Оглавление

[Введение 2](#_Toc12314852)

[1. Теоретическая часть 4](#_Toc12314853)

[1.1 Основы разработки приложений для ОС Android – основные понятия и определения 4](#_Toc12314854)

[1.2 Обзор языков программирования для создания приложений на Android 8](#_Toc12314855)

[1.3 Аналитический обзор инструментов для разработки приложений на Android 11](#_Toc12314856)

[2. Практическая часть 21](#_Toc12314857)

[2.1 Пример разработки мобильного приложения в Android Studio 21](#_Toc12314858)

[2.2 Примеры разработки мобильного приложения с использованием 31](#_Toc12314859)

[Visual Studio 2019 и Xamarin 31](#_Toc12314860)

[Заключение 47](#_Toc12314861)

[Литература 48](#_Toc12314862)

[Приложения 52](#_Toc12314863)

Введение

Разработка приложений под Android является одним из популярных направлений в программировании. Около 2/3 мобильных устройств используют именно эту операционную систему. Для нее разрабатывают игры, различные сервисы, стартапы. Крупные компании все чаще предлагают пользователям сервисные приложения. А программисты-одиночки вполне успешно зарабатывают на создании небольших программных продуктов (коммерческих или со встроенной рекламой) [9].

Первый iPhone был представлен в 2007-м. Через год появился магазин приложений от Apple – App Store, тогда же на сцену вышел Android, а вместе с ним магазин приложений Google Play (Android Market). Именно появление iPhone, Android (позже еще и Windows Phone) ознаменовало образование современного рынка мобильного ПО, каким мы его знаем сейчас [10].

Android является универсальной открытой платформой, используемой миллионами пользователей по всему миру. Существует множество инструментов для Android-разработчиков, которые помогут быстро начать работу. С каждым годом инструментов становится все больше, а их эффективность постоянно повышается [11].

С развитием и усложнением мобильных приложений стали появляться различные среды разработки и языки программирования, специально предназначенные для создания мобильного ПО. В итоге инструментов для разработки мобильных прикладных программ стало так много, что сейчас не только новичку, но и матерому программисту сложно вникнуть в это изобилие [10].

В этой связи, целью работы является аналитический обзор современных инструментов программирования для операционной системы Android, выбор средств для создания собственного мобильного приложения и его тестирование на реальном устройстве.

Задачами работы являются:

1. Обзор средств разработки приложений для мобильных устройств с операционной системой Android.
2. Сравнительный анализ интегрированных сред разработки мобильных приложений на ОС Android.
3. Создание тестового мобильного приложения в среде разработки Android Studio.
4. Создание тестового мобильного приложения в среде разработки MS Visual Studio с использованием фреймворка Xamarin.
5. Создание готового подписанного приложения и его запуск на реальном устройстве и в программе-эмуляторе для тестирования мобильных приложений.

1. Теоретическая часть

1.1 Основы разработки приложений для ОС Android –  
основные понятия и определения

Android – операционная система (ОС) [12] для мобильных устройств: смартфонов, планшетных компьютеров, КПК [1]. ОС Андроид была создана разработчиком Энди Рубином и поначалу развивалась в рамках компании Android Inc., а в 2005 году Google покупает Android Inc.

Архитектура ОС Android показана на рисунке 1 [13].

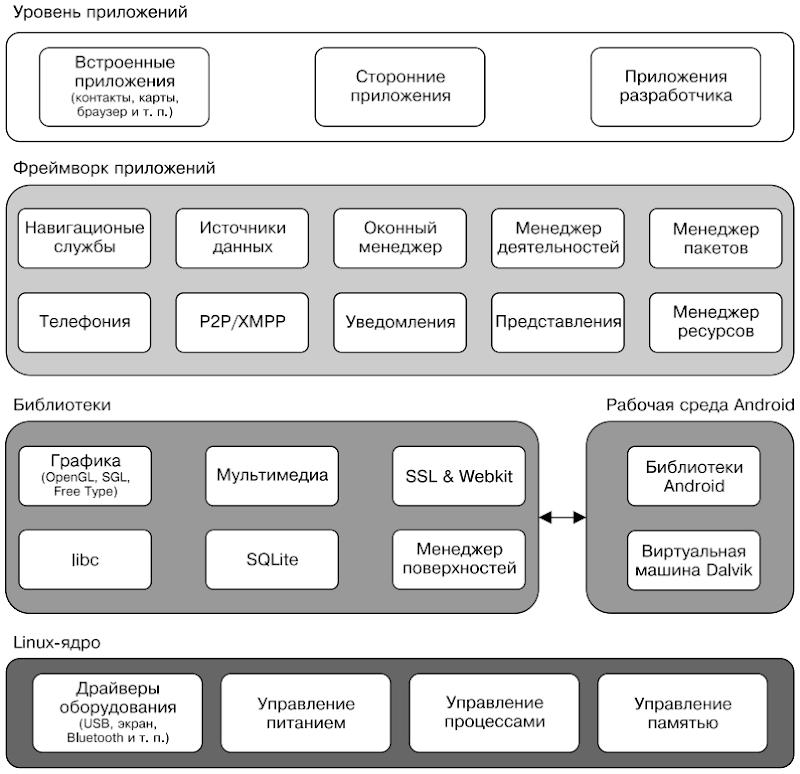


Рисунок 1 – Архитектура операционной системы Android

Внизу данной модели располагается ядро операционной системы как самая фундаментальная и базовая составляющая. Android основан на урезанном ядре ОС Linux. Оно обеспечивает функционирование системы, отвечает за безопасность, управление памятью, энергосистемой и процессами и т.д. [2].

Над ядром лежит набор библиотек, предназначенный для обеспечения базового функционала для приложений. Этот уровень отвечает за предоставление реализованных алгоритмов для вышележащих уровней, поддержку файловых форматов, осуществление кодирования и декодирования информации, отрисовку графики и многое другое. Библиотеки реализованы на C/C++ и скомпилированы под конкретное аппаратное обеспечение устройства, вместе с которым они и поставляются производителем в предустановленном виде.

Здесь же располагается Android Runtime – среда выполнения. Ключевыми её составляющими являются набор библиотек ядра (Core Libraries) и виртуальная машина Dalvik. Библиотеки обеспечивают большую часть низкоуровневой функциональности, доступной библиотекам ядра языка Java. Каждое приложение в ОС Android запускается в собственном экземпляре виртуальной машины Dalvik. Таким образом, все работающие процессы изолированы от операционной системы и друг от друга.

Выше располагается Application Framework, иногда называемый уровнем каркаса приложений [14]. Именно через каркасы приложений разработчики получают доступ к API, предоставляемым компонентами системы, лежащими ниже уровнем. Благодаря архитектуре фреймворка [6], любому приложению предоставляются уже реализованные возможности других приложений, к которым разрешено получать доступ.

В самом верху лежит уровень приложений. Сюда относится набор базовых приложений, который предустановлен на ОС Android. В него входят браузер, почтовый клиент, программа для отправки SMS, карты, календарь, менеджер контактов и многие другие [7]. Список интегрированных приложений может меняться в зависимости от модели устройства и версии Android. К уровню приложений относятся, в принципе, все приложения под платформу Android, в том числе и установленные пользователем. В целом, под мобильным приложением понимают компонент, устанавливаемый на мобильное устройство (телефон, планшет, фаблет, смартфон и т.п.) под конкретную мобильную платформу (Android, iOS, BlackBerry, HP webOS, Bada от Samsung и WindowsMobile), подключающийся к мобильному серверу и управляющий пользовательским интерфейсом и бизнес-логикой устройства [15].

Считается, что приложения под Android пишутся на языке Java, но существует возможность разрабатывать программы также на C/C++ (с помощью Native Development Kit), Basic (с помощью Simple), C# и с использованием других языков.

К настоящему моменту сложилась устойчивая классификация мобильных приложений, состоящая из трех больших групп: нативные, веб- и гибридные приложения [16]. Особенностью *нативных мобильных приложений* является то, что они разрабатываются специально под конкретную платформу (например, под iOS для iPhone, под Android для устройств под управлением ОС Android или под Windows для WindowsPhone и т.д.) и требуют от разработчика специальных знаний и умений для работы в конкретной среде разработки (xCode, Objective C для iPhone, Java,eclipse для устройств на Android и т.д.). Как правило, для написания таких приложений используются только «родные» языки программирования высокого уровня, затем они компилируются в нативный код мобильной платформы (операционной системы), дающий максимальную производительность и возможность использования всех функций устройства, таких как камера, геолокация, адресная книга пользователя и др.

*Веб-приложения* часто называют html5-приложениями, поскольку это, по сути, сайт, оптимизированный под смартфон. Пользовательский интерфейс создается при помощи стандартных веб-технологий. Их не нужно загружать из магазина приложений, но они могут находиться в специальных магазинах веб-приложений, которые есть в браузерах.

*Гибридные мобильные приложения* сочетают в себе некоторые функции нативных и веб-приложений: таких как кросс-платформенность и возможность использования аппаратных средств устройства. Такие приложения могут быть загружены через магазины приложений, и при этом имеют возможность независимого обновления информации. Гибридные приложения всегда требуют подключения к интернету, поскольку их веб-компонента обновляется через интернет. Такой способ построения мобильных приложений постепенно становится самым популярным, так как у него органическая среда распространения, но при этом разработка происходит быстрее и дешевле, чем в случае с нативными приложениями.

В современных условиях разработка ПО в большинстве случаев ведется с использованием *интегрированных сред разработки (IDE)* [17]. IDE – набор инструментов для разработки и отладки программ, имеющий общую интерактивную графическую оболочку, поддерживающую выполнение всех основных функций жизненного цикла разработки программы – набор и редактирование исходного текста (кода), компиляцию (сборку), исполнение, отладку, профилирование и др.

Также стоит отметить, что основным набором инструментом которым должна обладать платформа для программирования мобильных приложений, является Android SDK [18]. В состав Android SDK входят такие виды инструментов как:

− SDK manager (загружает и устанавливает компоненты Android SDK) [3];

− Debug Monitor (предназначен для отладки графического интерфейса);

− Android Emulator (инструмент для тестирования приложения непосредственно на компьютере);

− AVD manager (создает виртуальные Android устройства) [4];

− Android Debug Bridge (инструмент для управления эмулятором) [5].

1.2 Обзор языков программирования для создания приложений  
на Android

Рассмотрим основные языки для написания Android-приложений.

1. *Java*.

*Основная IDE: Android Studio.*

Java – строго типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Это официальный язык программирования данной ОС. Сложность Java мешает программировать на нём всем желающим. Как у объектно-ориентированного языка программирования у него куча особенностей в виде конструкторов классов, исключений, приводящих к падению приложений во время работы и других моментов, которые всегда необходимо учитывать при разработке. Тем не менее, код на Java легко читается и структурируется, особенно при соблюдении принятых стандартов его оформления [19].

1. *С++.*

*Основные IDE: Android Studio (версия 1.3 и выше), Visual Studio, QtCreator.*

**Более низкоуровневый язык. Подходит для** игровых движков и ресурсоёмких приложений. Был изобретён еще в 1985 году Бьёрном Страуструпом и до сих пор занимает верхние позиции самых популярных языков программирования. Данный язык предлагает полную свободу производимых действий, но имеет некоторые специфические особенности, на изучение которых может уйти немало времени.

1. C#.

*Основная IDE: Visual Studio.*

C# — объектно-ориентированный язык программирования. Разработан в 1998-2001 годах группой инженеров в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework в области разработки мобильных приложений и используется во фреймворке Xamarin [20]. Это упрощенная версия C/C+, разработанная компанией Microsoft, к тому же, язык является более современным и чистым в отличие от вышеупомянутых языков программирования. C# отлично подходит для разработки игр, но ограничивает ваши возможности на пути к профессиональному Android-разработчику.

1. Kotlin.

*Основные IDE: IntelliJ IDEA, Android Studio, Eclipse и NetBeans.*

Kotlin — это статически типизированный язык. Он поддерживает как объектно-ориентированное, так и процедурное программирование. По аналогии с вышеупомянутыми языками, основной код Kotlin-программы пишется в функции main, которой передаётся массив аргументов командной строки. Разработан петербургской компанией JetBrain. Язык был официально представлен в мае 2017 года на Google I/O и позиционируется Google как второй официальный язык программирования под Android после Java, только чуть более простой для понимания [21].

1. HTML5 + JavaScript + CSS.

*Основная IDE: PhoneGap.*

Подходит для создания нересурсоёмких приложений. PhoneGap позволит собирать полноценные приложения, разработанные на вышеупомянутых языках программирования и разметки. Приложения компилируются на сервере и затем уже доступны для использования на iOS, Android, Windows Phone, Web OS и BlackBerry OS. С такой широкой кроссплатформенностью разработка приложений может значительно ускориться [22].

1. Python.

*Основные IDE: Eclipse, Visual Studio, PyCharm.*

Не является языком мобильной разработки, а больше используется в веб-проектах. Самым популярным фреймворком является Kivy, который позволит создать приложение для Play Market на чистом Python.

1. LUA Script.

*Основная IDE: Corona.*

**Подойдёт для** создания игр и простых приложений. Corona — это достаточно легковесная среда разработки, код в которой необходимо писать на достаточно лёгком LUA. Данный тулкит поможет вам при написании простых 2D-игр, для коих здесь присутствуют библиотеки для 2D=объектов, звуков, сети и игрового движка. Созданные игры работают с OpenGL, что означает высокую эффективность.

1. JavaScript и UX.

*Основная IDE: Fuse.*

Основная логика Fuse-приложений строится на JavaScript — простом и понятном языке с низким порогом вхождения.

1. Visual Basic.

*Основная IDE: Basic4Android.*

Для BASIC есть специальная среда разработки B4A, в которой можно создавать Android-приложения. B4A воплощает концепцию визуального прототипирования RAD, которая подразумевает быстроту и удобство программирования.

1. Delphi/Object Pascal [8].

*Основная IDE: Embarcadero RAD Studio.*

Существенный недостаток в том, что данная IDE является платной, в то время как большинство конкурирующих платформ бесплатны и предоставляют лучший функционал.

В целом, приложения под Android можно создать практически на любом популярном языке – фреймворки и утилиты найдутся под все. Для тех, кто хочет стать профессиональным Android-разработчиком, использовать все возможности операционной системы, то лучше выбрать Java, Kotlin или C#.

1.3 Аналитический обзор инструментов для разработки приложений  
на Android

Рассмотрим основные инструменты для написания Android-приложений (таблица 1).

