теоретически можно использовать один и тот же xml-файл для установки макетов для двух разных классов Java.

Как видно, Android-приложение на самом деле состоит из нескольких файлов, и обязанность Android Studio состоит в том, чтобы держать их все в одном месте для вас. Главное окно справа от экрана позволит вам просматривать отдельные скрипты и файлы, а вкладки вверху позволяют переключаться между тем, что открыто в любой момент времени.

И в некоторых случаях будет несколько XML-файлов, описывающих различные аспекты макета вашей activity. Если выбрать "Basic Activity" вместо "Empry Activity", например, тогда будет activity_main.xml, который установил бы положение FAB и других элементов пользовательского интерфейса и content_main.xml, в котором будет размещаться контент, который можно добавить в середину экрана. В конечном итоге вы можете добавить "views" (такие элементы, как кнопки, текстовые поля и списки), и некоторые из них также могут иметь свои собственные XML-макеты.

Если хотите открыть что-то новое, можно сделать это через файловую иерархию слева. Здесь вы найдете все папки и папки внутри них. Ваши файлы Java размещаются под java, а затем имя пакета вашего приложения. Дважды щелкните по MainActivity.Java (предполагая, что вы используете Java), и он выйдет на первый план в окне справа – рисунок 2.1.5

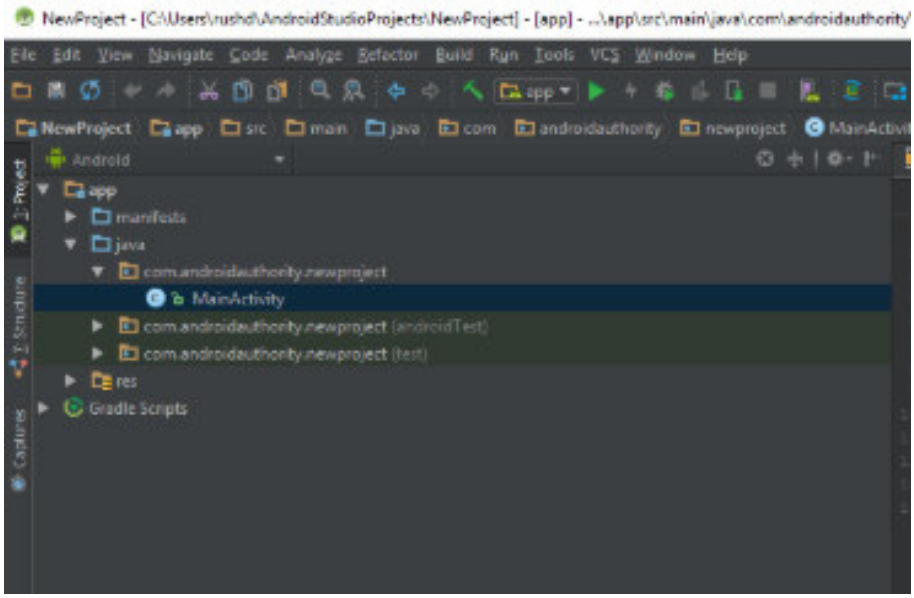


Рисунок 2.1.5 – Файловая йерархия.

При редактировании XML-файлов можно заметить две вкладки внизу. Они позволяют переключаться между представлением "текст" и представлением "дизайн". В текстовом представлении можно вносить изменения в XML-код напрямую, добавляя и редактируя строки. В режиме конструктора вы сможете добавлять, удалять и перетаскивать отдельные элементы по экрану и видеть, как они будут выглядеть. Текстовый вид имеет окно предварительного просмотра, а также для визуализации того, что вы создаете – пока ваш монитор достаточно широк!

Еще одна полезная папка − папка "res". Это сокращение от "resource", и это включает в себя "drawables" (изображения, которые вы разместите в своем приложении), а также "layout", где ваши XML-файлы идут. Все в папке ресурсов должно быть в нижнем регистре, поэтому подчеркивание часто используется для разделения имен файлов на читаемые заголовки в отсутствие верблюжьего регистра.

"Values" также является полезной папкой для поиска. Это содержит больше XML-файлов, которые содержат значения переменных-такие вещи, как имена приложений и значения цвета.

AndroidManifest.xml − еще один очень важный файл, найденный в папке "manifests". Его задача − определить важные факты о вашем приложении, например, какие действия будут включены, имя приложения, как оно будет видно пользователям, разрешения приложения и т. д.

Вы можете создать дополнительные классы Java, XML-файлы или целые действия в любой момент, чтобы добавить больше функциональности в приложение. Просто щелкните правой кнопкой мыши на соответствующем каталоге, а затем выберите "Создать", а затем все, что вы хотите добавить. Вы также можете открыть каталог своего проекта, щелкнув правой кнопкой мыши и выбрав "показать в Проводнике". Это удобно, если вы хотите отредактировать изображение, например. На рисунке 2.1.6 изображен файл AndroidManifest.xml

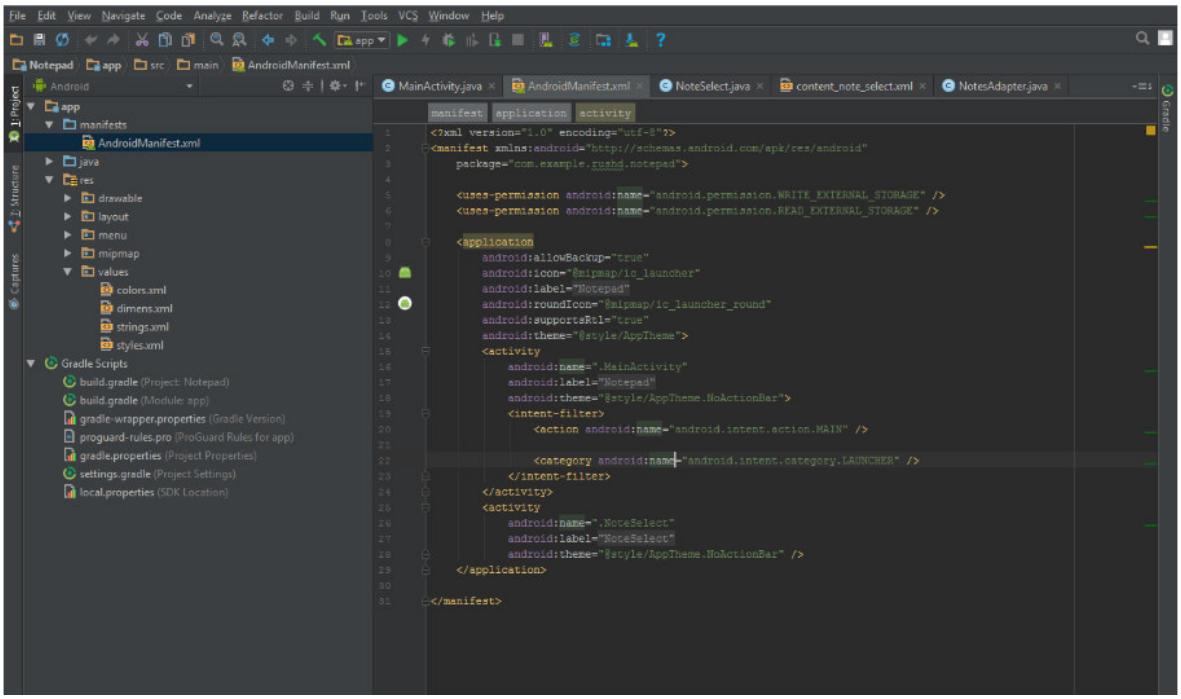


Рисунок 2.1.6 − Файл AndroidManifest.xml.

Android Studio пытается сохранить вещи хорошими и простыми для пользователей, предоставляя все необходимые инструменты и функции в

одном месте. Все становится только сложнее, когда вам нужно взаимодействовать с некоторыми из этих других элементов.

Например, вы можете заметить, что Android Studio иногда упоминает "Gradle". Это "build automation tool", который по существу помогает Android Studio превратить все эти разные файлы в один APK. Вы должны быть в состоянии оставить Gradle делать свое дело большую часть времени, но иногда вам нужно будет прыгать в build.gradle файлы, если вы хотите добавить новую "dependency", позволяющую дополнительные функции для вашего приложения. Иногда, если вещи перестают работать, вы можете выбрать Build> Clean Project, и это по существу подтвердит, где находятся все файлы и каковы их роли. Обычно будет два из этих файлов сборки Gradle, один для всего проекта и один для "модуля" (приложение). На рисунке 2.1.7 изображен build gradle.

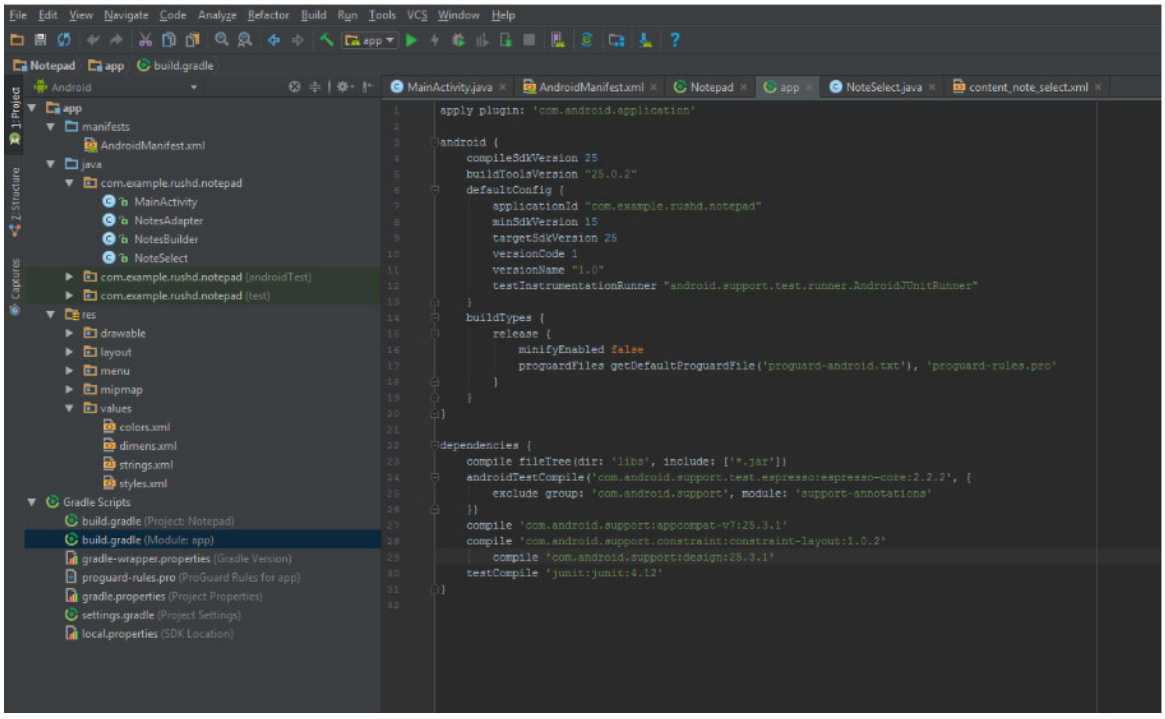


Рисунок 2.1.7. – Build gradle.

После того, как будете готовы протестировать приложение, имеется есть два варианта. Один из них-запустить его на физическом устройстве, а другой-создать виртуальное устройство (эмулятор) для его тестирования.

Запустить его на устройстве просто. Просто для этого надо подключить его через USB, и убедиться, что на устройстве разрешено отладка через USB и установки из неизвестных источников в настройках телефона, а затем надо нажать зеленую кнопку воспроизведения вверху или "Run> Run app".

