1.8. Установка инструментов разработчика, первая программа на Андроид

До сих пор мы изучали теоретические основы Котлин, теперь же посмотрим, как это все применимо к разработке на Андроид. Для полноценной разработки приложений онлайн-тренажера будет уже недостаточно, нужно скачать и поставить на компьютер специальную программу. Программы для разработки называются IDE – integrated development environment (интегрированная среда разработки). Для разработки на Андроид есть специальная IDE – Android Studio (Андроид Студио), скачать ее можно с официального сайта разработчика: https://developer.android.com/studio/.

После того как вы скачали Андроид Студию, ее нужно установить. Для этого запустите только что скачанный установочник.

Важный нюанс: перед установкой убедитесь, что имя пользователя компьютера не содержит никаких символов, кроме латинских букв и обычных цифр (кириллица недопустима!). При необходимости создайте нового пользователя, в имени которого не будет лишних символов. При установке следуйте инструкциям установочника.

По окончании установки запустите Андроид Студию, перед вами откроется стартовое окно с возможностью создать новый проект, открыть уже имеющийся или изменить какие-либо настройки. Нажмите кнопку *New Project*, у вас откроется окно настройки проекта, промотайте до пункта *Empty Activity* и выберите его (окно может немного отличаться от приведенного ниже) (рис. 1).

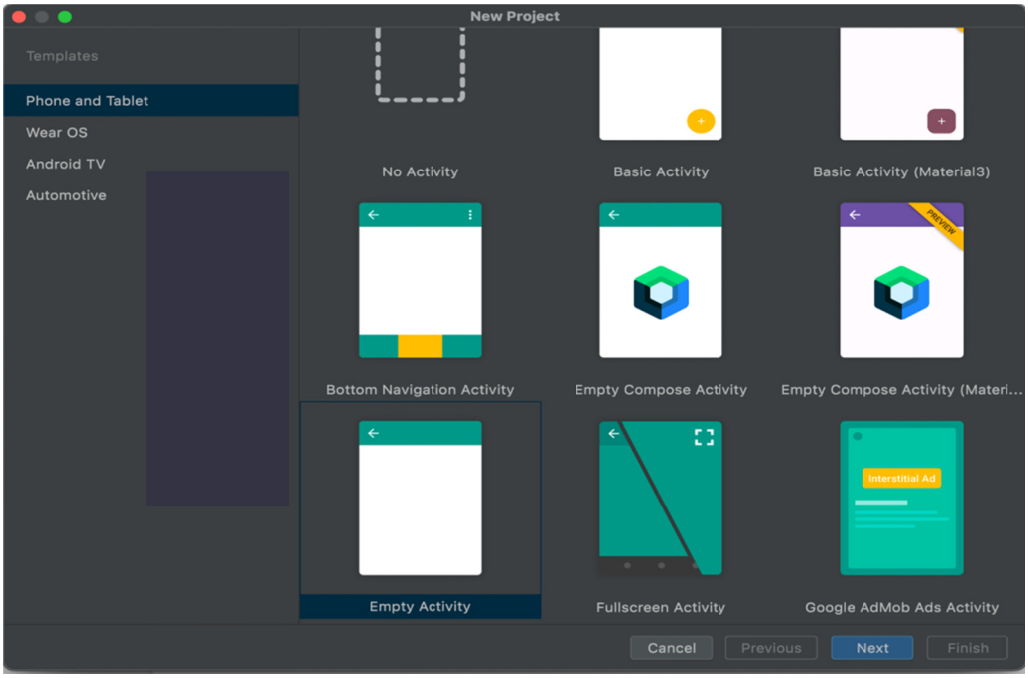

Рис. 1. Окно настройки проекта

Нажмите кнопку *Next*, в появившемся окне в поле *Name* введите название вашего проекта, например Hello World. В пункте *Package name* введите название пакета для вашего проекта, например ru.school.helloworld. В пункте *Save location* настройте путь к вашему проекту, убедитесь, что никаких лишних символов в пути нет, допустимы только латинские буквы и цифры. Пункты *Language* и *Minimum SDK* оставьте без изменений, нажмите *Finish*. Дождитесь, пока Андроид Студия настроит ваш проект, она скачает недостающие элементы для разработки, синхронизирует их, и вскоре можно будет приступить к непосредственной разработке на Андроид. После завершения настройки окно Андроид Студии будет выглядеть примерно так, как на рис. 2.

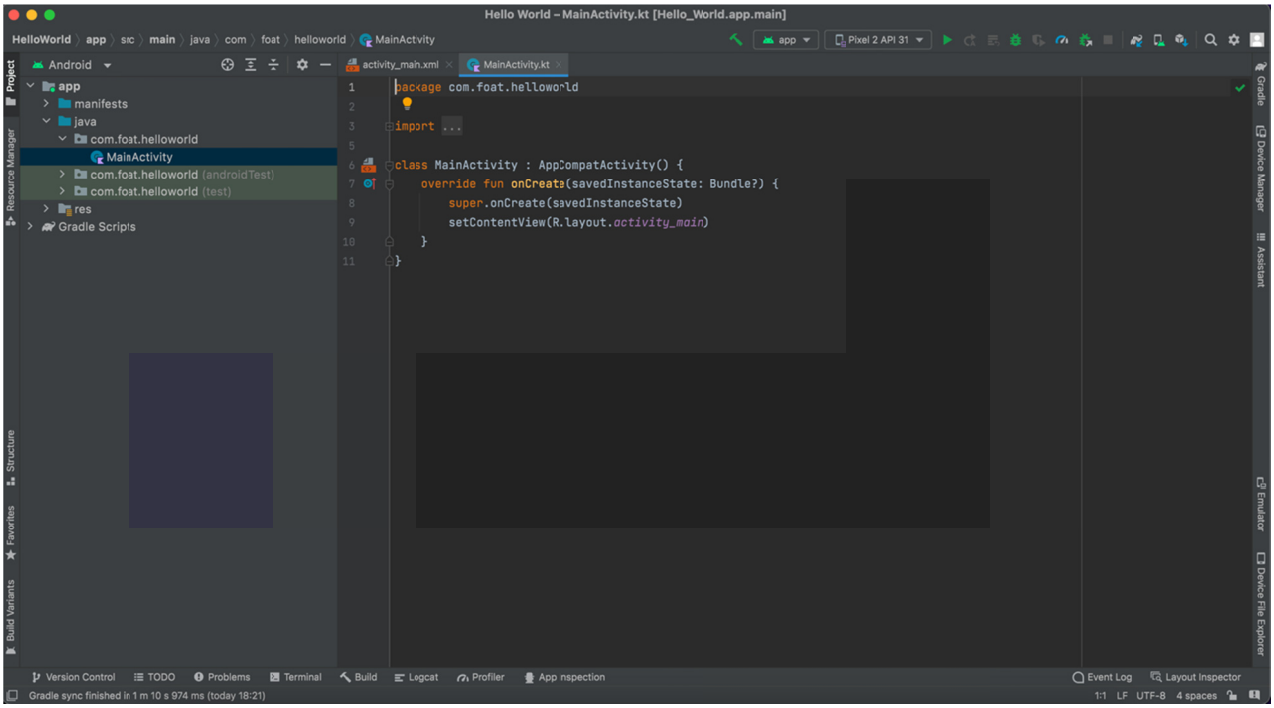

Рис. 2. Окно Андроид Студии с созданным проектом

Далее необходимо создать виртуальное устройство, если такого еще нет. Для этого перейдите в пункт меню Tools --> Device Manager. В открывшемся окне отобразится список имеющихся на данный момент виртуальных устройств (рис. 3).

Если ни одного устройства нет либо нужно создать еще одно, то нажмите кнопку *Create device*. В открывшемся окне необходимо выбрать модель устройства, которую нужно создать (на рис. 4 выбрана модель Pixel 5). Нажмите кнопку *Next*, в новом открывшемся окне выберите версию операционной системы Андроид, устройство, с которым вы хотите создать (рис. 5), рекомендуется выбирать первый или второй вариант сверху списка (на рис. 5 выбрана версия

API 32, которая соответствует версии Андроид 12.1). Нажимая кнопку *Next*, переходим в последнее окно настройки виртуального устройства. Здесь необходимо задать название устройства, которое будет отображаться в списке устройств, а также имеется возможность изменить некоторые другие свойства. Оставьте все настройки без изменений и нажмите кнопку *Finish*. После создания виртуального устройства можно наконец-то попробовать запустить наш проект и посмотреть, что получится. Для этого необходимо в меню с кнопками найти пункт, где указано виртуальное устройство (на рис. 7 это Pixel 2 API 31), и нажатием кнопки рядом (зеленый треугольник) запустить наше приложение на устройстве. Результат выполнения будет примерно такой же, как на рис. 8, отличия могут быть во внешнем виде устройства, по умолчанию оно запускается в виде прикрепленного окна внутри окна Device Manager (рис. 3).

