

**Учебное пособие по выполнению практических заданий по курсу
"Разработка облачных и мобильных приложений на платформе Google
Android" в среде Android Studio**



Автор: Макаров С.Л.

Москва 2017-2023

Содержание

Лабораторная работа №1. Создание первого приложения.....	3
Лабораторная работа №2. Работа с элементами.....	18
Лабораторная работа №3. Работа с экранами	23
Лабораторная работа №4. Стили и темы.....	30
Лабораторная работа №5. Работа со списками	34
Лабораторная работа №6. Работа с анимацией	45
Лабораторная работа №7. Работа с картами	50
Лабораторная работа №8. Создание виджета	56
Лабораторная работа №9. Работа с меню	63
Лабораторная работа №10. Работа с диалоговыми окнами	69
Лабораторная работа №11. Работа с уведомлениями.....	73
Лабораторная работа №12. Работа с аудио	77
Лабораторная работа №13. Работа с видео	78
Лабораторная работа №14. Работа с камерой.....	80
Лабораторная работа №15. Работа с настройками и БД.....	83

Лабораторная работа №1. Создание первого приложения

Задание: скачать и установить последние версии Java Development Kit и Android Studio, установить необходимые компоненты SDK, сконфигурировать эмулятор, создать новый проект, изменить TextView с надписью "Hello world" на свои ФИО и запустить проект на эмуляторе.

Для того, чтобы начать программировать под Android, необходимо скачать и установить две вещи: Java Development Kit (JDK) и Android Studio. Тут дело в том, что на многих компьютерах уже установлена Java какой-то версии, поэтому первый шаг необязателен. Но лучше всего всё-таки установить последнюю версию, на всякий случай, чтобы получить доступ к последней версии языка и development kit-а. Для этого нужно перейти по адресу oracle.com – Products – Hardware and Software – Java – Download Java – – выбираем свою ОС и разрядность и нажимаем на ссылку, чтобы скачать файл установки (в моём случае это был файл jdk-8u381-windows-x64.exe и jdk-11.0.20_windows-x64_bin.exe) и установить с параметрами по умолчанию, для установки нужны права администратора.

Вторым шагом идёт установка Android Studio. Тут у нас есть дилемма: дело в том, что этот курс традиционно читается на Java, а последняя (и не только) версия Android Studio предлагает программировать на Kotlin по умолчанию. С другой стороны, последние (свежие, вышедшие недавно) версии Android Studio, как правило, нестабильные и в них очень много багов – как визуальных, так и на уровне программирования. Поэтому, если вам нравится Kotlin, и вы не боитесь багов и нестабильности, перейдите по адресу <https://developer.android.com/studio/>, нажмите на кнопку «Download Android Studio (Giraffe на данный момент)», примите лицензию, скачайте и установите приложение с параметрами по умолчанию (версия 2022.3.1.19, Android Studio Giraffe, Patch 1, август 17, 2023, см. рис. 0), для установки нужны права администратора. Если же вам больше подходит стабильность,

отсутствие багов и Java для шаблона Empty Activity, надо найти проверенную версию Android Studio под названием Electric Eel (последний патч для которой тоже вышел в 2023 году), и установить её, например, отсюда:

<https://redirector.gvt1.com/edgedl/android/studio/install/2022.1.1.20/android-studio-2022.1.1.20-windows.exe>. Electric Eel позволяла выбирать язык программирования между Java и Kotlin даже для шаблона Empty Activity. Если хотите, можно найти ещё более старую версию Android Studio, например Dolphin или Arctic Fox, или даже использовать другую IDE типа IntelliJIDEA, Eclipse или даже Visual Studio – для прохождения данного курса этого будет вполне достаточно. Но вы должны установить и сконфигурировать соответствующие компоненты в сторонних IDE самостоятельно, если будете использовать их – Android SDK + Android Emulator.

После установки запустите Android Studio - появится окно приветствия, которое показано на рисунке 1 (окно показано для старой версии, для новой оно будет примерно таким же). Но появится оно только в том случае, если у вас ещё нет ни одного проекта в предыдущих версиях Android Studio, в которой вы никогда не работали, т.е. установка абсолютно чистая, с нуля. Иначе откроется последний проект.



Рисунок 0 – Версия Android Studio Giraffe

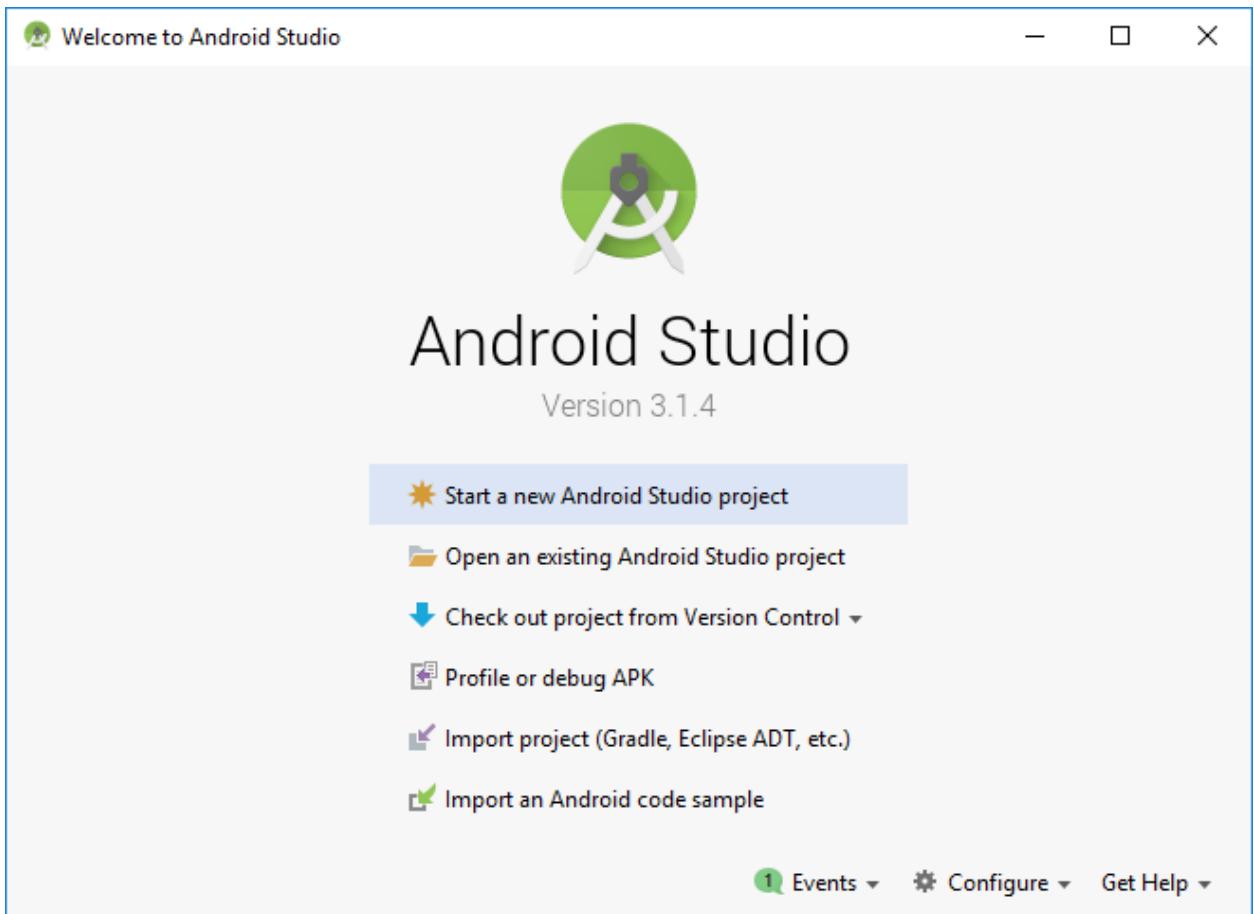


Рисунок 1 – Окно приветствия в Android Studio

Нажмите «Start a new Android Studio project», и в следующем окне выберите шаблон приложения, см. рисунок 2. В большинстве лабораторных работ необходимо выбирать «Empty Activity», в иных случаях об этом будет сказано дополнительно. Кроме того, на этой странице виден выбор платформы, для которой вы разрабатываете приложение (телефон и планшет, телевизор, часы, авто). В рамках данного курса будут разрабатываться приложения для смартфонов (Phone).

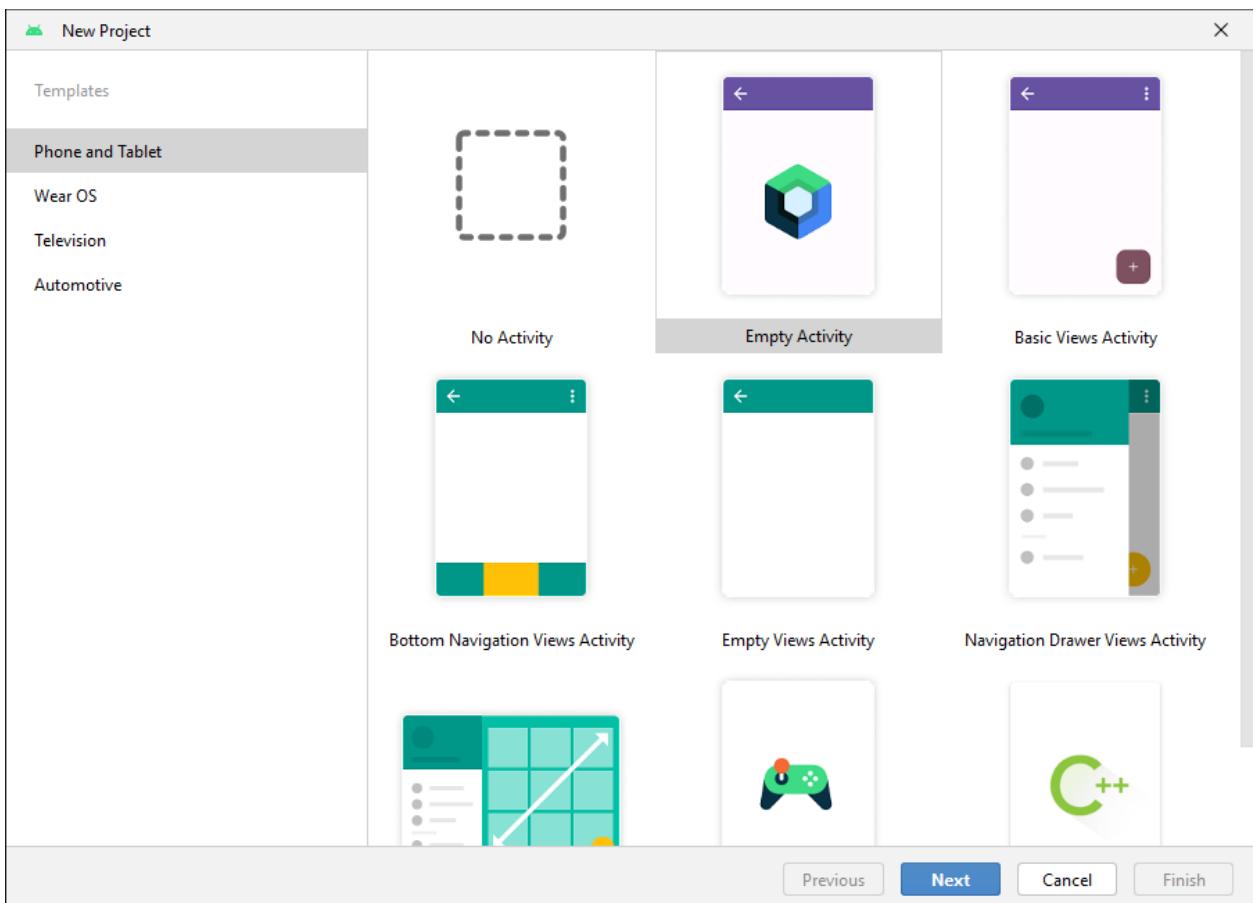


Рисунок 2 – Создание проекта в Android Studio, выбор шаблона и платформы

Нажмите «Next», и придумайте название проекта. Укажите название приложения в поле «Name», корневой пакет приложения в поле «Package Name», который является идентификатором приложения (по умолчанию формируется автоматически в виде com.example.myapplication, меняется в зависимости от значения поля Name), и путь для сохранения проекта «Save location» (лучше оставить по умолчанию). Далее можно выбрать минимальную версию SDK, на которой приложение будет работать (или по-другому – минимальную версию ОС Android). Пример заполнения показан на рисунке 3. Google рекомендует использовать как можно меньшую версию API, чтобы приложение можно было установить на большее количество устройств, однако, чем меньше версия API, тем меньше функций будет поддерживать приложение. С другой стороны, по умолчанию Android Studio ставит самую последнюю версию OS (API TiramisuPrivacySandbox Preview),

которая доступна менее, чем на 1% устройств, по сравнению с API 21 (99,5% устройств). Для корректного выполнения лабораторных работ не рекомендуется использовать API меньше 21, далее выбор за вами. Если у вас есть устройство под управлением Android, было бы логично выбрать соответствующую версию API. Соответствие версий API (SDK) и версий ОС Android можно посмотреть, например, здесь: <http://socialcompare.com/en/comparison/android-versions-comparison>.

Например, Tiramisu – это Android 13 и, соответственно, 33 версия API/SDK.

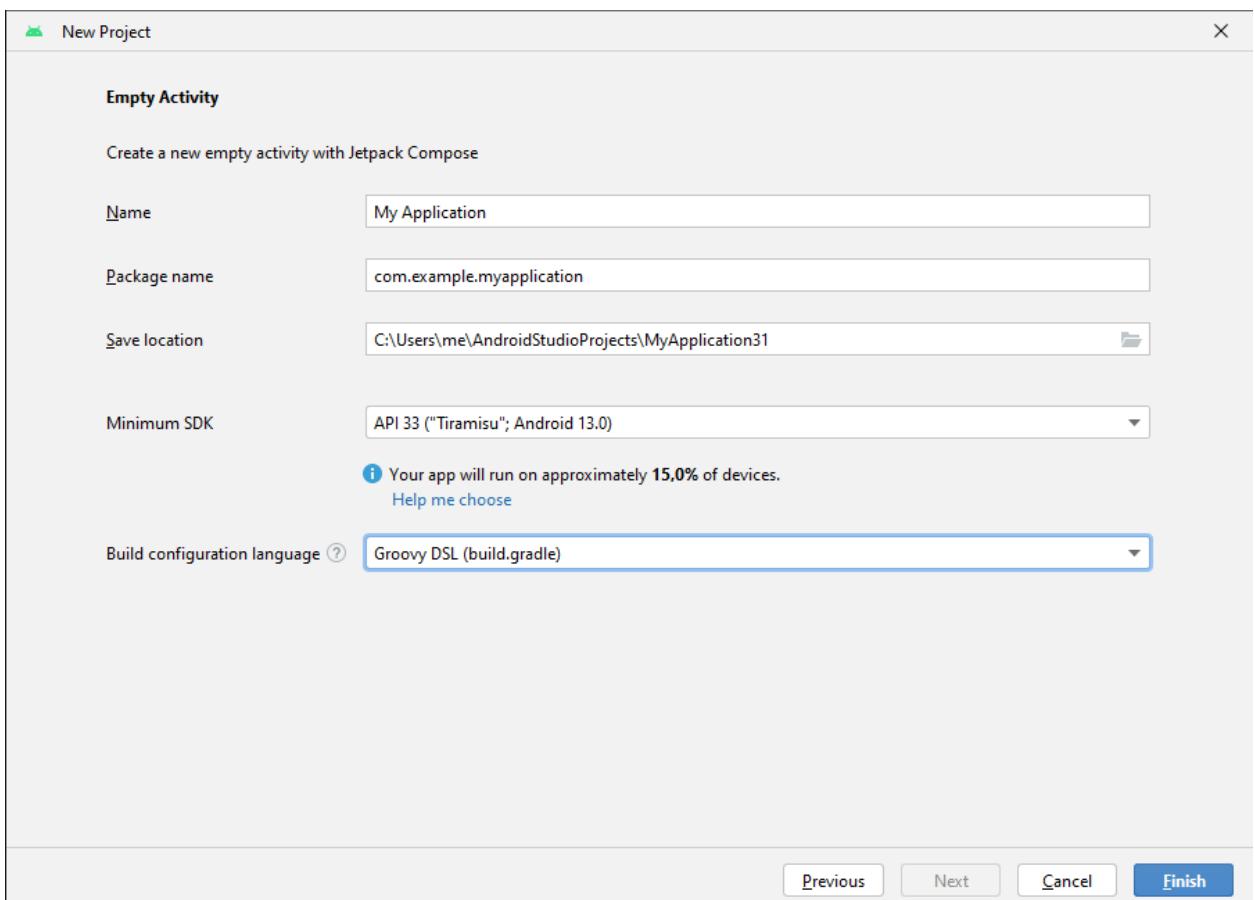


Рисунок 3 – Выбор названия проекта и других параметров

Нажмите «Finish», и через некоторое время, нужное на загрузку необходимых компонентов и построение проекта (рисунок 4), отобразится главное окно среды разработки, которое показано на рисунке 5.