Таблица 1 – Сравнительная характеристика сред разработки ПО для Android

| Android IDE/Framework | Язык  программирования | Доступно на платформе | Целевая платформа | Открытое ПО | Интерфейс |
| --- | --- | --- | --- | --- | --- |
| Eclipse | Java, C/C++, C#, JavaScript,  Python | Любая ОС, поддерживающая Java | Android,  iOS,  Linux,  MacOS,  Windows | Да (Eclipse Public License) | Нативный |
| IntelligJ IDEA | Java,  Scala,  Groovy,  Kotlin,  JavaScript,  TypeScript,  SQL | Windows  MacOS  Linux | Любая ОС, поддерживающая Java | Нет (Ultimate Ed.), Да (Community Ed., Apache License), | Нативный |
| Android Studio | Java,  C/C++, Kotlin | Windows, OS X, Linux | Android | Да (Apache License) | Нативный |
| NetBeans | Java,  C/C++,  HTML,  Python,  PHP,  JavaScript | Windows,  MacOS,  Linux,  Solaris | Кросс-платформенная | CDDL 1.0 и GPL2 | Нативный |
| Komodo | Java,  JavaScript,  Python,  PHP,  HTML,  Ruby | Кросс-платформенная | Windows,  MacOS,  Linux | Да (Mozilla  Public License) | Нативный |
| Xamarin | C#, Xaml | Windows,  OS X | iOS,  Android,  Windows,  Tizen | Да (Лицензия MIT) | Нативный |
| Unity | C#, UnityScript | Windows,  OS X,  Linux | iOS,  Android,  Windows,  Tizen | Ограниченная | UI Canvas |
| Phonegap | JavaScript, HTML5, CSS3 и нативные языки (Java, Objective-C, C#) | Windows,  OS X,  Linux | Android,  iOS,  Windows,  Blackberry, WebOS,  Symbian,  Bada,  Ubuntu,  Firefox OS | Да (Apache License). | Web |
| Qt | C++ QML | Кроссплатформенная | Android,  iOS,  Windows,  Symbian,  Linux, QNX | Да (GNU LGPL) | Нативный |
| Appcelerator Titanium | JavaScript, Python, Ruby, PHP | Windows,  OS X,  Linux | iOS,  Android,  BlackBerry, Windows, Tizen,  Denso | Да (Apache License). | Нативный |
| Telerik Platform | .Net, JavaScript, HTML5, Java | Windows,  OS X,  Linux | iOS,  Android,  BlackBerry, Windows | Нет | Web |
| Embarcadero RAD Studio | Delphi, C++ | Windows, MacOS | iOS,  Android | Нет | Нативный |

1. Eclipse.

Разработана некоммерческой компанией Eclipse Foundation. Данная среда разработки абсолютно нейтральна к платформе и языку программирования [23]. Eclipse поддерживает языки: Cobol, Java, C++, C. Но в добавок к этому есть функция добавления интересующего вас языка, к примеру таких как: C#, PHP, Python, Ruby. Среда Eclipse доступна, при помощи Eclipse Consortium под видом скомпилированного исполняемого файла для Windows, Linux и др. Eclipse представляет из себя платформу, в которой разрабатываются плагины, далее встраиваемые в неё. Одним из таких является Android Development Tools (ADT).

Преимущества Эклипс:

* интерфейс переведен на грамотный русский язык (документация прилагается);
* хорошо работает на компьютерах с низкой производительностью;
* имеет дополнительные функции (для серверной работы и анализа базы данных);
* может подключаться к модулям;
* может работать в групповом режиме (когда проект создают несколько человек одновременно).

Недостатки: многие параметры этой среды разработки могут запугать новичков. Eclipse не обладает всеми теми функциями, что и IntelliJ IDEA.

1. IntelligJ IDEA.

Разработана Jet Brains. Поддерживаемые языки программирования: AngularJS, CoffeeScript, HTML, JavaScript, LESS, Node JS, PHP, Python, Ruby, Sass, TypeScript и другие.

Особенности:

* Расширенный редактор баз данных и дизайнер UML;
* Поддержка нескольких систем сборки;
* Пользовательский интерфейс тестового запуска приложений;
* Интеграция с Git;
* Поддержка Google App Engine, Grails, GWT, Hibernate, Java EE, OSGi, Play, Spring, Struts и других;
* Встроенные средства развертывания и отладки для большинства серверов приложений;
* Интеллектуальные текстовые редакторы для HTML, CSS и Java;
* Интегрированный контроль версий;
* AIR Mobile с поддержкой Android и iOS.

Недостатки: эта среда разработки JavaScript требует времени и усилий на изучение, поэтому может оказаться не лучшим вариантом для начинающих. В ней есть много сочетаний горячих клавиш, которые нужно просто запомнить. Некоторые пользователи жалуются на неуклюжий интерфейс.

1. Android Studio.

Продукт компании Google, основан на IntelliJ Idea. Android Studio выступает в качестве редактора для выбранного вами языка программирования (он поддерживает Java, C++, а также Kotlin, который с недавних пор тоже стал официальным языком Android), компилятора, который может создавать APK файлы и файловой системы для организации проекта. Разработка интерфейса производится drag-n-drop методом, но также имеется возможность использовать XML.

Достоинства [24]:

* поддержка нескольких языков программирования, к которым относятся самые популярные – C/C++, Java.
* редактор кода, с которым удобно работать;
* позволяет разрабатывать приложения не только для смартфонов/планшетов, а и для портативных ПК, приставок для телевизоров Android TV, устройств Android Wear, новомодных мобильных устройств с необычным соотношением сторон экрана;
* тестирование корректности работы новых игр, утилит, их производительности на той или иной системе, происходит непосредственно в эмуляторе;
* достаточно большая библиотека с готовыми шаблонами и компонентами для разработки ПО;
* предварительная проверка уже созданного приложения на предмет ошибок в нем;
* большой набор средств инструментов для тестирования каждого элемента приложения, игры;
* для неопытных/начинающих разработчиков специально создано руководство по использованию Android Studio, размещенное на официальном сайте утилиты.

Недостатки: скупые возможности персонализации проявляются в редакторе кода и общих настройках. Все средства пакета требуют довольно много места на диске.

1. NetBeans.

Подходит для редактирования существующих проектов или создания новых. NetBeans предлагает простой drag-and-drop интерфейс, который поставляется с большим количеством удобных шаблонов проектов. Среда в основном используется для разработки Java приложений, но можно устанавливать пакеты, поддерживающие другие языки. Особенности:

* Интуитивный drag-and-drop интерфейс;
* Динамические и статические библиотеки;
* Интеграция нескольких сессий GNU-отладчика с поддержкой кода;
* Возможность осуществлять удаленное развертывание;
* Совместимость с платформами Windows, Linux, OS X и Solaris;
* Поддержка Qt Toolkit;
* Поддержка Fortan и Assembler;
* Поддержка целого ряда компиляторов, включая CLang / LLVM, Cygwin, GNU, MinGW и Oracle Solaris Studio.

Недостатки: эта бесплатная среда разработки потребляет много памяти, поэтому может работать медленно на некоторых ПК.

1. Komodo.

Особенности:

* Настраиваемый многооконный интерфейс;
* Интеграция контроля версий для Bazaar, CVS, Git, Mercurial, Perforce и Subversion;
* Профилирование кода Python и PHP;
* Возможность развертывания в облаке благодаря Stackato PaaS;
* Графическая отладка для NodeJS, Perl, PHP, Python, Ruby и Tcl;
* Автоматическое заполнение и рефакторинг;
* Стабильная производительность на платформах Mac, Linux и Windows.

Недостатки: бесплатная версия среды разработки программного обеспечения не включает в себя все функции.

1. Xamarin.

С помощью фреймворка Xamarin, который входит в состав Visual Studio, можно создавать кроссплатформенные приложения с помощью C#, а затем тестировать их на нескольких устройствах, подключенных к облаку. Это хороший и бесплатный выбор, если вы планируете выпустить приложение и для Android, и для IOS, но желаете писать свой код дважды. Также он является отличным выбором для тех, кто уже знаком с C# и/или Visual Studio. Минусом является то, что Xamarin неудобен в использовании Java библиотек и, как и с любой другой альтернативой Android Studio, теряется поддержка Google и расширенные встроенные функции.

Особенности IDE Visual Studio:

* Огромная библиотека расширений, которая постоянно увеличивается;
* IntelliSense;
* Настраиваемая панель и закрепляемые окна;
* Простой рабочий процесс и файловая иерархия;
* Статистика мониторинга производительности в режиме реального времени;
* Инструменты автоматизации;
* Легкий рефакторинг и вставка фрагментов кода;
* Поддержка разделенного экрана;
* Список ошибок, который упрощает отладку;
* Проверка утверждения при развертывании приложений с помощью ClickOnce, Windows Installer или Publish Wizard.

Недостатки: поскольку Visual Studio является супертяжелой IDE, для открытия и запуска приложений требуются значительные ресурсы. Поэтому на некоторых устройствах внесение простых изменений может занять много времени. Для простых задач целесообразно использовать компактный редактор или средство разработки PHP.

1. Unity.

Созданные с помощью Unity приложения работают под операционными системами Windows, OS X, Linux, Android, Apple iOS, Windows Phone, BlackBerry, а также на игровых приставках Wii, PlayStation и Xbox.

Преимущества:

* Отличный вариант для создания мобильных игр для целого ряда устройств
* 3D-движок дает высококачественные результаты без каких-либо сложных конфигураций
* Есть много хороших бесплатных плагинов
* Unity позволяет разработчику сделать свои собственные шейдеры и изменить путь, которым Unity визуализирует игру.

Недостатки: UI и сложность в использовании для новичков, исходный код недоступен. Компиляторы Unity не оптимизированы для ARM процессоров на некоторых мобильных устройствах [25].

1. Phonegap.

PhoneGap позволяет создавать мобильные приложения используя стандартные веб технологии (HTML5, JavaScript and CSS3).

Преимущества:

* PhoneGap имеет простое API, что позволит легко начать разработку, для тех кто сталкивался с HTML, CSS и JavaScript.
* Возможность использования любых существующих JavaScript библиотек (JQuery, Prototype, Sencha Touch)
* Поддержка всех мобильных платформ.

Недостатки: Пользовательский интерфейс визуализируется с помощью встроенного браузера. Это создает трудности в получении обратной связи по сравнению с нативным приложением. Часто существующие плагины оказываются устаревшими, поэтому иногда придется писать свои.

1. Qt.

Qt библиотека позволяет создавать кроссплатформенные оконные приложения на C++. Qt стоит рассматривать не столько как набор классов для создания GUI, а скорее как полноценный инструментарий классов на все случаи жизни. Есть возможность разрабатывать программы не только на C++, но и языке QML, сильно схожим с JavaScript. Это особая ветвь развития Qt, направленная на быстрое прототипирование и разработку мобильных приложений.

Преимущества:

* Qt имеет множество хороших инструментов, которые помогут в разработке, например, IDE QT Creator, Qt Designer и code profiling.
* Он имеет библиотеки, содержащие интуитивно понятные API интерфейсы для элементов, таких как сети, анимации и многое другое.

Недостатки: Qt сложен для начинающих.

1. Appcelerator Titanium.

Приложения, созданные с помощью данного SDK, будут нативными. Причем не только вид, но и сам код приложения будет тоже нативный. Но это не мешает создавать и классический WebView [26] и наполнить его желаемым web контентом.

Преимущества:

* JavaScript позволяет легко разрабатывать приложения без использования языков платформы.
* Appcelerator позволяет делать аналитику в режиме реального времени
* Использование native API даст более высокую производительность для приложений, которые не очень велики.

Недостатки: Есть задержки при запуске приложения из-за загрузки библиотеки. Трудно создавать сложные приложения, так как использование JavaScript отрицательно сказывается на производительности приложений.

1. Telerik Platform.

Одной из основных причин использовать AppBuilder является полноценная онлайн IDE. Она позволяет создавать, тестировать и даже публиковать гибридные приложения с любого компьютера или мобильного устройства, без необходимости в его загрузке.

Преимущества:

* Telerik предоставляет плагины Visual Studio и Sublime Text для AppBuilder.
* AppBuilder предлагает быстрый способ импорта плагинов Cordova.
* Полноценная онлайн IDE.
* Легок в использовании и изучении/

Недостатки: Небольшое сообщество.

Нет идеального решения, каждая IDE или фреймворк имеют свои плюсы и минусы. Для очень простых приложений можно использовать PhoneGap пока отзывчивость не станет ключевым критерием. А для более серьезной разработки лучше использовать Xamarin или Android Studio. Для каждого человека главным критерием при выборе нужного инструментария будут выступать используемая операционная система, язык программирования и те платформы, которые он планирует развивать.

2. Практическая часть

2.1 Пример разработки мобильного приложения в Android Studio

В первую очередь для создания приложений должен быть установлен пакет JDK 8, который необходим для разработки на языке Java. JDK 8 можно найти на сайте компании Oracle [27].

Создадим проект в Android Studio. При создании проекта нам будет предложено несколько шаблонов для различных ситуаций, но самыми распространенными являются Basic Activity и Empty Activity. Это удобные шаблоны для старта для создания большинства приложений. Выберем шаблон Empty Activity, который нам предлагается по умолчанию. Назовем наше приложение My1App и выберем язык Java (рисунок 2).

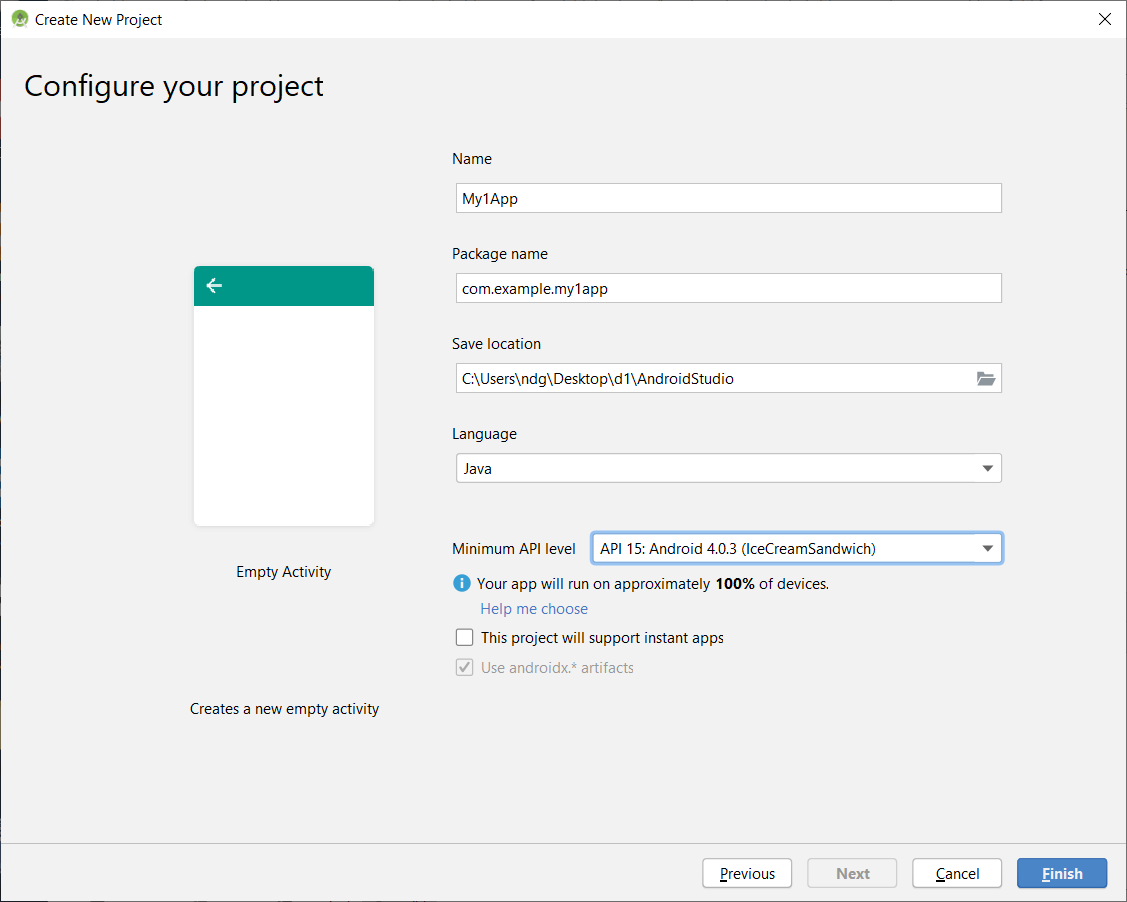


Рисунок 2 – Создание нового проекта в Android Studio

После создания проекта структура проекта Android отображается в следующем виде (рисунок 3).