После появится сообщение о том, что выполняется сборка Gradle (т. е. имеющийся код превращается в полное приложение), а затем он должен ожить на подключенном устройстве. Это быстрее, чем когда-либо прямо сейчас,

благодаря функции мгновенного запуска. На рисунке 2.1.8 изображен запуск на подключенном устройстве

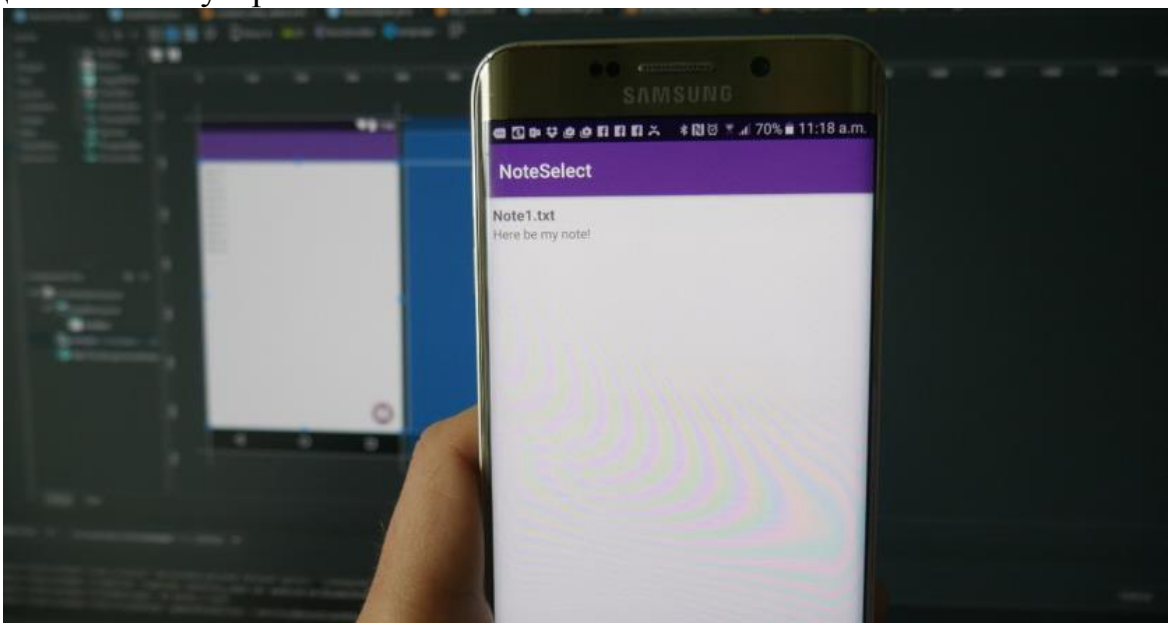


Рисунок 2.1.8 – Запуск на подключенном устройстве.

Пока приложение работает, вы сможете получать живые отчеты через вкладку "logcat" на мониторе Android (изображен ниже на рисунке 2.1.8), находящемся в нижней половине экрана. Если что-то пойдет не так, что приведет к сбою вашего приложения или станет невосприимчивым, появится красный текст, и это даст вам описание проблемы. Вы можете обнаружить, что это просто вопрос забытых разрешений или что-то еще, что легко исправить. Это существенно экономит вам массу времени, а не слепо пытается угадать, что пошло не так. Не забудьте отфильтровать типы сообщений, которые вы хотите увидеть здесь.

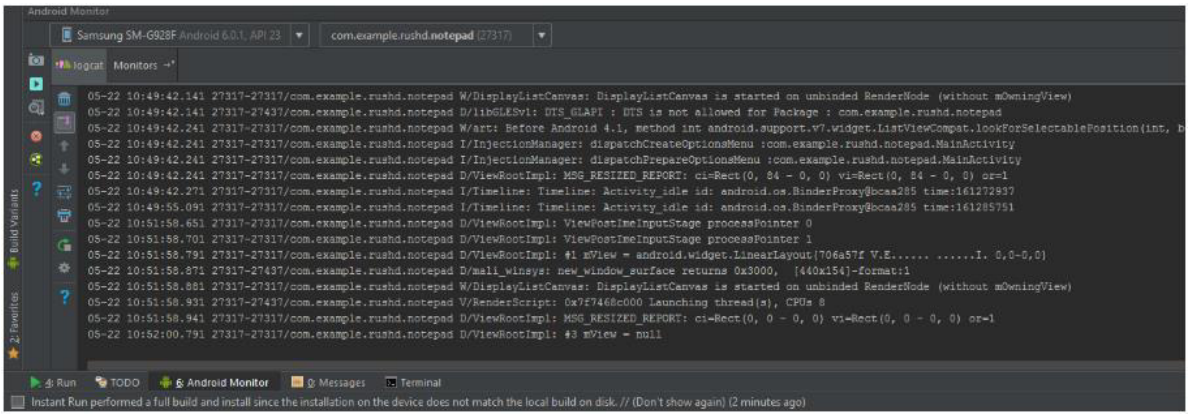


Рисунок 2.1.8 – Отчеты в реальном времени через вкалдку "logcat".

Также можно переключаться на вкладку мониторы и увидеть полезную информацию, такую как загрузка процессора и т. д. Android Device Monitor

(рисунок 2.1.9.) делает этот мониторинг шаг дальше и позволяет контролировать все сразу, в комплекте с удобным интерфейсом.

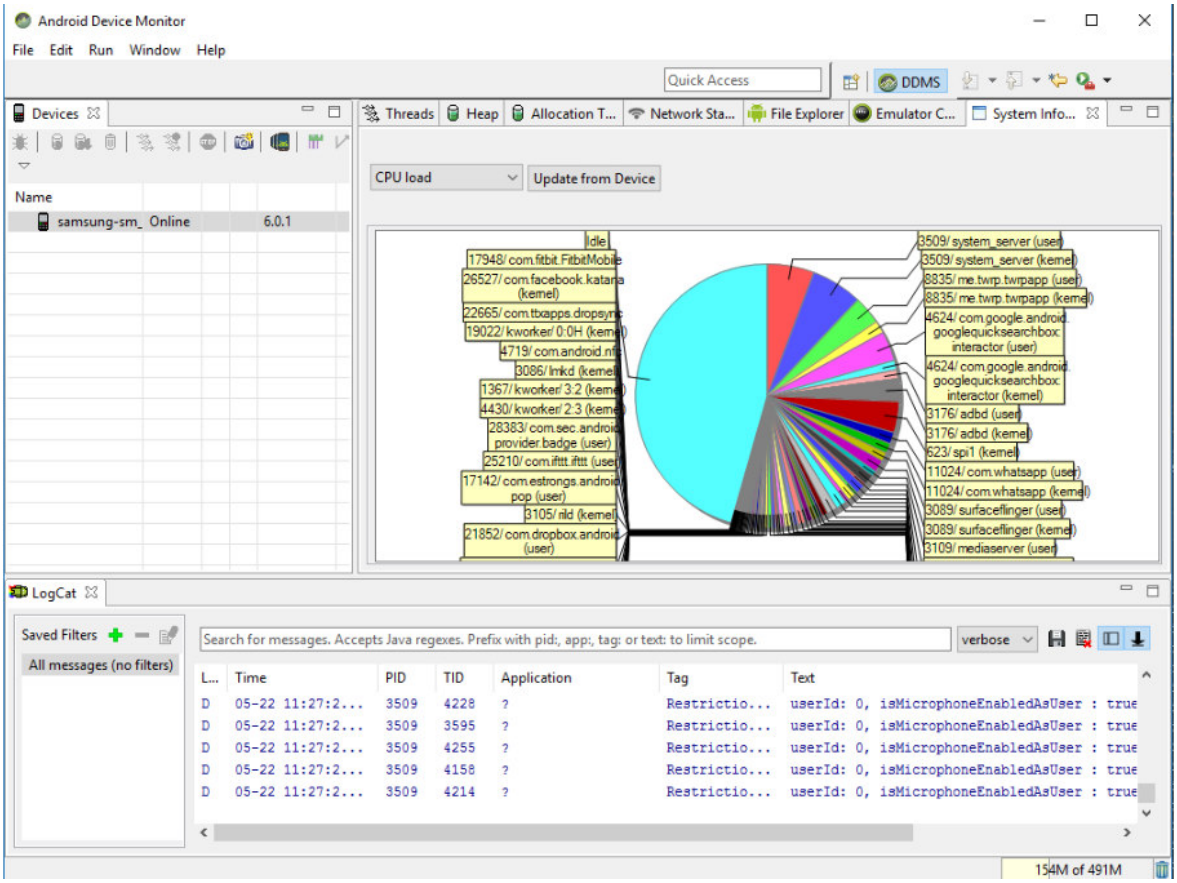


Рисунок 2.1.9 – Android Device Manager(AVD).

Вряд ли когда-нибудь разраб захочет разработать для Android без какого-либо устройства Android в вашем распоряжении. Однако, одна из самых больших проблем для разработчиков Android является фрагментация. Другими словами: недостаточно, чтобы ваше приложение работало на вашем устройстве, оно также должно работать на 10" и 15" устройствах. И он должен работать на устройствах под управлением более старых версий Android или недостаточна.

Это где ‘виртуального устройства Android появилась. Это по существу эмулятор, который вы можете использовать, чтобы имитировать внешний вид и производительность любого другого устройства Android, устанавливая такие вещи, как размер экрана, мощность и версию Android.

Чтобы использовать виртуальное устройство, сначала нужно создать его, загрузив необходимые компоненты и установив спецификации по своему усмотрению. Для этого перейдите в меню сервис > Android > AVD Manager.

Затем вы выберете свое оборудование и выберите платформу Android, которую вы хотите запустить. Если версия Android, которую вы хотите запустить, еще не загружена, то опция будет представлена рядом с ней. На

рисунке 2.1.10. изображен выбор устройства для нашего эмулярота. Он выбирается в вкладке сверху нажимая на кнопку AVD Manager.

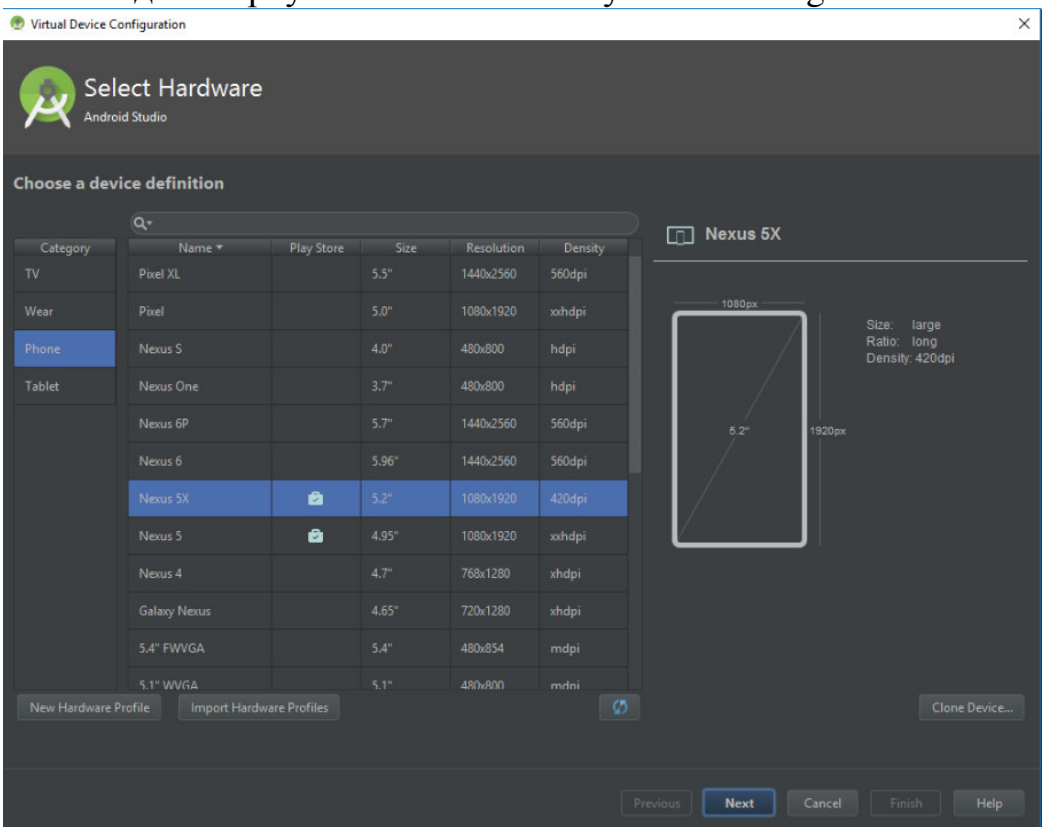


Рисунок 2.1.10 – Выбор устройства AVD.