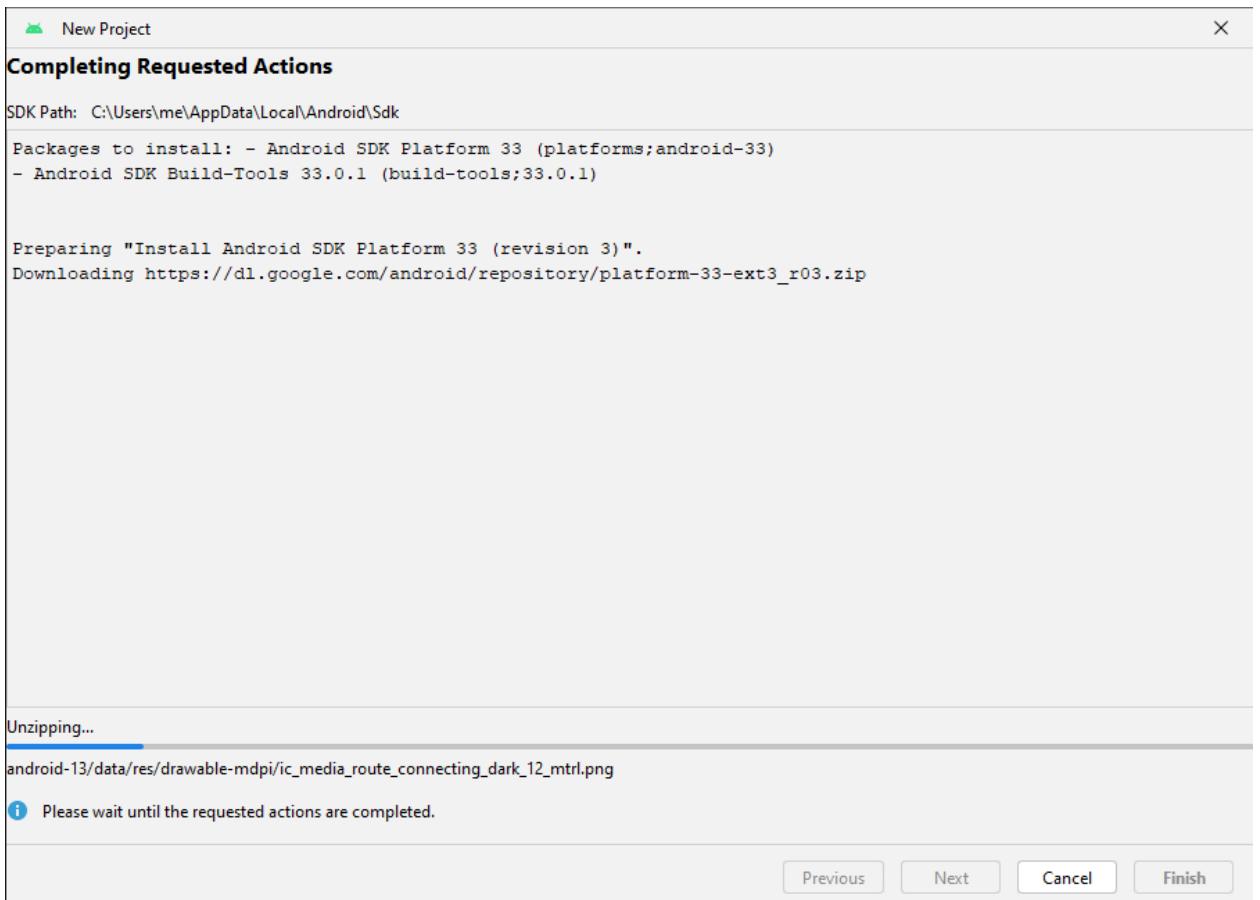


Рисунок 4 – Создание проекта

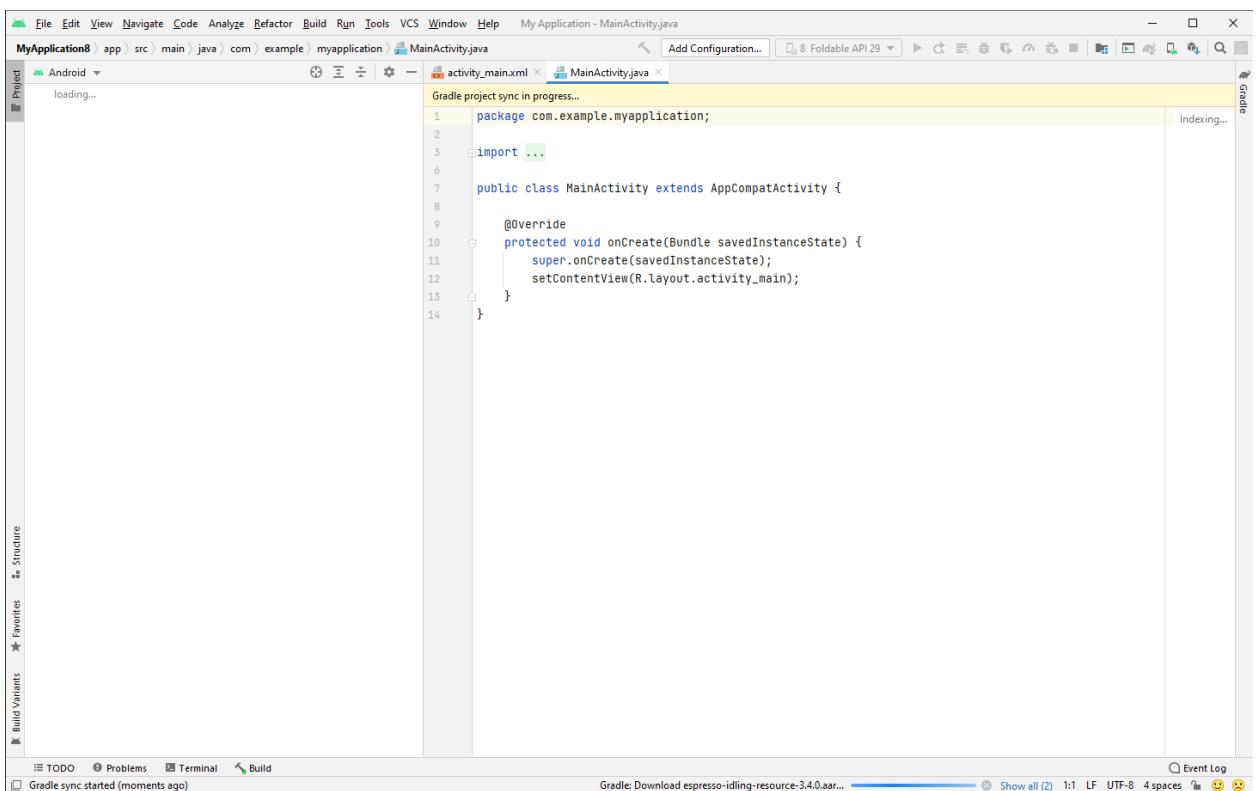


Рисунок 5 – Рабочая область Android Studio

Основные файлы и папки проекта:

- manifests/AndroidManifest.xml - данный файл предоставляет основную информацию о программе системе и является центральным для приложения, в нём перечисляются все используемые Activity, атрибуты и разрешения, которыми обладает приложение, указывается его иконка, название и т.д.
- java/com.example.myapplication – содержит файлы с исходным кодом на языке Java. Именно в этой папке размещаются все классы, создаваемые в процессе разработки приложения
- res/ – содержит структуру папок ресурсов приложения, рассмотрим некоторые из них:
 - layout - в данной папке содержатся xml-файлы, которые описывают внешний вид форм и их элементов, пока там находится только activity_main.xml – интерфейс MainActivity;
 - values - содержит XML файлы, которые определяют простые значения таких ресурсов, как строки, числа, цвета, темы, стили, которые можно использовать в данном проекте.

При создании проекта сразу же открывается файл MainActivity.java, в котором находится исходный код главного класса проекта. Чтобы открыть интерфейс приложения, нужно два раза кликнуть на res/layout/activity_main.xml файл, содержащий интерфейс первого экрана (Activity) приложения, или просто кликнуть на соседнюю с MainActivity.java уже открытую закладку. В зависимости от версии Android Studio для отображения интерфейса приложения может понадобиться дополнительно скачать и установить различные компоненты. Поскольку ранее мы выбрали Empty Activity, в окне интерфейса приложения ничего нет, кроме TextView с надписью «Hello World!». Добавлять интерфейсные элементы можно двумя

способами: в графическом (справа вверху кнопка «Design») и текстовом (справа вверху кнопка «Code») представлениях (можно и совмещать – кнопка «Split»). Интерфейсные элементы расположены в контейнере Palette слева от отображения интерфейса приложения. В контейнере элементы интерфейса сгруппированы по категориям, например Common (наиболее используемые), Text и т.д. В другом контейнере ниже, Component Tree, расположено дерево всех элементов интерфейса, которые есть в данный момент. На экране уже есть компонент TextView со значением «Hello World!»! Если его не видно, значит, у вас возникли проблемы с рендерингом окна дизайна. Справа вверху есть восклицательный знак в красном кружке, если нажать его, вы увидите суть проблемы.

Нажмите один раз левой кнопкой мыши по компоненту TextView, и справа появится окно, в котором можно задать текст и id для данного элемента. Поле «text» уже заполнено фразой «Hello world!». Если нажать на многоточие справа от этой фразы, откроется окно ресурсов, показанное на рисунке 6. Дело в том, что в файле res/values/strings.xml хранятся все текстовые данные приложения, что позволяет довольно быстро изменять названия различных элементов, а также с лёгкостью создавать локализацию под различные языки. В это окно выводятся значения ресурсов, в том числе и из этого файла. Если нажать кнопку + – String Value слева вверху, можно с помощью диалоговых окон создать новый строковый ресурс (рисунок 7). После нажатия на кнопку ОК ссылка на ресурс автоматически добавляется в поле text элемента TextView.

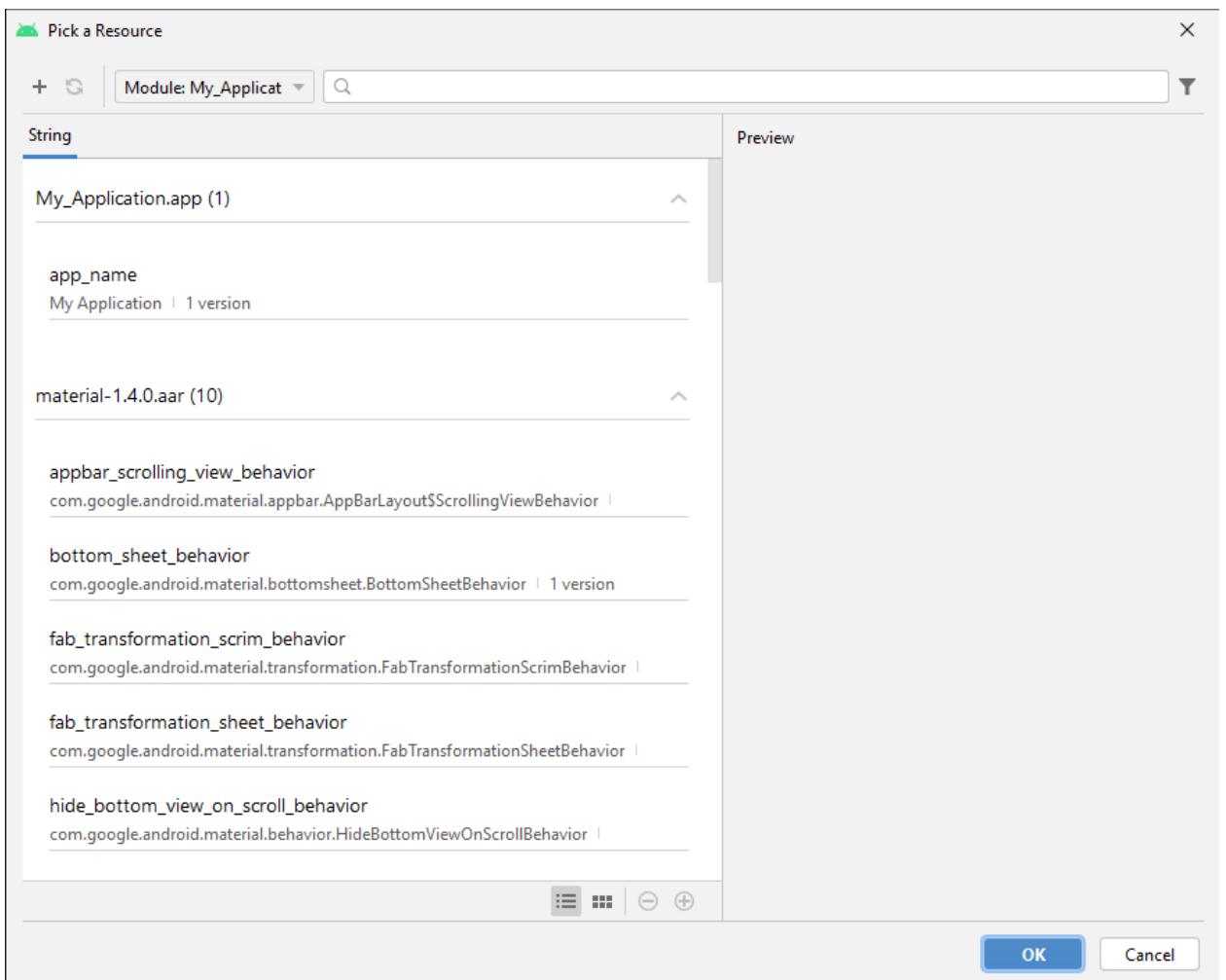


Рисунок 6 – Окно выбора ресурсов

Второй способ создания строкового ресурса – это редактирование непосредственно файла strings.xml. Для этого в файл res/values/strings.xml добавьте новую надпись с именем «fio», для этого необходимо написать следующее:

<string name="fio">Иванов Иван Иванович, гр. БПИ 171</string>

После того, как вы создали новую строковую переменную с помощью редактирования файла strings.xml, перейдите в файл activity_main.xml и измените название надписи на «@string/fio», а в поле id вставьте «fio» и нажмите Enter.

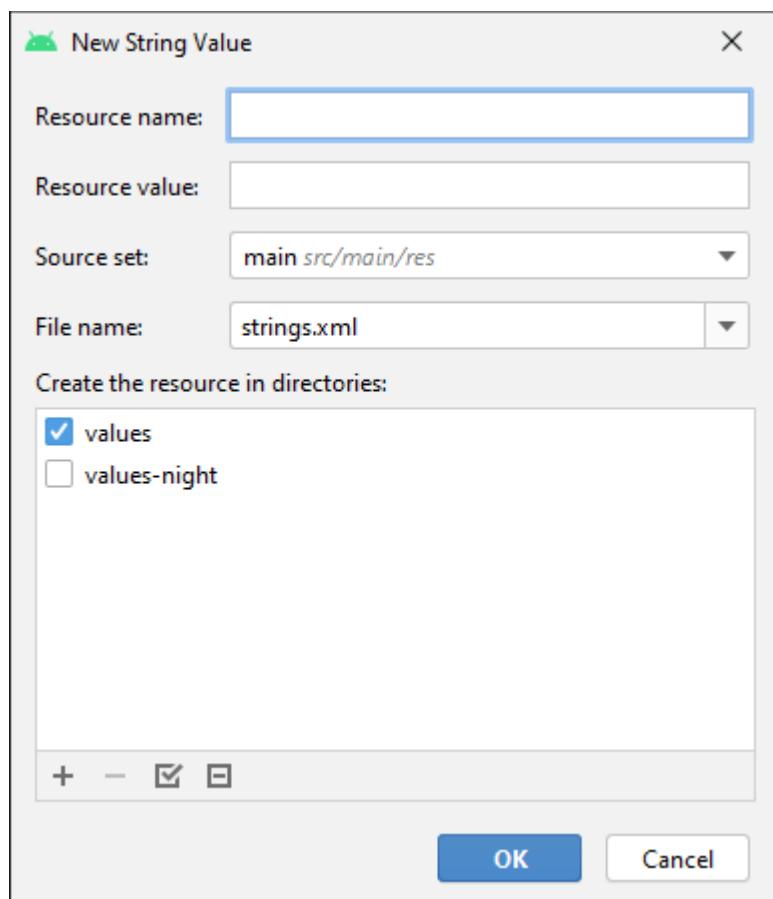


Рисунок 7 – Окно создания нового строкового ресурса

Если всё сделано правильно, то теперь на экране смартфона должны отобразиться ФИО и группа. Осталось запустить приложение на эмуляторе, но для начала его необходимо создать. Нажмите наверху Tools -> AVD Manager (AVD - Android Virtual Device), затем в появившемся окне (рисунок 8) «Create Virtual Device...», выберите модель телефона (Phone), которую хотите эмулировать, например, Nexus 5X, нажмите «Next», выберите образ и скачайте его, нажав на Download. Лучше во избежание излишней "тормозухи" выбрать API 16 (Android 4.1, armeabi-v7a) на закладке Other Images (рисунок 9), однако если у вас мощный компьютер, да ещё и на основе процессора Intel, можно выбрать образы из вкладки Recommended или тот, который соответствует вашему реальному устройству под управлением Android. Закладка x86 Images служит для счастливых обладателей

комьютеров на базе процессоров Intel – для этих эмуляторов существует встроенное средство для десятикратного ускорения работы эмулятора (Intel HAXM – Hardware Accelerated Execution Manager). Однако, начиная с версии 4.1, даже если у вас процессор AMD, работа эмулятора будет ускорена за счёт ресурсов видеокарты, поэтому не важно, на базе какого процессора построен ваш компьютер или ноутбук. Вариант образа, у которого в скобках написано Google API, служит для запуска приложений, использующих Google Play Services; нам это пока не требуется. Если у эмулятора есть значок Google Play, значит на эмуляторе можно будет использовать этот магазин для установки новых приложений, и на эмуляторе установлены Google Play Services.