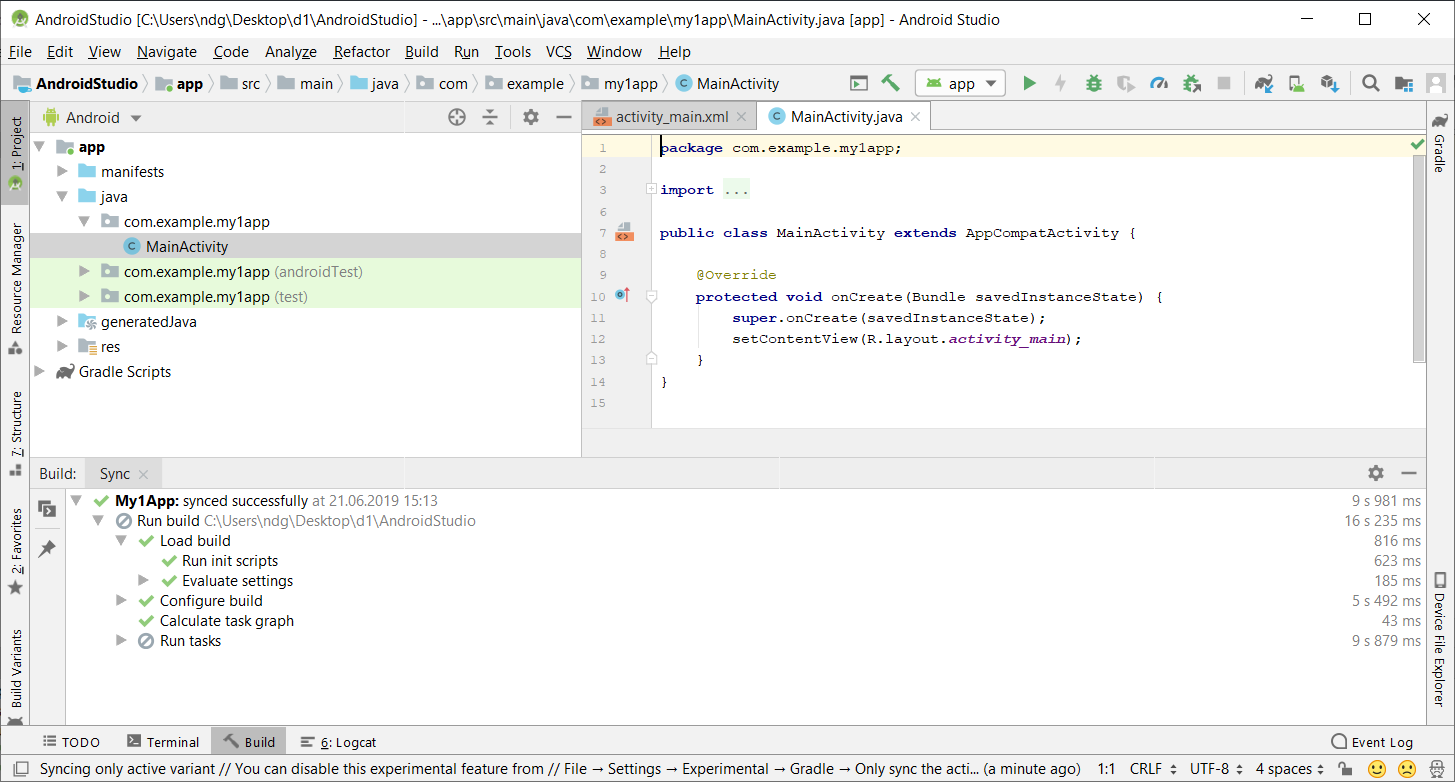


Рисунок 3 – Структура проекта

Проект Android может состоять из различных модулей [28]. По умолчанию, когда мы создаем проект, создается один модуль – app. Он имеет четыре подпапки:

1. manifests: хранит файл манифеста AndroidManifest.xml, который определяет конфигурацию;
2. java: хранит файлы кода на языке java, которые структурированы по отдельным пакетам;
3. res: содержит используемые в приложении ресурсы;
4. generatedJava: содержит сгенерированные исходные файлы.

Отдельный элемент Gradle Scripts содержит ряд скриптов gradle (как для модуля app или других возможных модулей, так и для всего проекта), которые используются при построении приложения.

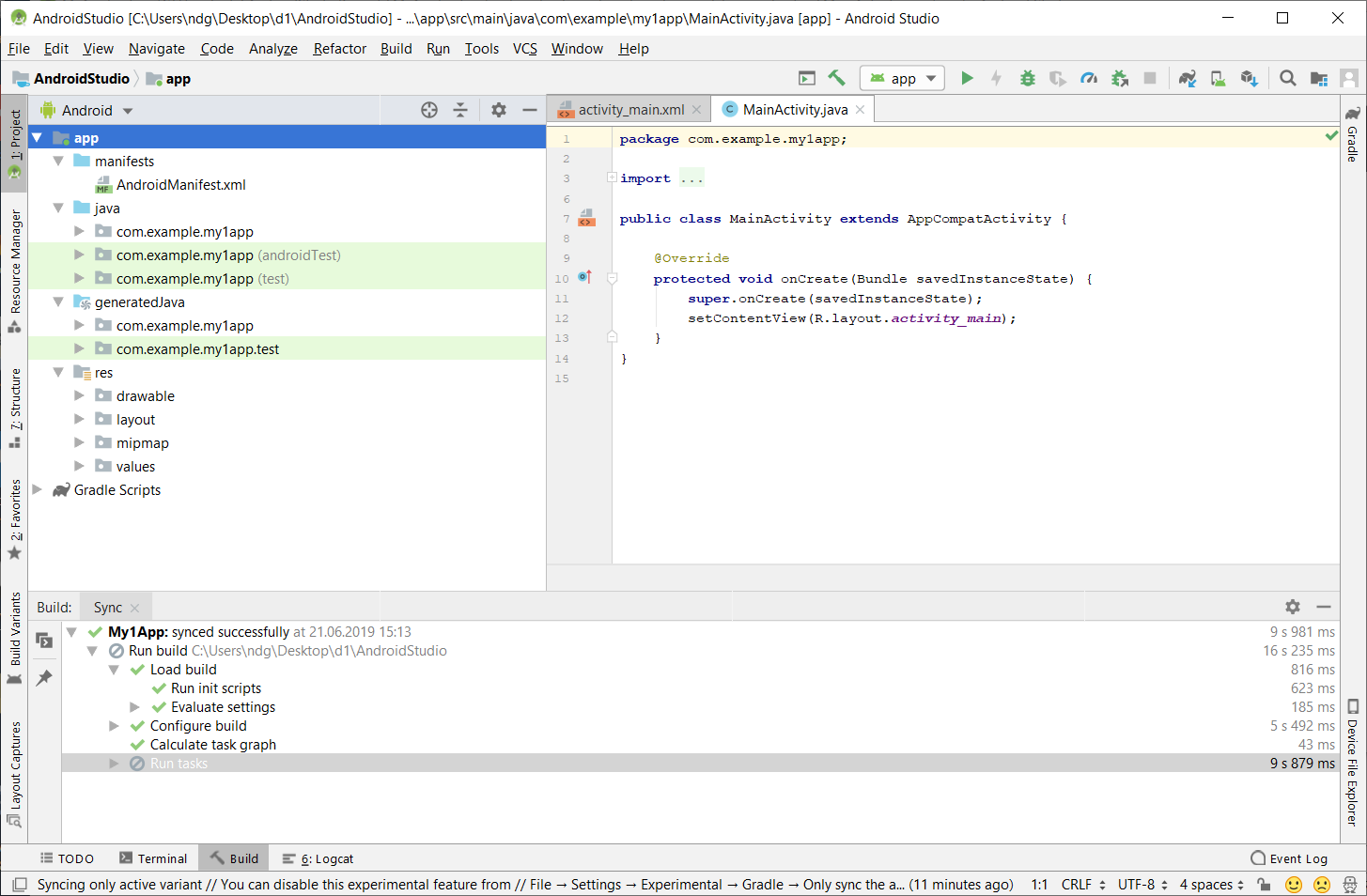


Рисунок 4 – Модули проекта

В студии по умолчанию должен быть открыт файл activity\_main.xml, который содержит определение графического интерфейса приложения – здесь размещаются компоненты. Стандартный проект уже содержит единственный размещенный компонент – TextView – с текстом "Hello world!". Соответственно, единственный функционал приложения – вывод этой текстовой строки на экране (рисунок 5).

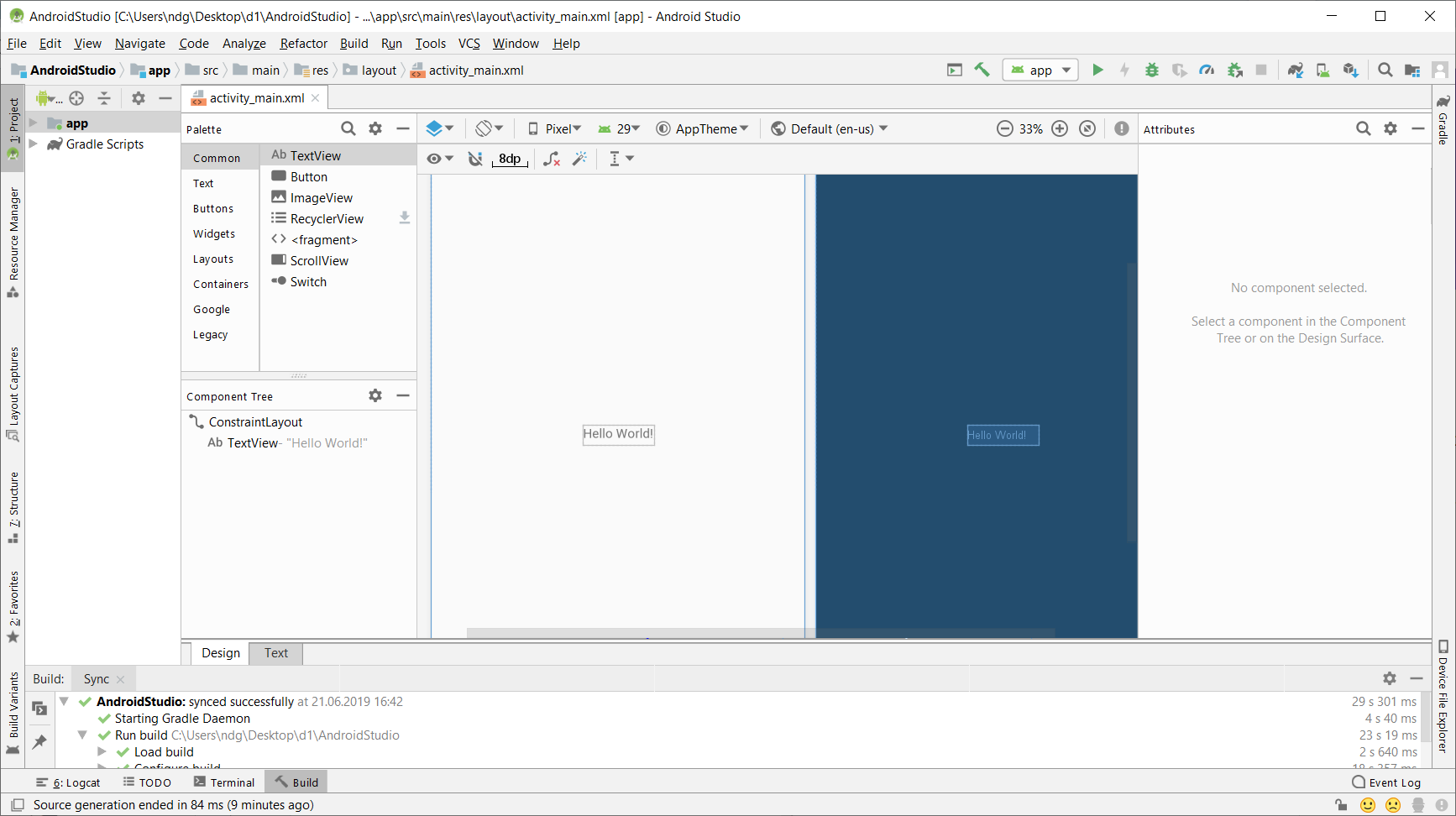


Рисунок 5 – Содержание проекта по умолчанию

Создадим новое приложение, которое будет представлять собой пополняемый список покупок, состоящий из наименования товара и его количества. Для этого заполним экран компонентами, просто перенося их на форму, указав при этом им нужный цвет и текстовое содержание. Таким образом, графическое содержание будет определено файлом item.xml (рисунок 6).