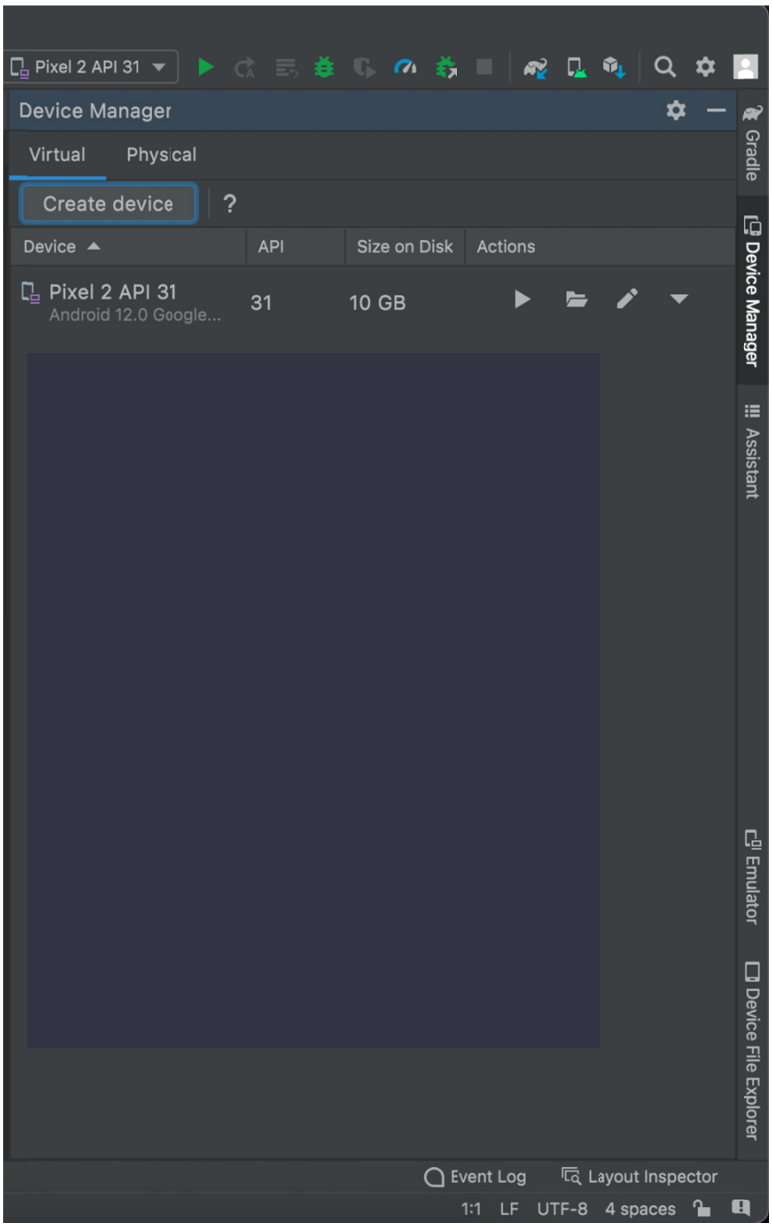

Рис. 3. Окно со списком виртуальных устройств

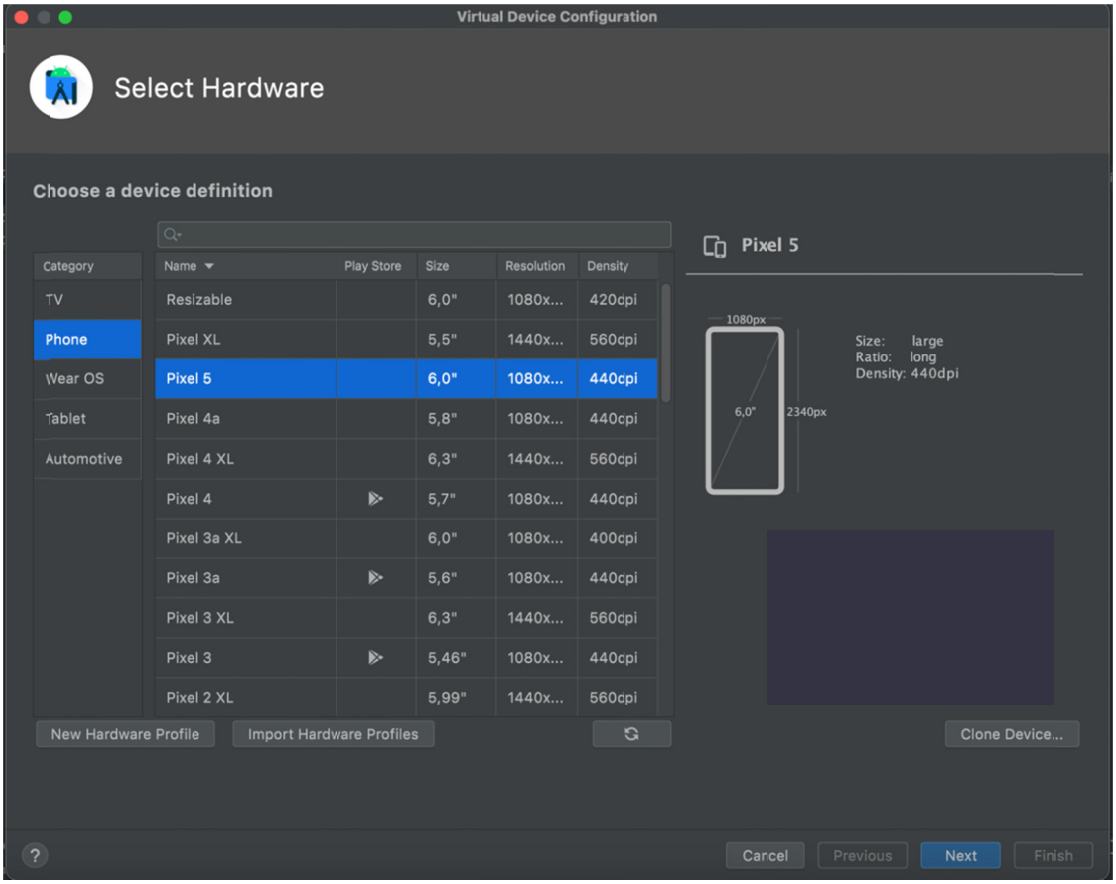

Рис. 4. Окно выбора модели устройства

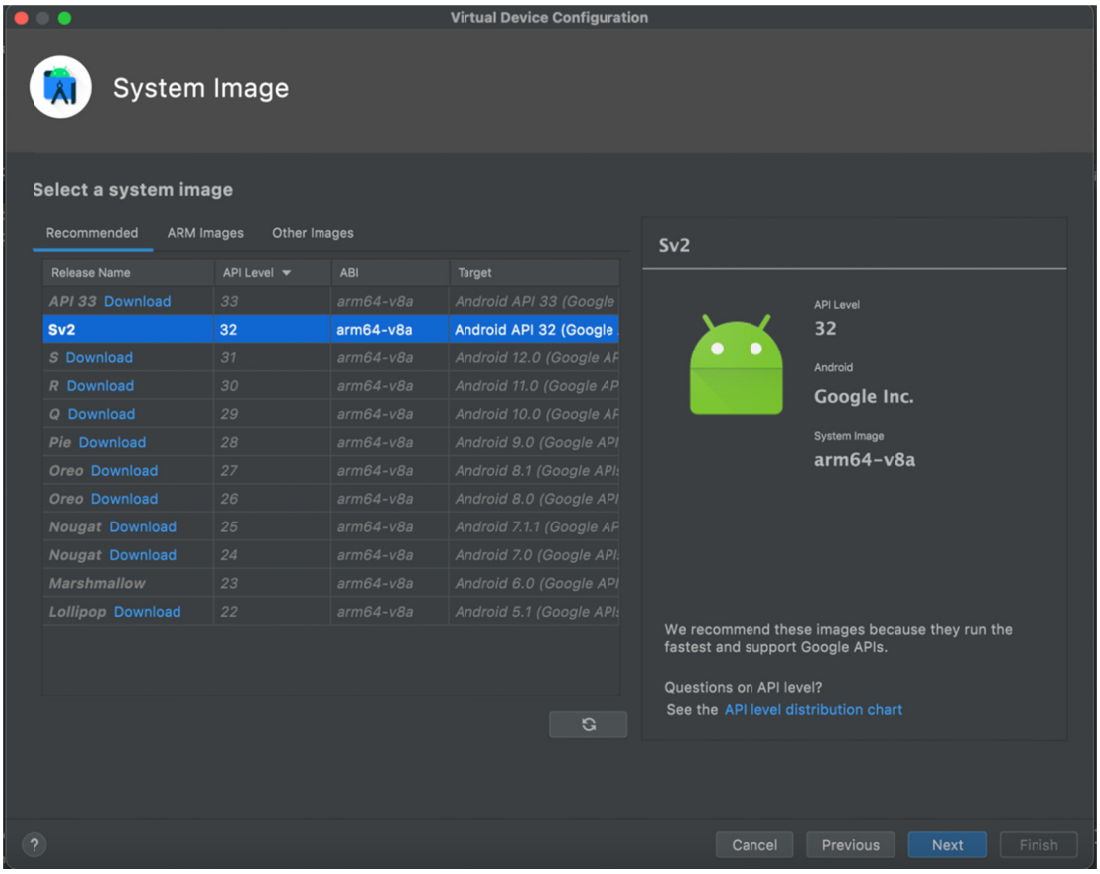

Рис. 5. Окно выбора версии Андроид

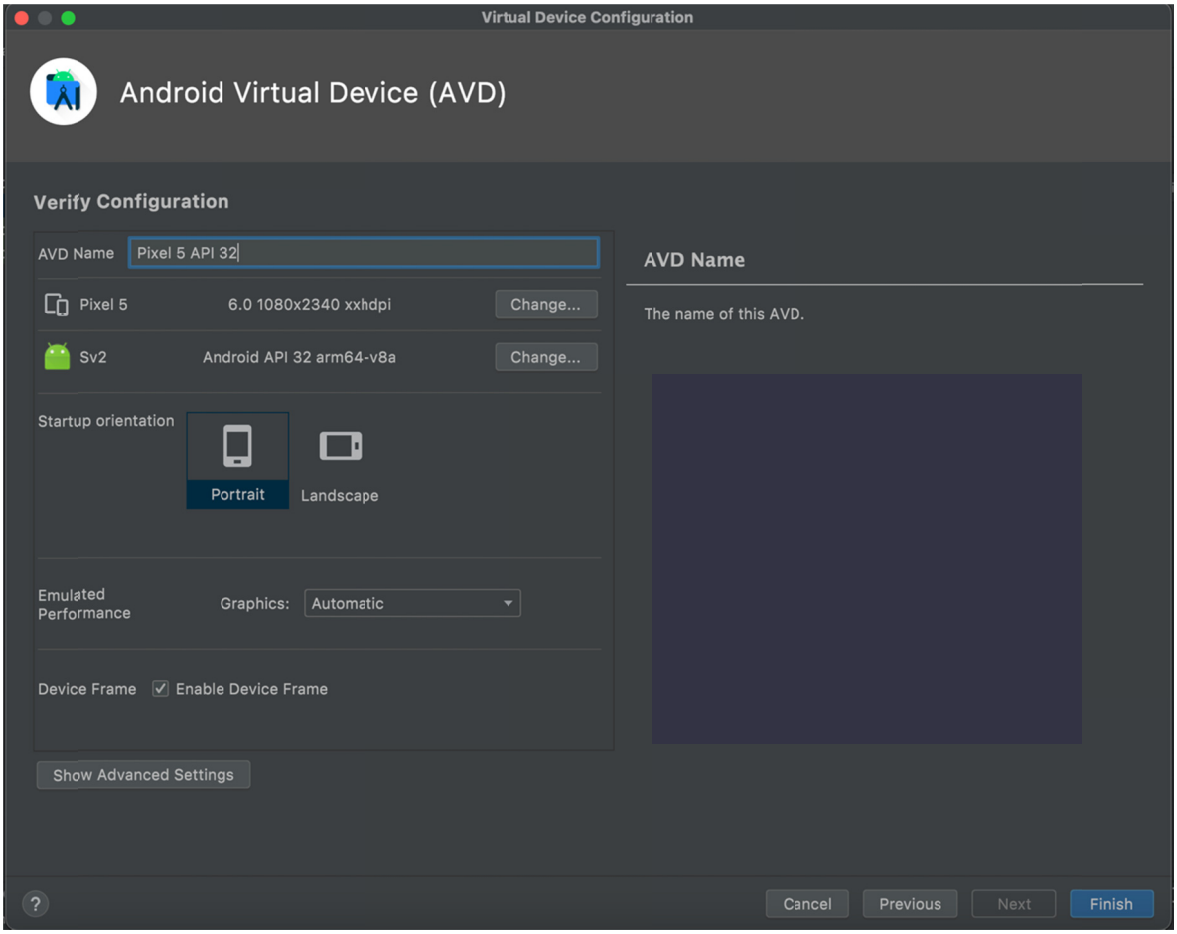

Рис. 6. Окно завершения создания устройства

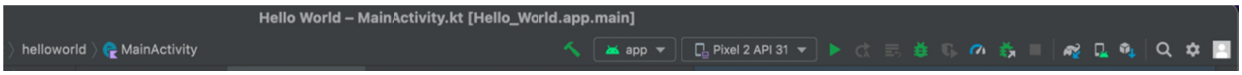

Рис. 7. Меню с кнопками

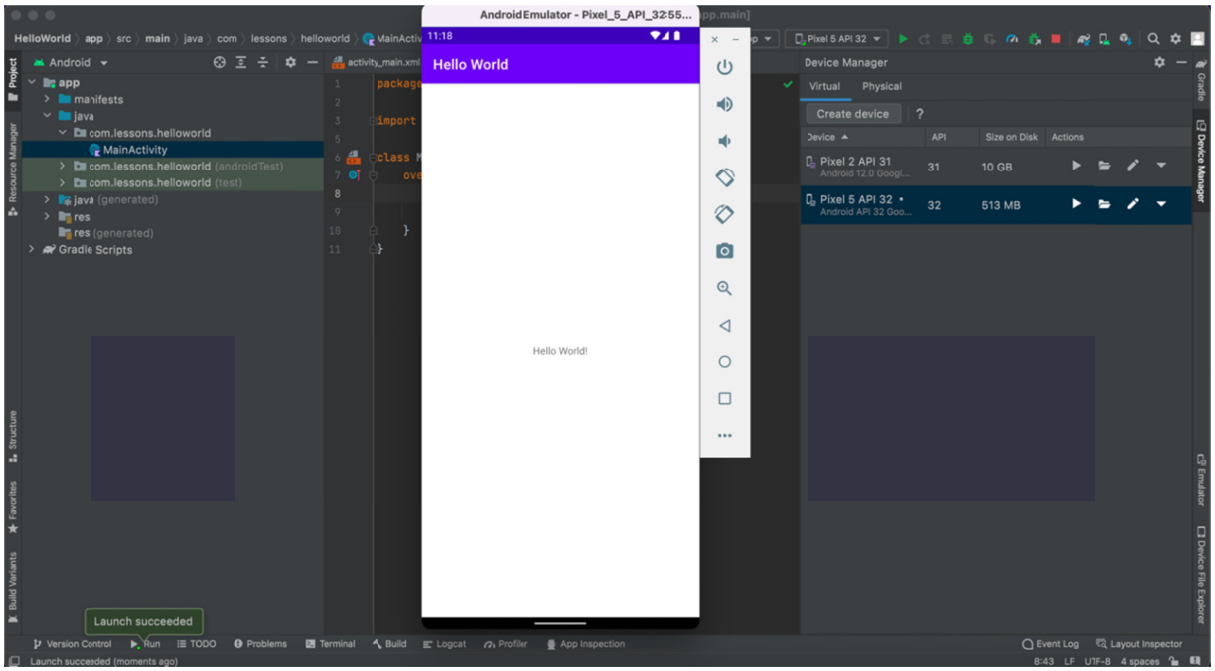

Рис. 8. Виртуальное устройство с запущенной программой

1.9. Структура Андроид проекта

Первая программа успешно запущена, результат виден на устройстве, теперь пришло время разобраться, что же добавлено в код и как это все работает.

В левом верхнем окне отображается структура всего нашего проекта – можно попробовать рассмотреть разные виды отображения:

– Android – режим по умолчанию, отображает основные «кирпичики» проекта;

– Project – режим отображения, как в файловой системе нашего компьютера;

– packages – режим отображения пакетов;

– другие.

В основном самым полезным будет режим отображения “Android”.

При открытии самого верхнего элемента в этом окне – app – мы увидим общую структуру проекта. Самый первый элемент в этом списке – папка **manifests**, которая содержит всего один элемент – **AndroidManifest.xml**. Этот файл будет содержать в себе описание всех элементов нашего приложения. На начальном этапе нам будет достаточно того, что уже содержится в файле, прежде всего – информация о главном окне нашего будущего приложения **MainActivity** (рис. 9).