После настройки некоторых устройств можно выбрать одно из них при запуске приложения и отладке так же, как на физическом устройстве. Стоит обратить внимание, однако, что понадобятся довольно приличные спецификации для запуска виртуального устройства. Для тех, кто проявляет интерес, можно относиться к этому так же, как и к любому другому эмулятору и даже получить доступ к Play Маркету для загрузки ваших приложений. Если у вас есть оборудование, это жизнеспособный способ запуска некоторых приложений на ПК с Windows.

Если настроить таргетинг на конкретную версию Android или создать виртуальное устройство под управлением определенной версии, для этого потребуется загрузить необходимые инструменты платформы и SDK. Это можно сделать с помощью диспетчера SDK, который можно найти, выбрав сервис > диспетчер SDK. Здесь также можно найти дополнительные ресурсы, такие как Google Glass Development Kit или репозиторий Android, который предоставляет вам дополнительные функции для использования в вашем приложении. На рисунке 2.1.11 изображена настройка и выбор Android SDK. Для установки достаточно поставить флажок рядом с тем, что нужно загрузить, а затем нажать "ОК". Android Studio также будет предупреждать время от времени, когда пришло время обновить саму IDE или любой из этих элементов. Удостоверьтесь, чтобы держать вверх-к-дата!

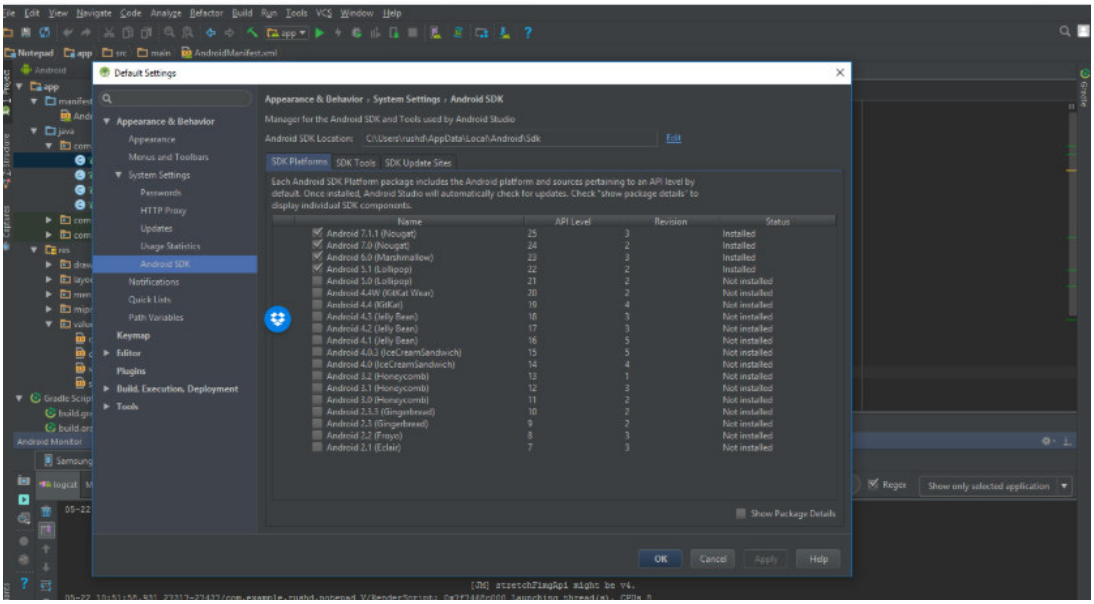


Рисунок 2.1.11 – Окно выбора SDK.

Наконец, как только завершиться тестирование приложения и он будет готов выйти в большой мир, надо выбрать Build > Generate Signed APK. Это даст файл, который нужно будет загрузить в Google Play и который будет содержать все различные файлы, ресурсы и многое другое. На рисунке 2.1.12 изображено окно генерации APK.

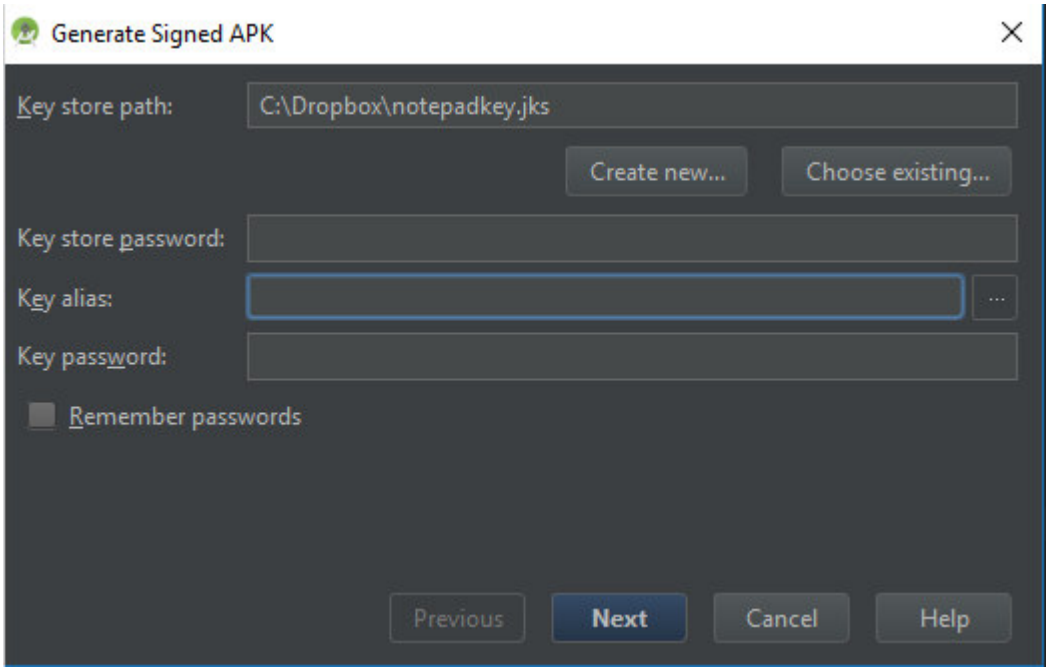


Рисунок 2.1.12 – Окно генерации APK.

После будет предложено создать или ввести хранилище ключей. Это своего рода "сертификат подлинности", который доказывает, что APK,

который загружаете, − это приложение, о котором говориться. Это предотвращает взлом вашей учетной записи Google Play, а затем загрузку вредоносного APK в качестве "обновления" в приложение. Для этого нужно будет сохранить этот файл в безопасности, так как после его потери нет возможности обновить приложение снова. Выберать "release" в качестве типа сборки, если имеется желание сделать это что-то, что можно освободить, а затем нажать кнопку "finish".

3 Экспериментальная часть

3.1 Интерфейс

Основной интерфейс включает в себя:
– изучение материала;
– чат;
– инструменты;
– профиль.

На рисунке 3.1.1 представлен один из разделов мобильного приложения котором находятся материал для изучения той или иной предметной области. В этом окошке можно выбирать материал, который хотите изучить.

Включает в себя:
– концепцию ключей;
– кибератаки;
– конфиденциальность.

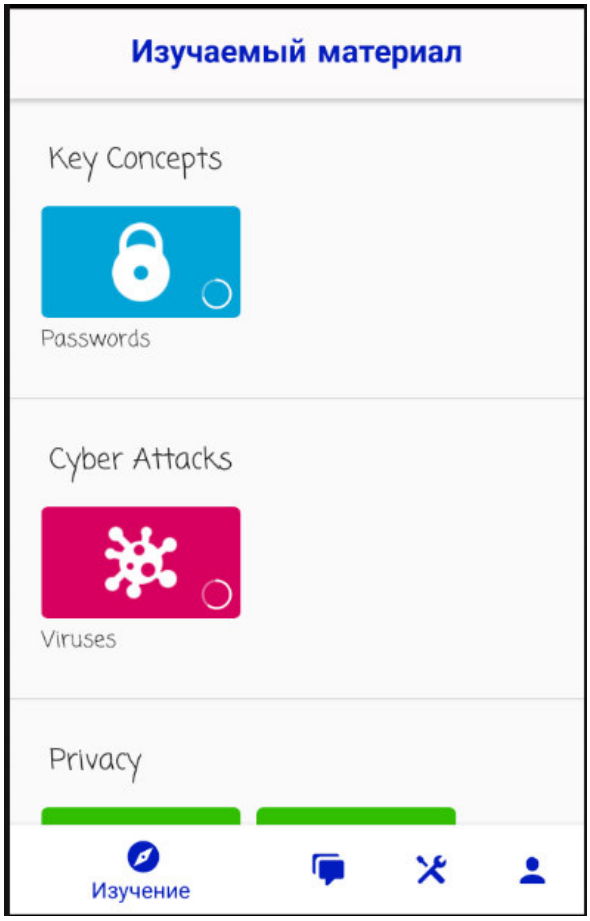


Рисунок 3.1.1. – Окно раздела изучение курсов.

На рисунке 3.1.2 происходит оценка знаний пройденного материала

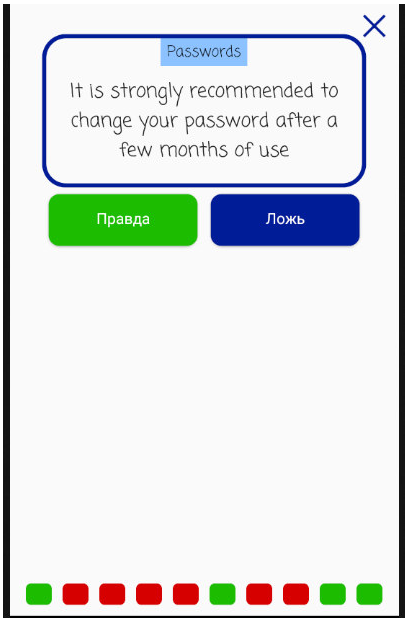
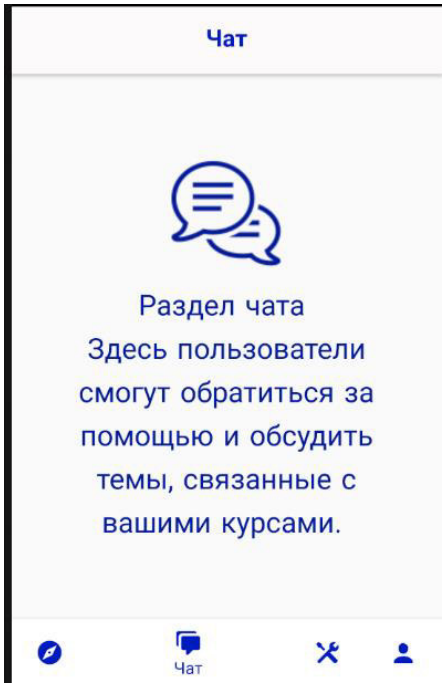


Рисунок 3.1.2 – Проверка знаний с помощью теста.

На рисунке 3.1.3 имеется чат, который позволяет связываться с разработчиком, обращаться за помощью и обсуждать темы.



Рисуонк 3.1.3 – Окно нашего чата.

На рисунке 3.1.4 предоставлен набор инструментов, которые облегчают ту или иную задачу, связанную с киберзащитой.