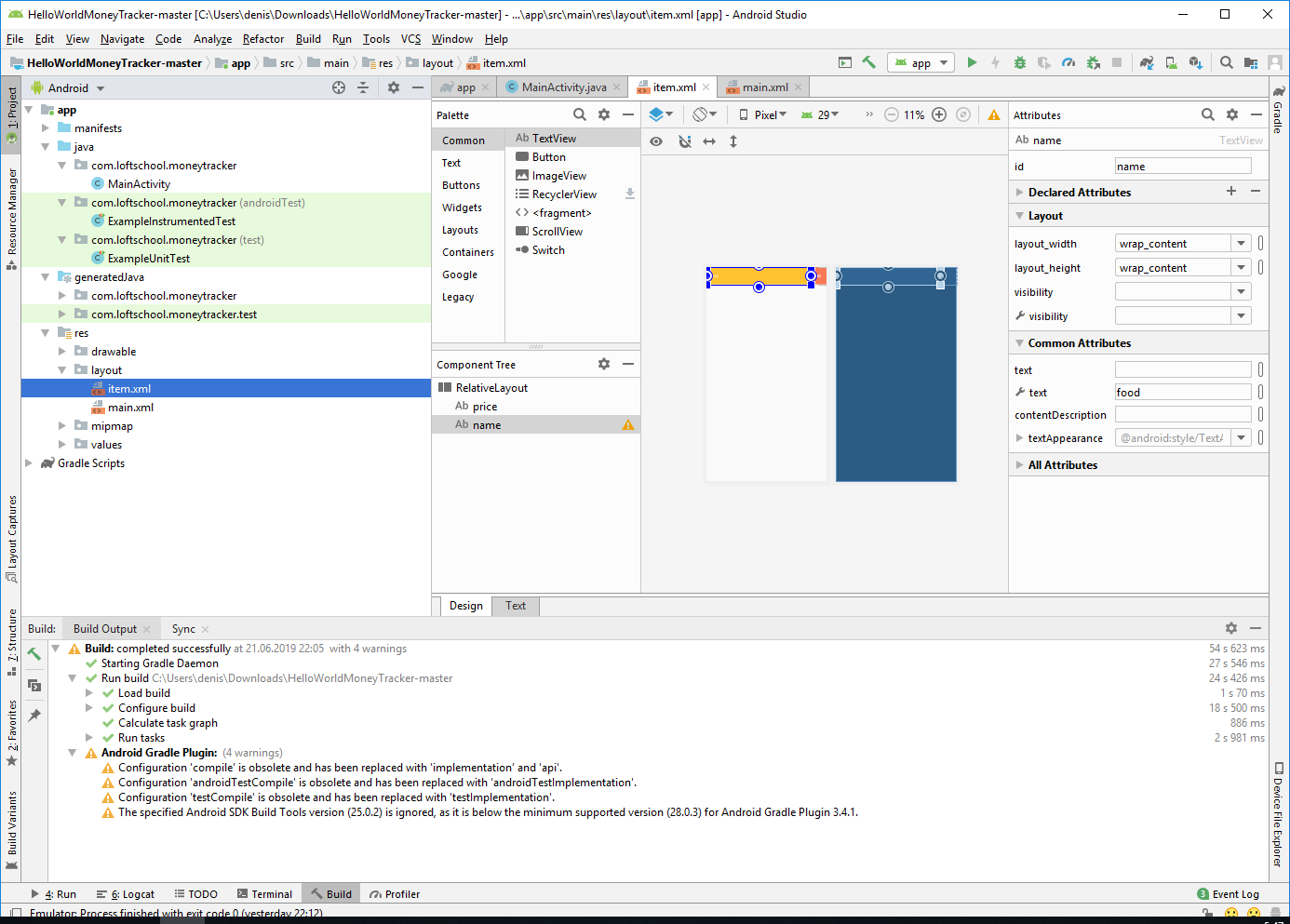


Рисунок 6 – Графическое содержание, определенное файлом item.xml

Графическое содержание после добавления элементов списка определено в файле main.xml (рисунок 7).

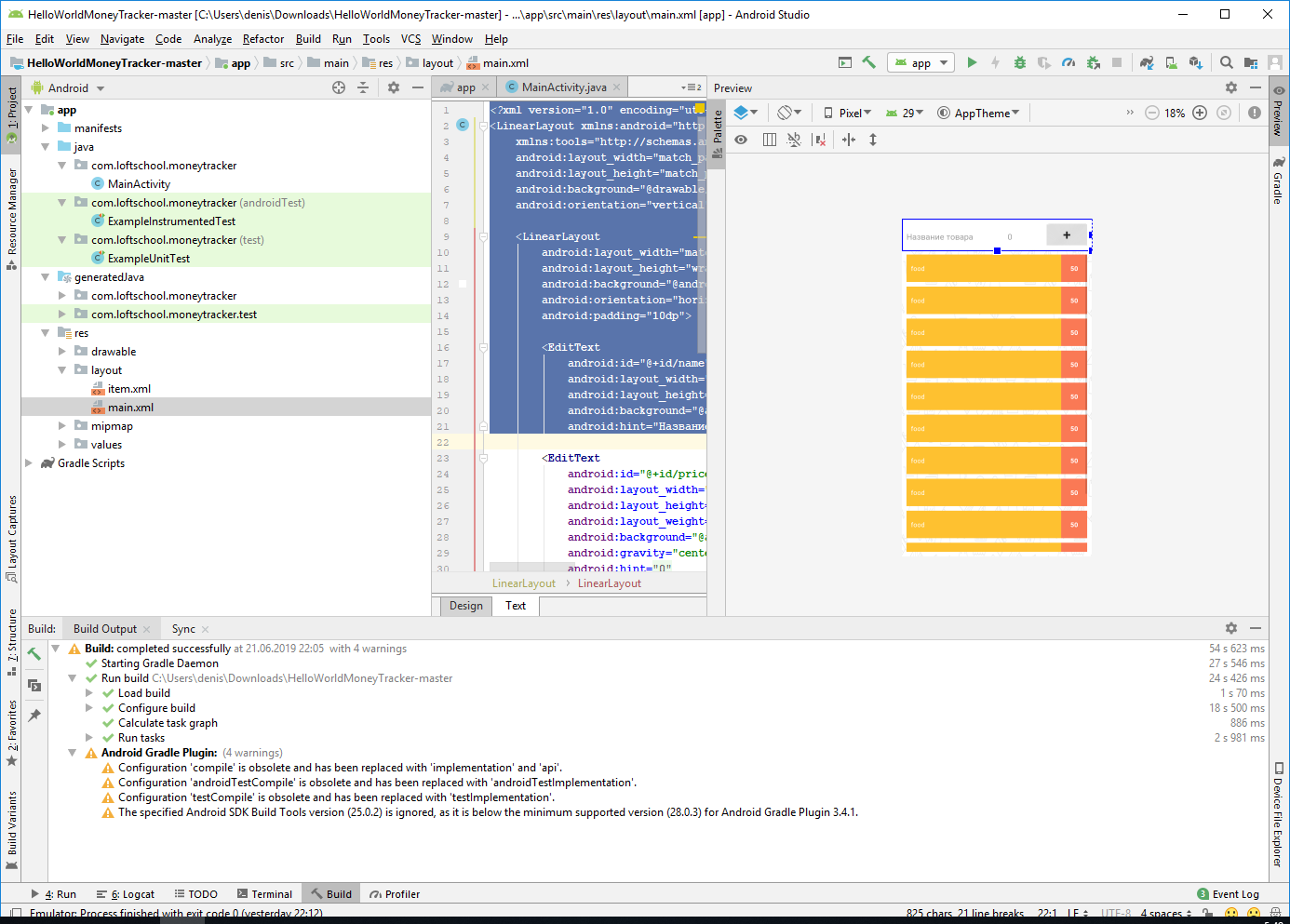


Рисунок 7 – Графическое содержание, определенное файлом main.xml

Логика работы приложения описывается в файле MainActivity.java

**package** com.loftschool.moneytracker;  
  
**import** android.os.Bundle;  
**import** android.support.annotation.NonNull;  
**import** android.support.annotation.Nullable;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.ListView;  
**import** android.widget.TextView;  
  
*/\*\*  
 \* Copyright (c) 2017.  
 \* All Rights Reserved.  
 \* <p>  
 \* Author: Dmitry Gordeev <netimen@gmail.com>  
 \* Date: 05.04.17  
 \*/***public class** MainActivity **extends** AppCompatActivity {  
 **static class** Item {  
 String **name**;  
 **int price**;  
  
 Item(String name, **int** price) {  
 **this**.**name** = name;  
 **this**.**price** = price;  
 }  
 }  
  
 @Override  
 **protected void** onCreate(@Nullable Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***main***);  
 **final** EditText name = (EditText) findViewById(R.id.***name***);  
 **final** EditText price = (EditText) findViewById(R.id.***price***);  
 **final** Button add = (Button) findViewById(R.id.***add***);  
 **final** ListView items = (ListView) findViewById(R.id.***items***);  
 **final** ItemsAdapter adapter = **new** ItemsAdapter();  
  
 items.setAdapter(adapter);  
 add.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 adapter.add(**new** Item(name.getText().toString(), Integer.*valueOf*(price.getText().toString())));  
 }  
 });  
 }  
  
 **private class** ItemsAdapter **extends** ArrayAdapter<Item> {  
 ItemsAdapter() {  
 **super**(MainActivity.**this**, R.layout.***item***);  
 }  
  
 @NonNull  
 @Override  
 **public** View getView(**int** position, @Nullable View convertView, @NonNull ViewGroup parent) {  
 **final** View view = getLayoutInflater().inflate(R.layout.***item***, **null**);  
 **final** Item item = getItem(position);  
 ((TextView) view.findViewById(R.id.***name***)).setText(item.**name**);  
 ((TextView) view.findViewById(R.id.***price***)).setText(String.*valueOf*(item.**price**));  
 **return** view;  
 }  
 }  
}

Запустим проект, нажав на зеленую стрелочку на панели инструментов. Начнется построение проекта. Данный процесс может занять некоторое время, после чего отобразится диалоговое окно для выбора устройства для запуска. Здесь мы можем выбрать подключенный к компьютеру гаджет, либо эмулятор. В нашем случае выберем эмулятор Galaxy Nexus с образом системы Oreo API level 26 (Android 8.0) (рисунок 8).

Выберем устройство и нажмем на кнопку OK. И после запуска мы увидим наше приложение на экране устройства (рисунок 9).

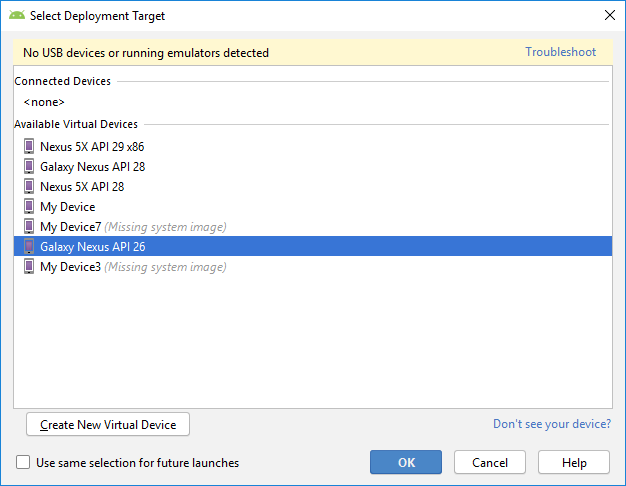


Рисунок 8 – Окно создания и выбора эмуляторов

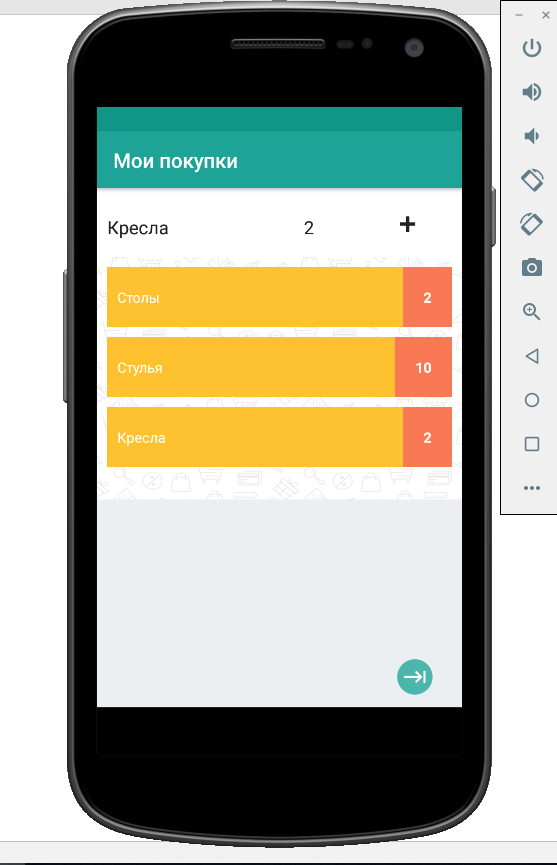


Рисунок 9 – Запуск приложения на эмуляторе

Теперь можно создать исполняемый файл-приложение для Android – apk-файл. Для этого необходимо сделать следующее.

1. Откроем окно Build Variants и напротив нашего модуля в поле Build Variant переключим режим сборки с debug на release.
2. В основном меню открываем Build → Generate Signed APK.
3. Открывается окно помощника генерации подписи для apk-файла. Все распространяемые приложения для Android должны иметь цифровую подпись автора, иначе приложение не установится на устройства пользователей. Сгенерированная подпись должна храниться в специальном файле-хранилище, расположенном на вашем компьютере, откуда потом она будет извлекаться для подписания apk-файлов. Для создания хранилища нажимаем кнопку Create New… и заполняем открывшиеся в окне поля:

* Путь для размещения файла хранилища
* Пароль и подтверждение для доступа к хранилищу
* Имя подписи, по которому она будет вызываться
* Пароль и подтверждение для доступа к подписи
* Срок действия подписи (по умолчанию 25 лет, оставляем без изменений)
* Хотя бы одно из полей сертификата. Обычно заполняют имя и фамилию, город и страну (RU) (рисунок 10).

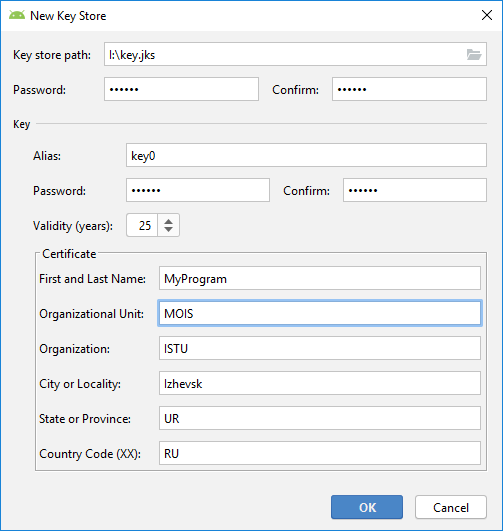


Рисунок 10 – Добавление авторских данных в окне New Key Store

1. Нажимаем OK. Окно закрывается, и мы видим, что все поля в предыдущем окне автоматически заполнились введёнными нами данными. Ставим галочку в поле Remember Password, чтобы каждый раз не набирать пароль, и нажимаем OK.
2. В следующем окне контролируем путь и имя apk-файла. По умолчанию оно равно имени модуля. Выбираем версию подписей – V1 (подпись Jar) и V2 (полная подпись APK) при генерации подписанного apk в AndroidStudio. Нажимаем Finish и ждём, посматривая в строку состояния.
3. Когда сборка проекта закончится, отобразится окошко с предложением открыть папку с полученным apk-файлом. Открываем её и видим наш файл.

Мы можем переименовать файл, сохранив его расширение, и выложить в Play Market или на любой сайт или сразу загрузить на мобильное устройство. После загрузки на телефон/планшет достаточно нажать на него, и с помощью стандартного установщика пакетов приложение будет установлено.

Проверим работоспособность apk-файла на программе эмуляторе для тестирования приложений Nox [30] или BlueStack, которые хорошо зарекомендовали себя как эмуляторы Android устройств для тестирования созданных приложений, игр и т.д. Для запуска приложения просто перетаскиваем созданный apk-файл в область экрана устройства, после чего начнется установка приложения (рисунок 11).

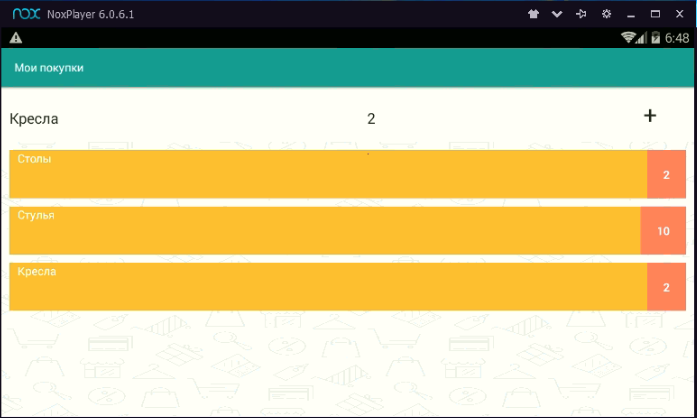


Рисунок 11 – Работа приложения в программе-эмуляторе Nox

2.2 Примеры разработки мобильного приложения с использованием

Visual Studio 2019 и Xamarin

Создадим проект в Microsoft Visual Studio Community 2019, предварительно доустановив Xamarin с использованием инсталлятора Visual Studio Installer [29]. Также потребуется установка в системе комплекта разработчика Java Development Kit (JDK) и платформа Android SDK.