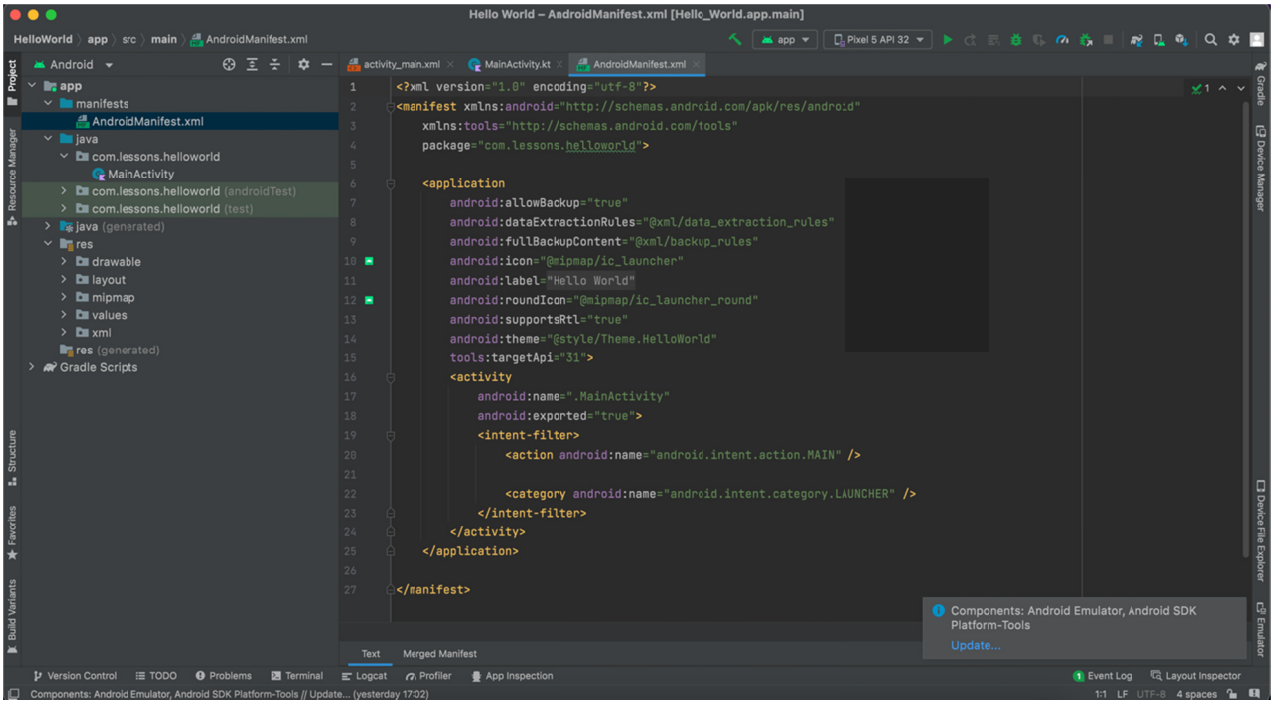

*Рис. 9. Открытый файл **AndroidManifest.xml***

Далее раскройте папку **java**, эта папка в любом проекте содержит основную часть кода приложения, в нашем случае написанного на Котлин. На данный момент в папке один файл – **MainActivity.kt**. Если он не отображается на главном окне, то его можно открыть, найдя в **пакете ru.school.helloworld** (здесь должно быть название пакета, который вы указали при создании проекта). Пакет в Котлин – это логическая общность, которая объединяет в себе определенный функционал (чаще всего это классы и функции), используемый для решения близкой по смыслу задачи. Традиционно пакет начинается с крупной общности типа «ru», «com» и т. д., далее добавляется название компании или организации разработчика (у нас это lessons), далее название проекта без пробелов и маленькими буквами (в нашем случае helloworld). В стандартной библиотеке Андроида часто пакеты начинаются со слов android/android. Две другие подпапки (подписанные соответственно **androidTest** и **test**) с таким же названием содержат тесты (это особый вид кода, который позволяет проверять наш код на корректность) – пока что созданные автоматически.

Далее раскройте папку **res**, эта папка в любом проекте содержит все ресурсы приложения. Так называются любые файлы с контентом – будь то изображения, звуки, текстовые данные, файлы интерфейсов нашего приложения или различные другие вспомогательные файлы. Например:

− в подпапке **drawable** будут содержаться все изображения;

− в подпапке **layout** – файлы интерфейсов приложения, сейчас в ней находится единственный файл **activity_main.xml** с описанием главного окна нашего приложения;

− **mipmap** содержит главную иконку приложения, по умолчанию это изображение зеленого робота;

− **values** содержит строковые ресурсы, цвета, темы со стилями приложения и т. д.

В текущем пособии мы не будем углубляться в сложные проекты с несколькими окнами, рассмотрим только случай приложения с одним главным окном, код которого будет в **MainActivity.kt**, а описание интерфейса – в **activity_main.xml.**

Далее откройте файл **activity_main.xml**.

Справа от панели Project отображаются панели, общее название которых **Редактор макета** (Layout Editor) (рис. 10). В правой части редактора сверху находится панель с палитрой, которая содержит элементы интерфейса, необходимые для приложения. В самом верхнем пункте палитры находятся самые популярные и нужные элементы – **TextView** (для отображения текста), **Button**

(обычная кнопка), **ImageView** (контейнер для вставки изображений), **RecyclerView** (для отображения списков), **ScrollView** (контейнер для отображения проматываемого контента – используется, если контент может не уместиться на экране устройства), **Switch** (элемент с возможностью выбора вариантов).

Снизу от палитры располагается панель со структурой макета – **Дерево макета** (Component tree). Здесь отображаются элементы, которые на данный момент добавлены в наш файл макета.

Далее справа находится основное окно редактора – собственно, панель редактора макета (Design editor). На этой панели в явном виде отображается дизайн нашего макета – либо в визуальном виде, либо в текстовом виде формата XML, либо как комбинация их обоих. Вид отображения можно сменить нажатием одной из кнопок в правом верхнем углу – Code, Split, Design. Попробуйте изменить вид отображения макета. В текстовом режиме отображения панели палитра и дерево макета пропадают. Обычно удобнее всего пользоваться режимом Split, чтобы описывать макет явно в текстовом виде и сразу видеть изменения в дизайне.

1.10. Макет приложения

Теперь посмотрим, какие элементы добавлены в нашем макете. Для начала перейдем в режим отображения макета Split.

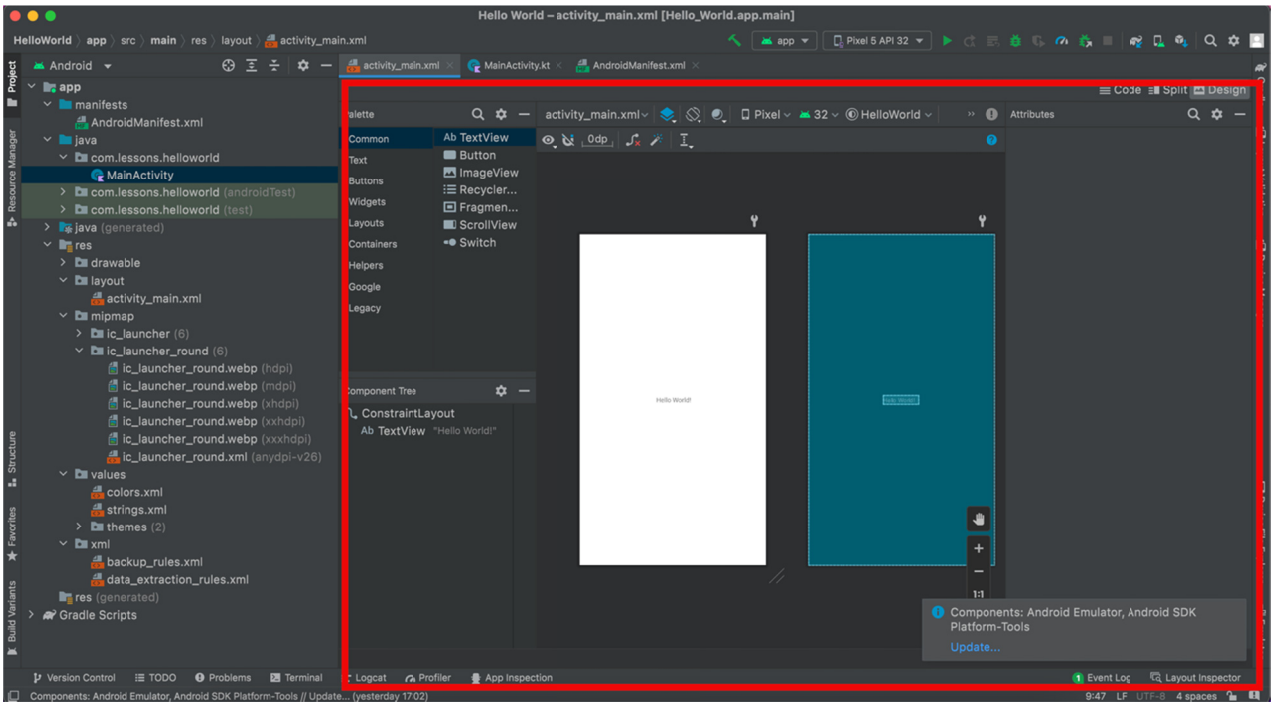

Рис. 10. Редактор макета

На самой верхней строчке файла указан так называемый корневой элемент (в терминологии формата XML) – `androidx.constraintlayout.widget.ConstraintLayout`, где `ConstraintLayout` – собственно, название элемента, а `androidx.constraintlayout.widget` – название пакета, где находится этот элемент. Вообще все элементы в XML выделяются знаками «<» и «>», либо конец элемента может обозначаться, как и начало, но с добавлением знака «/». А Андроид Студия любезно подсвечивает нам границы каждого элемента. Например, для того, чтобы увидеть границы корневого элемента, нажмите в любом месте на название элемента (рис. 11). Кроме `ConstraintLayout` файл содержит еще один элемент – `TextView`, который используется для отображения текста “Hello World”.