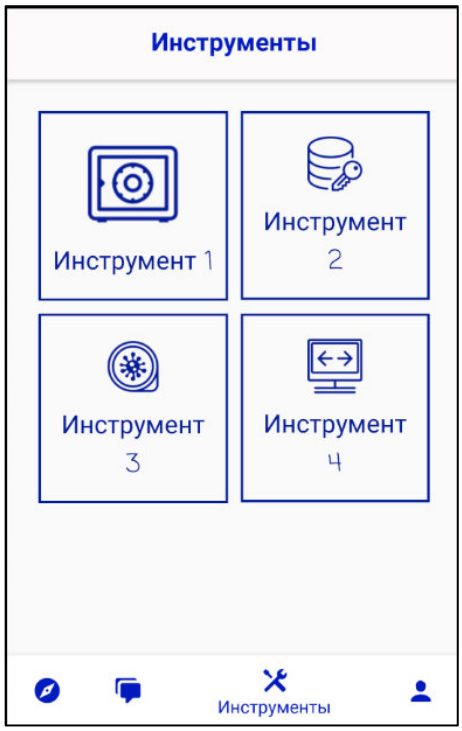


Рисунок 3.1.4 – Инструменты.

На рисунке 3.1.5 показан профиль, в котором отображен прогресс в изученном материале и какое вы имеете «звание».

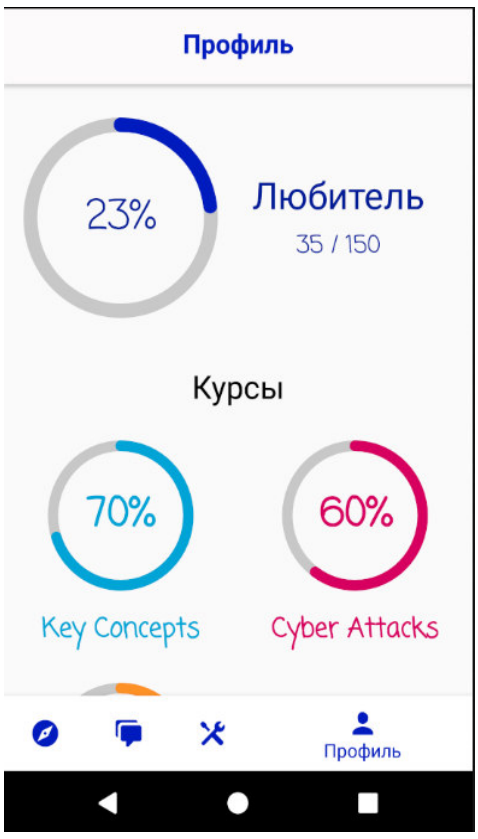


Рисунок 3.1.5. – Профиль пользователя.

3.2 Разработка Activity и классов

Проект состоит из нескольких папок и файлов. Основные из них:

− src является исходным кодом для Java. Вот основной файл для работы. Появятся новые классы;

− gen-файлы, созданные самой Java;

− файлы ресурсов res.

Содержит несколько подкаталогов:

– res / drawable- * dpi - эти пять папок содержат ресурсы для различных расширений экрана. Если зайти в каждую папку, то можно найти там файл ic_launcher.PNG, который является значком вашего приложения. В папке drawable-ldpi ничего нет, так как это папка для старых телефонов, которые вам больше не нужно поддерживать.

– res / layout - эта папка содержит xml-файлы, описывающие внешний вид форм и различных элементов формы. После создания проекта activity_main уже существует.xml и fragment_main.XML;

– res / menu - вот ресурсы для меню;

– в res / значения - здесь у нас есть некоторые строковые ресурсы, ресурсы цветов, тем, стилей и измерений, которые мы можем использовать в нашем проекте. Также есть похожие папки valuesw820dp, values-v11 (для планшетов Android 3.0), values-14 (для Android 4), предназначенные для определенных типов устройств;

− при разработке приложения отдельные компоненты приложения были прорисованы в Adobe Illustrator CC:

– значок приложения;

– маркеры начала и конца обрезанного файла;

– значки для кнопок справки, сброса, сохранения, добавления нового файла, записи файла и т. д.

3.2.1 Создание и реализация Activity

Activity является важным компонентом приложения Android, а способ запуска и связывания действий является фундаментальной частью модели приложения платформы. В отличие от программных парадигм, в которых приложения запускаются с использованием метода main (), система Android инициирует код в экземпляре Activity, вызывая специальные методы обратного вызова, соответствующие конкретным этапам ее жизненного цикла.

В папке /activities было создано небольшое количество Activity, которые состоят из следующих файлов:

− ChatActivity;

− HomeActivity;

− ProfileActivity;

− QuizActivity;

− SectionActivity.

3.2.2 Создание и реализация классов

Java является объектно-ориентированным языком, поэтому такие понятия как "класс" и "объект" играют в нем ключевую роль. Любую программу на Java можно представить, как набор взаимодействующих между собой объектов.

В папке /objects было создано небольшое количество классов, которые состоят из следующих файлов:

− Course;
− Lesson;
− Level;
− Question;
− Section.

3.2.3 Создание базы данных

Приложение считывает сведения о курсах и уроках из базы данных SQLite, сохраненной локально. При первом открытии приложения база данных создается на основе инструкций, приведенных в lessonsdb.sql, который хранится в папке assets.

Чтобы создать базу данных (вы можете найти пример, с которым .sql файл создается в репозитории проекта), можно использовать любое программное обеспечение, которое хотите, я воспользовался DB Browser для SQLite, или даже написать непосредственно SQL-код в активах .sql-файл. С помощью DB Browser for SQLite откройте данные БД в представителе проекта добавьте свои курсы и уроки в соответствии со схемой ниже, а затем экспортируйте ее, перейдя в Файл -> Экспорт -> база данных в файл SQL -> выберите таблицы курса и урока -> OK -> сохраните ее в папке Project assets с именем lessonsdb SQL.

Для того, чтобы приложение работало автоматически, вам нужно будет сохранить ту же структуру таблиц и использовать соглашения. Достаточно всего 2 таблицы, в которых описано структура и построение базы данных, чтоб автоматизировать работу приложения. В рисунке 3.2.1 показана наша таблица с курсами.

Field	Example	Explanation
_id	`2`	Id of the course, should be autoincremented: numbers from 1 to the number of courses
title	`Privacy`	Name of the course, it will appear in the titles

Рисунок 3.2.1 – Таблица Course.

В рисунке 3.2.2 идет пояснение построение таблицы lesson. Подробное разъяснение того, как надо делать базу данных к каждому теоретическому материалу. Она может иметь в себе до 9 секции теоретического материала.

Field	Example	Explanation
_id	`3`	Id of the lesson, should be autoincremented: numbers from 1 to the number of lessons
courseid	`2`	Id of the course this lesson belongs to, it needs to have a corresponding entry in the course table
questions	`GDPR`	Name of the lesson, it will appear in the titles
result	`9`	Number of questions the user got correct in his best attempt of the current lesson
nsections	`4`	Number of sections that the current lesson is composed by
section0 ... section9	`Title of the section<<-->>Text of the section`	Each of the fields contains one section, title and text of the section are separated by this `<<-->>` set of characters. If the lesson is composed by three sections only the fields 0, 1, 2 should be used and the others left empty
questions	`Question 1<->Correct Ans<->Wrong 1<->Wrong 2<->Wrong 3<<-->>Question 2<->T<<-->> ...`	Details of the 10 questions at the end of the lesson, questions are divided by `<<-->>`. Each question can either be: multiple choice (`Text Question<->Text Correct Answer<->Wrong Answer 1<->Wrong Answer 2<->Wrong Answer 3`) or true or false (`Text Question<->T or F`)

Рисунок 3.2.2 – Таблица lesson

3.2.4 Разработка инструментов

Это единственная часть приложения, которую придется разрабатывать самостоятельно, если вы предпочитаете, этот раздел можно легко удалить, однако, мы настоятельно рекомендуем не делать этого, потому что это хороший стимул для пользователей, которые завершили все доступные в настоящее время курсы, чтобы сохранить приложение. Этот репозиторий также предлагает набор общих инструментов, которые можно легко добавить в проект.

Инструменты доступны из таблицы, присутствующей в ToolsActivity, и могут быть разработаны как независимый набор действий или даже просто диалог, который открывается в действии Tools.

Как вы можете видеть на скриншотах шаблона, каждый урок имеет значок, а каждый раздел-изображение. Все они сохраняются в папке приложения drawable со следующими соглашениями:

Значки уроков: с именем z<id урока>.png (например, для урока с идентификатором 3 путь к изображению должен быть app\src\main\res\drawable\z3.формат PNG.) Значок должен быть квадратным белым логотипом с прозрачным фоном, рекомендуемый размер ≈ 100x100 пикселей.

Изображение раздела: имя s<id урока><номер раздела>.jpg (например, для первого раздела урока с id 3 путь к изображению должен быть app\src\main\res\drawable\s30.формат JPG.) Изображение должно быть связано с содержимым раздела, рекомендуемый размер не менее 300x200 пикселей.

4 Экономическая часть

4.1 Расчет трудоемкости разработки

Определение трудоемкости разработки программного продукта начнем с составления перечни основных этапов и видов работ, выполненные в ходе разработки. Трудоемкость программного продукта производиться на основании опытно-статистического метода.

Таблица 4.1.1 − Распределение работ по этапам и видам и оценка их трудоемкости

Этапы разработки ПП	Вид работы	Трудоемкость, чел. час.
Этап 1	Определение цели	10
Этап 2	Выбор средств для разработки ПП	15
Этап 3	Создание блок-схемы для ПП	35
Этап 4	Составление таблиц и связей	10
Этап 5	Разработка графического дизайна ПП	40
Этап 6	Создание функционала	70
Этап 7	Отладка программной части продукта	35
Этап 8	Налаживание работы	35
Итого: трудоемкость выполнения дипломного проекта		250

4.2 Расчет затрат на разработку ПП

В данном этапе включены такие статьи, как затраты:
− на необходимые материальные ресурсы;
− затраты на оплату труда;
− затраты на расход электроэнергии;
− общие затраты на амортизационные отчисления;
− на социальный налог;
− прочие расходы.

Расчет затрат на материальные ресурсы описан в таблице 4.2.1, которая представлена ниже.

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

$$З_М = \sum_{i=1}^{n} Р_i * Ц, \quad (4.1)$$

где P_i - расход i-го вида материального ресурса, натуральные единицы;

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

i - вид материального ресурса;

n - количество видов материальных ресурсов.

Таблица 4.2.1 − Затраты на материальные ресурсы

Наименование материального ресурса	Единица измерения	Количество израсходованного материала	Цена за единицу, тг	Сумма, тг
Бумага А4 для блок схем	Штук	1	1500	1500,
Тетрадь 96 л.	Штук	1	200	200
Ручки	Штук	3	100	300
ИТОГО затраты на материальные ресурсы				2000,00

$$З_{общее} = З_{бумага} + З_{тетрадь} + З_{ручки} = 2000{,}00 \text{ тг.} \quad (4.2)$$

Расчет затрат на необходимое оборудование приведены в таблице 4.2.2.

Таблица 4.2.2. – Расчет затрат на оборудование и ПО, необходимое для проекта

Наименование материала	Марка	Ед. измерения	Количество	Цена за ед. в тенге	Сумма в тенге
Ноутбук	Acer Aspire E5 – 771G	Штук	1	139 200	139 200
Принтер	Epson	Штук	1	34 990	34 990
Моб. телефон	Xiaomi Redmi 4A	Штук	1	33 990	33 990
Компьютерная мышь проводная	A4TECH N-708X	Штук	1	4 000	4 000
Итого					212 180

$$З_{общее} = 139\,200 + 34\,990 + 33\,990 + 4000 = 212180 \quad (4.3)$$

Так как для разработки программного продукта используются техника, которая потребляет электроэнергию, то необходимо рассчитать затраты на электроэнергию, которые представлены в Таблице 4.2.3.