При создании проекта выбираем пункт Мобильное приложение (Xamarin.Forms) (рисунок 12).

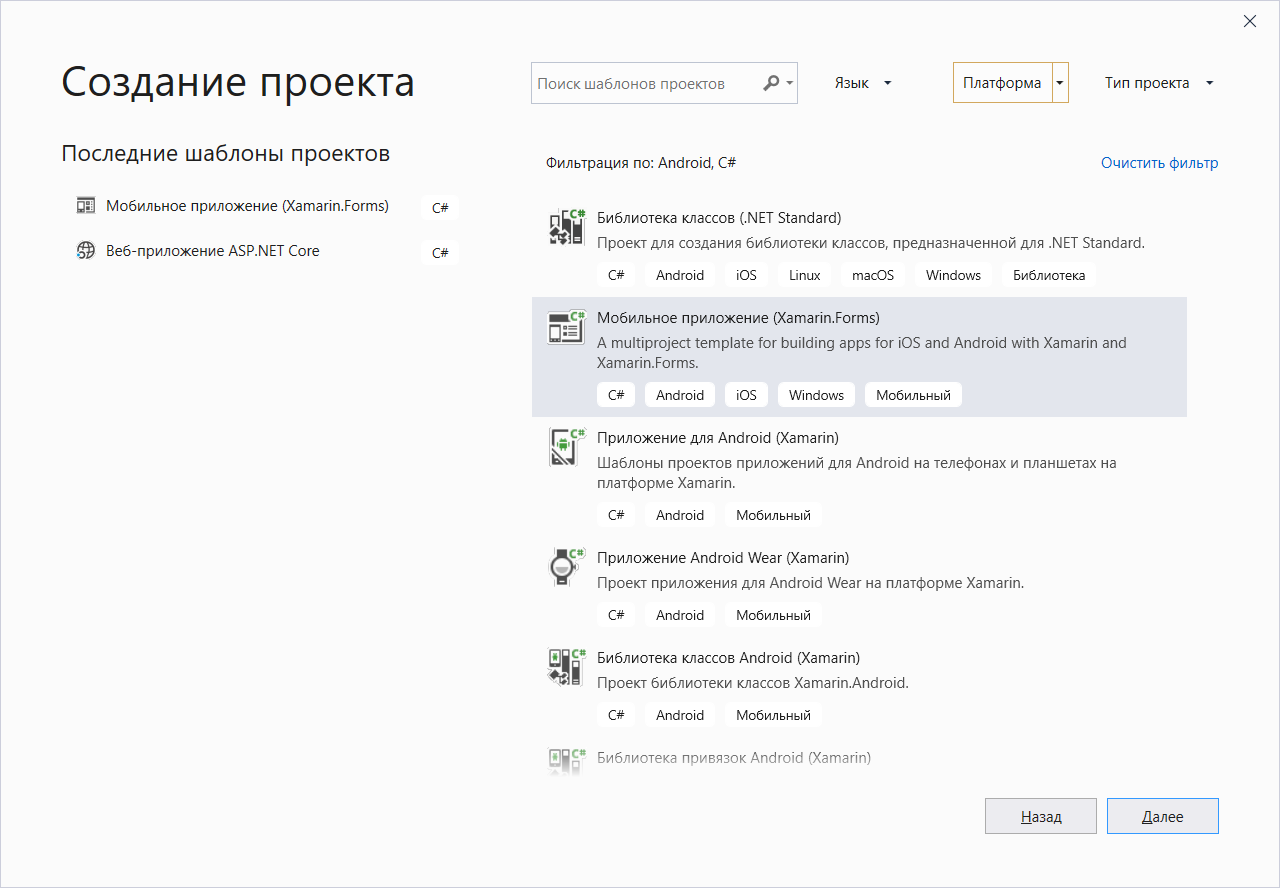


Рисунок 12 – Создание проекта в MS Visual Studio

Назовем приложение MyApp. После этого на выбор будет предложено несколько шаблонов приложения. Выберем «Пустой», а в качестве целевой платформы оставим только Android (рисунок 13).

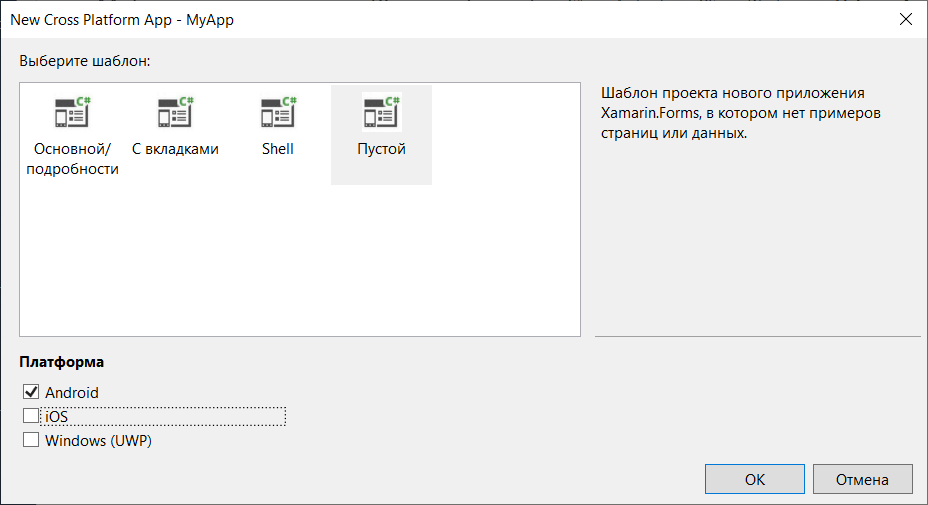


Рисунок 13 – Шаблоны проектов

И Visual Studio сгенерирует новое решение, в которой будет два проекта:

1. MyApp: главный проект библиотеки, которая и будет содержать всю основную логику приложения;
2. MyApp.Android: проект для Android.

В главном проекте уже будет четыре основных файла:

* App.xaml: файл, который определяет ресурсы, общие для всего приложения;
* App.xaml.cs: файл с кодом C#, с которого начинается выполнение приложения;
* MainPage.xaml: файл с визуальным интерфейсом для единственной страницы MainPage в виде xaml;
* MainPage.xaml.cs: файл, который содержит логику MainPage на языке C#

То есть имеются файлы, которые определяют логику всего приложения - App.xaml и App.xaml.cs и файлы с интерфейсом и логикой единственной страницы приложения - MainPage.

Фактически проект по умолчанию уже представляет простейшее приложение, которое просто выводит на экран некоторый текст.

При добавлении новой страницы XAML в проект также одновременно добавляется файл кода C# (в нашем случае это файл MainPage.xaml.cs), где, как предполагается, должна находится логика приложения, связанная с разметкой из файла XAML. Файлы XAML позволяют нам определить интерфейс окна, но для создания логики приложения, например, для определения обработчиков событий элементов управления, нам все равно придется воспользоваться кодом C#.

Пусть наше приложение представляет собой простейший альбом, содержащий несколько картинок. Наше приложение будет содержать кнопку, по нажатию на которую будут пролистываться картинки.

Выберем MainPage.xaml-файл, после чего раскроется его содержимое (рисунок 14), в котором, по умолчанию, описывается вывод строки, помещаемой в контейнер StackLayout, располагаемой по центру экрана, и содержащей надпись «Welcome to Xamarin.Forms!».

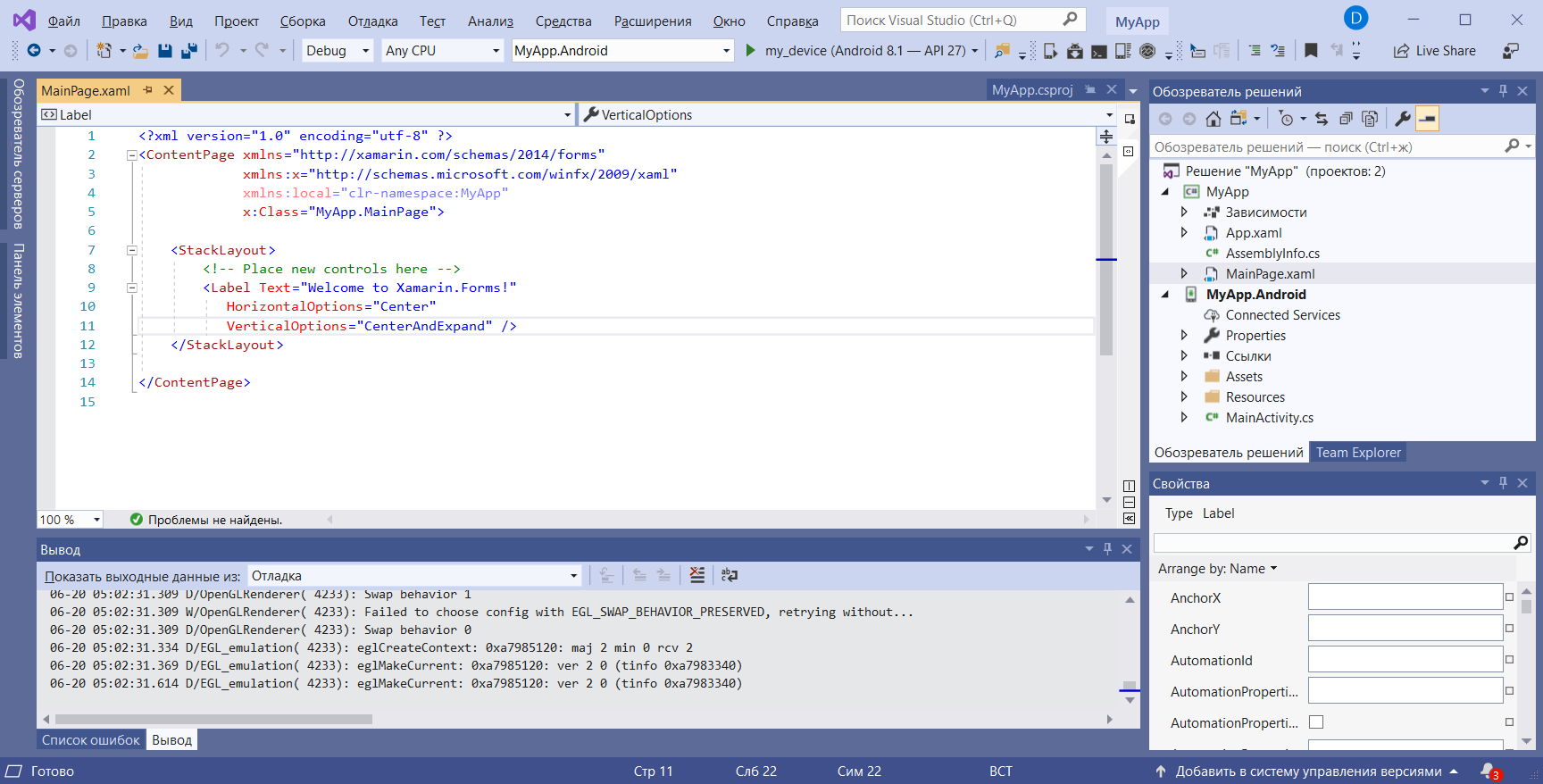


Рисунок 14 – Содержимое MainPage.xaml-файла

В нашем случае xaml-код писать не понадобится. Поэтому убираем контейнер StackLayout вместе с его содержимым. В результате содержимое MainPage.xaml-файла останется следующим

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

xmlns:local="clr-namespace:MyApp"

x:Class="MyApp.MainPage">

</ContentPage>

Теперь перейдем к файлу MainPage.xaml.cs и пропишем там логику нашей MainPage-страницы. По умолчанию, здесь как таковой, логики нет, кроме вызова в конструкторе InitializeComponent(). Через этот вызов на странице формируется интерфейс, определенный в файле MainPage.xaml. Поместим в разделе public MainPage() контейнер stackLayout, в который поместим кнопку.

public MainPage()

{

InitializeComponent();

StackLayout stackLayout = new StackLayout();

Button button = new Button

{

Text = "Нажми!",

FontSize = Device.GetNamedSize(NamedSize.Large, typeof(Button)),

BorderWidth = 1,

HorizontalOptions = LayoutOptions.Center,

VerticalOptions = LayoutOptions.CenterAndExpand

};

button.Clicked += OnButtonClicked;

stackLayout.Children.Add(button);

this.Content = stackLayout;

}

Здесь прописывается первоначальный текст “Нажми!” на кнопке, определяется ее расположение и создается событие OnButtonClicked, в котором описывается действие при нажатии на кнопку.

Осталось описать событие OnButtonClicked. Создадим переменную imageindex, отвечающую за номер нажатия и номер картинки. И в зависимости от ее значения будет менять текстовое содержание и изображение на кнопке.

public static int imageindex;

private void OnButtonClicked(object sender, System.EventArgs e)

{

Button button = (Button)sender;

if (button.Text == "Нажми!")

{

string pic = "egek" + Convert.ToString(MainPage.imageindex) + ".jpg";

button.ImageSource = pic;

imageindex++;

if (imageindex == 4)

{

imageindex = 1; button.Text = "Просмотрено!";

}

}

else

{

button.Text = "Нажми!";

button.ImageSource = "";

}

}

Осталось перед выводом изображения добавить их в соответствующий проект для Android. Для этого в обозревателе решений перейдем в проект MyApp.Android и перетащим файлы изображений в папку Resources/Drawable (рисунок 15).