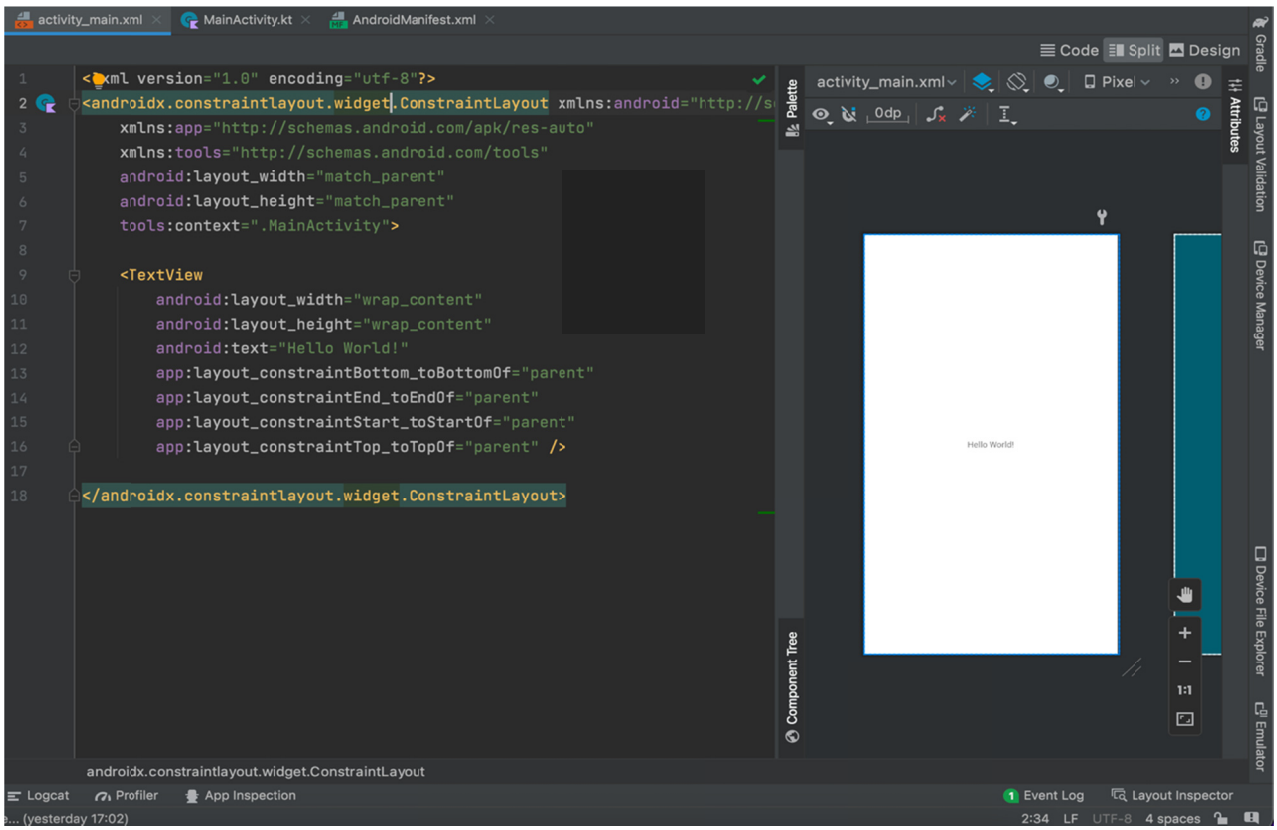

Рис. 11. Подсветка корневого элемента

Давайте рассмотрим подробнее атрибуты `TextView`. Основной атрибут этого элемента – `android:text="Hello World!"`, здесь указывается текст для отображения. Попробуйте изменить текст и запустить приложение.

Также важными и необходимыми атрибутами являются:

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

Как можно понять из названия, эти атрибуты используются для задания размеров нашего элемента (длинны и высоты соответственно), это стандартные атрибуты для большинства элементов – и для текстовых полей, и для кнопок, и для контейнеров (например, у элемента `ConstraintLayout` они тоже указаны). Значение `wrap_content` означает, что размер элемента будет подстроен под контент, который в нем содержится, т. е. в случае текстового поля размер элемента будет определен размером текста. Другим возможным значением этого атрибута является `match_parent`, который означает, что элемент займет все свободное пространство по высоте или длине.

Оставшиеся элементы используются для расположения элемента относительно других элементов, в нашем случае – внутри контейнера `ConstraintLayout`:

```
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
```

Как можно понять из названия, первый из них определяет положение нижней части элемента относительно нижней части элемента из правой части выражения (в нашем случае это элемент верхнего уровня, так называемый родительский элемент – `parent`). Следующий описывает положение правой части нашего элемента относительно правой части родительского элемента, далее – положение левой части относительно левой родительского элемента, и, наконец, последний – верхнюю часть нашего элемента относительно верхней родительского. Важное замечание: части ключевого слова `End` и `Start` ориентируются на язык приложения, в котором задаются все тексты, например, для русского языка они будут соответствовать правой и левой частям элемента, а для языков с направлением текста «справа налево» (арабский) они будут соответствовать левой и правой частям.

В правом верхнем углу панели редактора можно найти панель атрибутов выделенного в редакторе элемента (вызывается нажатием кнопки **Attributes**) (рис. 12), прокрутите панель до самого низа и найдите выпадающий список **All attributes** (рис. 13).