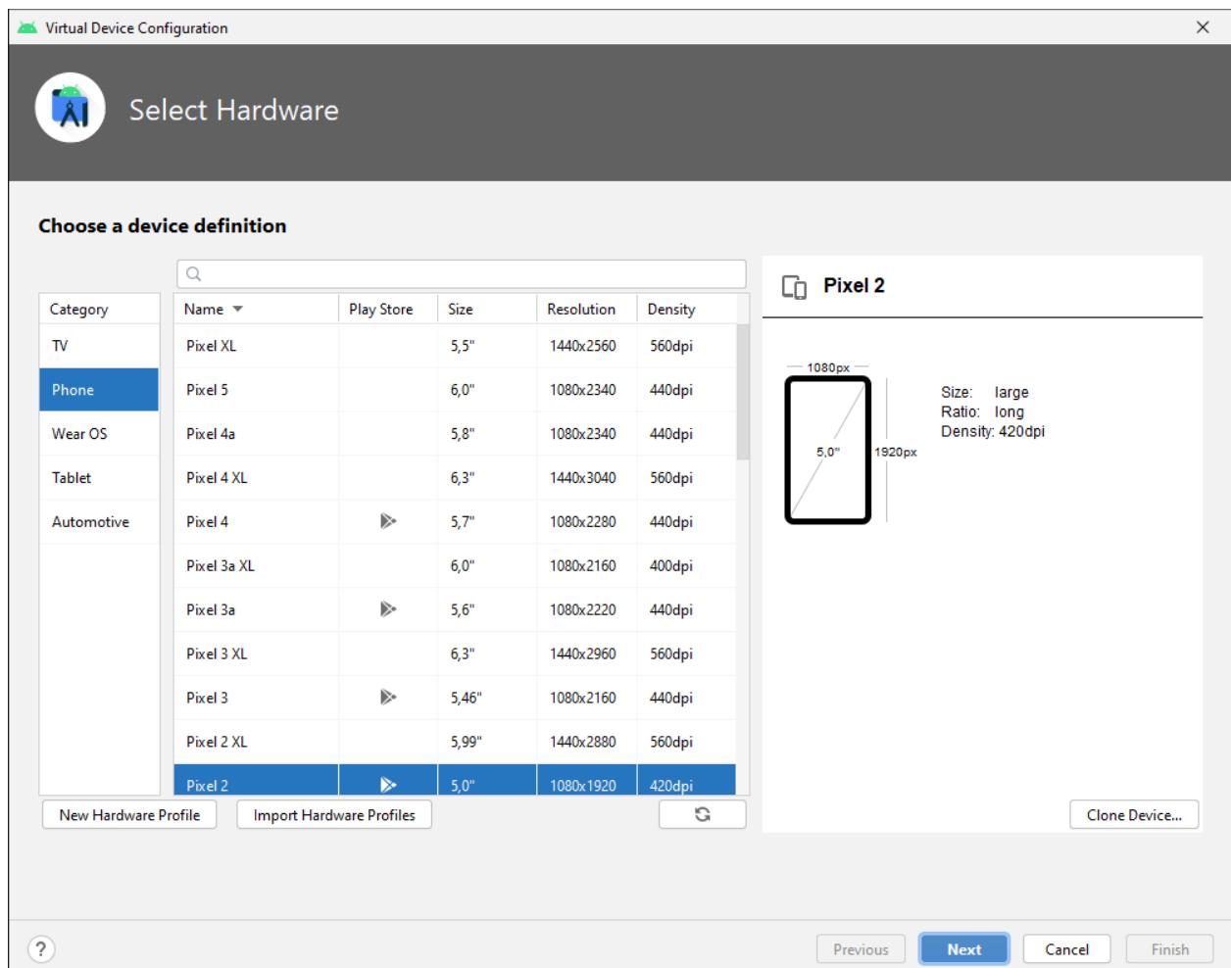


Рисунок 8 – Окно создания эмулятора (AVD) в Android Studio

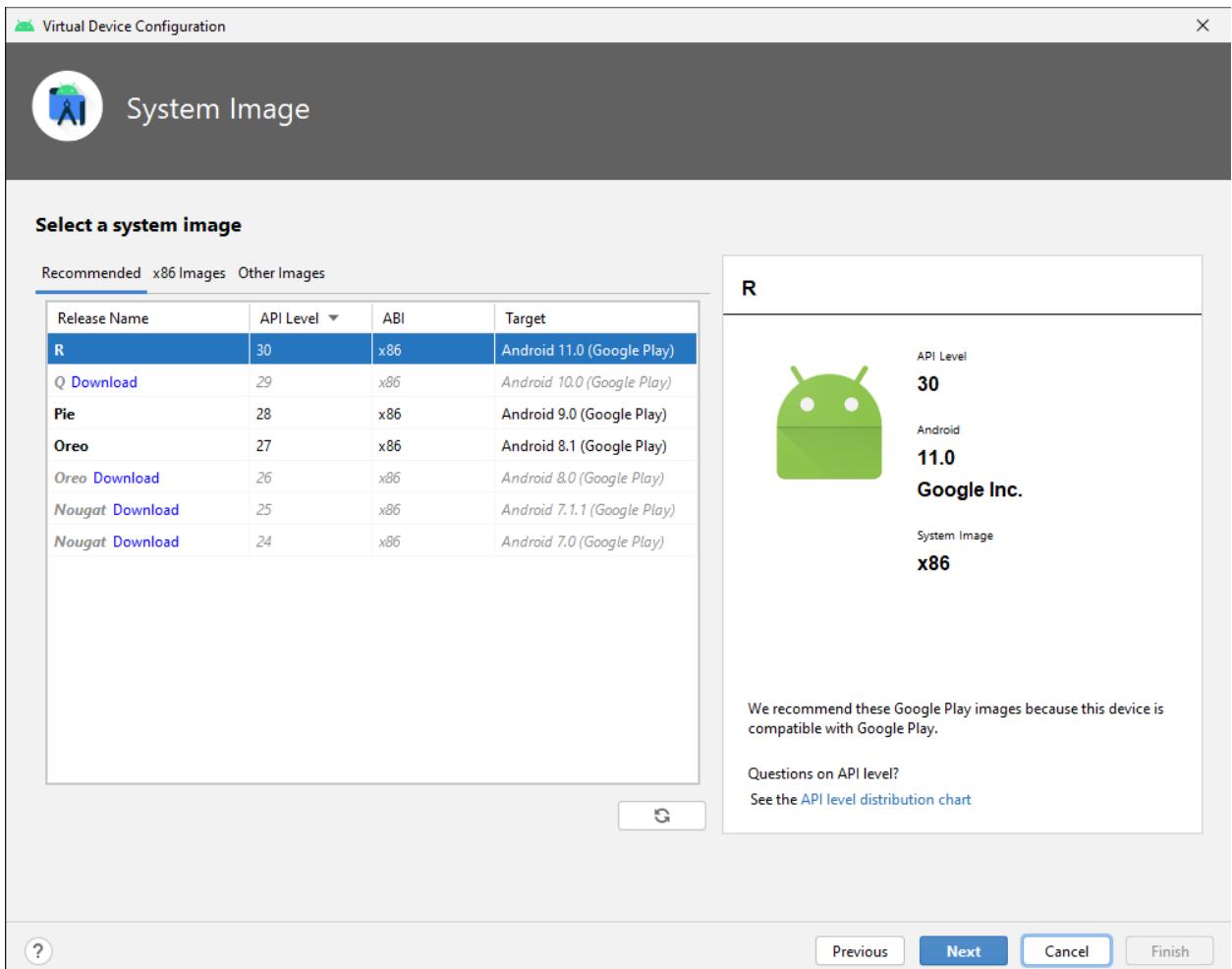


Рисунок 9 – Окно выбора и скачивания образа эмулятора

После скачивания образа устройства и нажатия кнопки Finish в окне скачивания выберите из списка скачанный образ, который теперь выделяется жирным шрифтом (см. рисунок 9). Нажмите «Next», в следующем окне нажмите «Show Advanced Settings» в левом нижнем углу и в поле камера выберите оба поля «Emulated» (рисунок 10). Остальное менять нет необходимости, нажмите «Finish» и закройте окно AVD Manager. Осталось выбрать эмулятор из списка слева от кнопки «Run ‘app’», нажать эту зелёную кнопку «Run ‘app’» вверху справа (или Shift+F10) и ждать загрузки эмулятора. После того, как эмулятор загрузится, разблокируйте экран смартфона (или нажмите кнопку OK), и сразу же должно запуститься приложение, как показано на рисунке 11. Справа располагается панель

управления эмулятором, где можно менять его ориентацию в пространстве, управлять звуком, нажимать аппаратные кнопки, вводить координаты локации и многое другое.

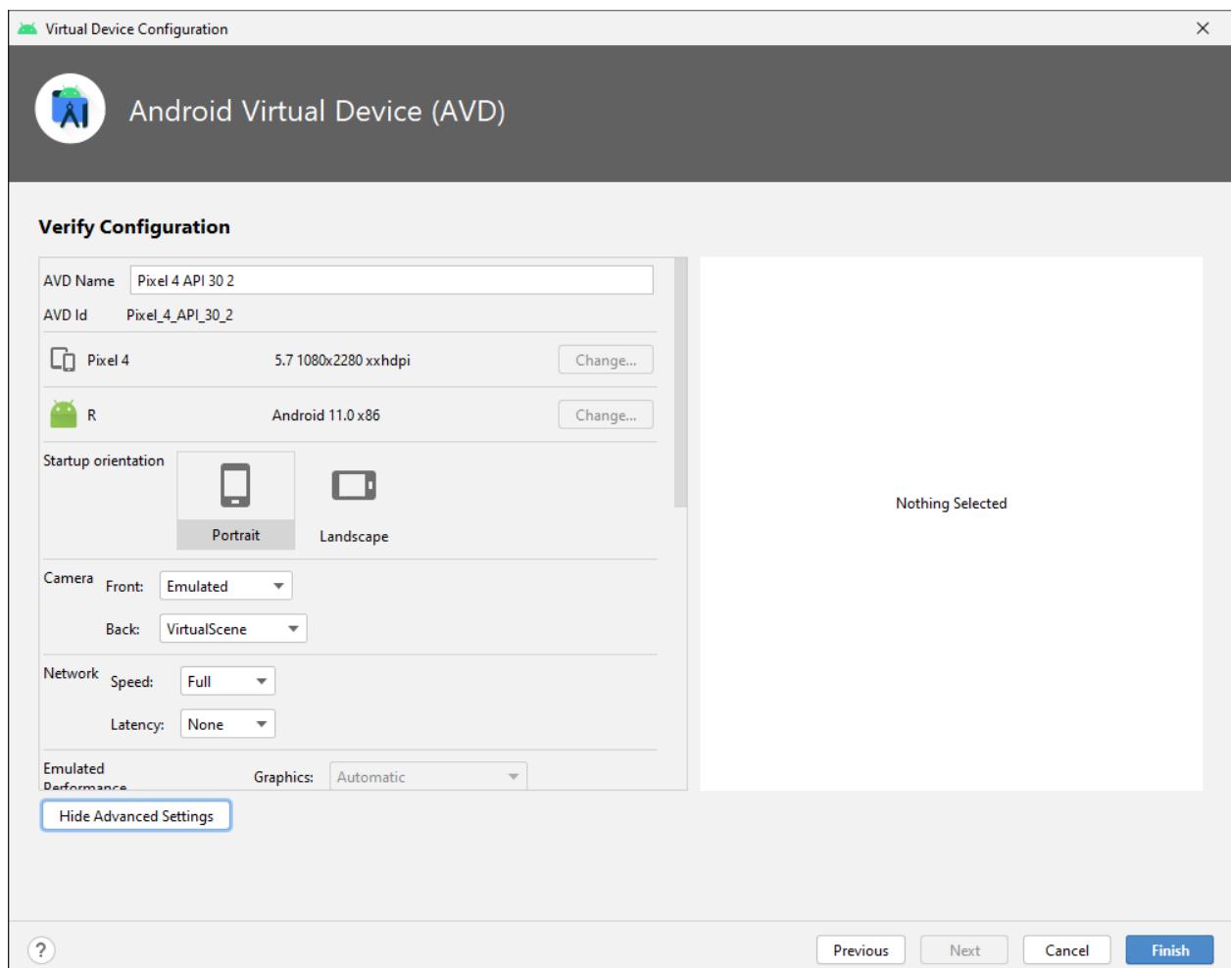


Рисунок 10 – Окно дополнительных свойств эмулятора

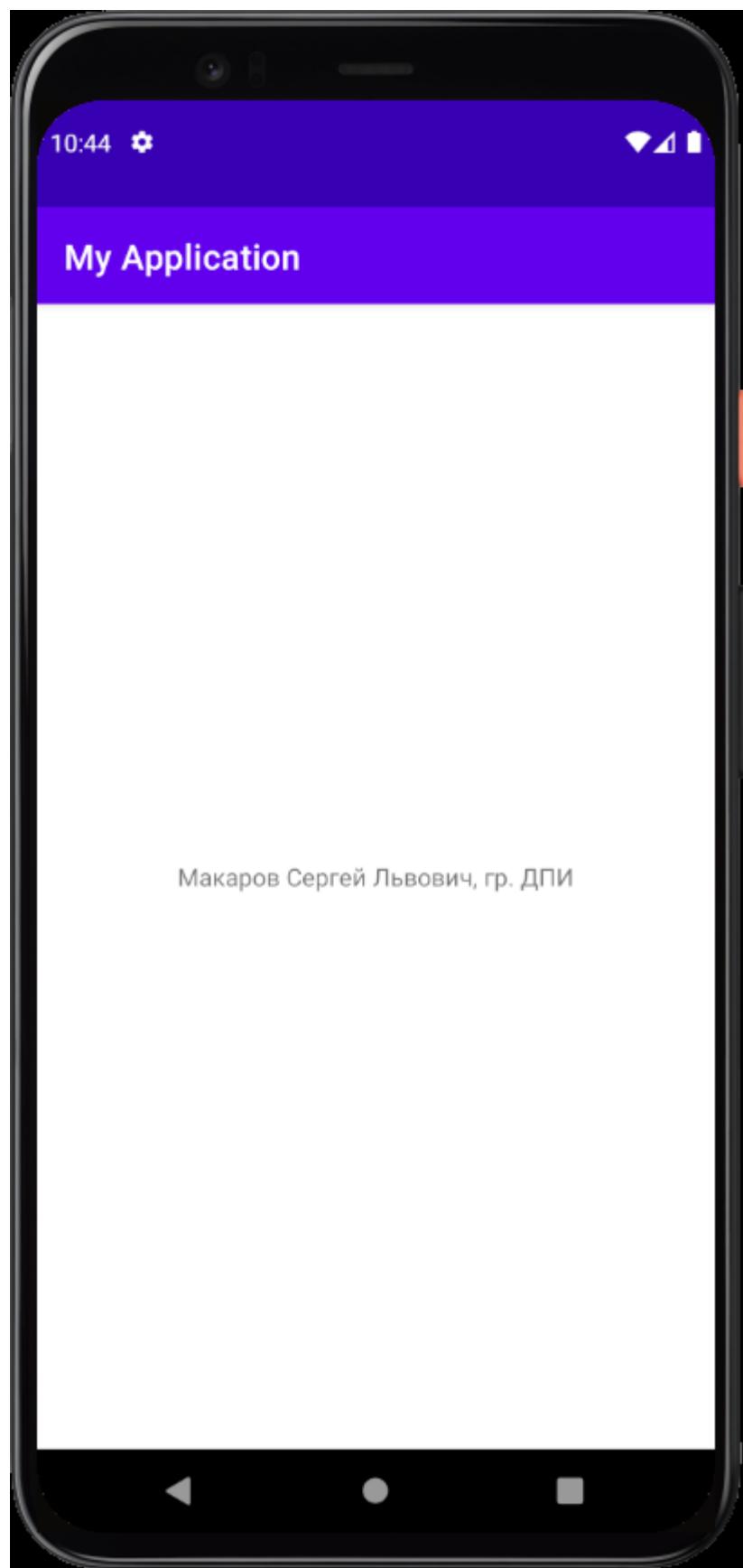


Рисунок 11 – Эмулятор Android с запущенным приложением

Теперь необходимо проверить установленные компоненты SDK. Для этого перейдите Tools -> SDK Manager и сверьтесь со следующим списком. Если каких-то компонентов не хватает, то следует их установить.

- SDK Tools
 - Android SDK Tools
 - Android SDK Platform-tools
 - Android SDK Build-tools (последней версии)
 - Google Play services
 - Android Support Repository
 - Google Repository
 - Intel x86 Emulator Accelerator (HAXM installer)
 - Google USB Driver
 - Google Play Instant Development SDK (если хотите использовать Android Instant Apps)
- SDK Platforms
 - Android API 16 (или той версии, которая у вас на реальном устройстве)

Лабораторная работа №2. Работа с элементами

Задание: создать приложение, которое выводит в элемент TextView надпись, введённую пользователем в текстовом поле EditText после нажатия на кнопку Button. Помимо этого, в Activity должен быть TextView с ФИО студента и группой. Запустить на эмуляторе и убедиться, что всё работает.

Создайте новый проект (File -> New -> New Project) с Empty Activity, удалите стандартный TextView с фразой «Hello World!» и поместите поле EditText. Для этого из панели слева (Palette) перетащите на экран смартфона объект под названием Plain Text из категории Text. Plain Text - самый простой вариант текстового поля для пользовательского ввода. Затем расположите поле TextView и элемент Button. Пример расположения элементов показан на рисунке 12. Не забудьте про элемент с ФИО и группой.