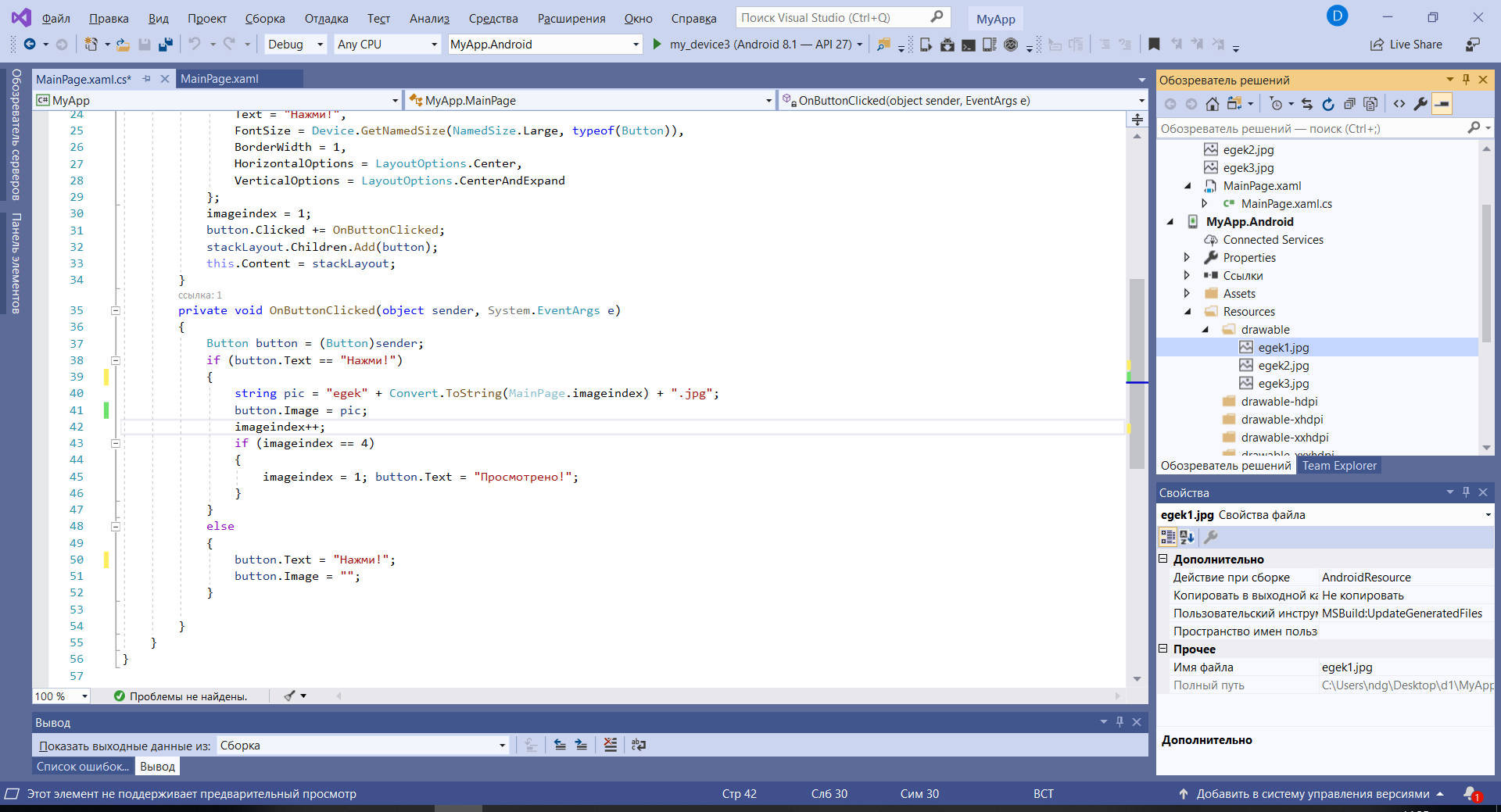


Рисунок 15 – Добавление файлов изображений

Запустим приложение на проверку. Для этого необходимо выбрать соответствующий эмулятор, на котором будет запущено компилируемое приложение. Нужный нам эмулятор надо создать (должен быть установлен Android SDK). Заходим в Android Device Manager (справа от кнопки запуска находим «Открыть Android Device Manager»), после чего запустится окно с возможностью добавления виртуальных Android-устройств. Нажмем «Создать» и выбираем предпочтительные настройки устройства (рисунки 16-17).

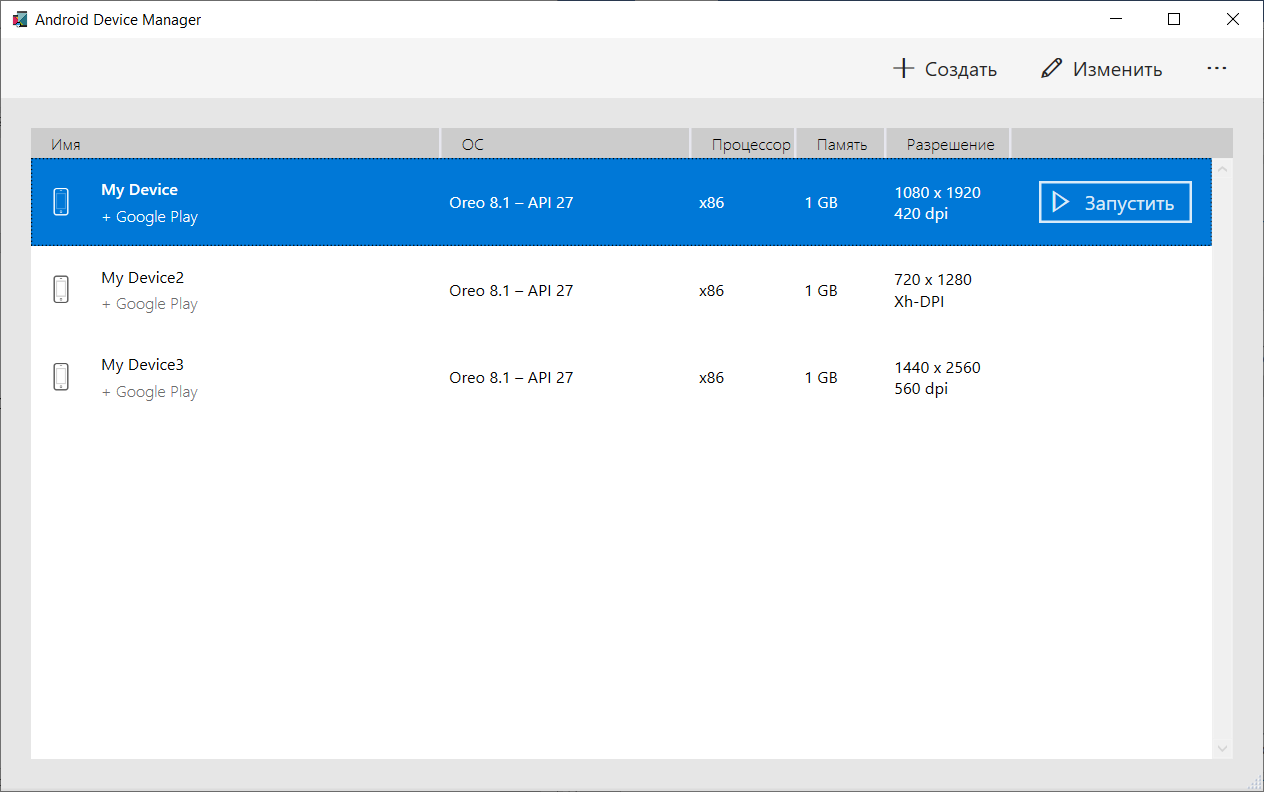


Рисунок 16 – Создание виртуального устройства

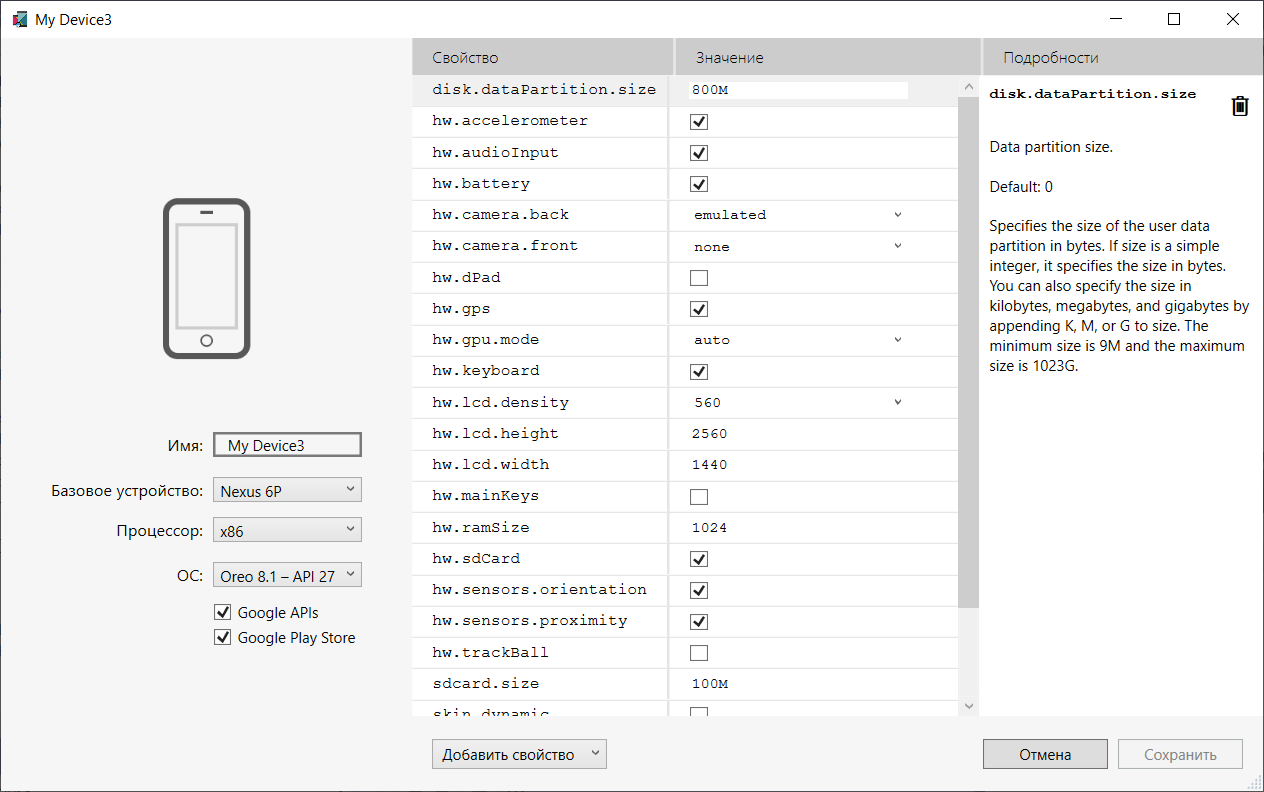


Рисунок 17 – Настройка параметров виртуального устройства

Дожидаемся, когда наш эмулятор появится в списке доступных для компиляции устройств, после чего выбираем его и нажимаем на кнопку с зеленым треугольником для начала отладки.

Если все пошло удачно, то отобразится эмулятор устройства с нашим запущенным приложением. При нажатии на кнопку мы сможем просматривать загруженные в альбоме картинки (рисунок 18).

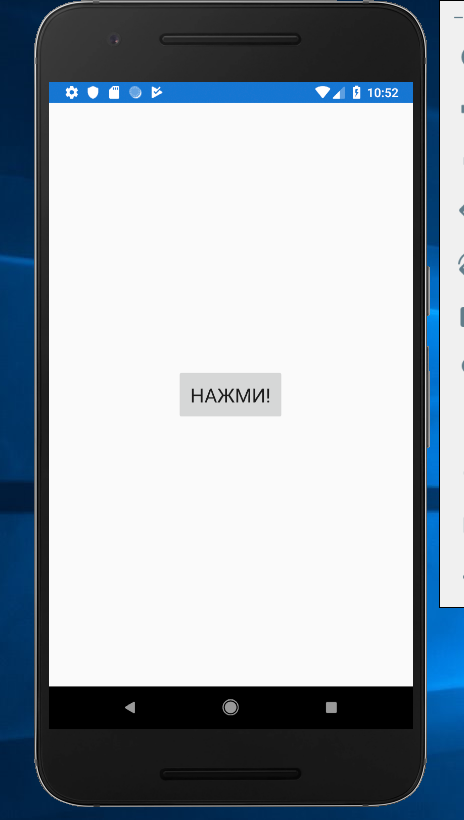
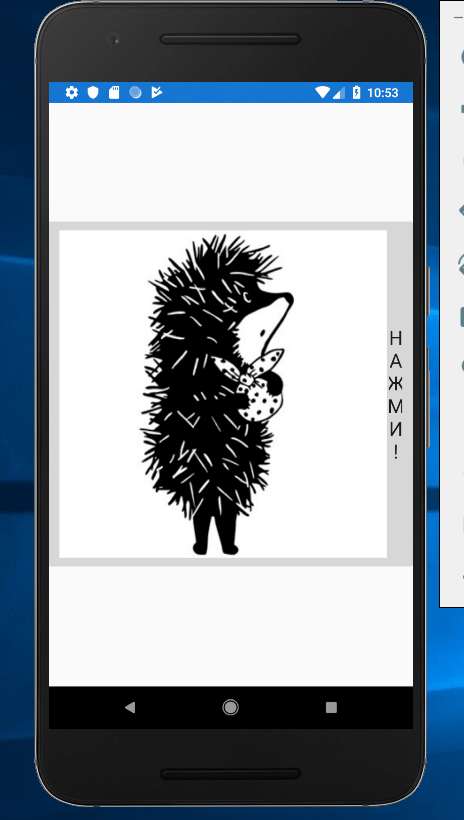
 

Рисунок 18 – Запуск и работа приложения на виртуальном устройстве

Приложение успешно создано и проверено на встроенном эмуляторе. Теперь можно создать «подписанный» apk-файл – исполняемый файл-приложение для Android. Для этого меняем конфигурацию решения c Debug на Release, выбираем платформу MyApp.Android и ПКМ (правой кнопкой мыши) кликаем на Свойства. Отобразится область параметров Android-приложения. Изменим свойства на вкладках Манифест Android (рисунок 19)

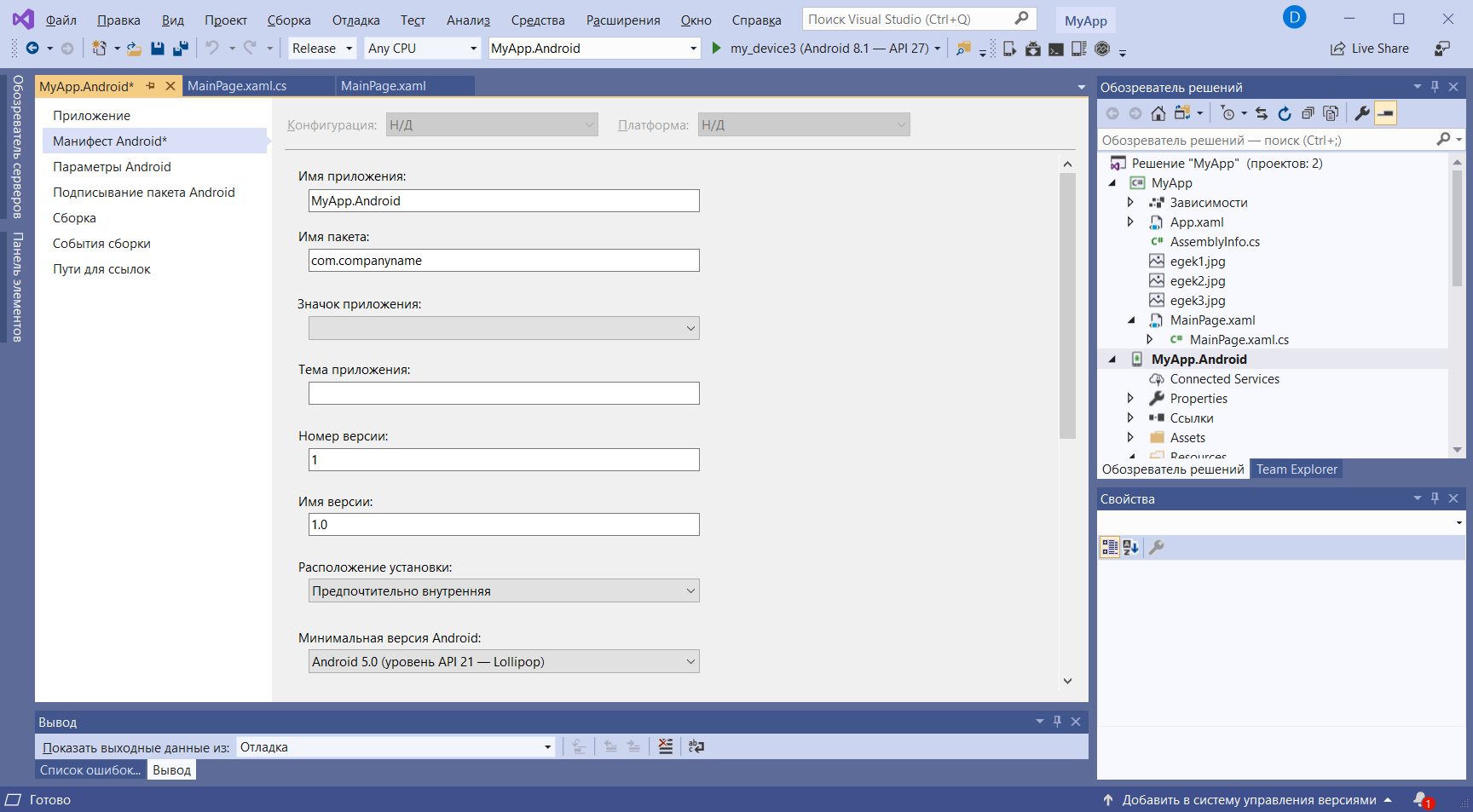


Рисунок 19 – Настройка параметров Android-приложения

и Параметры Android, где на закладке Связывание устанавливаем свойство Нет.