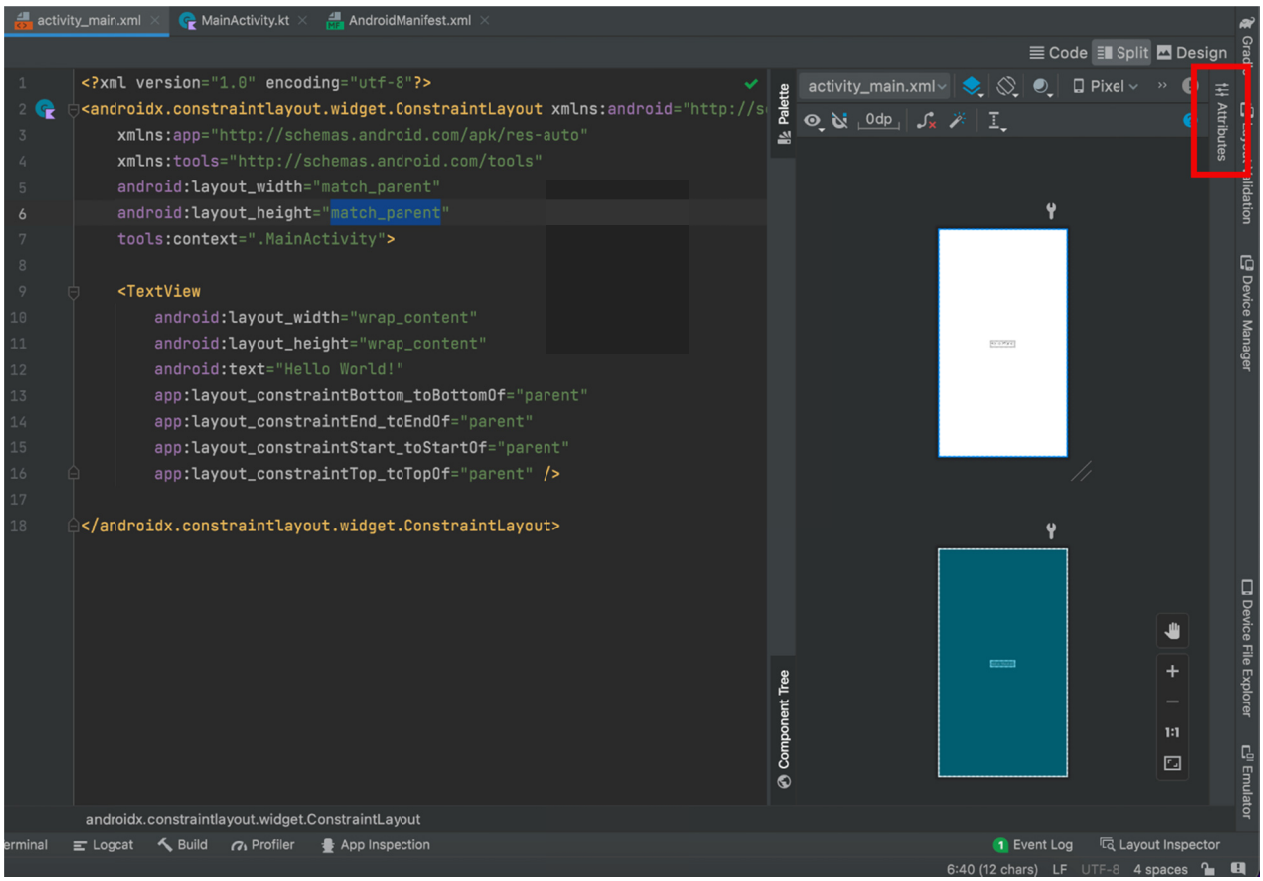

Рис. 12. Кнопка для вызова панели атрибутов

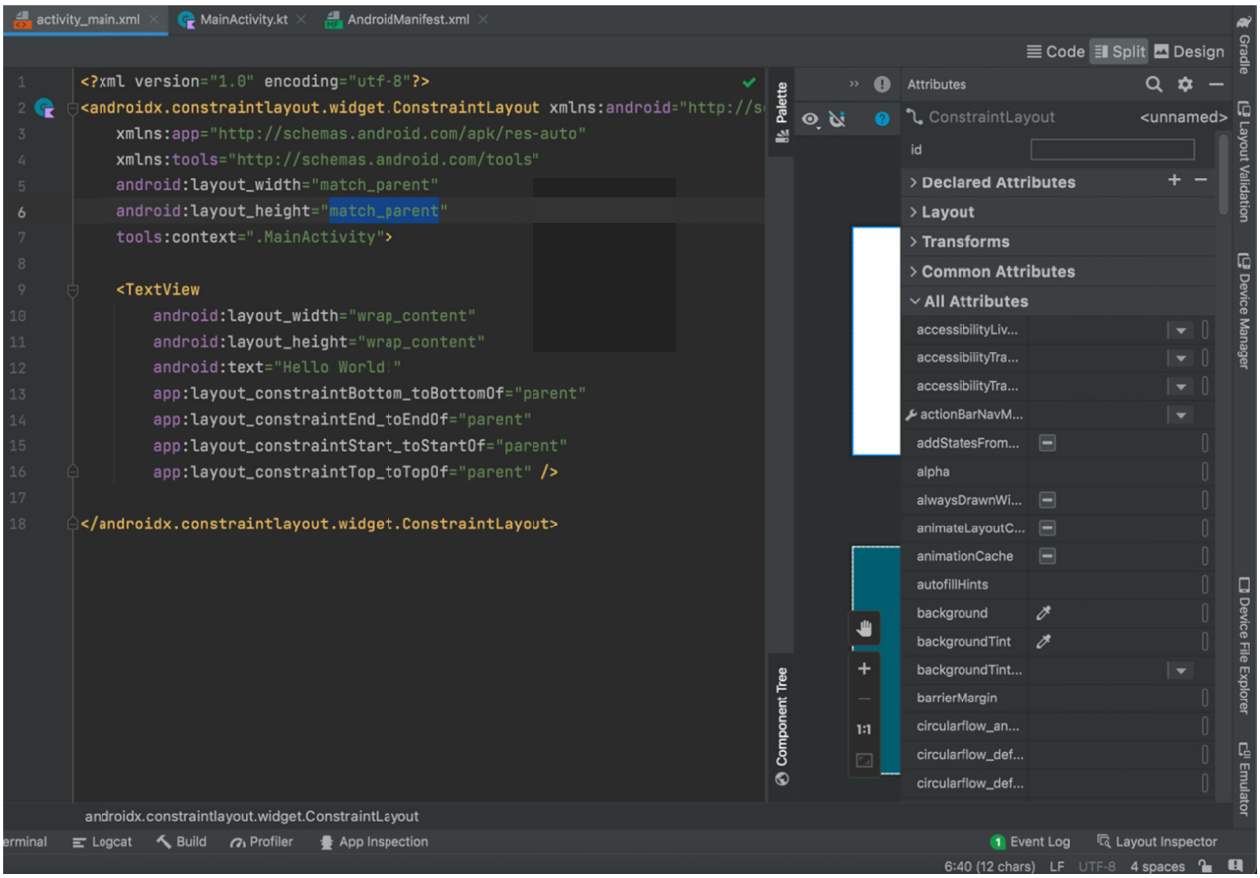

Рис. 13. Панель с атрибутами выделенного в редакторе элемента

Через эту панель можно менять свойства элементов, не внося изменений в формате XML. У разных элементов атрибуты могут отличаться, но некоторые являются общими для всех. Например, `background` отвечает за фоновую заливку элемента, он есть и у контейнера `ConstraintLayout`, и у `TextView`. А вот атрибут `textColor` есть только у элементов, работающих с текстом. Самостоятельно исследуйте панель атрибутов и попробуйте изменить цвет, стиль, размер текста у `TextView`, фоновую заливку у `TextView` и `ConstraintLayout` (Андроид Студия предоставляет инструмент для выбора цвета из палитры), попробуйте задать длинный текст в `TextView`. Как можно заметить, при изменении атрибутов, которых нет в текстовом виде, они сразу же добавляются и в редакторе, это происходит автоматически.

При изменении текста в `TextView` можно было заметить, что Андроид Студия подсвечивает это текстовое поле, при наведении на него курсора появится подсказка: `Hardcoded string "тут_ваш_текст", should use @string resource`. Это означает, что текстовые значения лучше писать не прямо в редакторе макета, а в специальном файле **strings.xml**, который содержится в папке **res**, подпапке **values**. Для поддержки нескольких языков в приложении нужно использовать разные файлы **strings.xml**, каждый из которых должен находиться в своей подпапке **values** с суффиксом языка (например, суффикс **-ru** для русского языка, **-es** – для испанского и т. д.). Папка без суффиксов используется по умолчанию, там располагают файл, из которого берутся все строковые ресурсы, если нет отдельных языковых папок. Поскольку в нашем приложении не будет поддержки разных языков, мы будем использовать эту папку по умолчанию для строк на русском языке. Откройте файл **strings.xml**, пока в этом файле только одна строка `<string name="app_name">Hello World</string>`, она используется для названия приложения (`app_name` – это имя строки, по этому имени к ней можно будет обратиться с любой другой точки проекта, а текст между квадратными скобками – значение этой строки). Создайте еще один элемент с названием radius_label и значением «Введите значение радиуса:». Вернитесь в файл макета и измените значение текста у `TextView` следующим образом:

```
"@string/radius_label"
```

Ключевое слово `@string` означает, что значение нужно будет взять из строковых ресурсов. Далее после знака «слеш» указывается имя строки. Запустите приложение и посмотрите, изменился ли текст.

Добавим еще несколько элементов на макет. Для этого перейдем в режим Design, в левом верхнем углу в панели палитры найдем элемент **Plain Text** (он находится в группе Text) и перетащим его курсором на макет, оставив пониже предыдущего TextView. В группе Buttons найдем элемент **Button** и перетащим его на макет пониже предыдущего элемента Plain Text (рис. 14).