Таблица 4.2.3 − Затраты на электроэнергию

Наименование оборудования	Паспортная мощность, кВт	Коэффициент использования мощности	Время работы оборудования для разработки ПП, ч	Цена электроэнергии, тг/кВт*ч	Сумма , тг
Ноутбук Acer Aspire E5-771G	0,6	0,7	250	18,32	1923,60
Моб. Телефон Xiaomi Redmi 4A	0,08	0,7	10	18,32	10,26
Принтер Epson	0,5	0,7	5	18,32	32,06
ИТОГО затраты на электроэнергию					1965,92

Общая сумма затрат на электроэнергию ($З_э$) рассчитывается по формуле:

$$З_э = \sum_{i=1}^{n} M_i * K_i * T_i * Ц, \quad (4.4)$$

где M_i - паспортная мощность i-го электрооборудования, кВт;

K_i - коэффициент использования мощности i-го электрооборудования (принимается K_i=0.7¸0.9);

T_i - время работы i-го оборудования за весь период разработки ПП ч;

Ц - цена электроэнергии, тг/кВт×ч;

i - вид электрооборудования;

n - количество электрооборудования.

$$З_{э\ общие} = (0,6 \times 0,7 \times 250 \times 18,32=) + \\ +(0,5 \times 0,7 \times 5 \times 18.32) + \\ +(0,08 \times 0,7 \times 10 \times 18.32) = 1965,92 \text{ тенге} \quad (4.5)$$

Для статьи «Затраты на оплату труда» необходимо включить расходы по оплате труда всех работников (студента, дипломного руководителя и консультантов, привлеченных к дипломному проекту), которые вовлечен и заняты разработкой самого программного продукта.

Средняя заработок программиста, начинающего по данным, представленным составляет 160000 тенге, средний заработок руководителя проекта составляет примерно 180000 не учитывая всякие бонусы, относящиеся к его званию кандидата технических наук и профессора.

Затраты на оплату труда рассчитываются по форме, приведенной в Таблице 4.2.4.

Таблица 4.2.4 − Затраты на оплату труда

Категория работника	Квалификация	Трудоемкость разработки ПП, чел.×ч	Часовая ставка, тг/ч	Сумма, тг
Служащий	Программист	250	925,4	231 350
Руководитель	Руководитель проекта	100	1071,43	107143
ИТОГО затраты на оплату труда				338493

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

$$З_{тр} = \sum_{i=1}^{n} ЧС_i * Т_i, \quad (4.6)$$

где $ЧС_i$ - часовая ставка i-го работника, тг;

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

i - категория работника;

n - количество работников, занятых разработкой ПП.

Часовая ставка работника может быть рассчитана по формуле:

$$ЧС_i = \frac{ЗП_i}{ФРВ_i}, \quad (4.7)$$

Фонд рабочего времени при 40 часовой неделе в 5 рабочих дней равняется 21 дням в месяц по 8 часов:

$ФРВ_i = 21 \times 8 = 168$ ч.

$$ЧС_i = \frac{ЗП_i}{ФРВ_i} = \frac{160\,000}{168} = 925{,}40 \quad (4.8)$$

$$ЧС_{рук} = \frac{ЗП_{рук}}{ФРВ_{рук}} = \frac{180\,000}{168} = 1071{,}43 \quad (4.9)$$

$$З_{тр.общее} = (925{,}4 \times 250) + (11071{,}43 \times 100) = 338493 \quad (4.10)$$

Трудоемкость разработки ПП определяется по данным таблицы 1.

В статью «Социальный налог» включается сумма, которая рассчитывается как 9,5% и медицинское страхование как 1,5% итого 11% от затрат на оплату труда всех работников ($З_{тр}$), занятых разработкой ПП. При расчете необходимо учесть, что пенсионные отчисления (10% от $З_{тр}$) не облагаются социальным налогом (ставки указаны на 2019 год).

$$З_{по} = З_{тр} \times 0,1 = 338493 \times 0,1 = 33849,30 \quad (4.11)$$

$$З_{тр}\,(\text{с уч п. о. }) = З_{тр} - З_{по} = \ 338493 - 33849,3 = 304643,70 \quad (4.12)$$

$$Н_{с} = З_{тр}(\text{с уч п. о. }) \times 0,11 = 304643,7 \times 0,11 = 33510,81 \quad (4.13)$$

В статью «Амортизация основных фондов» включается сумма амортизационных отчислений от стоимости оборудования и программного обеспечения (ПО), используемых при разработке ПП. Амортизационные отчисления рассчитываются по форме, приведенной в таблице 5.

Общая сумма амортизационных отчислений определяется по формуле:

$$З_{АМ} = \sum_{i=1}^{n} \frac{Ф_i * Н_{А_i} * Т_{НИРi}}{100 * Т_{ЭФi}} \quad (4.14)$$

где $Ф_i$ - стоимость i-го ОФ, тг;

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

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

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

i - вид ОФ;

n - количество ОФ.

При определении стоимости ОФ необходимо учесть также затраты на доставку и монтаж, установку ПО. Эти затраты могут быть приняты в размере 10-25 % от затрат на приобретение ОФ.

Годовые нормы амортизации ОФ принимаются по налоговому кодексу РК или определяются, исходя из возможного срока полезного использования ОФ:

$$Н_{Ai} = \frac{100}{Т_{Ni}} \quad (4.5)$$

$$Н_{Ai} = \frac{100}{5} = 20\%$$

$$Н_{Ai} = \frac{100}{3} = 33,3\%$$

$$Н_{Ai} = \frac{100}{7} = 14,3\%$$

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

Возможный срок полезного использования ОФ может быть принят:5 лет для ноутбука, 3 года для мобильного телефона и 7 лет для принтера (ОФ может быть принят от 3 до 10 лет).

$$З_{АМ1}=\frac{139\ 200\times20\times250}{100\times1730}=4023,12 \text{ тенге} \quad (4.16)$$

$$З_{АМ2}=\frac{34\ 990\times14,3\times10}{100\times1730}=28,90 \text{ тенге} \quad (4.17)$$

$$З_{АМ3}=\frac{33\ 990\times33,3\times20}{100\times1730}=130,85 \text{ тенге} \quad (4.18)$$

$$З_{АМобщие}=4023,12+28,9+130,85=4182,87 \text{ тенге} \quad (4.19)$$

Таблица 4.2.5 - Амортизация основных фондов (ОФ)

Наименование оборудования и ПО	Стоимость оборудования и ПО, тг	Годовая норма амортизации, %	Эффективный фонд времени работы оборудования и ПО, ч/год	Время работы оборудования и ПО для разработки ПП, ч	Сумма, тг
Ноутбук Acer Aspire E5-771G	139 200	20	1730	250	4023,12
Принтер Epson	34 990	14,3	1730	10	28,90
Моб. Телефон Xiaomi Redmi 4A	33 990	33,3	1730	20	130,85
ИТОГО					4182,87

В статью «Прочие затраты» включаются расходы на коммунальные платежи, затраты на лицензирование и сертификацию, расходы на рекламу, канцелярские и прочие хозяйственные расходы.

Таблица 4.2.6 – Затраты на интернет

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

На основании полученных данных по отдельным статьям составляется смета затрат на разработку (материальные затраты, затраты на оплату труда, отчисления на социальные нужды, амортизация основных фондов, прочие затрты) ПП по форме, приведенной в таблице 4.2.7.

Таблица 4.2.7 − Смета затрат на разработку программы

Статьи затрат	Сумма, тг
1. Материальные затраты, в том числе:	
- материалы	2000,00
- оборудование и ПО	212180,00
- электроэнергия	1965,92
2. Затраты на оплату труда.	338493,00
3. Отчисления на социальные нужды.	33510,81
4. Амортизация основных фондов.	4182,87
5. Прочие затраты.	4500,00
ИТОГО по смете	596832,60

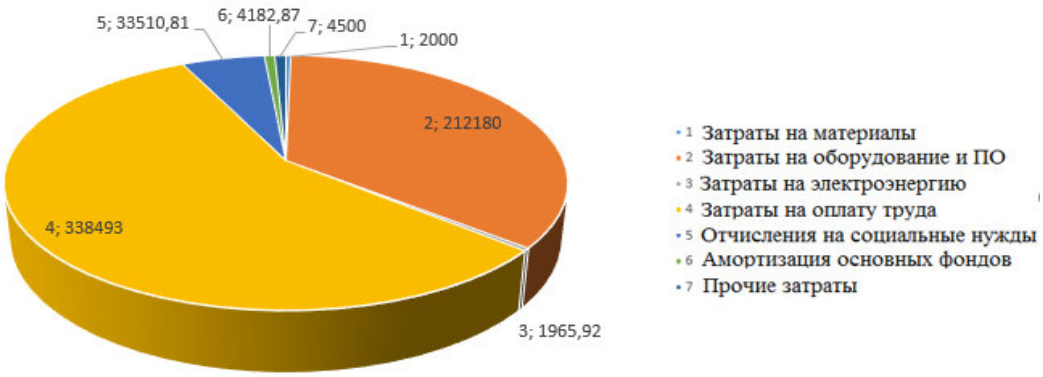


Диаграмма 4.2.1 – Смета затрат на разработку программы.

4.3 Определение возможной (договорной) цены ПП

Величина возможной (договорной) цены программы должна устанавливаться с учетом эффективности, качества и сроков ее выполнения на уровне, отвечающем экономическим интересам заказчика (потребителя) и исполнителя. Договорная цена ($Ц_Д$) для прикладных ПП рассчитывается по формуле:

$$Ц_д = З_{НИР} \times \left(1 + \frac{Р}{100}\right), \quad (4.20)$$

где $З_{НИР}$ - затраты на разработку ПП, тг;

Р - средний уровень рентабельности ПП. % (принимаем в размере 25 %, по согласованию с консультантом по экономической части).

$$Ц_д = З_{НИР} \times \left(1 + \frac{Р}{100}\right) = 596832{,}60 + 596832{,}60 \times 0{,}25 =$$
$$= 596832{,}60 + 149208{,}15 = 746040{,}75 \text{ тенге} \quad (4.21)$$

Далее определяется цена реализации с учетом налога на добавленную стоимость (НДС), ставка НДС устанавливается законодательно Налоговым Кодексом РК. На 2017 год ставка НДС установлена в размере 12%. Цена реализации с учетом НДС рассчитывается по формуле:

$$Ц_р = Ц_д + Ц_д \times НДС$$

$$Ц_р = 746040,75 + 746040,75 \times 0,12 = 835565,64 \text{ тенге} \qquad (4.22)$$

Таким образом, себестоимость программного продукта составляет – 596832,60 тенге.

Прибыль составляет – 149208,15 тенге.

Договорная цена − 835565,64 тенге.

5 Охрана труда и безопасность жизнедеятельности

5 .1 Анализ условий труда

Рабочее место разработчиков должно обеспечивать комфортное положение для работы, обеспечивая высокую производительность труда. Работа над дипломным проектом велась в офисе, в котором работают 2 служащих. Так как работа выполняется в офисе на ноутбуке с хорошим шумоподавлением, то работа с шумом исключается. В офисе установлен кондиционер, обеспечивающий отличный воздухообмен, поддерживающий приемлемую температуру. Из этого исходит, что в офисе основная работа над дипломным проектом ведется без проблем с шумом и вентиляцией. В офисе находится 2 светильника ЛД65 и имеется одно окно(3,5х1,5). Недостаток освещения неблагоприятно сказывается на работе разработчика и в этом разделе я привел расчеты по решению данной задачи.

5.2 Расчет естественного освещения

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

В данном случае, работа предполагает собой чтение, письмо и работу за компьютером, из этого следует, что освещенность, которая необходима для выполнения работ в данном помещении составляет 400 лк.

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

Тип помещения – офис.

Характеристики помещения:

длина L = 6 м, ширина B = 4 м, высота H = 3 м.

Высота рабочей поверхности над уровнем пола $h_{рп}$=0,725, окна начинаются с высоты $h_{рп}$=0,725 м, высота окна h_0=1,5м.

Рабочее помещение находится в IV часовом поясе – в г. Алматы (пояс светового климата – IV северной широты и южнее).