Далее снова заходим в MyApp.Android и ПКМ кликаем сначала на Очистить, где сначала выбираем «Очистить», а потом «Пересобрать». Дожидаемся успешного завершения операции. Теперь необходимо архивировать проект в apk-файл, для чего там же, в MyApp.Android, ПКМ выбираем пункт Архивация (рисунок 20).

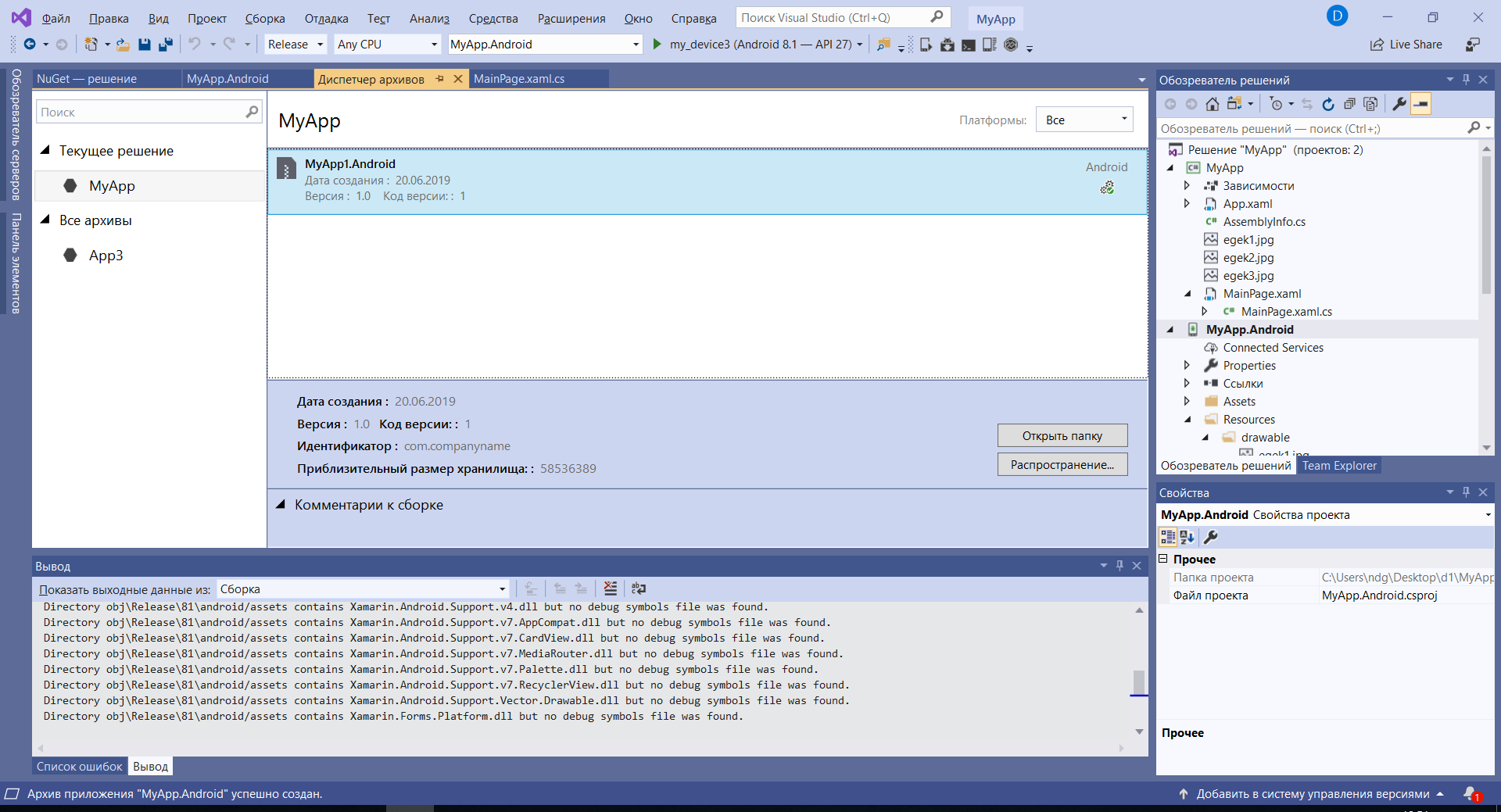


Рисунок 20 – Архивация приложения

В этом окне необходимо нажать на Распространение, потом на Специальный. Добавляем авторские данные через «+» (рисунки 21-22).

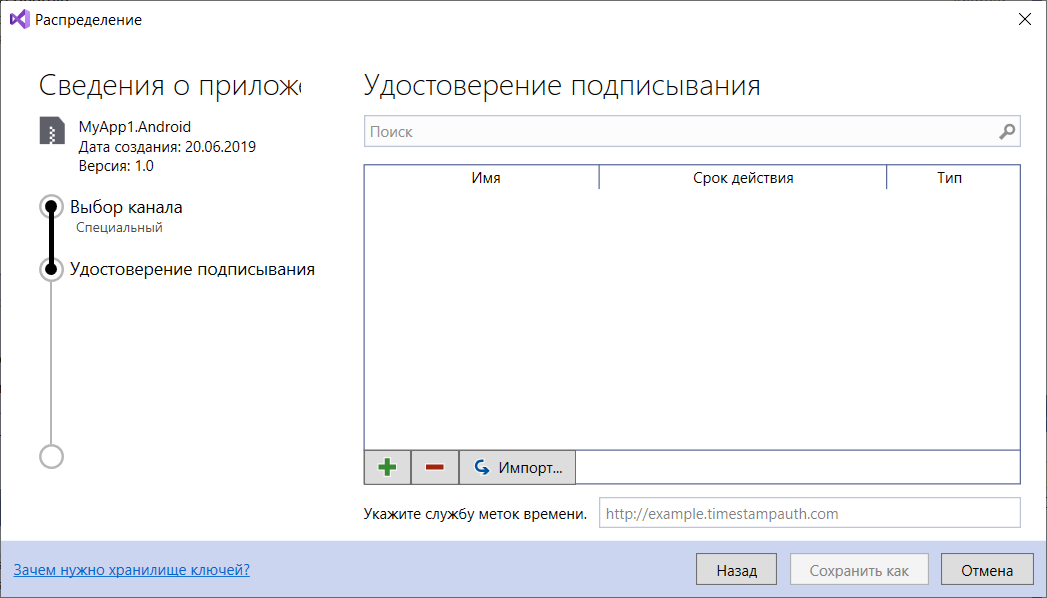


Рисунок 21 – Создание удостоверения

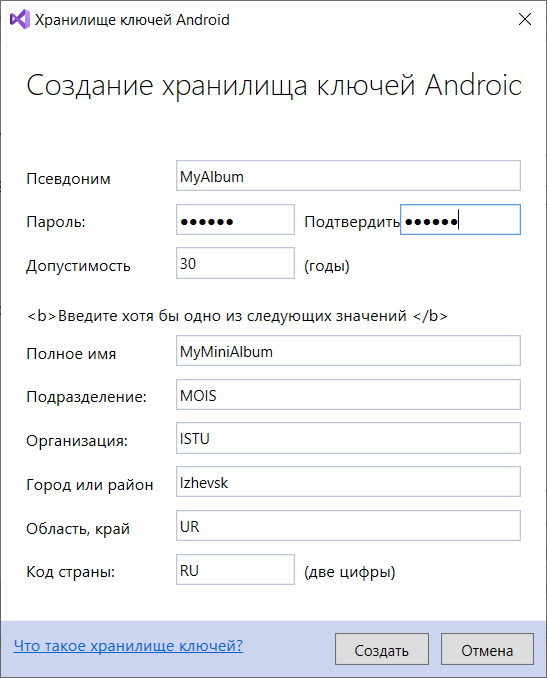


Рисунок 22 – Добавление авторских данных

Сохраняем через «Сохранить как» и указываем место расположения сохраняемого файла. В итого получен полный apk-файл, который можно запустить на реальном андроид устройстве.

Проверим работоспособность данного файла. Для этого можно использовать свой собственный телефон. Пусть это будет Xiaomi Redmi 4 с версией ОС Android 6.0 Marshmallow. Просто перекачиваем файл на наше устройство и разрешаем запуск приложения. В итоге получаем следующее (рисунок 23).

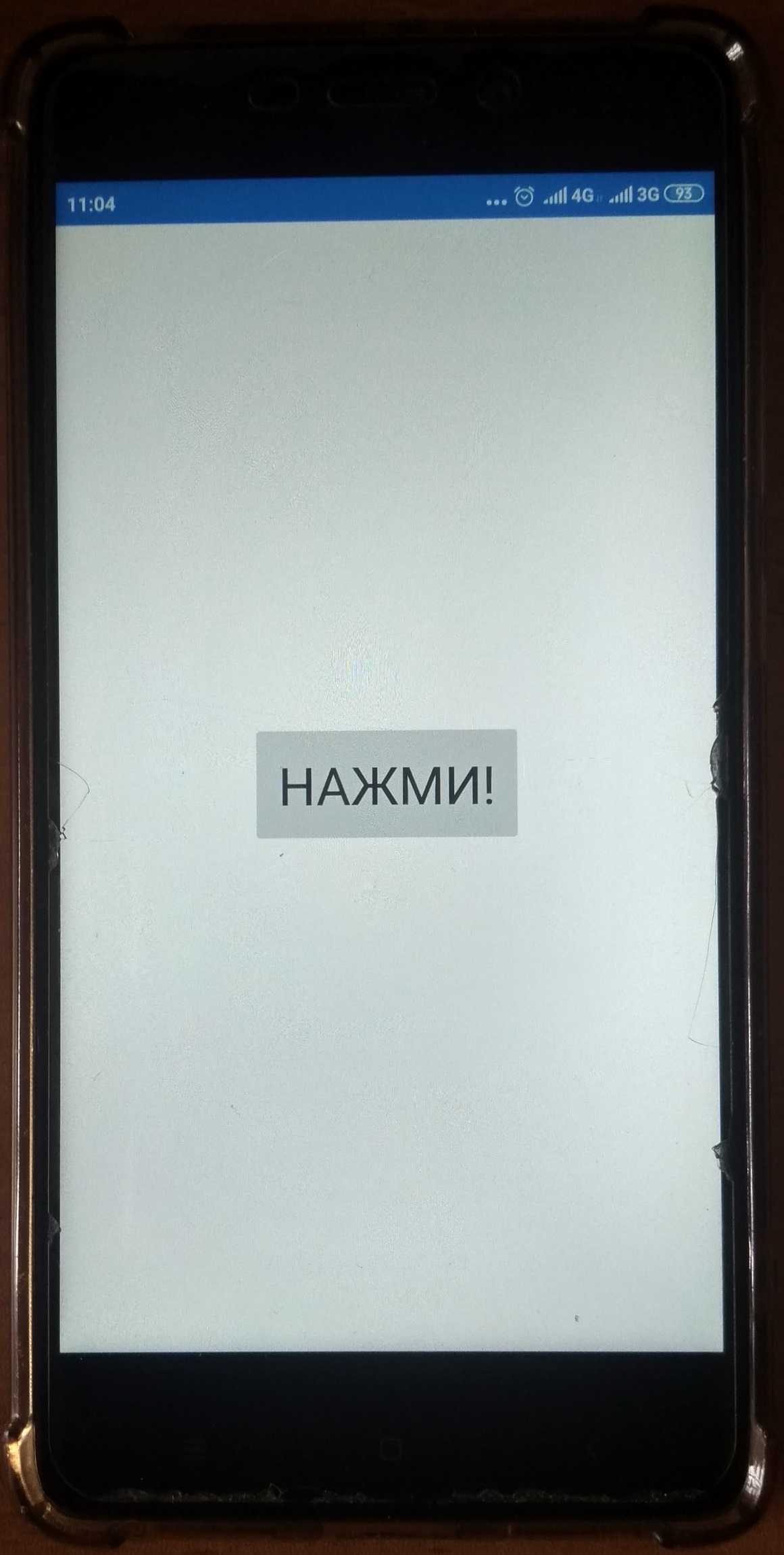


Рисунок 23 – Работа приложения на реальном устройстве

Либо через утилиту типа Nox или BlueStack. Перетаскиваем созданный apk-файл в область экрана устройства, после чего начнется установка приложения (рисунок 24).



Рисунок 24 – Работа приложения с программе-эмуляторе

Возьмем пример посложнее – создание часов. Здесь при написании приложения мы будет использовать код на обоих языках в Xamarin – С# и xaml. Так содержание файла MainPage.xaml будет следующим

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

xmlns:local="clr-namespace:Clock"

x:Class="Clock.MainPage">

<AbsoluteLayout x:Name="absoluteLayout"

SizeChanged="OnAbsoluteLayoutSizeChanged">

<BoxView x:Name="hourHand"

Color="Black" />

<BoxView x:Name="minuteHand"

Color="Black" />

<BoxView x:Name="secondHand"

Color="Black" />

</AbsoluteLayout>

</ContentPage>

Здесь используется AbsoluteLayout – контейнер компоновки с абсолютным позиционированием, т. е. нужно явно указывать X и Y координаты вложенных элементов. Для добавления элементарной графики используется объект BoxView, который отображает простой прямоугольник с заданной шириной, высотой и цветом. Здесь этих объектов 3 штуки, отвечающих, соответственно за отображение часовой, минутной и секундной стрелок.