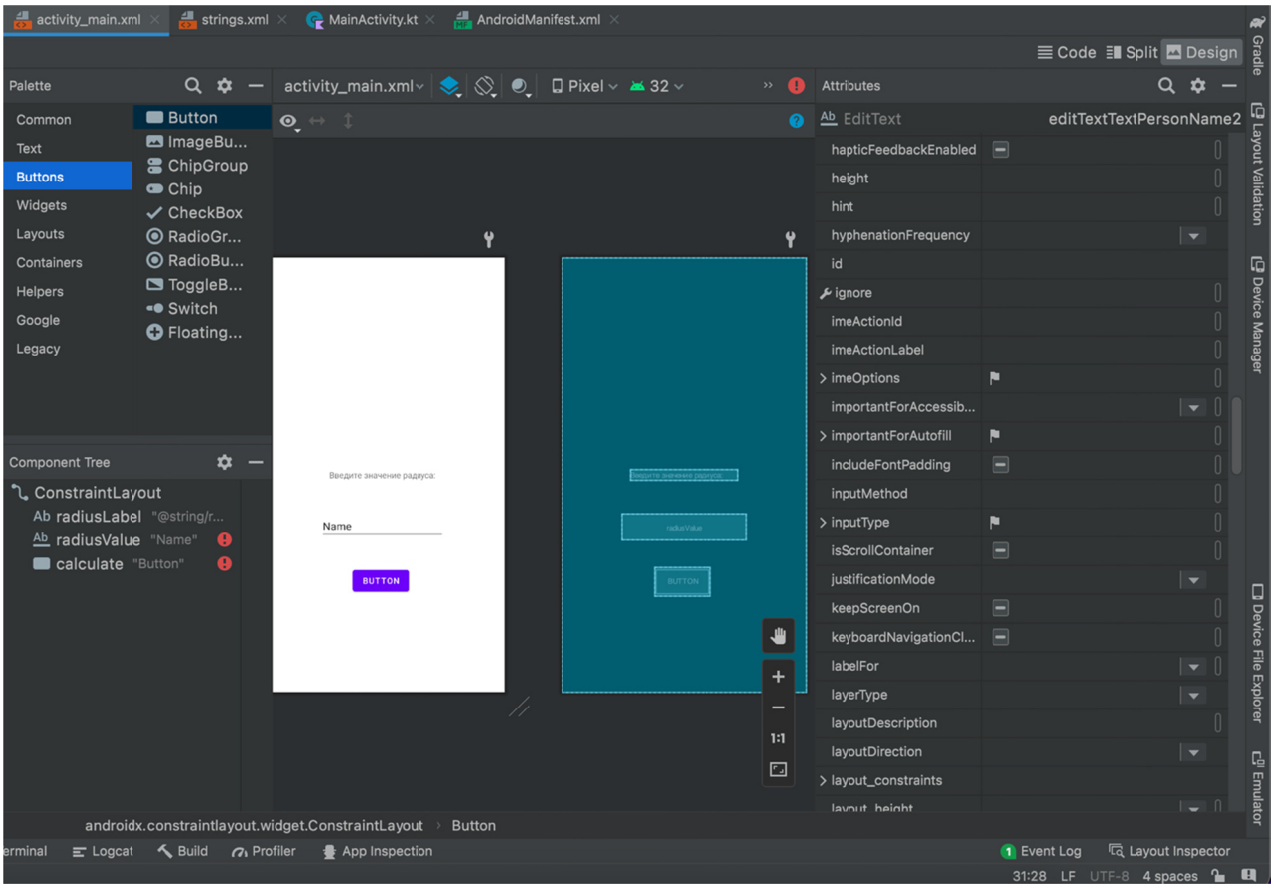

Рис. 14. Окно редактора макета с новыми добавленными элементами

Перейдем в режим Split и откроем панель Attributes. Андроид Студия подсвечивает наши новые добавленные элементы, подсказывая, что им не выставлено положение относительно соседних элементов. Для того чтобы расположить их в нужном нам порядке, зададим каждому элементу `id` – название элемента, по которому мы будем к нему обращаться (по аналогии с именами строковых ресурсов). По очереди выделяя каждый элемент в макете, найдите для каждого атрибут `id` в списке атрибутов и задайте нужные значения, например, для TextView – `radiusLabel`, EditText – `radiusValue`, Button – `calculate` (рис. 15). Далее каждый элемент будет обозначаться его именем.

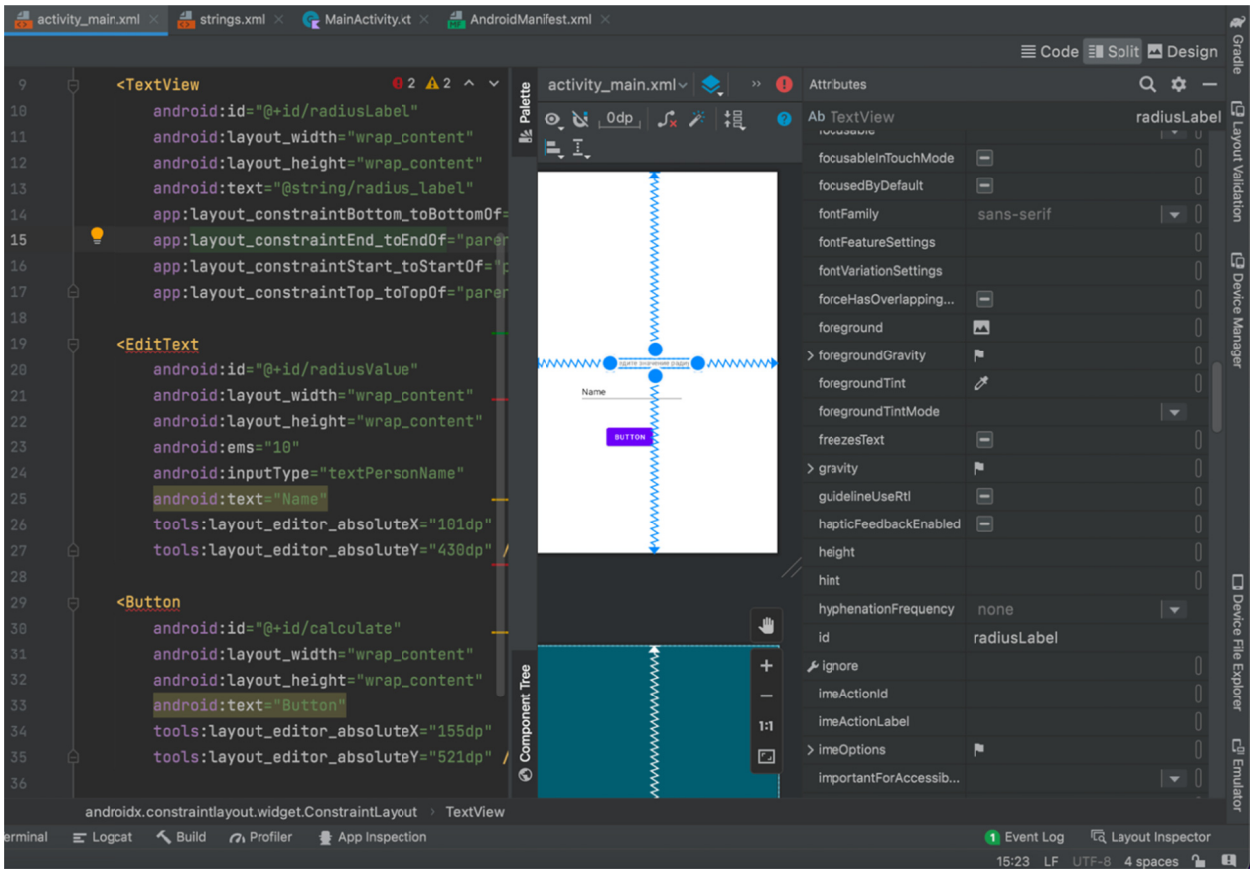

Рис. 15. Окно редактора макета с заданными id

Теперь можно определить для каждого элемента его положение.

Выделим элемент `radiusLabel` и найдем среди его атрибутов выпадающий список **layout_constraints**, раскроем его двойным нажатием и найдем в нем атрибут `layout_constraintBottom_toBottomOf` – изменим его значение на `@id/radiusValue`, это можно сделать, выбрав в выпадающем списке значений нужное имя.

Выделим элемент `radiusValue`, найдем среди его атрибутов `layout_constraintEnd_toEndOf` и `layout_constraintStart_toStartOf` – зададим им значение `parent`, т. е. левый и правый концы элемента будут привязаны к левому и правому краю родительского элемента. Атрибуту `layout_constraintTop_toBottomOf` установим значение `@id/radiusLabel`, это расположит наш элемент сверху по центру родительского элемента, чуть ниже элемента `radiusLabel`. Далее нам нужно, чтобы пользователь мог вводить в это поле только числа, для этого изменим атрибут `inputType`: `android:inputType="numberDecimal"`.

Далее аналогичным образом установите расположение для кнопки `calculate` – сверху по центру и чуть ниже элемента `radiusValue`, а также

удалите текст у элемента `radiusValue`, а текст у кнопки задайте «Вычислить площадь круга».

Самостоятельно добавьте так же еще два TextView:

1. С текстом «Площадь круга =», расположите его между кнопкой `calculate` и полем ввода `radiusValue`, размеры задайте `wrap_content` (не забудьте, что текстовое значение нужно задать в strings.xml).

2. С текстом «Калькулятор площади круга», расположите его на самый верх окна, выше элемента `radiusLabel`. Задайте элементу размер текста `textSize = «24sp», textStyle = «bold»`.

После всех действий у вас должно получиться так же, как на рис. 16. Запустите приложение на эмуляторе и проверьте, что получилось, убедитесь, что в поле ввода можно вводить только числа (целые и дробные).