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

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

$$100\frac{S_0}{} = \frac{e_н \times \eta_0 \times К_{зд} \times К_з}{} \quad (5.1)$$

$$S_0=\frac{S_n\times e_н\times\eta_0\times К_{зд}\times К_з}{\tau_0\times r_1} \quad (5.2)$$

где S_n- площадь помещения; $S_n = 6\times4=24$ м^2;

$e_н$ – нормируемое значение КЕО;

$К_з$ – коэффициент запаса;

m_N – коэффициент светового климата;

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

$$e_x^{IV}=e_n\times m\times c \quad (5.3)$$

где m = 0,7;

c = 0,75 в наружных стенах зданий;

e_n - 1,2 для работ высокой точности III разряда зрительной работы;

$$e_x^{IV}=1,2\times0,7\times0,75 = 0,63 \quad (5.4)$$

Учитывая тип помещения и находим коэффициент k_3 = 1,2 (учебные помещения, лаборатории, конструкторские бюро).

τ_0- общий коэффициент светопропускания,

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

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

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

$\tau_3 = 0,8$ (железобетонные фермы и арки); $\tau_4 = 1$ (регулируемые жалюзи и шторы);

η_0 − световая характеристика окон.

$$\tau_0 = 0,5 \cdot 0,6 \cdot 0,8 \cdot 1 = 0,24 \quad (5.5)$$

Определим по формуле 5.4

$$L = B - 1$$

$$h_{расч} = h_{но} + h_0 - h_{рп} \quad (5.6)$$

$$L = 4 - 1 = 3 \text{ м}$$

$$\frac{L}{l}=\frac{L}{B-1}=\frac{6}{3}=2$$

$$h_{расч} = 0,8 + 1,5 - 0,725 = 1,575$$

$$\frac{B}{h_{расч}}=\frac{4}{1,575}=2,54$$

По таблице 5.2 определим $\eta_0 = 10,5$.

r_1 – коэффициент, учитывающий повышение КЕО при боковом освещении благодаря свету, отраженному от поверхностей помещения и подстилающего слоя.

Средний коэффициент отражения в помещении ρср = 0,5, принимаем одностороннее боковое освещение.

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

Тогда $r_1 = 1,05$.

$k_{зд}$ – коэффициент, учитывающий затенение окон противостоящими зданиями.

Поскольку затеняющих зданий поблизости нет, $k_{зд} = 1$. Вычислим общую площадь окон:

$$S_0=\frac{24\times1,2\times10,5\times1\times0,63}{100\times0,24\times1,05}=7,56 \text{ м}^2$$

Площадь световых проемов равна $S_{сп} = 7,56$ м^2

$$l_{ok}=\frac{S_o}{h_{ok}}=\frac{7,56}{3}=2,5 \qquad (5.7)$$

Исходя из этого, площадь световых проемов (5,75 <7,56 м^2) никак не обеспечивает необходимых условий труда на рабочих местах.

Так как основная цель – создание благоприятных условий труда в помещение с площадью S=24 м^2, исходя из этого совместно с естественным освещением используется искусственное освещение.

5.3Расчет искусственного освещения

Разряд зрительной работы – V. Нормируемая освещённость – 400 лк. В качестве светильника был взят Л33. Длина светильника 1514 мм, мощность – 65 ВТ. В помещении имеются 2 светильника.

В таблицу 5.3.1 изображены технические характеристики используемых светильников.

Схема освещенности представлена на рисунке 5.3.1.

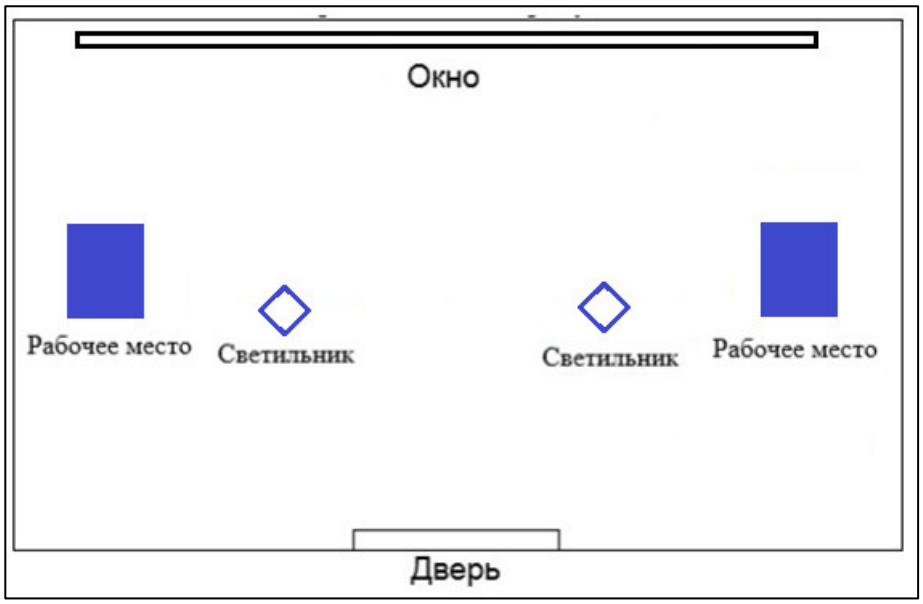


Рисунок 5.3.1 − Схема расчета освещенности

Таблица 5.3.1 – Технические характеристики Л33

Тип лампы	Мощность, Вт	Напряжение, В	Световой поток, лм	Длина L, не более, мм	Диаметр D, мм	Тип цоколя
Л33	65	110	3750	1514	38	G13d

Для начала идет проверка соответствия минимальной заданной освещенности при имеющихся 2 светильниках.

$$E=\frac{N\cdot n\cdot \Phi_л\cdot \eta}{k_z\cdot S_{ос}\cdot Z} \quad (5.9)$$

Где $S_{ОС}$ −площадь помещения;

k_z − коэффициент запаса;

N – Количество светильников;

Z −коэффициент неравномерности освещения, Z = 1,1;

n − количество ламп в светильнике;

$\Phi_л$ − световой поток выбранной лампы, $\Phi_л$ = 3750 лм

η − коэффициент использования, η = 65%

$$E=\frac{2\cdot 2\cdot 3750\cdot 0{,}65}{1{,}3\cdot 24\cdot 1{,}1}\approx 284 \text{ лк}$$

При 2 светильниках минимальная освещенность равняется примерно 284 лк, что не соответствует условиям труда. В следствие чего возникает

необходимость, увеличить количество светильников для обеспечения приемлемой освещенности. Для этого необходимо произвести расчеты для реконструкции:

Коэффициенты отражения от потолка стен и пола соответственно равны:

$$\rho_{\text{пот}} = 70\ \%$$

$$\rho_{\text{ст}} = 50\ \%$$

$$\rho_{\text{пол}} = 30\ \%$$

Вычислим высоту подвеса светильника над рабочей поверхностью по формуле 5.7:

$$H = h - h_p - h_c \qquad (5.10)$$

где: h_c – расстояние от светильника до перекрытия, h_c = 0,1 м;
h_p – высота рабочей поверхности над полом, h_p = 0,725 м;
h – высота помещения, h=3 м.

$$H = 3 - 0{,}1 - 0{,}725 = 2{,}175 \text{ м}$$

Расстояние между рядами светильников:

$$L_b = \lambda \cdot H \qquad (5.10)$$

$$L_b = 0{,}8 \cdot 2{,}175 = 1{,}74 \text{ м}$$

Расстояние между светильниками:

$$L_a = L_b - 0{,}1 = 1{,}74 - 0{,}1 = 1{,}64 \text{ м}$$

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

$$l_1 = (0{,}4 \div 0{,}8) \cdot L \qquad (5.11)$$

$$l_1 = 0{,}6 \cdot 1{,}74 = 1{,}044 \text{ м}$$

Определяем индекс помещения по формуле 5.10

$$i = \frac{L \times B}{h_p \times (L+B)} \qquad (5.12)$$

$$i=\frac{4\times 2}{2.175\times(4+2)}=1{,}319$$

Коэффициент использования в данном случае равен η = 65%, коэффициент запаса равен $k_з$ = 1,2

Определим количество люминесцентных ламп по формуле 5.11

$$N=\frac{E\times k_z\times S_{OC}\times Z}{n\times Ф_л\times \eta} \qquad (5.11)$$

Где S_{OC} −площадь помещения;

k_z − коэффициент запаса;

E − заданная минимальная освещенность, E = 400 лк;

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

n − количество ламп в светильнике;

$Ф_л$ − световой поток выбранной лампы, $Ф_л$ = 3570 лм;

η − коэффициент использования, η = 65%.

$$N=\frac{400\times 1{,}2\times 24\times 1{,}4}{2\times 3750\times 0{,}65}\approx 4$$

Для создания нормируемой освещенности 400 лк необходимо 4 люминесцентных ламп серии ЛД, мощность каждой лампы должна быть не меньше 65 Вт, что соответствует действительности, а значит имеющегося в наличии освещения достаточно для соответствия санитарным нормам.

Для решения данной проблемы следует провести реконструкцию офиса и добавить 2 люминесцентных ламп серии Л33для обеспечения необходимого освещения в помещении. На рисунке 5.3.2 изображен офис после реконструкции.

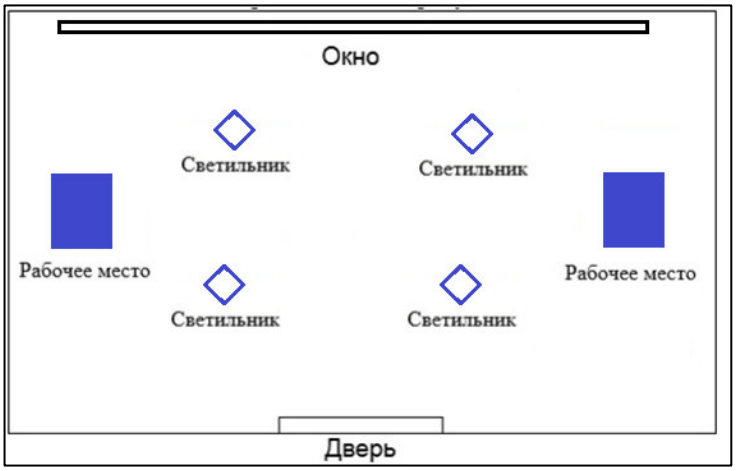


Рисунок 5.3.2 – Окончательный вариант расстановки светильников.

Заключение

Преимущества автоматизации в электронном обучении не только экономические. Поскольку преподаватели переходят от традиционного сосредоточения на группах к индивидуальному подходу, автоматизация дает много преимуществ. Это может изменить способ обучения студентов-точно адаптировать материал к группам и пользователям.

Допустим, пользователь не прошел тест. Это может привести к тому, что платформа отправит им электронное письмо с некоторыми дополнительными ресурсами или зачислит их в догоняющее подразделение с мини-модулями, ориентированными на области, которые нуждаются в работе. Наблюдение за тем, сколько пользователей перенаправлено в эту систему, также может стать отличным инструментом отслеживания для преподавателей, которые могут видеть, какие части курса нуждаются в более подробной информации и какие методы обучения работают лучше всего.

Другими словами, автоматизация электронного обучения может работать в виде петли обратной связи для проверки и измерения знаний и, в свою очередь, создавать эффективный и релевантный новый контент для учащихся.

Нетрудно понять, как автоматизация может изменить электронное обучение и в процессе этого изменить способ создания и потребления всех видов образовательного контента. Более специализированный контент означает больше вовлеченных учащихся, что приводит к лучшим результатам, более высоким показателям завершения и более эффективным курсам.

Автоматизация становится все более важным инструментом для преподавателей в крупных организациях. Вот почему менеджерам по персоналу важно учитывать, как выбранная ими система управления обучением адаптирует автоматизацию, позволяя им адаптировать инновационные решения, которые упрощают процесс обучения и обогащают опыт учащихся.

Чтобы узнать больше о том, как технология электронного обучения может помочь подготовиться к грядущим тенденциям в образовании рабочей силы в этом дипломном проекте создан шаблон приложения на Android Studio, который можно модифицировать под себя.

В этой работе рассмотрена создание шаблона, которое находится в открытом доступе и каждый желающий может модифицировать его под себя. Также рассмотрена расчет освещения помещения необходимого для комфортной работы в рабочем месте.

Список литературы

1 "Android Studio Release Updates". Android Developers Official Website. May 11, 2017. Retrieved May 12, 2017.