Логика приложения описывается в файле MainPage.xaml.cs. Для него написан следующий код

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

namespace Clock

{

// Learn more about making custom code visible in the Xamarin.Forms previewer

// by visiting https://aka.ms/xamarinforms-previewer

[DesignTimeVisible(true)]

public partial class MainPage : ContentPage

{

struct HandParams

{

public HandParams(double width, double height, double offset) : this()

{

Width = width;

Height = height;

Offset = offset;

}

public double Width { private set; get; } // fraction of radius

public double Height { private set; get; } // ditto

public double Offset { private set; get; } // relative to center pivot

}

static readonly HandParams secondParams = new HandParams(0.02, 1.1, 0.85);

static readonly HandParams minuteParams = new HandParams(0.05, 0.8, 0.9);

static readonly HandParams hourParams = new HandParams(0.125, 0.65, 0.9);

BoxView[] tickMarks = new BoxView[60];

public MainPage()

{

InitializeComponent();

// Create the tick marks (to be sized and positioned later).

for (int i = 0; i < tickMarks.Length; i++)

{

tickMarks[i] = new BoxView { Color = Color.Black };

absoluteLayout.Children.Add(tickMarks[i]);

}

Device.StartTimer(TimeSpan.FromSeconds(1.0 / 60), OnTimerTick);

}

void OnAbsoluteLayoutSizeChanged(object sender, EventArgs args)

{

// Get the center and radius of the AbsoluteLayout.

Point center = new Point(absoluteLayout.Width / 2, absoluteLayout.Height / 2);

double radius = 0.45 \* Math.Min(absoluteLayout.Width, absoluteLayout.Height);

// Position, size, and rotate the 60 tick marks.

for (int index = 0; index < tickMarks.Length; index++)

{

double size = radius / (index % 5 == 0 ? 15 : 30);

double radians = index \* 2 \* Math.PI / tickMarks.Length;

double x = center.X + radius \* Math.Sin(radians) - size / 2;

double y = center.Y - radius \* Math.Cos(radians) - size / 2;

AbsoluteLayout.SetLayoutBounds(tickMarks[index], new Rectangle(x, y, size, size));

tickMarks[index].Rotation = 180 \* radians / Math.PI;

}

// Position and size the three hands.

LayoutHand(secondHand, secondParams, center, radius);

LayoutHand(minuteHand, minuteParams, center, radius);

LayoutHand(hourHand, hourParams, center, radius);

}

void LayoutHand(BoxView boxView, HandParams handParams, Point center, double radius)

{

double width = handParams.Width \* radius;

double height = handParams.Height \* radius;

double offset = handParams.Offset;

AbsoluteLayout.SetLayoutBounds(boxView,

new Rectangle(center.X - 0.5 \* width,

center.Y - offset \* height,

width, height));

// Set the AnchorY property for rotations.

boxView.AnchorY = handParams.Offset;

}

bool OnTimerTick()

{

// Set rotation angles for hour and minute hands.

DateTime dateTime = DateTime.Now;

hourHand.Rotation = 30 \* (dateTime.Hour % 12) + 0.5 \* dateTime.Minute;

minuteHand.Rotation = 6 \* dateTime.Minute + 0.1 \* dateTime.Second;

// Do an animation for the second hand.

double t = dateTime.Millisecond / 1000.0;

if (t < 0.5)

{

t = 0.5 \* Easing.SpringIn.Ease(t / 0.5);

}

else

{

t = 0.5 \* (1 + Easing.SpringOut.Ease((t - 0.5) / 0.5));

}

secondHand.Rotation = 6 \* (dateTime.Second + t);

return true;

}

}

}

Коротко о содержании данного файла. Здесь структурой HandParams задаются относительные размеры каждой из стрелок по отношению к размеру самих часов. В обработчике OnAbsoluteLayoutSizeChanged определяется положение центра и радиуса контейнера AbsoluteLayout, задаются положение, размер и поворот 60 объектов BoxView, отвечающих за линии (деления) для указания минут и секунд. Обработчик LayoutHand вызывается с указанными размерами и расположением для каждой из стрелок часов. Для изменения положение стрелок используется обработчик тика таймера OnTimerTick.

Запустив наше приложение во встроенном эмуляторе, получим следующий результат (рисунок 25).

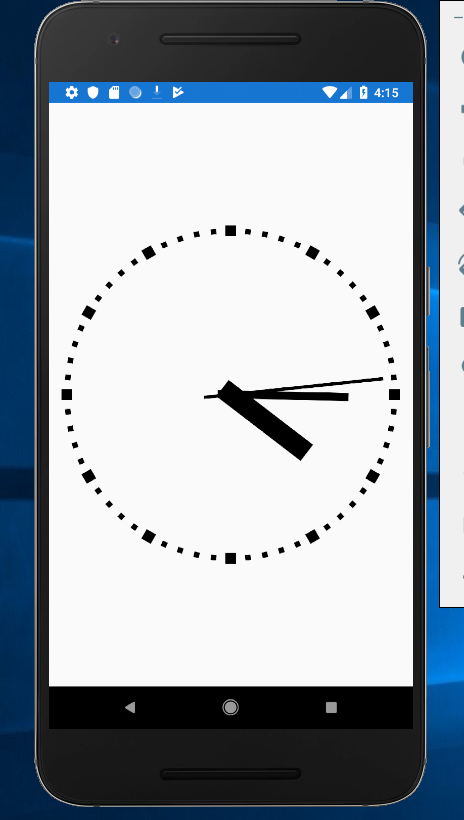


Рисунок 25 – Запуск приложения на эмуляторе

Заключение

1. Проведен обзор фреймворков и интегрированных сред разработки мобильных приложений. Данный обзор показал достаточное разнообразие инструментов разработки мобильных приложений, включая нативные и веб-приложения.
2. При выборе инструмента программирования под Android немаловажным является знание используемого языка. Официальным языком программирования для Android является Java, но существует возможность разрабатывать программы также на C/C++ (с помощью Native Development Kit), Basic (с помощью Simple), C# и с использованием других языков
3. Интегрированная среда разработки Android Studio является наиболее предпочтительным вариантом при выборе инструментария для разработки мобильных приложений. Данный среда содержит полный набор инструментов для создания готового Android приложения.
4. Для тех, кто владеет современным и очень удобным языком C#, хорошей альтернативой Android Studio является Microsoft Visual Studio с установленным фреймворком Xamarin. Преимуществом данной среды является ее кроссплатформенность, недостатком – большой размер готовых apk-файлов (минимальный размер больше 5 Мбайт).
5. Приведены примеры создания мобильных приложений с использованием Android Studio и MS Visual Studio+Xamarin, Проведен запуск готовых apk-файлов в программе тестирования мобильных приложений Nox и на реальном телефоне.

Литература

Основная литература

1. Введение в разработку приложений для ОС Android [Электронный ресурс] / Ю. В. Березовская, О. А. Юфрякова, В. Г. Вологдина [и др.]. – 2-е изд. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 433 c. – Режим доступа: http://www.iprbookshop.ru/73669.html.
2. Семакова А. Введение в разработку приложений для смартфонов на ОС Android [Электронный ресурс]/ Семакова А. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 102 c. – Режим доступа: http://www.iprbookshop.ru/73670.html.
3. Филлипс, Б. Android. Программирование для профессионалов. 3-е изд. / Б. Филлипс, К. Стюарт, К. Марсикано. – СПб.: Питер, 2017. – 688 с.
4. Дейтел, П. Android для разработчиков. 3-е изд. / П. Дейтел, Х. Дейтел, А. Уолд. – СПб.: Питер, 2016. – 512 с.
5. Гриффитс, Дон. Head First. Программирование для Android / Дон Гриффитс, Дэвид Гриффитс. – СПб.: Питер, 2016. – 704 с.

Дополнительная литература

1. Медникс, З. Программирование под Android. 2-е изд. / З. Медникс, Л. Дорнин, Б. Мик, М. Накамура. – СПб.: Питер, 2013. – 560 с.
2. Коматинени, Сатия. Android 4 для профессионалов. Создание приложений для планшетных компьютеров и смартфонов: Пер. с англ. / Сатия Коматинени, Дэйв Маклин. – М.: ООО «И.Д. Вильямс», 2012. – 880 с.
3. Хвощев, С.В. Основы программирования в Delphi для ОС Android [Электронный ресурс]/ Хвощев С.В. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 85 c. – Режим доступа: http://www.iprbookshop.ru/73694.html.

Интернет-ресурсы

1. ТОП 10 языков программирования для Android / редакция techrocks.ru. – Текст: электронный //techrocks.ru: [сайт]. – 2018. – 22 авг. – URL: techrocks.ru/2018/08/22/top-10-programming-languages-for-android/ (дата обращения: 16.06.2019).
2. Мобильная разработка: выбираем инструментарий. Silo, black box или cross-platform? / Юрий Язев. – Текст: электронный // xakep.ru: [сайт]. – 2017. – 08 март. – URL: https://xakep.ru/2017/03/08/mobile-development-tools/ (дата обращения: 16.06.2019).
3. Лучшие инструменты для разработки приложений под Android / Перевод статьи Adam Sinicki: Best Android developer tools. – Текст: электронный // nuancesprog.ru: [сайт]. – 2018. – 09 нояб. – URL: https://nuancesprog.ru/p/2414/ (дата обращения: 16.06.2019).
4. Что такое операционная система/ Evgen2sat. – Текст: электронный //guidepc.ru: [сайт]. – 2018. – 18 авг. – URL: https://guidepc.ru/articles/chto-takoe-operatsionnaya-sistema/ (дата обращения: 17.06.2019).
5. Архитектура операционной системы Андроид. – Текст: электронный //pr0andr0id.blogspot.com: [сайт]. – 2014. – 21 февр. – URL: http://pr0andr0id.blogspot.com/2014/02/blog-post\_21.html (дата обращения: 17.06.2019).
6. Фреймворк – важный инструмент программиста. Обзор HTML/CSS, PHP и Python-фреймворков. – Текст: электронный // fructcode.com: [сайт]. – 2018. – URL: https://fructcode.com/ru/blog/features-of-popular-frameworks-html-css-php-and-python-frameworks/ (дата обращения: 17.06.2019).
7. Сравнительный анализ видов мобильных приложений / Ерошенко М.Н. – Текст: электронный // www.rusnauka.com: [сайт]. – URL: http://www.rusnauka.com/46\_PWMN\_2015/Informatica/2\_205079.doc.htm (дата обращения: 17.06.2019).
8. Разработка гибридных веб-приложений, способных использовать аппаратные средства мобильных устройств / Шейн Черч. – Текст: электронный // msdn.microsoft.com: [сайт]. – 2012. – URL: https://msdn.microsoft.com/ru-ru/magazine/hh852592.aspx (дата обращения: 17.06.2019).
9. 10 лучших IDE. – Текст: электронный // www.internet-technologies.ru: [сайт]. – 2017. – 23 апр. – URL: https://www.internet-technologies.ru/articles/10-luchshih-ide.html (дата обращения: 17.06.2019).
10. Android SDK: официальный сайт. – URL: https://android-sdk.ru.uptodown.com/windows (дата обращения: 18.06.2019).
11. На чём пишут приложения под Android/ Екатерина Якимова. – Текст: электронный //livetyping.com: [сайт]. – 2018. – 31 мая. – URL: https://livetyping.com/ru/blog/na-chem-pishut-prilozhenija-pod-android (дата обращения: 17.06.2019).
12. Разработка под iOS и Android: рейтинг языков программирования. – Текст: электронный //apptractor.ru: [сайт]. – URL: https://apptractor.ru/razrabotka-pod-ios-i-android-reyting-yazyikov-programmirovaniya (дата обращения: 17.06.2019).
13. За что Kotlin так полюбили в Google и кому нужны две тысячи языков программирования/Наталья Блинникова. – Текст: электронный //news.ifmo.ru: [сайт]. – 2017. – 25 мая. – URL: http://news.ifmo.ru/ru/news/6683/ (дата обращения: 17.06.2019).
14. Я хочу стать Android-разработчиком — какие языки программирования нужно знать/ NickMeller. – Текст: электронный //trashbox.ru: [сайт]. – 2016. – 4 февр. – URL: https://trashbox.ru/topics/94183/ya-hochu-stat-android-razrabotchikom-kakie-yazyki-programmirovaniya-nuzhno-znat (дата обращения: 17.06.2019).
15. Фадеев, А.Ю. Сравнительный анализ программного обеспечения для разработки мобильных приложений / А.Ю. Фадеев, Е.А. Волкова. – Текст: электронный // Электронный научный журнал «Наука и перспективы». – 2016. – №3. – URL: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-programmnogo-obespecheniya-dlya-razrabotki-mobilnyh-prilozheniy (дата обращения: 18.06.2019).
16. Android Studio: среда разработки мобильных приложений/ публикации. – Текст: электронный // arduinoplus.ru: [сайт]. – 2019. – 14 дек. – URL: https://arduinoplus.ru/android-studio/ (дата обращения: 18.06.2019).
17. Обзор кросс-платформенных решений для разработки мобильных приложений / Денис Зарицкий. – Текст: электронный // habr.com: [сайт]. – 2017. – 12 янв. – URL: https://habr.com/ru/post/319348/ (дата обращения: 18.06.2019).
18. Android WebView: актуальные проблемы и их решение/ Михаил Чабанов. – Текст: электронный // habr.com: [сайт]. – 2015. – 24 июля. – URL: https://habr.com/ru/company/mailru/blog/262167 (дата обращения: 18.06.2019).
19. Oracle: Java SE Development Kit 8 Downloads: официальный сайт. – URL: https://www.oracle.com/technetwork/java/javase/downloads/2133151 (дата обращения: 20.06.2019).
20. Введение в платформу Android. – Текст: электронный // metanit.com: [сайт]. – 2017. – 14 нояб. – URL: https://metanit.com/java/android/1.1.php (дата обращения: 18.06.2019).
21. Visual Studio: официальный сайт компании Microsoft. – URL: https://visualstudio.microsoft.com/ru/ (дата обращения: 20.06.2019).
22. Компания Nox Limited: официальный сайт. – URL: https://ru.bignox.com (дата обращения: 20.06.2019).

Приложения

Понятийно-терминологический словарь.

*Мобильное приложение* – компонент, устанавливаемый на мобильное устройство (телефон, планшет, фаблет, смартфон и т.п.) под конкретную мобильную платформу (Android, iOS, BlackBerry, HP webOS, Bada от Samsung и WindowsMobile), подключающийся к мобильному серверу и управляющий пользовательским интерфейсом и бизнес-логикой устройства.

*Операционная система* (ОС) – программное обеспечение, управляющее всем аппаратным и другим программным обеспечением на компьютере и предоставляющее службы, которые могут использовать приложения.

*Фреймворк* – программная среда, своеобразный каркас, используемый для того, чтобы существенно облегчить разработку и объединение разных модулей программного проекта.

*API* (application programming interface) – это набор готовых классов, функций, процедур, структур и констант.

*APK* – формат исполняемых файлов-приложений для Android.

*IDE* (integrated development environment) – интегрированная среда разработки – набор инструментов для разработки и отладки программ, имеющий общую интерактивную графическую оболочку, поддерживающую выполнение всех основных функций жизненного цикла разработки программы – набор и редактирование исходного текста (кода), компиляцию (сборку), исполнение, отладку, профилирование и др.

*WebView* – это компонент платформы Android, который позволяет встраивать web-страницы в Android-приложения.