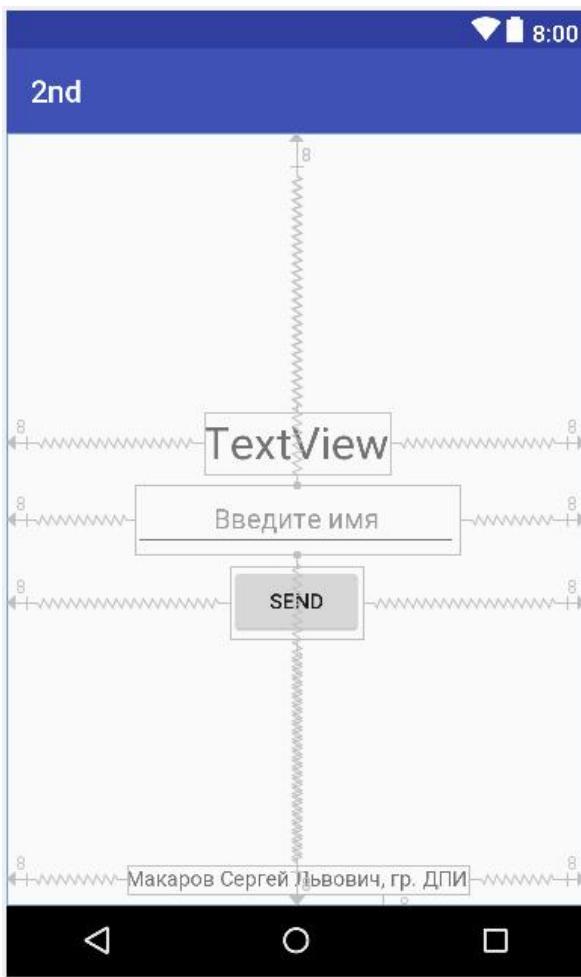


Рисунок 12 – Расположение элементов на экране приложения

Что касается интерфейса, нужно помнить, что если вы работаете с `ConstraintLayout`, вам нужно в окне дизайна привязать используемые 3 элемента интерфейса к границам экрана и связать их между собой, иначе весь интерфейс будет в одной куче в левом верхнем углу экрана, как будто вы используете `FrameLayout`.

В поле `EditText` мы будем вводить текст, и после нажатия на кнопку этот текст будет отображаться в элементе `TextView`. Дополнительно в поле `EditText` можно указать какой-нибудь фоновый текст, чтобы пользователь видел, куда же ему вводить информацию. Чтобы добавить фоновый текст в `EditText` необходимо выделить этот элемент и в правой нижней панели его свойств найти пункт `hint`, туда можно вписать любую фразу. Можно написать

текст просто словами, либо же через файл strings.xml. В нём достаточно добавить строчку:

```
<string name="hint_editText">Введите имя</string>
```

А в поле hint нужно сослаться на данную строчку следующим образом:

```
@string/hint_editText
```

Можете поступать так, как вам удобнее, но всё же правильнее писать все текстовые поля в файле strings.xml. Затем также переименуйте кнопку, и переходим к написанию кода в файл MainActivity.java.

В объявлении класса MainActivity необходимо убедиться, что класс наследуется от AppCompatActivity. Так нужно будет делать всегда, если в лабораторной работе не указано иное.

Далее в методе onCreate необходимо написать код для взаимодействия элементов. Метод onCreate выполняется всегда при загрузке Activity. Для начала определим в коде нужные нам элементы, с которыми мы будем работать, для этого в методе onCreate пишем следующее:

```
final TextView textView = (TextView) findViewById(R.id.textView);
final EditText editText = (EditText) findViewById(R.id.editText);
Button button = (Button) findViewById(R.id.button);
textView.setText(" "); // чтобы при старте не был виден
```

Так как перенос текста из EditText в TextView должен происходить после нажатия на кнопку, то для кнопки необходимо создать событие onClick. Для этого напишите следующее:

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
```

```

public void onClick(View v) {
    textView.setText("Привет, " +
editText.getText().toString() + " !");
}
);

```

На самом деле среда разработки может написать большую часть кода за вас, просто начните писать button.setOnClickListener(), а дальше можно нажать Enter, так как у вас в подсказке будет нужная вам конструкция, получится следующее:

```
button.setOnClickListener();
```

Теперь в скобках начните писать «new View.OnClickListener()» и на первой строчке в подсказке будет OnClickListener{...} (android.view.View), нажимайте Enter и основная часть конструкции написана автоматически:

```

button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
    }
});

```

Осталось в метод onClick написать нужный нам код:

```
textView.setText("Привет, " + editText.getText().toString() + "!");
```

Всё довольно легко и к данным подсказкам легко привыкнуть. В данной строчке кода элементу textView присваивается текст «Привет, [текст из editText]!». Запустите приложение и проверьте его работу, введя имя и нажав на кнопку. Результат должен быть примерно таким, как изображено на рисунках 13 и 14.

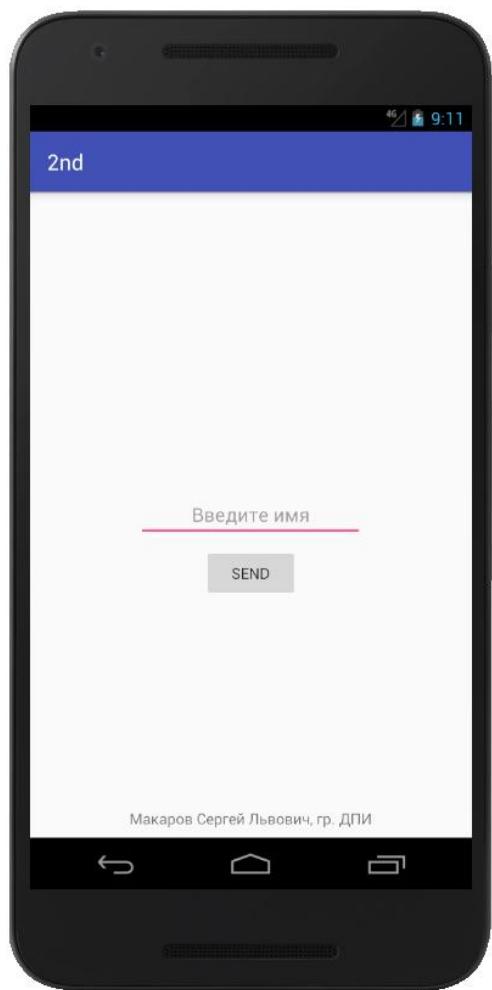


Рисунок 13 – Вид до ввода текста

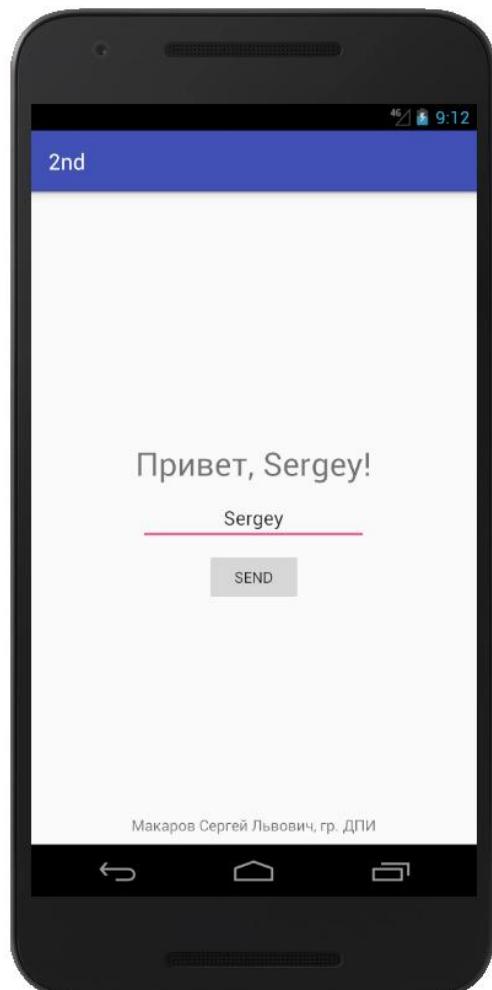


Рисунок 14 – Вид после ввода текста

Лабораторная работа №3. Работа с экранами

Задание: создать приложение, которое состоит из нескольких activities. Первое activity содержит элемент TextView с названием или номером activity, текстовое поле EditText для ввода какой-то информации, кнопку Button с названием "Next" или "Перейти на 2 activity/экран/окно" или просто "2". Помимо этого, в 1 activity должен быть TextView с ФИО студента и группой. После нажатия на эту кнопку происходит переход на второе activity, где содержится TextView с названием или номером activity, TextView с надписью что-то вроде "В первом окне вы напечатали:" и под ним - ещё один TextView с содержимым EditText с первого activity, и, разумеется, кнопка "1" или "Вернуться на 1 экран" или "Вернуться к вводу текста", нажав на которую пользователь может перейти обратно к 1 activity. Запустить на эмуляторе и убедиться, что всё работает.

Пока обновлялась эта методичка, вышла новая версия 3.2 Android Studio, поэтому первое, что нужно сделать, это обновить среду (Help – Check for Updates...), включая системы сборки Gradle, Android SDK Platrofm-Tools и т.д. После апдейта уже нет проблем с интерфейсом, описанных в начале лабораторной работы №1.

Создайте новый проект с Empty Activity и удалите стандартный TextView с фразой «Hello World!». Теперь в первом activity (файл main_activity.xml) расставьте элементы TextView с текстом «Activity 1», EditText с фоновым текстом «Введите имя» и кнопку Button с текстом «Go». Не забудьте свои ФИО и группу. Пример расстановки элементов показан на рисунке 18.

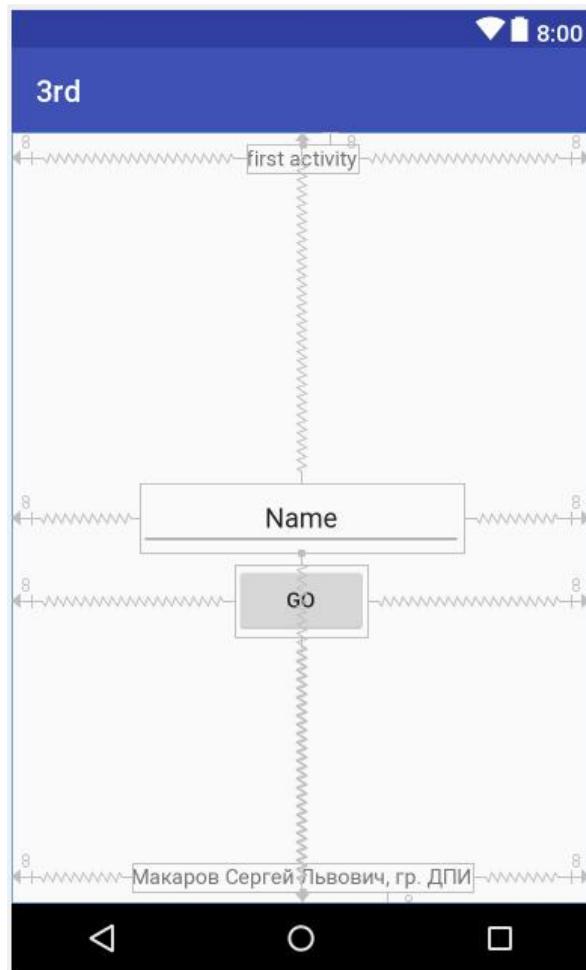


Рисунок 18 – Пример расстановки элементов в файле activity_main.xml

Начнём с интерфейса. Добавим второе activity, для этого слева наверху нажмите по папке app правой кнопкой мыши и выберите пункт New -> Activity -> Empty Activity, как показано на рисунке 19.

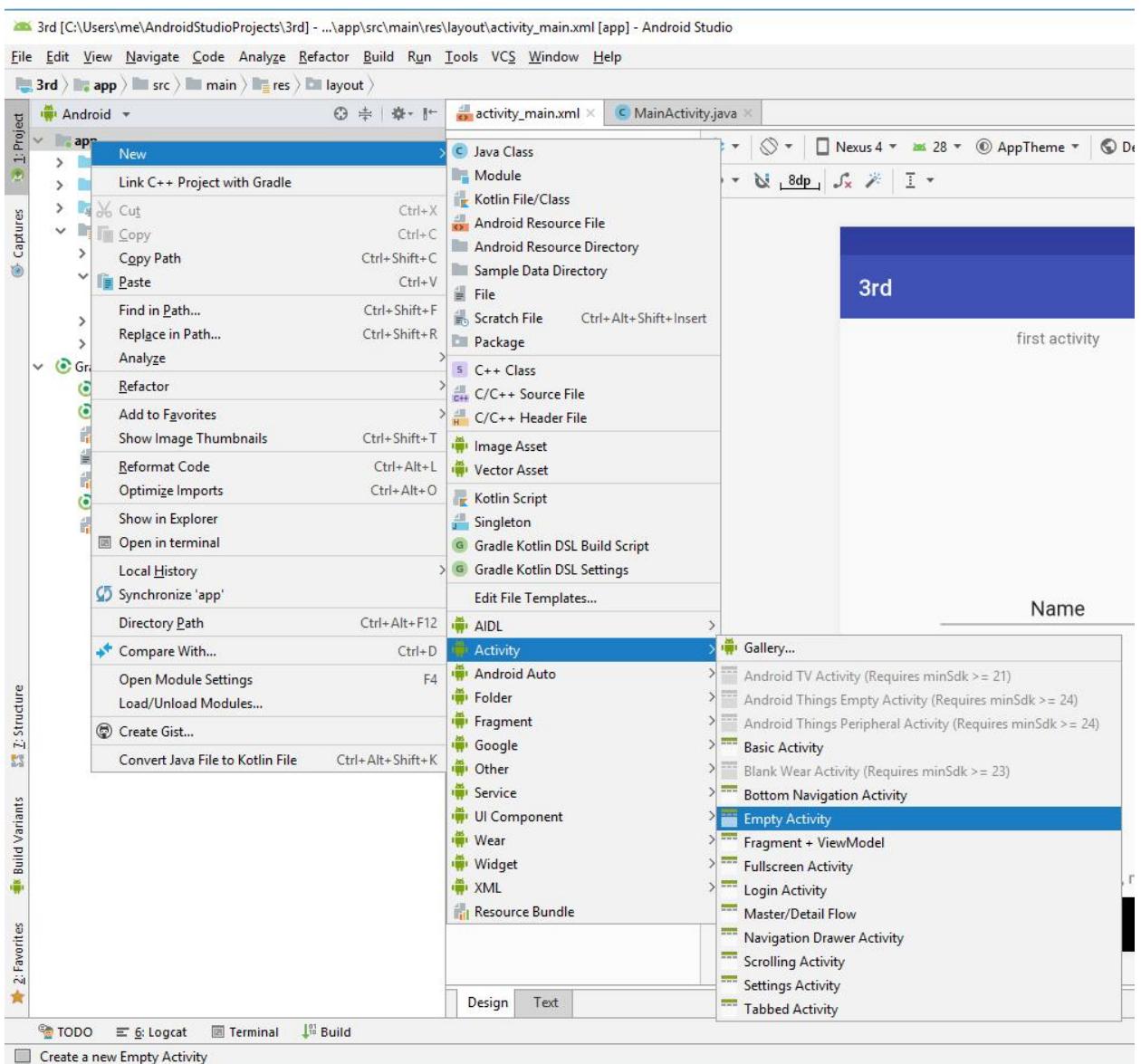


Рисунок 19 – Добавление нового Activity

После этого появится стандартное окно создания activity, можете оставить все названия по умолчанию и нажать Finish, либо поменять название в поле Activity Name на «SecondActivity», остальные поля подстроятся под это название автоматически. Таким образом, создалось два новых файла: SecondActivity.java и activity_second.xml.

Перейдите в файл activity_second.xml и расставьте там следующие элементы: TextView с текстом «Activity 2», TextView с текстом «Вы ввели:», TextView без текста и кнопку Button для перехода обратно в первое activity. Пример расстановки элементов показан на рисунке 20.

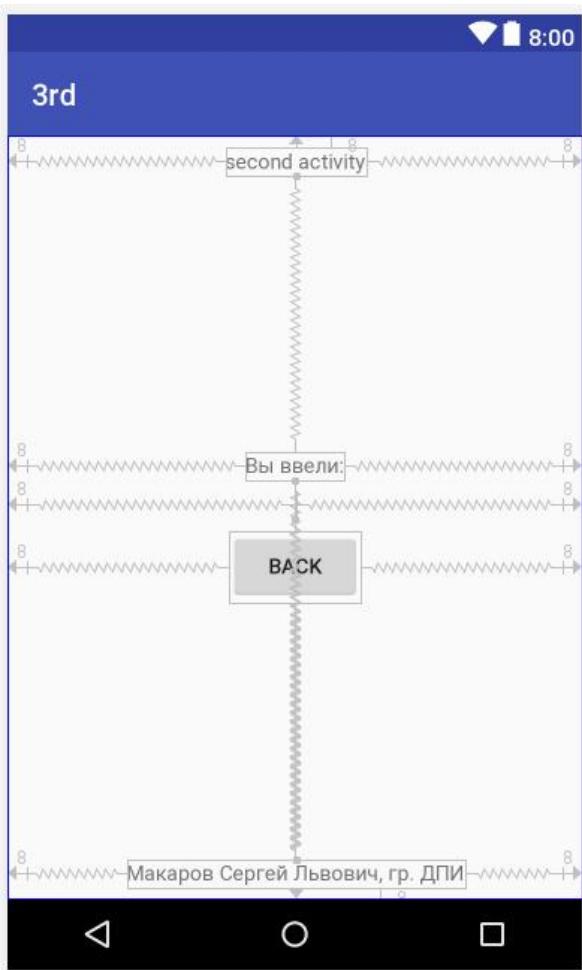


Рисунок 20 – Пример расстановки элементов в файле activity_second.xml

После расстановки элементов в двух activity необходимо написать код для взаимодействия элементов с activity. Перейдите в файл MainActivity.java и создайте переменную:

```
public static String text2remember;
```

В ней будет храниться текст из поля EditText в первом activity и использоваться из неё в TextView во втором activity. Далее в методе onCreate определите нужные элементы (не забудьте про зависимости, импортировать их можно по подсказке в Android Studio после написания этого кода):

```
final EditText editText = findViewById(R.id.editText);  
Button button = findViewById(R.id.button);
```

Заметьте, что если раньше каст (cast, или (EditText), (Button) и т.д.) перед методом findViewById нужно было указывать обязательно, как мы сделали в лабораторной работе №2, в новой версии Android Studio это необязательно. Теперь присвойте текст полю editText текст из переменной text2remember:

```
editText.setText(text2remember);
```

Делается это для того, чтобы при возвращении из второго activity в первое в поле TextView было написано то, что было написано до перехода во второе activity. Теперь создайте для кнопки listener и метод onClick (по подсказкам, показано во второй лабораторной работе) и внутри метода onClick напишите следующее:

```
text2remember = editText.getText().toString();
Intent intent = new Intent(MainActivity.this,
SecondActivity.class);
startActivity(intent);
```

Здесь переменной text2remember присваивается текст из поля EditText и определяется понятие Intent для запуска второго activity.

Теперь перейдите в файл SecondActivity.java и в методе onCreate определите элементы и присвойте полю TextView текст из переменной text2remember:

```
TextView textView = findViewById(R.id.textView6);
textView.setText(MainActivity.text2remember);
Button button = findViewById(R.id.button2);
```

Осталось только создать для кнопки listener и метод onClick для возвращения на первое activity. Создайте их и внутри метода onClick напишите:

```
Intent intent = new Intent(SecondActivity.this,  
MainActivity.class);  
startActivity(intent);
```

Не забывайте про импорт зависимостей, среда разработки вам об этом напомнит и поможет это сделать.

Заметьте, при создании второго activity в файл манифеста AndroidManifest.xml автоматически добавилась строка

```
<activity android:name=".SecondActivity"></activity>
```

Каждое activity приложения должно быть упомянуто в файле AndroidManifest.xml.

Теперь запустите приложение и протестируйте, всё должно работать. Пример работы приложения показан на рисунках 21 и 22.



Рисунок 21 – Вид первого activity

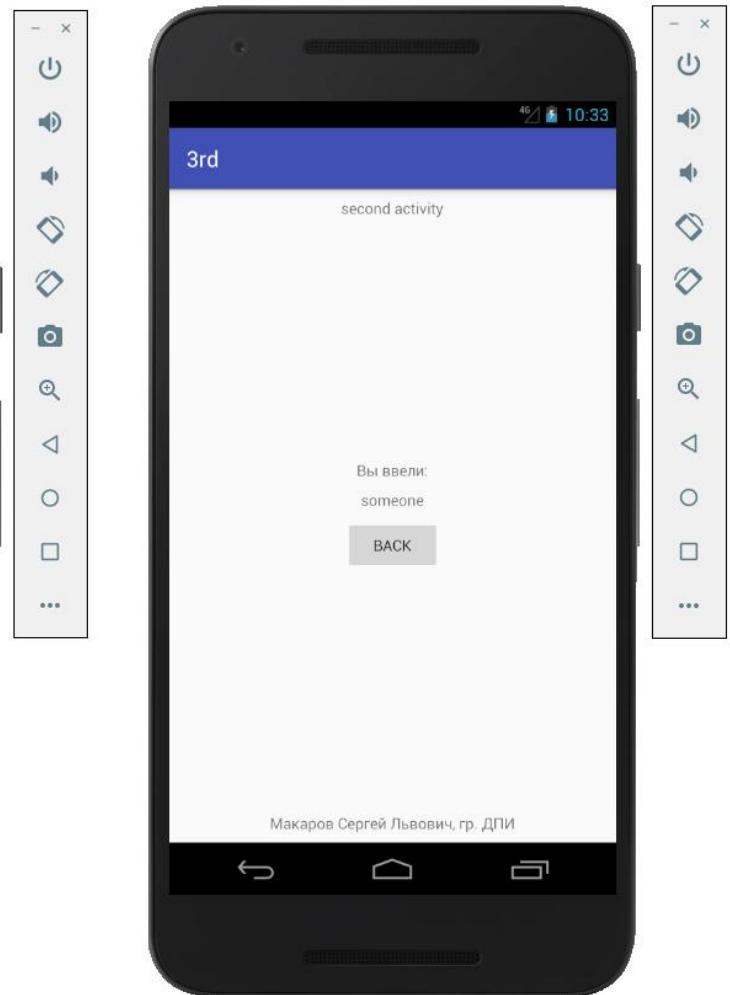


Рисунок 22 – Вид второго activity

Лабораторная работа №4. Стили и темы

Задание: написать приложение, работающее с разными темами/стилями. Сначала создать свой стиль и применить его к какому-нибудь интерфейсному элементу, затем - свою тему, которая применяется ко всем интерфейсным элементам. Приложение при этом должно выглядеть нестандартно. Запустить на эмуляторе и убедиться, что всё работает.

Создайте новый проект с Empty Activity, перейдите в activity_main.java, удалите стандартное текстовое поле с фразой «Hello world!» и поместите друг под друга 4 TextView, а затем поменяйте у них размер шрифта (свойство textSize) на 10, 14, 28 и 36 sp соответственно (не забудьте про ФИО и группу). Как мы видим, все они отличаются друг от друга. Бывают ситуации, когда всем элементам необходимо иметь один стиль (один размер шрифта, один цвет текста и т.д.). В данном случае можно поменять параметры вручную, а что делать, есть у вас 100 таких элементов? В таком случае можно создать стиль и применить его ко всем нужным элементам. Переходим в файл values/styles.xml и ниже существующего стиля под названием AppTheme создаём следующий стиль:

```
<style name="style1" parent="@android:style/TextAppearance">
    <item name="android:textColor">#000000</item>
    <item name="android:textSize">30sp</item>
    <item name="android:typeface">monospace</item>
</style>
```

TextAppearance – это стиль для текста в Android по умолчанию. Далее мы устанавливаем цвет текста чёрным, делаем текст 30 размера и тип указываем monospace. Теперь нужно применить этот стиль к созданным четырём элементам. Выделите все четыре TextView и в панели Attributes (справа) найдите свойство style, где пропишите стиль:

@style/style1

Также стиль можно выбрать из списка, нажав на 3 точки справа от текстового поля (в появившемся окне он будет последним в списке). Примените стиль. Вы увидите, что текстовые поля, кроме одного (у которого и так было 14sp по умолчанию, поэтому вы не стали ничего вводить в поле textSize), не поменялись. Это происходит потому, что поле textSize непустое и поэтому имеет приоритет над стилем. Если очистить все значения textSize у всех полей, то и все выделенные текстовые поля приведутся к единому виду.

Создадим подстиль, который будет соответствовать предыдущему стилю во всём, кроме размера шрифта. В файле styles.xml ниже стиля style1 напишите:

```
<style name="style1.bigfont">
    <item name="android:textSize">50sp</item>
</style>
```

Применяем данный подстиль к любому текстовому полю и видим, что размер шрифта увеличился.

Со стилями разобрались, теперь переходим к созданию собственной темы для приложения. В файле styles.xml уже создана базовая тема под названием AppTheme:

```
<style name="AppTheme"
parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>
```

Мы можем кастомизировать данную тему. Изменим родительскую тему приложения на следующую:

```
<style name="AppTheme"
parent="Base.Theme.AppCompat.Light.Dialog.Alert">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>
```

Теперь приложение будет иметь вид диалогового окна. Добавим какие-нибудь параметры, чтобы тема отличалась от стандартной:

```
<item name="android:background">#ff0e6647</item>
<item name="android:textColor">#fffffff</item>
<item name="android:textSize">32sp</item>
<item name="android:textAllCaps">true</item>
<item name="android:typeface">monospace</item>
```

Свойства довольно понятны из названий, поэтому расписывать их не имеет смысла. Удалите из текстовых полей предыдущие стили и увидите, что текстовые поля сразу подстроились под параметры из темы. Дело в том, что тема AppTheme уже прописана как базовая в приложении, поэтому любые изменения сразу же отображаются на всех элементах. В поле с ФИО можно поставить свойство gravity в значение center_horizontal.

Ну и в заключение переходим в файл MainActivity.java и в объявлении класса оставим следующее:

```
public class MainActivity extends Activity {
```

Делается это для того, чтобы в приложении не выводилась так называемая «шапка» с названием приложения.

Запустите проект на эмуляторе, он должен иметь вид, как показано на рисунке 23.

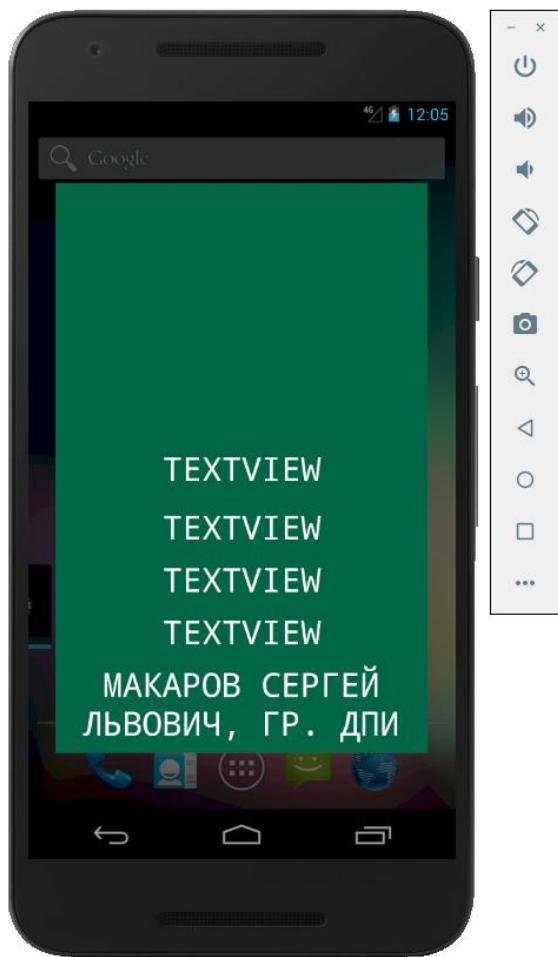


Рисунок 23 – Результат выполнения лабораторной работы №4

Приложение запустилось без «шапки» и в виде уведомления. Теперь придумайте собственную тему, чтобы всем не сдавать одно и то же приложение.

Лабораторная работа №5. Работа со списками

Задание: создать пользовательский список. Например, создать свой список в виде твиттера (картинка и текст), элементы которого просто статически задать в массиве (как и картинки).

Список можно создать, как минимум, двумя способами: с помощью устаревшего элемента ListView, или с помощью более сложного, но более надёжного и избавленного от недостатков элемента RecyclerView, появившегося с выходом Android Lollipop. Также можно использовать элемент GridView, но это уже на любителя. Рассмотрим сначала ListView, потом RecyclerView.

Создайте новый проект с Empty Activity, перейдите в файл activity_main.java и удалите стандартное текстовое поле с фразой «Hello World!». Для начала необходимо создать xml-файл, который будет содержать список. Нажмите правой кнопкой мыши по папке values и создайте файл так, как показано на рисунке 24.

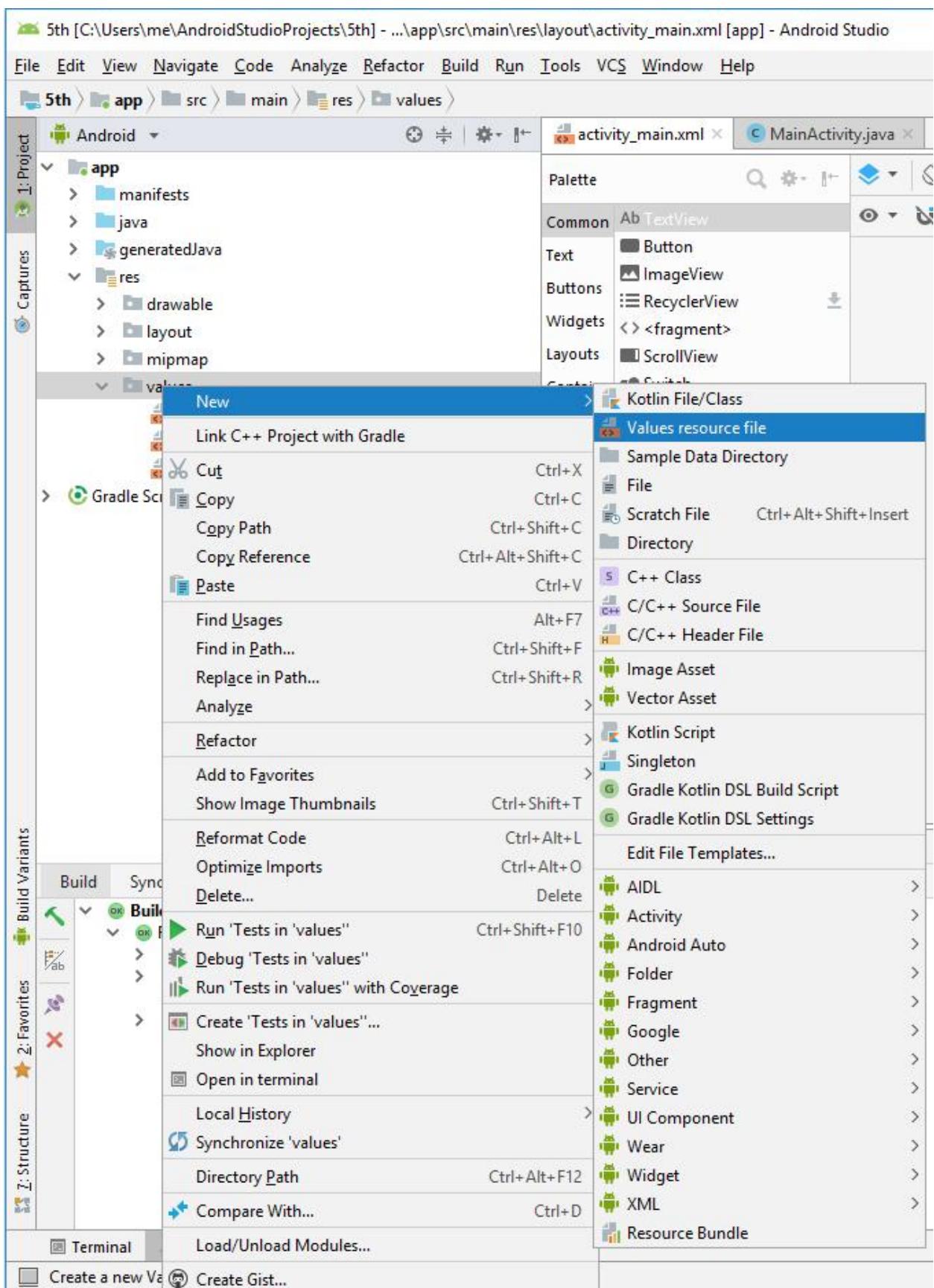


Рисунок 24 – Создание нового файла ресурсов

В появившемся окне оставьте все параметры по умолчанию и введите название файла, например, «images», после чего нажмите OK. После проделанных действий в папке values появился файл images.xml, в котором нужно будет создать список. Переходим в этот файл и внутри тэга resources пишем:

```
<string-array name="images">
    <item>Image 1</item>
    <item>Image 2</item>
    <item>Image 3</item>
    <item>Image 4</item>
</string-array>
```

В item можно использовать любые слова и фразы на любом языке. Теперь перейдите в файл MainActivity.java и исправьте объявление класса:

```
public class MainActivity extends ListActivity {
```

Не забудьте импортировать нужные классы, среда разработки подскажет об этом. Затем добавьте в метод onCreate следующий код:

```
setListAdapter(new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_1,
    getResources().getStringArray(R.array.images)) );
```

Теперь перейдите в файл activity_main.xml и из раздела Legacy панели Palette перенесите объект ListView на экран. Обратите внимание, что ListView необходимо присвоить id «@android:id/list». Сделать это можно в текстовом представлении, добавив в тэг ListView следующую строчку:

```
android:id="@android:id/list"
```

Запустите приложение и увидите список, который был написан в файле images.xml. Однако, чтобы список был с картинками, нужно воспользоваться Custom Lists. Для этого в папке layout необходимо добавить ещё один файл. Нажмите правой кнопкой мыши по папке layout и создайте новый Layout resource file. В появившемся окне оставьте все параметры по умолчанию и введите название файла «list_item». Переходим в этот файл и помещаем элементы ImageView и TextView на своё усмотрение. Пример расположения элементов в данном файле показан на рисунке 25. Так будет выглядеть один элемент нашего списка.

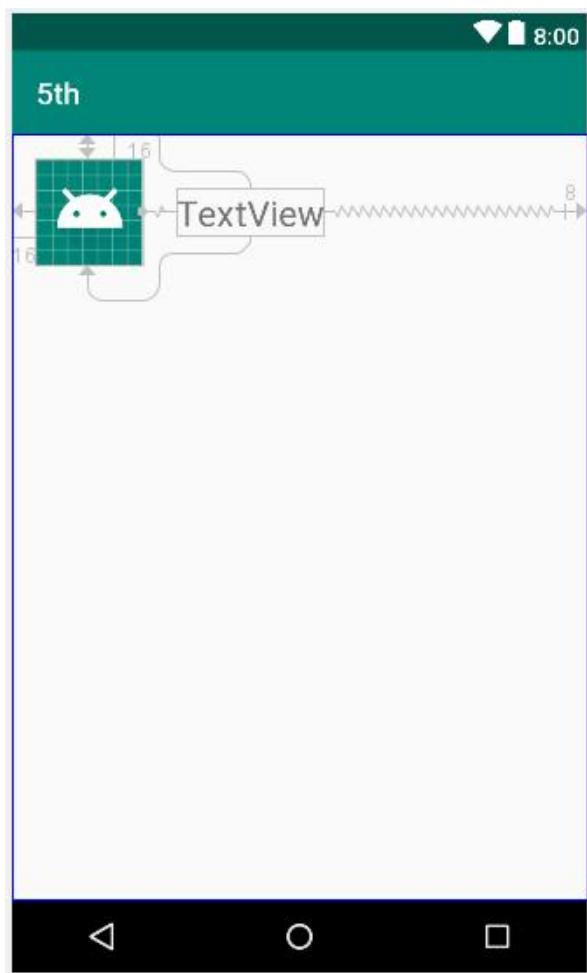


Рисунок 25 – Пример расположения элементов в файле list_item.xml

Теперь вернитесь в файл MainActivity.java и измените код в методе onCreate на следующий:

```
setListAdapter(new MyAdapter(this,
    android.R.layout.simple_list_item_1, R.id.textView,
    getResources().getStringArray(R.array.images)));
```

Далее создайте класс MyAdapter внутри класса MainActivity:

```
public class MyAdapter extends ArrayAdapter<String> {
    public MyAdapter(Context context, int resource, int textViewResourceId, String[] string) {
        super(context, resource, textViewResourceId, string);
    }

    @Override
    public View getView(int position, View convertView,
    ViewGroup parent) {
        LayoutInflator inflater = (LayoutInflator)
        getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        View row = inflater.inflate(R.layout.list_item, parent,
        false);
        String[] items =
        getResources().getStringArray(R.array.images);
        ImageView image = (ImageView)
        row.findViewById(R.id.imageView);
        TextView text = (TextView)
        row.findViewById(R.id.textView);
        text.setText(items[position]);
        return row;
    }
}
```

Но это ешё не всё. Необходимо создать конструкцию switch-case, чтобы в каждой строке была соответствующая ей картинка. Перед return row вставьте следующий код:

```
switch (items[position]) {
    case "Image 1":
        image.setImageResource(R.drawable.image1);
        break;
    case "Image 2":
        image.setImageResource(R.drawable.image2);
        break;
    case "Image 3":
        image.setImageResource(R.drawable.image3);
        break;
    case "Image 4":
```

```
    image.setImageResource (R.drawable.image4) ;  
    break;  
}
```

Теперь осталось поместить любые изображения в папку drawable. Изображения либо должны называться image1.jpeg, image2.jpeg и т.д., либо в конструкции switch-case необходимо поменять путь до картинок в строке R.drawable.image1 и т.д. Не забудьте про текстовое поле с ФИО. Запустите приложение и увидите собственный список с картинкой и текстом. Пример показан на рисунке 26.

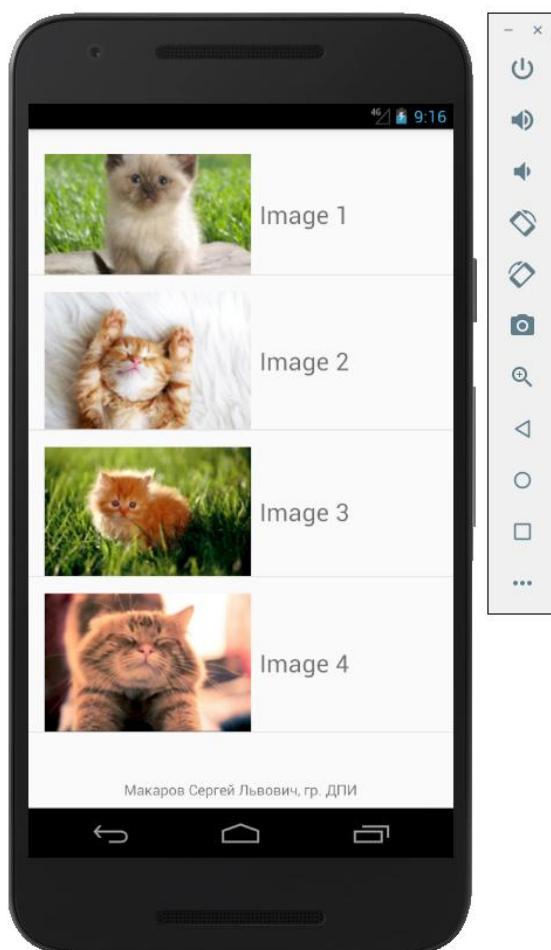


Рисунок 26 – Пример выполнения лабораторной работы №5

Теперь рассмотрим более мощный и современный элемент RecyclerView¹. Этот элемент создаёт ровно столько элементов списка, сколько видно на экране пользователю, поэтому он значительно экономит память, если список большой – а в приложениях списки, как правило, длинные. Когда пользователь прокручивает список вниз, верхний элемент уходит за пределы экрана и становится невидимым, при этом его содержимое очищается, а сам этот более ненужный на экране элемент помещается вниз экрана и заполняется новыми данными следующих элементов списка, т.е. переиспользуется (is being **recycled**). Общая модель работы RecyclerView изображена ниже.

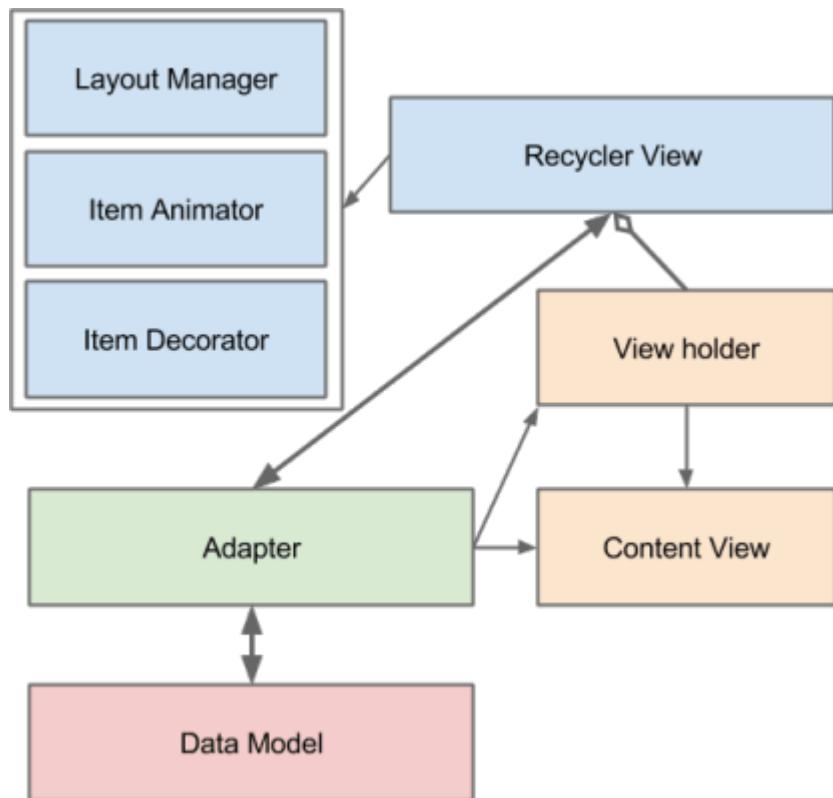


Рисунок 26.2 – Модель работы RecyclerView²

¹ <https://metanit.com/java/android/5.11.php>

² <http://developer.alexanderklimov.ru/android/views/recyclerview-kot.php>

Создадим новый проект с EmptyActivity. Первое, что нужно сделать – это добавить следующую зависимость в gradle-файл уровня модуля:

```
implementation 'androidx.recyclerview:recyclerview:1.2.1'
```

Далее нужно построить макет для отдельного элемента списка, но он у нас уже есть в проекте с ListView и лежит в файле list_item.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="16dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@mipmap/ic_launcher" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:text="TextView"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="@+id/imageView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.077"
        app:layout_constraintStart_toEndOf="@+id/imageView"
        app:layout_constraintTop_toTopOf="@+id/imageView" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Также нам понадобится файл images.xml в папке res/values, содержащий названия картинок, и сами эти картинки в папке drawable. Теперь удалим TextView "Hello world" и добавим компонент RecyclerView в activity_main.xml. Чтобы видеть дизайн элемента списка, который мы

создали в `list_item.xml`, добавим значение `@layout/list_item` в свойство `RecyclerView` под названием `listitem`. Кроме того, нужно добавить следующее свойство в код описания `RecyclerView`:

```
app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
```

Это нужно для того, чтобы `RecyclerView` знал, как отображать список. Оптимизируем наш предыдущий проект: создадим 2 дополнительных класса, обслуживающих `RecyclerView`. Первый класс назовём `Image`, он будет содержать основные методы для задания необходимых данных элемента списка:

```
public class Image {

    private String name; // название
    private int imageRes; // картинка

    public Image(String name, int image) {
        this.name=name;
        this.imageRes=image;
    }

    public String getName() {
        return this.name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getImageResource() {
        return this.imageRes;
    }

    public void setImageResource(int flagResource) {
        this.imageRes = flagResource;
    }
}
```

Второй класс назовём `CustomRecyclerAdapter`, он нужен для того, чтобы определить свой кастомный адаптер для `RecyclerView`, и чтобы в итоге связывать нужные данные и отображать список.

```

public class CustomRecyclerAdapter extends
RecyclerView.Adapter<CustomRecyclerAdapter.ViewHolder>{

    private final LayoutInflator inflater;
    private final List<Image> images;

    CustomRecyclerAdapter(Context context, List<Image> images) {
        this.images = images;
        this.inflater = LayoutInflator.from(context);
    }

    public void onBindViewHolder(CustomRecyclerAdapter.ViewHolder holder, int
position) {
        Image image = images.get(position);
        holder.text.setText(image.getName());
        holder.image.setImageResource(image.getImageResource());
    }

    @NonNull
    @Override
    public CustomRecyclerAdapter.ViewHolder onCreateViewHolder(@NonNull
 ViewGroup parent, int viewType) {
        View view = inflater.inflate(R.layout.list_item, parent, false);
        return new ViewHolder(view);
    }

    @Override
    public int getItemCount() {
        return images.size();
    }

    public class ViewHolder extends RecyclerView.ViewHolder {
        final ImageView image;
        final TextView text;
        ViewHolder(View view) {
            super(view);
            image = view.findViewById(R.id.imageView);
            text = view.findViewById(R.id.textView);
        }
    }
}

```

И теперь, когда у нас есть всё необходимое, в MainActivity.java достаточно добавить код, наполняющий с помощью кастомного адаптера список RecyclerView. Данные добавляются в методе setData():

```

public class MainActivity extends AppCompatActivity {

    ArrayList<Image> images = new ArrayList<Image>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

```

```
        setData();
        RecyclerView recyclerView = findViewById(R.id.review1);
        CustomRecyclerAdapter adapter = new CustomRecyclerAdapter(this,
images);
        recyclerView.setAdapter(adapter);
    }

    private void setData() {

        images.add(new Image ("cat1", R.drawable.image1));
        images.add(new Image ("cat2", R.drawable.image2));
        images.add(new Image ("mycat", R.drawable.image3));
        images.add(new Image ("nyancat", R.drawable.image4));

    }
}
```

Таким образом, получим тот же список из котов, но теперь он пролистывается вниз, каждый кот расположен на отдельной странице, и список можно значительно расширить.

Лабораторная работа №6. Работа с анимацией

Задание: создать приложение, содержащее анимированные интерфейсные элементы (например, увеличивающиеся при клике на них кнопки, вращающиеся TextView и т.д.).

Самым простым видом анимации является покадровая анимация, когда картинки (кадры) сменяют друг друга, создавая эффект движения или изменения изображения. Это слишком простой, но утомительный процесс, поэтому рассмотрим анимацию другого типа, так называемую tween-анимацию. Создайте новое приложение с Empty Activity, перейдите в файл activity_main.xml, удалите текстовое поле и поместите в центр экрана кнопку, а внизу – как обычно, текстовое поле с ФИО. Затем необходимо создать новый xml-файл для описания анимации. Для этого нажмите правой кнопкой мыши по папке res и создайте новый файл Android resource с параметрами, которые изображены на рисунке 27.

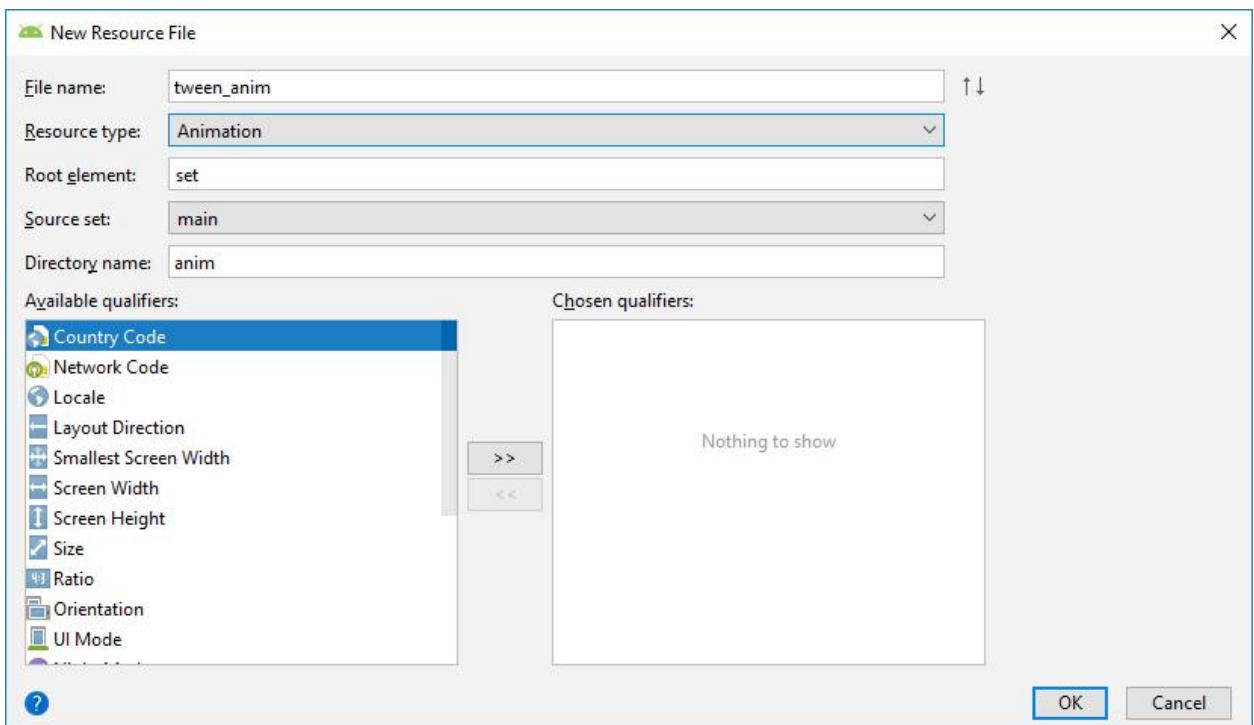


Рисунок 27 – Создание файла анимации

После данных действий в папке res появится новая папка anim с файлом tween_anim.xml внутри. Переходим в этот файл и добавляем следующий код внутрь тэга set:

```
<scale
    android:fromXScale="2.0"
    android:toXScale="0.5"
    android:fromYScale="2.0"
    android:toYScale="0.5"
    android:pivotX="50%"
    android:pivotY="50%"
    android:duration="2000" />
```

Здесь описывается уменьшение объекта с длительность анимации в 2 секунды (`android:duration`). Так как анимация будет отображаться на кнопке, нужно будет вернуть её в исходный размер, для этого добавляем ещё немного кода:

```
<scale
    android:fromXScale="0.5"
    android:toXScale="2.0"
    android:fromYScale="0.5"
    android:toYScale="2.0"
    android:pivotX="50%"
    android:pivotY="50%"
    android:duration="2000"
    android:startOffset="2000" />
```

Теперь кнопка сможет обратно увеличиться до своих размеров, анимация увеличения также будет длиться 2 секунды, но она будет отрабатывать не сразу, а с задержкой в 2 секунды (`android:startOffset`). То есть сначала идёт анимация уменьшения объекта, а затем анимация увеличения. Ну и в заключение добавим анимацию вращения кнопки:

```
<rotate
    android:pivotX="50%"
    android:pivotY="50%"
    android:fromDegrees="0"
    android:toDegrees="360"
```