2 "Android Studio 3.0 Canary 3 is now available". Android Studio Official Blog. June 2, 2017. Retrieved June 3, 2017.

3 Forbes, How Much Do Average Apps Make.

4 Forbes, Мобильные услуги достигнут 32 млрд. долларов США к 2018 году

5 "Download Options". developer.android.com. Google. Retrieved 16 February 2017.

6 Forrester Research, Тед Шадлер.

7 Ben Evans, «Мобильники Съедят Мир», 2013.

8 "Terms and Conditions". developer.android.com. Google. Retrieved 24 April 2017.

9 "Build Overview". android.com.

10 "Building Android Studio". android.com.

11 "Android Studio website".

12 "Download Android Studio". Android Developers. Retrieved June 13, 2015.

13 "Google Launches Android Studio And New Features For Developer Console, Including Beta Releases And Staged Rollout". VentureBeat. December 8, 2014. Retrieved December 9, 2014.

14 Ducrohet, Xavier; Norbye, Tor; Chou, Katherine (May 15, 2013). "Android Studio: An IDE built for Android". Android Developers Blog. Google. Retrieved May 16, 2013.

15 "Getting Started with Android Studio". Android Developers. Google. Retrieved May 14, 2013.

16 Haslam, Oliver (May 16, 2013). "Download Android Studio IDE For Windows, OS X And Linux". Redmond Pie. Retrieved May 16, 2013.

17 Honig, Zach (May 15, 2013). "Google intros Android Studio, an IDE for building apps". Engadget. AOL. Retrieved May 16, 2013.

18 Dobie, Alex (May 15, 2013). "Android Studio unveiled at Google I/O keynote". Android Central. Mobile Nations. Retrieved May 16, 2013.

19 Olanoff, Drew (May 15, 2013). "Google Launches Android Studio And New Features For Developer Console, Including Beta Releases And Staged Rollout". TechCrunch. AOL. Retrieved May 16, 2013.

20 "Android Studio BETA". Google. Google. May 15 2013. Retrieved August 15, 2014

Приложение А
(обязательное)

Техническое задание

1. Основные требования:

– название разрабатываемой программы: автоматизация разработки электронных обучающих программ.

– цель разработки: создание шаблона образовательного приложения Android Academy, который позволяет сосредоточиться на содержании курсов. Предоставление пользователям инструмента для создания своей учебной платформы, не тратя на них много времени и ресурсов и не имея больших знаний о разработке проектов на платформе Android Studio;

Рекомендуемые платформы для разработки приложения (на выбор разработчика):

– Android Studio;
– Android SDK;
– Kotlin;
– Java;
– C#;
– SQLite;
– общий объем программной части системы, Мб;
– не более 50 Мб.

2. Технические требования:

– платформа работы приложения: Android Virtual Device;
– приложение Android;

3. Экономические требования.

Расчет сметы разработки программного продукта (подлежит обсуждению):

– стоимость готового продукта 835565,64 тг;
– стоимость разработки 596832,60;

Аудитория направлена на преподатавателей и студентов, стремящиеся к знаниям.

Приложение Б
(обязательное)