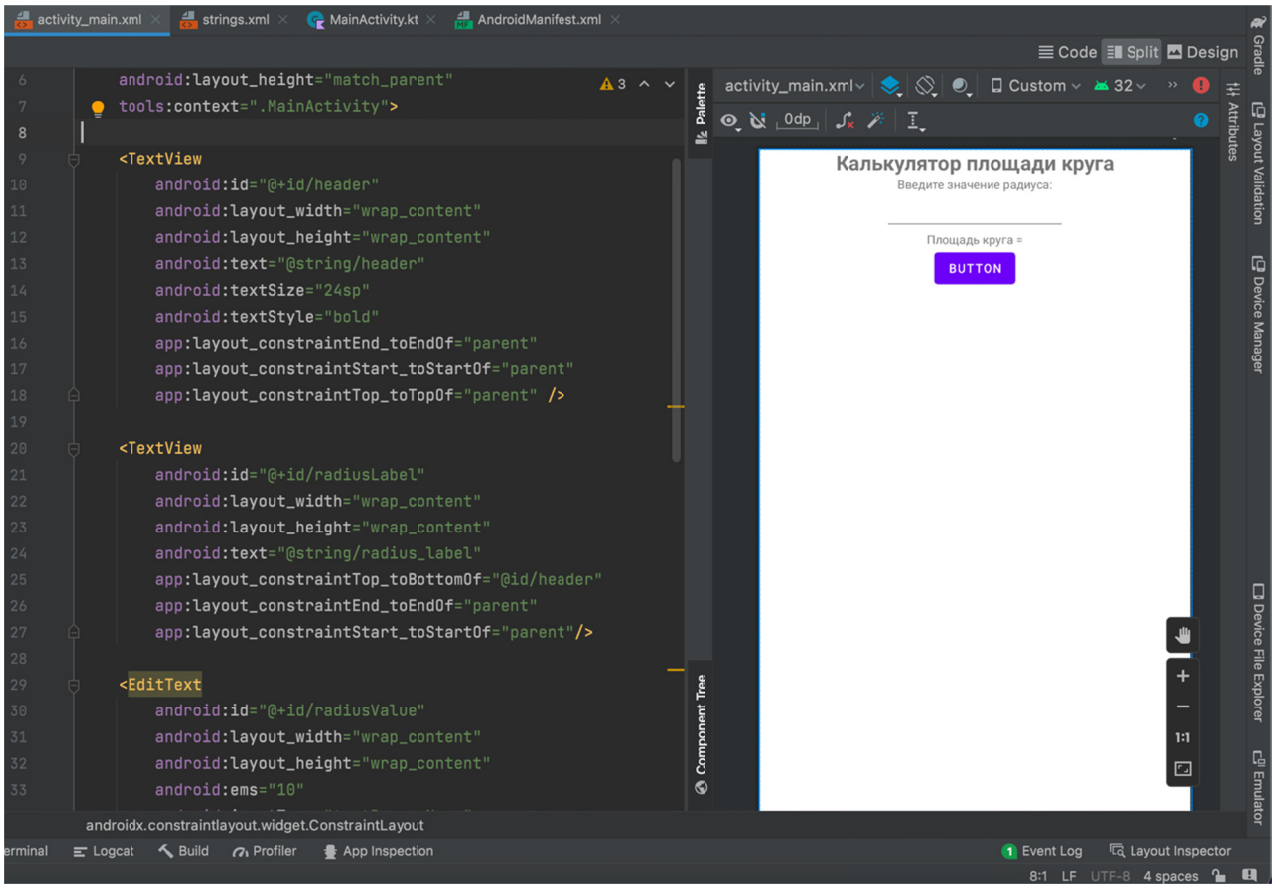

Рис. 16. Готовый макет приложения

1.11. Взаимодействие с интерфейсом

Макет нашего приложения готов, но приложение еще ничего не делает. Для того чтобы приложение реагировало на действия пользователя, нужно добавить обработчики действий – прежде всего, нажатие кнопки `calculate`. Перейдем в файл MainActivity.kt. Немного о классе, который находится в этом файле – **активность MainActivity**. Активность – основной класс структуры Андроид-приложения. Этот класс отвечает за отображение и взаимодействие с главным экраном приложения, его же имя указано в манифесте как точка запуска. В этом классе реализована одна функция `onCreate`, которая вызывается автоматически при запуске приложения и инициализации активности. В этом же методе активности задается макет окна, с помощью метода `setContentView(R.layout.activity_main)`. Все изменения в нашем тестовом проекте мы будем делать внутри этого метода, но вообще хорошим тоном считается создавать отдельные функции с отдельным функционалом.

Далее определим, как в этом файле получить доступ к элементам макета, которые мы создали в `activity_main`. Андроид предоставляет стандартную функцию получения любого элемента из макета путем обращения к ее имени `findViewById`:

```
val button = findViewById<Button>(R.id.calculate)
```

В квадратных скобках функции указывается тип элемента, который нужно достать из макета (в примере выше это `Button`), а в круглых скобках – имя элемента (`R.id.calculate`). Это имя образуется путем добавления приставки `R.id` к идентификатору `calculate`, который был задан в макете (рис. 15). Приставка `R.id` используется для всех идентификаторов макетов. После добавления этого кода Андроид Студия будет подчеркивать строку красным, подсказывая, что здесь есть какая-то ошибка – это ошибка отсутствия пакета с нужным элементом. При наведении курсора на строку выйдет подсказка, что можно импортировать нужный пакет, нажмите Import, после чего добавится строка `import android.widget.Button` наверху файла. Аналогичным образом добавьте переменные для доступа к `radiusValue` и `circleArea`.

Далее создадим обработчик нажатия на кнопку. Для этого вызовем функцию `setOnClickListener` у кнопки:

```
button.setOnClickListener { }
```

Этот метод добавляет обработчик нажатия для кнопки, внутри которой будут описаны действия, которые произведутся при нажатии на кнопку. Для начала получим данные из строки ввода:

```
val radius = input.text.toString().toDoubleOrNull()
```

`input` – название переменной для поля ввода (у вас может отличаться), `text` – стандартная переменная у этого объекта, которую далее нужно преобразовать в `String`, который, в свою очередь, уже преобразуется в `Double`. Если поле ввода содержит данные, которые невозможно преобразовать в `Double`, то в переменную `radius` запишется значение `null`. Проверяем значение `radius` на `null`, и если оно не `null`, то вычисляем размер площади круга по известной формуле и записываем полученное значение в текстовое поле `result`:

```
if (radius != null) {
val res = pi * radius * radius
val label = getString(R.string.circle_area)
result.text = "$label$res"
}
```

Чтобы получить строковый ресурс, воспользовались методом `getString`, который принимает в себя имя строки из strings.xml. Кроме того, для присвоения текстовому полю значения использовали **строку с аргументами –** в Котлин можно прямо в строковое значение вписывать значения других переменных, поставив предварительно перед ними знак доллара «`$`».

Общий код активности должен получиться примерно таким, как на рис. 17.

Запустите приложение и попробуйте вычислить площадь круга для разных значений радиуса, убедитесь, что все работает, площадь вычисляется верно.

Контрольные вопросы и задачи

1. Какие основные преимущества языка Котлин?

2. Чем отличаются объекты, объявленные с использованием ключевых слов val и var?

3. Назовите основные операции над типом Boolean.

4. В чем отличие анонимных функций от обычных?

5. Что подразумевается, когда говорят, что переменная nullable?

6. Дайте определение оператора range.

7. Что такое конструктор класса?

8. Назовите три основных вида коллекцией в Котлин.

9. Установите все инструменты разработчика – Андроид Студию и эмулятор, пользуясь разделом 1.8.

10. Создайте приложение «Калькулятор площади круга», используя в качестве руководства разделы 1.10 и 1.11.

11. В качестве самостоятельного задания реализуйте новое приложение – простой калькулятор для арифметических операций:

- *10 цифр, от 0 до 9, все можно реализовать через Button;*
- *4 арифметические операции – «+, -, *, /», реализовать через Button;*
- *2 поля ввода – для первого аргумента и для второго, EditText;*
- *одно текстовое поле TextView, для вывода результата;*
- *кнопка очистки полей ввода.*

activity_main.xml | strings.xml | MainActivity.kt | AndroidManifest.xml

```
package com.lessons.helloworld

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.TextView

class MainActivity : AppCompatActivity() {

    private val pi = 3.1415

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val input = findViewById<EditText>(R.id.radiusValue)
        val result = findViewById<TextView>(R.id.circleArea)
        val button = findViewById<Button>(R.id.calculate)
        button.setOnClickListener {  it: View!
            val radius = input.text.toString().toDoubleOrNull()
            if (radius != null) {
                val res = pi * radius * radius
                val label = getString(R.string.circle_area)
                result.text = "$label$res"
            }
        }
    }
}
```

erminal | Logcat | Build | Profiler | App Inspection

Рис. 17. Полный код активности ***MainActivity***