```
    android:duration="2000"
    android:startOffset="4000" />
```

Анимация вращения будет ждать 4 секунды (пока отработают первые две анимации), а затем за 2 секунды повернёт кнопку на 360 градусов по часовой стрелке.

Ещё одним важным свойством анимации является параметр `interpolator`. Интерполятор позволяет ускорить или замедлить выполнение анимации на устройстве, а также добавить некоторые эффекты. Добавим следующий код в конец открывающего тэга `set`:

```
    android:interpolator="@android:anim/bounce_interpolator"
```

Набрав `@android:anim` в кавычках, можно увидеть, какие типы интерполяторов бывают, см. рисунок 28. Можно поэкспериментировать с разными типами интерполяторов.



Рисунок 28 – Варианты интерполяторов

Итак, в нашем случае открывающий тэг set должен выглядеть следующим образом:

```
<set
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/bounce_interpolator">
```

С описанием анимации закончили, теперь необходимо перейти в файл MainActivity.java и выполнить анимацию после нажатия на кнопку. Для этого в методе onCreate создаём ссылку на кнопку:

```
final Button button = findViewById(R.id.button);
```

И создаём для неё listener и onClick. В onClick добавляем анимацию:

```
Animation animation =
AnimationUtils.loadAnimation(MainActivity.this,
R.anim.tween_anim);
button.startAnimation(animation);
```

Импортируем все зависимости и исправляем ошибки, после чего запускаем проект на эмуляторе. Теперь необходимо изменить параметры в файле tween_anim.xml на собственные, чтобы все не сдавали одну и ту же анимацию. Поэкспериментируйте с различными параметрами, можете также производить анимацию не с кнопкой, а с каким-нибудь другим элементом. Результат выполнения работы показан на рисунке 29.

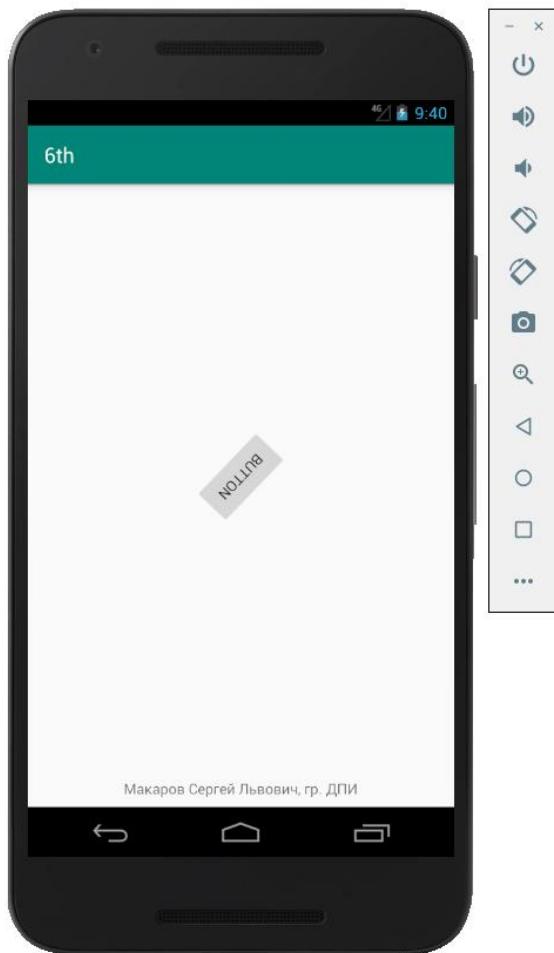


Рисунок 29 – Результат анимации

Лабораторная работа №7. Работа с картами

Задание: создать приложение, отображающее после запуска карты Google или какие-нибудь другие карты.

Создайте новый проект, но на этот раз выберите Google Maps Activity вместо Empty Activity. Для того, чтобы пользоваться картами Google, необходимо получить персональный ключ для приложения. Переходим в файл res/values/google_maps_api.xml (открывается по умолчанию), и в комментариях видим инструкцию на английском языке о том, как получить ключ для приложения. Если коротко, то необходимо перейти по персональной ссылке из комментария, залогиниться в свой Google аккаунт, нажать кнопку Continue и затем Create API key, после чего перед вами появится окно API key created с ключом, начинающимся на "AIza". Копируем значение ключа из поля и вставляем в файл res/values/google_maps_api.xml вместо фразы YOUR_KEY_HERE.

Теперь переходим в файл MapsActivity.java, и в самом последнем методе onMapReady добавляем следующее в начало, после строки `mMap = googleMap;`:

```
LocationManager locationManager = (LocationManager)
this.getSystemService(Context.LOCATION_SERVICE);
```

Затем строчкой ниже начинаем писать LocationListener
`locationListener = new Loc` и дальше выбираем первую подсказку.
Появится 4 метода, в первом методе onLocationChanged пишем следующее:

```
Latlng latlng = new Latlng(location.getLatitude(),
location.getLongitude());
mMap.clear();
mMap.moveCamera(CameraUpdateFactory.newLatlng(latlng));
mMap.animateCamera(CameraUpdateFactory.zoomTo(15));
mMap.addMarker(new MarkerOptions().position(latlng).title("I am
here"));
```

После этих четырёх методов (после символа }; - не забудьте поставить точку с запятой) пишем следующее:

```
locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, locationListener);
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, locationListener);
```

Здесь мы запрашиваем обновление местоположения по вышкам сотовой связи и по GPS. Код просит добавить проверку разрешений для пользователя, добавляем эту проверку (выделяем строку с ошибкой, слева видим красную лампочку с восклицательным знаком внутри, нажимаем на неё и выбираем Add permission check) и одновременно в файл манифеста добавляем соответствующие разрешения. Разрешение **ACCESS_FINE_LOCATION** уже есть, добавим недостающее:

```
<uses-permission
    android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

Далее вернёмся в **MapsActivity.java** и в методе **onMapReady** закомментируем всё, что идёт в конце и относится к Сиднею (после *// Add a marker in Sydney and move the camera*), и после этого вставим две строчки:

```
mMap.setMyLocationEnabled(true);
mMap.getUiSettings().setZoomControlsEnabled(true);
mMap.setMapType(2);
```

Таким образом, на карте отобразится кнопка текущего местоположения и кнопки зума.

Прежде чем запускать приложение, необходимо создать новый эмулятор с поддержкой Google Play services (Google APIs), если у вас ещё нет

такого эмулятора, так как на обычном эмуляторе карту вы не увидите, а увидите вместо неё надпись "Your_app_name won't run without Google Play services, which are not supported by your device.". Затем, возможно, после запуска приложения вы можете получить сообщение о том, что "Your_app_name won't run unless you update Google Play services", так как для работы приложения нужна версия 2 этих сервисов – нажмите кнопку Update в этом случае (кнопка будет работать только в том случае, если ваш эмулятор поддерживает функциональность Google Play). Лучше всего сконфигурировать эмулятор с одной из последних версий API (начиная с Lollipop) – тогда не надо будет обновлять Google Play services до последней версии. Если после запуска приложения появляется предложение установить Instant Run, надо согласиться.

Также помните о добавлении вашего ФИО в приложение – в этот раз это будет сделать сложнее, чем обычно, так как надо самостоятельно создать layout в файле res -> layout -> activity_maps.xml, обернуть в него fragment с картой и добавить элемент TextView.

Запускаем приложение, однако эмулятор не имеет данных о местоположении, но их можно ему передать! Для этого в панели инструментов справа от эмулятора нажмите на три точки внизу, после этого откроется окно, показанное на рисунке 30. Справа в окне видно, что можно ввести значения широты и долготы и передать их приложению с помощью кнопки SEND. Введите 55.769794 в поле Longitude и 37.655640 в поле Latitude (координаты Москвы) и нажмите кнопку SEND.

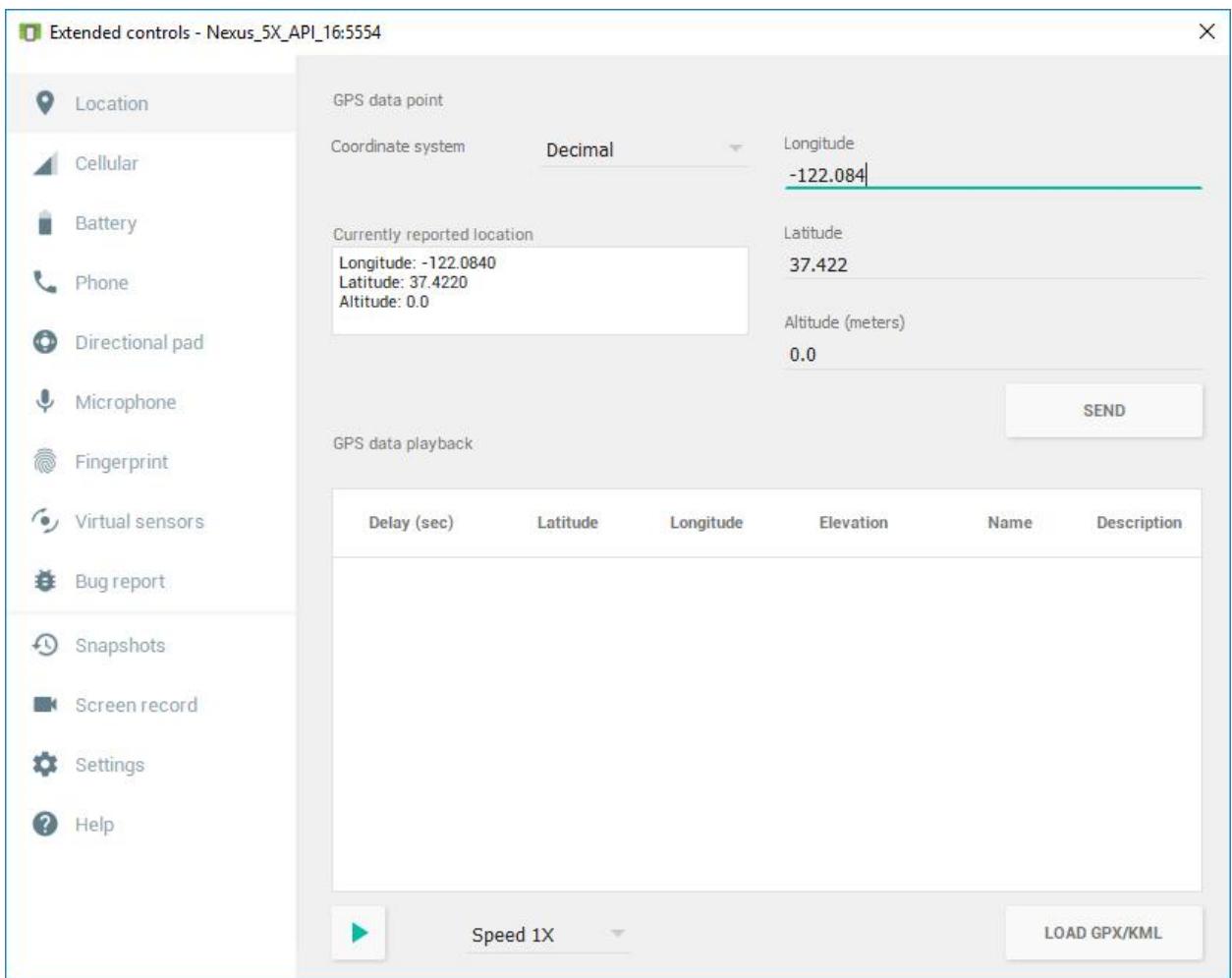


Рисунок 30 – Окно дополнительного управления эмулятором

Теперь эмулятор знает координаты и отображает своё местоположение по ним. Пример интерфейса работающего приложения показан на рисунках 31 и 32. Приложение можно запустить и на вашем реальном устройстве, на котором точно есть Google Play services.

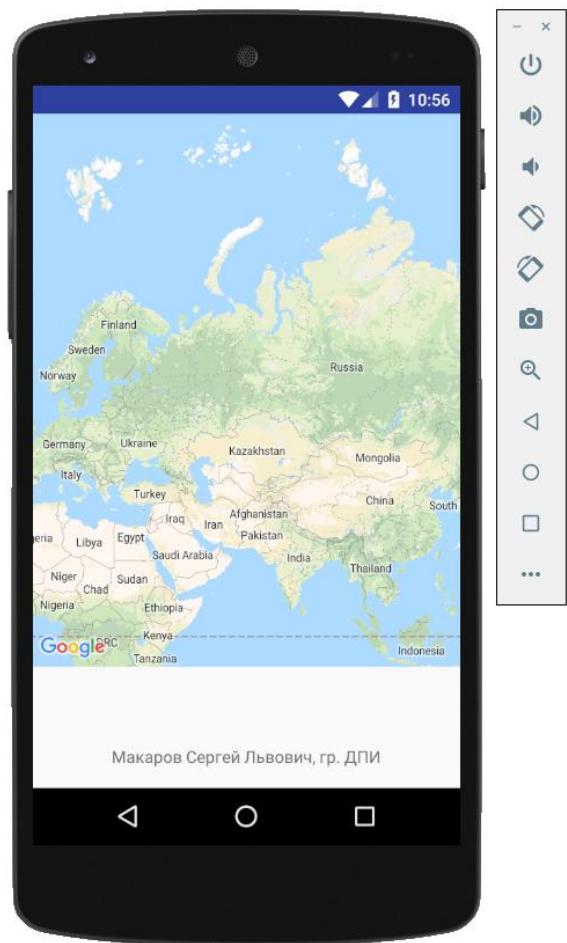
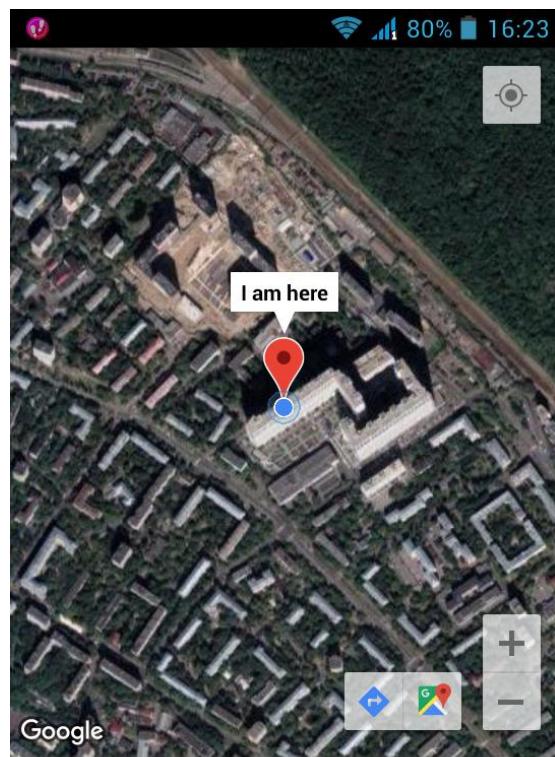


Рисунок 31 – Пример выполнения лабораторной работы №7 на эмуляторе



Макаров Сергей Львович, гр. ДПИ

Рисунок 31 – Пример выполнения лабораторной работы №7 на смартфоне

Лабораторная работа №8. Создание виджета

Задание: создать собственный виджет с настройками. Например, виджет, который открывает какой-то сайт, адрес которого можно поменять в настройках.

Создайте новый проект без Activity, то есть при создании необходимо выбрать пункт Add No Activity вместо обычного Empty Activity. Теперь необходимо создать activity для виджета. Для этого нажмите правой кнопкой мыши по папке app, затем New -> Widget -> App Widget. Здесь задайте необходимые параметры для виджета. Можно оставить все пункты по умолчанию, однако нужно поставить галочку в пункте Configuration Screen, чтобы в проект добавилось activity для настройки виджета.

Перейдите в файл res/xml/new_app_widget_info.xml и в параметре android:updatePeriodMillis смените значение «86400000» на «0», так как обновлять данный виджет не нужно. Затем перейдите в файл manifests/AndroidManifest.xml и перед тэгом application добавьте приложению разрешение пользоваться интернетом:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Теперь перейдите в файл NewAppWidget.java, в методе updateAppWidget удалите весь код и напишите следующий:

```
Intent intent=new Intent(Intent.ACTION_VIEW,
Uri.parse("https://google.com"));
PendingIntent pending= PendingIntent.getActivity(context, 0,
intent, 0);
RemoteViews views = new RemoteViews(context.getPackageName(),
R.layout.new_app_widget);
views.setOnClickListener(R.id.appwidget_text, pending);
appWidgetManager.updateAppWidget(appWidgetId, views);
```

Здесь устанавливается, что при нажатии на виджет необходимо открыть браузер и перейти по ссылке google.com.

Теперь необходимо найти и удалить в этом классе метод onDelete.

Затем перейдите в файл NewAppWidgetConfigureActivity.java, удалите весь код в этом классе и добавьте следующие строчки:

```
private NewAppWidgetConfigureActivity context;  
private int widgetID;
```

Строчкой ниже начните писать protected и в появившемся списке выберите protected void onCreate (savedInstanceState), метод onCreate создастся автоматически. Далее в методе onCreate необходимо написать:

```
setContentView(R.layout.new_app_widget_configure);  
setResult(RESULT_CANCELED);  
context = this;  
Bundle extras = getIntent().getExtras();
```

Затем добавляем следующее условие:

```
if (extras != null) {  
    widgetID =  
    extras.getInt(AppWidgetManager.EXTRA_APPWIDGET_ID,  
    AppWidgetManager.INVALID_APPWIDGET_ID);  
    final AppWidgetManager widgetManager =  
    AppWidgetManager.getInstance(context);  
    final RemoteViews views = new  
    RemoteViews(context.getPackageName(), R.layout.new_app_widget);  
    final EditText editText = (EditText)  
    findViewById(R.id.appwidget_text);  
    Button button = (Button) findViewById(R.id.add_button);  
}
```

Далее в условии задайте listener и onClick для объявленной кнопки и в методе onClick напишите следующее:

```
Intent intent=new Intent(Intent.ACTION_VIEW,  
Uri.parse(editText.getText().toString()));  
PendingIntent pending= PendingIntent.getActivity(context, 0,
```

```

intent, 0);
views.setOnClickListenerPendingIntent(R.id.appwidget_text, pending);
widgetManager.updateAppWidget(widgetID, views);
Intent resultValue = new Intent();
resultValue.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID,
widgetID);
setResult(RESULT_OK, resultValue);
finish();

```

Когда виджет добавляется на экран смартфона, появляется страница с настройками, где необходимо прописать желаемый адрес, начиная с «<http://>», и после нажатия на кнопку виджет добавится на экран смартфона и будет работать только с заданной ссылкой.