Листинг программы

```
package com.gcorso.academy.activities;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.design.widget.BottomNavigationView;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;
import com.gcorso.academy.R;
public class ChatActivity extends AppCompatActivity {
   private BottomNavigationView.OnNavigationItemSelectedListener
mOnNavigationItemSelectedListener
         = new BottomNavigationView.OnNavigationItemSelectedListener() {
      @Override
      public boolean onNavigationItemSelected(@NonNull MenuItem item) {
         Intent intent;
         switch (item.getItemId()) {
            case R.id.navigation_explore:
               intent = new Intent(ChatActivity.this, HomeActivity.class);
               startActivity(intent);
               return true;
            case R.id.navigation_chat:
               return true;
            case R.id.navigation_tools:
               intent = new Intent(ChatActivity.this, ToolsActivity.class);
               startActivity(intent);
               return true;
            case R.id.navigation_profile:
               intent = new Intent(ChatActivity.this, ProfileActivity.class);
               startActivity(intent);
               return true;
         }
         return false;
      }
   };
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_chat);
      final ActionBar abar = getSupportActionBar();
      View viewActionBar = getLayoutInflater().inflate(R.layout.actionbar_titletext_layout, null);
      ActionBar.LayoutParams params = new ActionBar.LayoutParams(
```

```
            ActionBar.LayoutParams.WRAP_CONTENT,
            ActionBar.LayoutParams.MATCH_PARENT,
            Gravity.CENTER);
        TextView textviewTitle = (TextView)
viewActionBar.findViewById(R.id.actionbar_textview);
        textviewTitle.setText("Чат");
        abar.setCustomView(viewActionBar, params);
        abar.setDisplayShowCustomEnabled(true);
        abar.setDisplayShowTitleEnabled(false);
        BottomNavigationView navigation = (BottomNavigationView)
findViewById(R.id.navigation);
        navigation.setOnNavigationItemSelectedListener(mOnNavigationItemSelectedListener);
        navigation.setSelectedItemId(R.id.navigation_chat);
    }
}
package com.gcorso.academy.activities;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.design.widget.BottomNavigationView;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.view.Gravity;
import android.view.MenuItem;
import android.view.View;
import android.widget.ListView;
import android.widget.TextView;
import com.gcorso.academy.adapters.CoursesListAdapter;
import com.gcorso.academy.persistence.LessonsLDH;
import com.gcorso.academy.objects.Course;
import com.gcorso.academy.R;
import java.util.List;
public class HomeActivity extends AppCompatActivity {
    CoursesListAdapter coursesListAdapter;
    LessonsLDH lessonsLDH;
    private BottomNavigationView.OnNavigationItemSelectedListener
mOnNavigationItemSelectedListener
        = new BottomNavigationView.OnNavigationItemSelectedListener() {
        @Override
        public boolean onNavigationItemSelected(@NonNull MenuItem item) {
            Intent intent;
            switch (item.getItemId()) {
                case R.id.navigation_explore:
                    return true;
                case R.id.navigation_chat:
                    intent = new Intent(HomeActivity.this, ChatActivity.class);
                    startActivity(intent);
                    return true;
                case R.id.navigation_tools:
                    intent = new Intent(HomeActivity.this, ToolsActivity.class);
```

```
                startActivity(intent);
                return true;
            case R.id.navigation_profile:
                intent = new Intent(HomeActivity.this, ProfileActivity.class);
                startActivity(intent);
                return true;
        }
        return false;
    }
  };
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_home);
    lessonsLDH = LessonsLDH.getInstance(this);
    List<Course> courses = lessonsLDH.getCourses();
    coursesListAdapter = new CoursesListAdapter(this, courses);
    ListView listCourses = findViewById(R.id.courses_list);
    listCourses.setAdapter(coursesListAdapter);
    BottomNavigationView navigation = (BottomNavigationView)
findViewById(R.id.navigation);
    navigation.setOnNavigationItemSelectedListener(mOnNavigationItemSelectedListener);
    final ActionBar abar = getSupportActionBar();
    View viewActionBar = getLayoutInflater().inflate(R.layout.actionbar_titletext_layout, null);
    ActionBar.LayoutParams params = new ActionBar.LayoutParams(
        ActionBar.LayoutParams.WRAP_CONTENT,
        ActionBar.LayoutParams.MATCH_PARENT,
        Gravity.CENTER);
    TextView textviewTitle = (TextView)
viewActionBar.findViewById(R.id.actionbar_textview);
    textviewTitle.setText("Изучаемый материал");
    abar.setCustomView(viewActionBar, params);
    abar.setDisplayShowCustomEnabled(true);
    abar.setDisplayShowTitleEnabled(false);
package com.gcorso.academy.activities;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.design.widget.BottomNavigationView;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.view.Gravity;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;
import com.gcorso.academy.adapters.CoursesProgGridAdapter;
import com.gcorso.academy.persistence.LessonsLDH;
import com.gcorso.academy.objects.Course;
import com.gcorso.academy.layout.ExpandableHeightGridView;
import com.gcorso.academy.layout.FitDoughnut;
import com.gcorso.academy.objects.Level;
```

```
import com.gcorso.academy.R;
import java.util.ArrayList;
import java.util.List;
public class ProfileActivity extends AppCompatActivity {
   private BottomNavigationView.OnNavigationItemSelectedListener
mOnNavigationItemSelectedListener
        = new BottomNavigationView.OnNavigationItemSelectedListener() {
      @Override
      public boolean onNavigationItemSelected(@NonNull MenuItem item) {
         Intent intent;
         switch (item.getItemId()) {
            case R.id.navigation_explore:
               intent = new Intent(ProfileActivity.this, HomeActivity.class);
               startActivity(intent);
               return true;
            case R.id.navigation_chat:
               intent = new Intent(ProfileActivity.this, ChatActivity.class);
               startActivity(intent);
               return true;
            case R.id.navigation_tools:
               intent = new Intent(ProfileActivity.this, ToolsActivity.class);
               startActivity(intent);
               return true;
            case R.id.navigation_profile:
               return true;
         }
         return false;
      }
   };
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_profile);
      final ActionBar abar = getSupportActionBar();
      View viewActionBar = getLayoutInflater().inflate(R.layout.actionbar_titletext_layout, null);
      ActionBar.LayoutParams params = new ActionBar.LayoutParams(
            ActionBar.LayoutParams.WRAP_CONTENT,
            ActionBar.LayoutParams.MATCH_PARENT,
            Gravity.CENTER);
      TextView textviewTitle = (TextView)
viewActionBar.findViewById(R.id.actionbar_textview);
      textviewTitle.setText("Профиль");
      abar.setCustomView(viewActionBar, params);
      abar.setDisplayShowCustomEnabled(true);
      abar.setDisplayShowTitleEnabled(false);
      BottomNavigationView navigation = (BottomNavigationView)
findViewById(R.id.navigation);
      navigation.setOnNavigationItemSelectedListener(mOnNavigationItemSelectedListener);
      navigation.setSelectedItemId(R.id.navigation_profile);
      LessonsLDH lessonsLDH = LessonsLDH.getInstance(this);
      Level level = lessonsLDH.getLevel();
```

```
        // set up the overall level
        FitDoughnut doughnut = (FitDoughnut) findViewById(R.id.doughnuttot);
        doughnut.animateSetPercent((float) level.getPerctot());
        TextView tvperctot = findViewById(R.id.tvpercentage);
        String p = Integer.toString(level.getPerctot())+ "%";
        tvperctot.setText(p);
        TextView tvLev = findViewById(R.id.tvlevel);
        tvLev.setText(level.getLiv());
        TextView tvProg = findViewById(R.id.tvprogress);
        String prog = Integer.toString(level.getProg()) + " / " + Integer.toString(level.getTot());
        tvProg.setText(prog);
        ExpandableHeightGridView gridCourses = findViewById(R.id.gridCoursesProg);
        List<Course> courses = new ArrayList<>();
        List<String> coursesTitles = lessonsLDH.getCourseNames();
        for(int i = 0; i<coursesTitles.size(); i++){
            courses.add(new Course(coursesTitles.get(i), level.getPerccourses()[i]));
        }
        CoursesProgGridAdapter coursesAdapter = new CoursesProgGridAdapter(courses);
        gridCourses.setAdapter(coursesAdapter);
        gridCourses.setExpanded(true);
    }
}
package com.gcorso.academy.activities;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.RelativeLayout;
import android.widget.TextView;
import com.gcorso.academy.persistence.LessonsLDH;
import com.gcorso.academy.layout.FitDoughnut;
import com.gcorso.academy.objects.Level;
import com.gcorso.academy.Preferences;
import com.gcorso.academy.R;
import java.util.ArrayList;
import java.util.List;
public class QuizActivity extends AppCompatActivity {
    String[] questions;
    String lessontitle;
    LessonsLDH lessonsLDH;
    int curq;
    int lessonid;
    public static final String Q_SEP = "<<-->>";
    public static final String A_SEP = "<->";
    boolean[] results;
    TextView lessontitleTv;
    TextView questiontitleTv;
```

```
    Button[] answersBt;
    ImageView[] resqIv;
    RelativeLayout emptyview;
    ImageView closeBt;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_quiz);
        Intent intent = getIntent();
        lessonid = intent.getIntExtra("lessonid", 0);
        lessonsLDH = LessonsLDH.getInstance(this);
        String q = lessonsLDH.getQuestions(lessonid);
        lessontitle = lessonsLDH.getLessonTitle(lessonid);
        questions = q.split(Q_SEP);
        curq = 0;
        lessontitleTv = findViewById(R.id.lesson_title);
        questiontitleTv = findViewById(R.id.question_title);
        answersBt = new Button[]{findViewById(R.id.answer1), findViewById(R.id.answer2),
findViewById(R.id.answer3), findViewById(R.id.answer4)};
        resqIv = new ImageView[]{findViewById(R.id.resq0), findViewById(R.id.resq1),
findViewById(R.id.resq2),
                findViewById(R.id.resq3),
findViewById(R.id.resq4),findViewById(R.id.resq5),findViewById(R.id.resq6),
                findViewById(R.id.resq7),findViewById(R.id.resq8),findViewById(R.id.resq9)};
        emptyview = findViewById(R.id.emptyview);
        emptyview.setVisibility(View.GONE);
        lessontitleTv.setText(lessontitle);
        results = new boolean[]{false, false, false, false, false, false, false, false, false, false};
        emptyview.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                nextQuestion();
            }
        });
        setCurrentQuestion();
        closeBt = findViewById(R.id.closebt);
        closeBt.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(QuizActivity.this, HomeActivity.class);
                startActivity(intent);
            }
        });
    }
    private void nextQuestion(){
        if(curq == 9){
            int tot = 0;
            for(int i = 0; i<10; i++){
                if(results[i]) {
                    tot++;
```

```
            }
        }
        int ch = 5*lessonsLDH.updateResult(lessonid, tot);
        LayoutInflater inflater = this.getLayoutInflater();
        View dialogView = inflater.inflate(R.layout.result_dialog, null);
        final android.app.AlertDialog dialog = new
android.app.AlertDialog.Builder(QuizActivity.this).setView(dialogView).show();
        TextView tvCompl = dialogView.findViewById(R.id.tvcompl);
        TextView tvScore = dialogView.findViewById(R.id.tvscore);
        TextView tvResult = dialogView.findViewById(R.id.tvresult);
        if(tot == 10){
            tvCompl.setText("Отлично!");
        } else if (tot>=7){
            tvCompl.setText("Хорошо!");
        } else {
            tvCompl.setText("Попробуйте снова!");
            tvCompl.setTextColor(getResources().getColor(R.color.wrong));
            TextView tvExtra = dialogView.findViewById(R.id.tvextra);
            tvExtra.setVisibility(View.VISIBLE);
        }

        String res = "Счёт:\n" + Integer.toString(tot) + "/10";
        tvResult.setText(res);
        String change = "+" + Integer.toString(ch) + "\n" + Preferences.SCORE_NAME;
        tvScore.setText(change);
        // level
        Level level = lessonsLDH.getLevel();
        FitDoughnut doughnut = (FitDoughnut) dialogView.findViewById(R.id.doughnuttot);
        doughnut.animateSetPercent((float) level.getPerctot());
        TextView tvperctot = dialogView.findViewById(R.id.tvpercentage);
        String p = Integer.toString(level.getPerctot())+ "%";
        tvperctot.setText(p);
        TextView tvLev = dialogView.findViewById(R.id.tvlevel);
        tvLev.setText(level.getLiv());
        TextView tvProg = dialogView.findViewById(R.id.tvprogress);
        String prog = Integer.toString(level.getProg()) + " / " + Integer.toString(level.getTot());
        tvProg.setText(prog);
        Button btTryAgain = dialogView.findViewById(R.id.btTryAgain);
        btTryAgain.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                dialog.dismiss();
                Intent intent = new Intent(QuizActivity.this, SectionActivity.class);
                intent.putExtra("sectionn", 0);
                intent.putExtra("lessonid", lessonid);
                startActivity(intent);
            }
        });
        Button btHome = dialogView.findViewById(R.id.btHome);
        btHome.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
            public void onClick(View view) {
                dialog.dismiss();
                Intent intent = new Intent(QuizActivity.this, HomeActivity.class);
                startActivity(intent);
            }
        });
    } else {
        emptyview.setVisibility(View.GONE);
        curq++;
        setCurrentQuestion();
    }
}
private void setCurrentQuestion(){
    String[] parts = questions[curq].split(A_SEP);
    questiontitleTv.setText(parts[0]);
    final int correctans;
    if(parts.length == 2){
        // true or false
answersBt[0].setBackground(getResources().getDrawable(R.drawable.answerbox));
        answersBt[1].setBackground(getResources().getDrawable(R.drawable.answerbox));
        answersBt[0].setText("Правда");
        answersBt[1].setText("Ложь");
        answersBt[2].setVisibility(View.GONE);
        answersBt[3].setVisibility(View.GONE);
        if(parts[1].charAt(0)== 'F'){
            correctans = 1;
        } else {
            correctans = 0;
        }
    } else {
        // 4 multiple choices
        List<Integer> list = new ArrayList<Integer>();
        list.add(0);
        list.add(1);
        list.add(2);
        list.add(3);
        java.util.Collections.shuffle(list);
        correctans = list.indexOf(0);
        for(int i = 0; i<4; i++){
            answersBt[i].setVisibility(View.VISIBLE);
            answersBt[i].setBackground(getResources().getDrawable(R.drawable.answerbox));
            answersBt[i].setText(parts[list.get(i)+1]);
        }
    }
    answersBt[correctans].setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
answersBt[correctans].setBackground(getResources().getDrawable(R.drawable.answerboxcorrect)
);
            results[curq] = true;
            resqIv[curq].setBackground(getResources().getDrawable(R.drawable.qcorrect));
```

```
            emptyview.setVisibility(View.VISIBLE);
        }
    });
    for(int i = 0; i<4; i++){
        if(i != correctans){
            final int finalI = i;
            answersBt[i].setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
answersBt[correctans].setBackground(getResources().getDrawable(R.drawable.answerboxcorrect)
);
answersBt[finalI].setBackground(getResources().getDrawable(R.drawable.answerboxwrong));
                    resqIv[curq].setBackground(getResources().getDrawable(R.drawable.qwrong));
                    emptyview.setVisibility(View.VISIBLE);
                }
            });
        }
    }
  }
}
package com.gcorso.academy.activities;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import com.gcorso.academy.persistence.LessonsLDH;
import com.gcorso.academy.objects.Section;
import com.gcorso.academy.R;
public class SectionActivity extends AppCompatActivity {
  TextView pageTv;
  ImageView navprecBt;
  ImageView navnextBt;
  ImageView closeBt;
  TextView sectiontitleTv;
  ImageView sectionimage;
  TextView sectiontextTv;
  Section section;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_section);
    Intent intent = getIntent();
    final int lessonid = intent.getIntExtra("lessonid", 0);
    final int sectionn = intent.getIntExtra("sectionn", 0);
    LessonsLDH lessonsLDH = LessonsLDH.getInstance(this);
    section = lessonsLDH.getSection(lessonid, sectionn);
    if(section == null){
      Toast.makeText(SectionActivity.this, "Проблемы при загрузке раздела."
```

```
            + Integer.toString(sectionn), Toast.LENGTH_LONG).show();
    }
    pageTv = findViewById(R.id.page);
    navprecBt  = findViewById(R.id.navprec);
    navnextBt = findViewById(R.id.navnext);
    closeBt = findViewById(R.id.closebt);
    sectiontitleTv = findViewById(R.id.sectiontitle);
    sectionimage = findViewById(R.id.sectionimage);
    sectiontextTv = findViewById(R.id.sectiontext);
    sectiontitleTv.setText(section.getTitle());
    String imagename = "s" + Integer.toString(lessonid) + Integer.toString(sectionn);
    sectionimage.setImageResource(getResources().getIdentifier(imagename,
          "drawable", getPackageName()));
    sectiontextTv.setText(section.getText());
    String page = section.getLessonTitle() + "  " + Integer.toString(sectionn+1)
          + "/" + Integer.toString(section.getLessonSections());
    pageTv.setText(page);
    closeBt.setOnClickListener(new View.OnClickListener() {
       @Override
       public void onClick(View view) {
          Intent intent = new Intent(SectionActivity.this, HomeActivity.class);
          startActivity(intent);
       }
    });
    if(sectionn == 0){
       navprecBt.setVisibility(View.GONE);
    } else {
       navprecBt.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View view) {
             changeSection(lessonid, sectionn-1);
          }
       });
    }
    navnextBt.setOnClickListener(new View.OnClickListener() {
       @Override
       public void onClick(View view) {
       if(sectionn<section.getLessonSections()-1){
          changeSection(lessonid, sectionn+1);
       } else {
          startQuiz(lessonid);
       }
       }
    });
}
void changeSection(int lessonid, int sectionn){
    Intent intent = new Intent(SectionActivity.this, SectionActivity.class);
    intent.putExtra("sectionn", sectionn);
    intent.putExtra("lessonid", lessonid);
    startActivity(intent);
}
```

```
    void startQuiz(int lessonid){
        Intent intent = new Intent(SectionActivity.this, QuizActivity.class);
        intent.putExtra("lessonid", lessonid);
        startActivity(intent);
    }
}
package com.gcorso.academy.activities;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.design.widget.BottomNavigationView;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.view.Gravity;
import android.view.MenuItem;
import android.view.View;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;
import com.gcorso.academy.R;
public class ToolsActivity extends AppCompatActivity {
    private BottomNavigationView.OnNavigationItemSelectedListener
mOnNavigationItemSelectedListener
            = new BottomNavigationView.OnNavigationItemSelectedListener() {
        @Override
        public boolean onNavigationItemSelected(@NonNull MenuItem item) {
            Intent intent;
            switch (item.getItemId()) {
                case R.id.navigation_explore:
                    intent = new Intent(ToolsActivity.this, HomeActivity.class);
                    startActivity(intent);
                    return true;
                case R.id.navigation_chat:
                    intent = new Intent(ToolsActivity.this, ChatActivity.class);
                    startActivity(intent);
                    return true;
                case R.id.navigation_tools:
                    return true;
                case R.id.navigation_profile:
                    intent = new Intent(ToolsActivity.this, ProfileActivity.class);
                    startActivity(intent);
                    return true;
            }
            return false;
        }
    };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_tools);
        final ActionBar abar = getSupportActionBar();
```

```
        View viewActionBar = getLayoutInflater().inflate(R.layout.actionbar_titletext_layout, null);
        ActionBar.LayoutParams params = new ActionBar.LayoutParams(
                ActionBar.LayoutParams.WRAP_CONTENT,
                ActionBar.LayoutParams.MATCH_PARENT,
                Gravity.CENTER);
        TextView textviewTitle = (TextView)
viewActionBar.findViewById(R.id.actionbar_textview);
        textviewTitle.setText("Инструменты");
        abar.setCustomView(viewActionBar, params);
        abar.setDisplayShowCustomEnabled(true);
        abar.setDisplayShowTitleEnabled(false);
        BottomNavigationView navigation = (BottomNavigationView)
findViewById(R.id.navigation);
        navigation.setOnNavigationItemSelectedListener(mOnNavigationItemSelectedListener);
        navigation.setSelectedItemId(R.id.navigation_tools);
        RelativeLayout item1 = findViewById(R.id.item1);
        item1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // click on item 1
            }
        });
        RelativeLayout item2 = findViewById(R.id.item2);
        item2.setOnClickListener(itemEmptyListener);
        RelativeLayout item3 = findViewById(R.id.item3);
        item3.setOnClickListener(itemEmptyListener);
        RelativeLayout item4 = findViewById(R.id.item4);
        item4.setOnClickListener(itemEmptyListener);
    }
    View.OnClickListener itemEmptyListener = new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Toast.makeText(ToolsActivity.this, "Этот инструмент еще недоступен.",
Toast.LENGTH_LONG).show();
        }
    };
}
```