Для того, чтобы изменить внешний вид виджета или его окна конфигурации/настроек, перейдите в файлы res -> layout -> new_app_widget.xml и new_app_widget_configure.xml соответственно. Например, в последнем файле можно добавить свои ФИО и поместить текстовое поле для адреса и кнопку по центру окна, изменить LinearLayout на RelativeLayout, в текстовом поле можно написать «<http://>», чтобы пользователю не нужно было набирать это вручную; а в первом – поменять слово EXAMPLE на другое, изменить цвет виджета, его размеры в файле res/xml/new_app_widget_info.xml и т.д.

Запустить проект не получится – в нём нет activity. Поэтому постройте проект (Build -> Build APK(s)) и скопируйте и установите apk файл на ваше устройство (справа внизу нажмите locate, и откроется папка с только что построенным apk файлом виджета). Затем скопируйте и установите виджет в соответствии с правилами вашей версии ОС Android на ваше мобильное устройство – виджет будет отображаться среди других виджетов, его надо найти и перетащить на один из рабочих столов, при этом откроется сконфигурированное в коде окно настройки, где надо будет ввести адрес и нажать кнопку Add Widget.

Что касается эмулятора, то сначала его надо запустить. Затем, чтобы скопировать готовый apk файл на эмулятор в Android Studio, нужно выбрать

пункт главного меню View -> Tool Windows -> Device File Explorer. В открывшемся окне появится структура каталогов и файлов эмулятора. Откройте папку data -> app. В этой папке вы можете увидеть ранее установленные файлы apk от наших предыдущих проектов. Нажмите правой кнопкой мыши по этой папке, выберите Upload, найдите только что созданный apk файл виджета (обычно он находится в папке имя_проекта\app\build\outputs\apk\debug) и нажмите кнопку OK, см. рисунок 32. После этого перейдите в окно эмулятора, нажмите на кнопку Все программы и перейдите на закладку WIDGETS, см. рисунки 33 и 34 (приведена инструкция и показан интерфейс для ОС Android 4.1.2) – вы должны увидеть виджет (внизу у него будет написано имя_проекта, например – 8th2try, рисунок 35; при этом перезагружать эмулятор не надо).

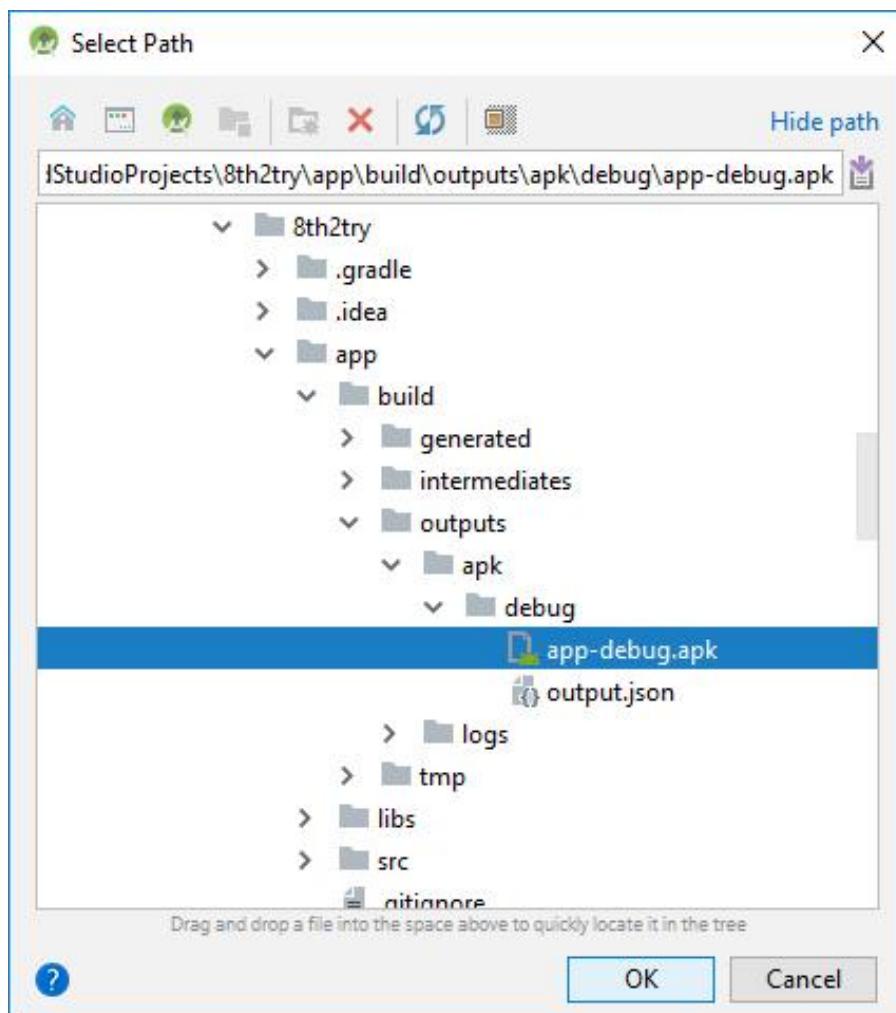


Рисунок 32 – Копирование файла виджета из папки проекта



Рисунок 33 – Все программы

Рисунок 34 – Установленные приложения

Перетащите виджет на один из экранов и в настройках введите любой адрес, который обязательно должен начинаться с «<http://>!» Затем нажмите кнопку Add Widget и кликните по виджету, после чего должен запуститься стандартный браузер с заданной в настройках ссылкой. Примеры страницы настройки и виджета представлены на рисунках 36 и 37.

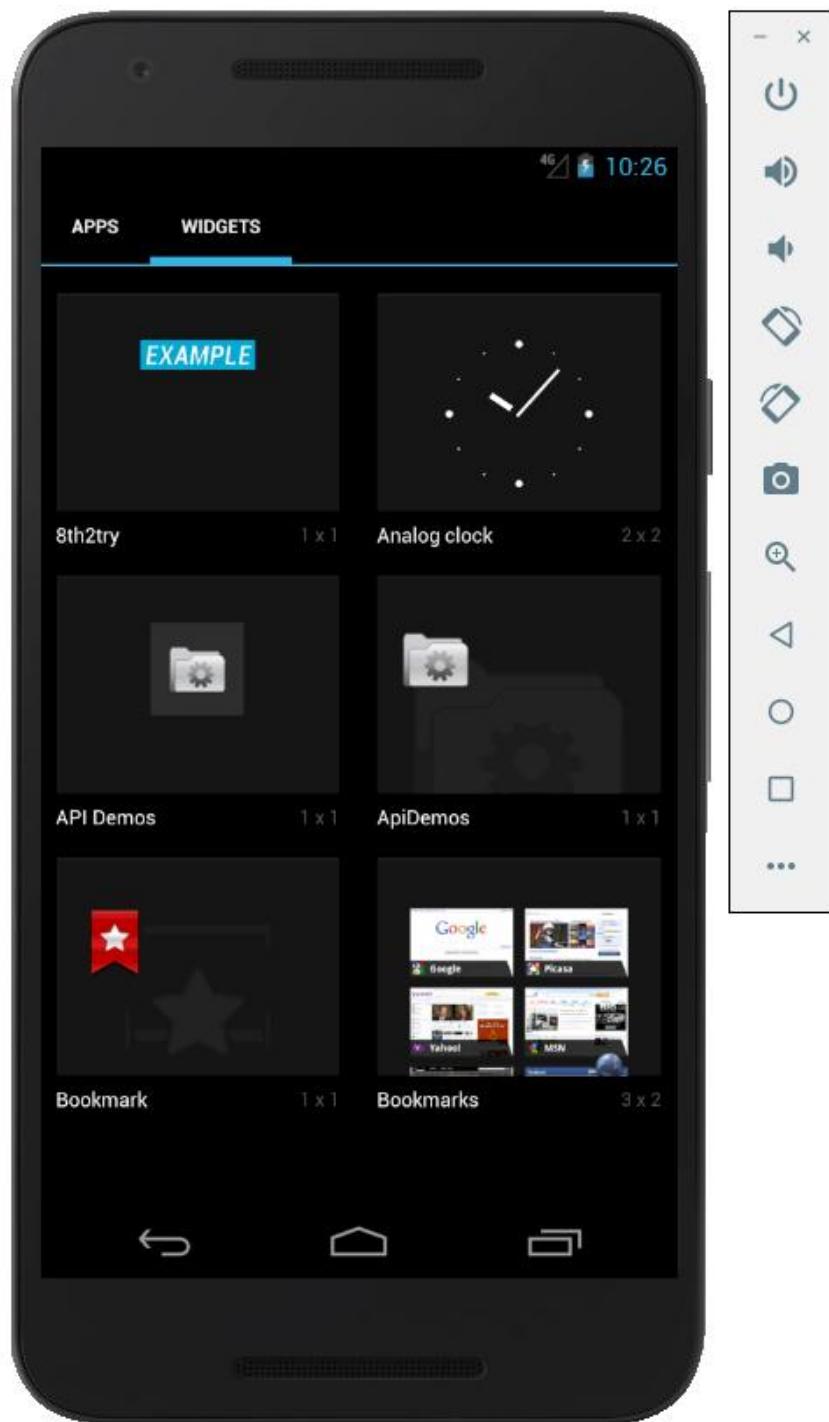


Рисунок 35 – Скопированный виджет 8th2try

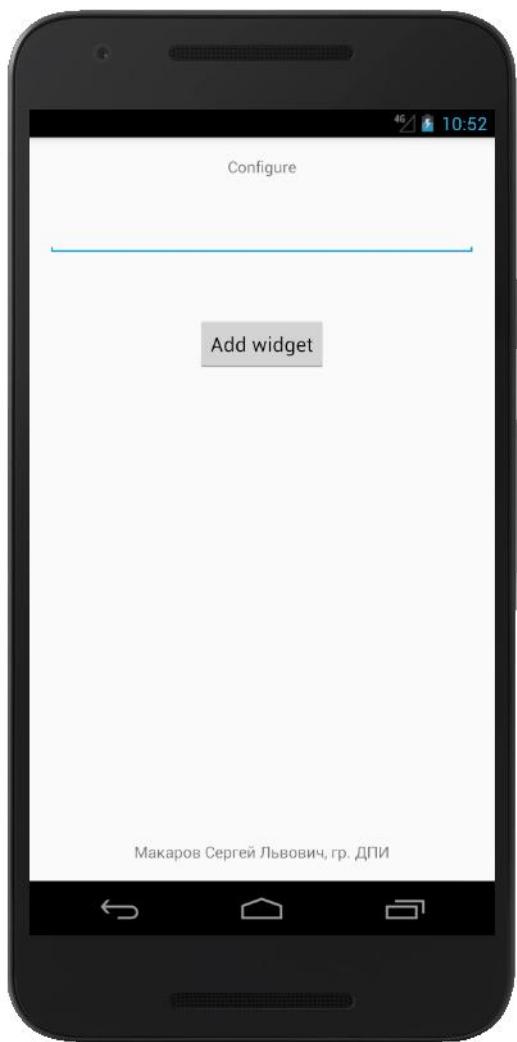


Рисунок 36 – Окно настройки виджета



Рисунок 37 – Готовый виджет

Лабораторная работа №9. Работа с меню

Задание: создать приложение, использующее опциональное и контекстное меню для какого-нибудь интерфейсного элемента. Естественно, выбор пунктов меню должен что-то менять в интерфейсных элементах или их отображении. Например, очистить поле ввода через контекстное меню, или отобразить невидимые интерфейсные элементы через установку галок в опциональном меню.

В операционной системе Android есть два вида меню: контекстное и опциональное. Опциональное меню появляется при нажатии на кнопку меню на устройстве, либо при нажатии кнопки меню в программе (три точки или три тире в правом или левом верхнем углу в зависимости от версии ОС). Контекстное меню появляется при длительном нажатии на каком-либо элементе пользовательского интерфейса.

Создайте новый проект с Empty Activity. Конечно, можно создать проект, выбрав Basic Activity, при этом опциональное меню добавится автоматически: в папке res каждого проекта будет отдельная папка menu, в которой располагается файл menu_main.xml, описывающий опциональное меню приложения по умолчанию. В то же время, для опционального меню в файле MainActivity.java добавится заготовленный код с двумя методами для опционального меню: onCreateOptionsMenu и onOptionsItemSelected; а в интерфейсе приложения и в коде по умолчанию будет один пункт меню Settings.

Однако наша цель – научится создавать меню самостоятельно, поэтому всё надо делать вручную – это недолго. Создадим папку menu в папке res (правой кнопкой мыши по папке res -> New -> Directory) и добавим в неё файл menu.xml: это можно сделать, нажав правой кнопкой мыши по папке menu и выбрав New и появившийся пункт Menu resource file. Заметьте, этот пункт контекстного меню папки menu отсутствует в контекстных меню

других папок. В открывшемся окне назовите файл также menu, см. рисунок 38.

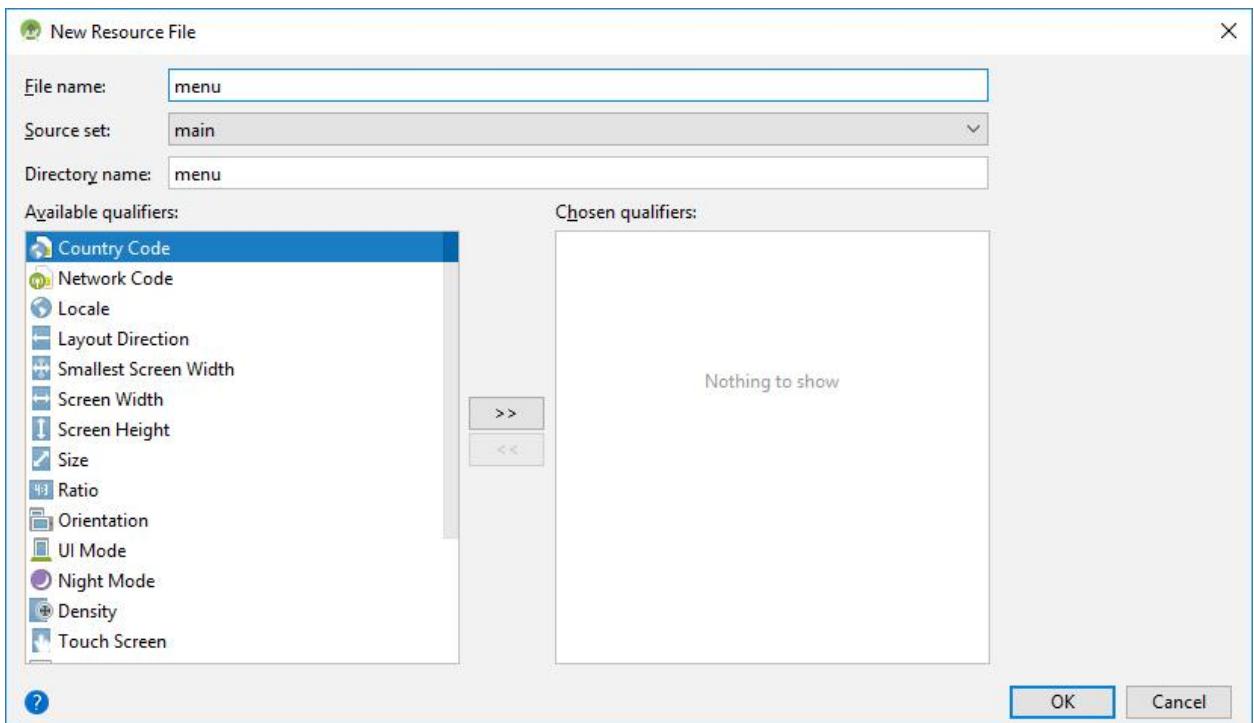


Рисунок 38 – Создание файла опционального меню

Перейдите в файл menu.xml (закладка Text) и добавьте новый пункт опционального меню:

```
<item android:id="@+id/show_text" android:title="Скрыть текст"
      android:checkable="true" />
```

Теперь необходимо этот пункт меню отобразить в приложении. Перейдите в файл MainActivity.java и добавьте метод onOptionsItemSelected после метода onCreate: начните печатать onOpt... и выберите конструктор для этого метода. В методе onOptionsItemSelected перед return напишите следующее:

```
int id = item.getItemId();
if (id == R.id.show_text) {
    if (item.isChecked()) {
```

```

        TextView textView = findViewById(R.id.text1);
        textView.setVisibility(TextView.VISIBLE);
        item.setChecked(false);
    }
    else {
        TextView textView = findViewById(R.id.text1);
        textView.setVisibility(TextView.INVISIBLE);
        item.setChecked(true);
    }
}

```

В данном коде в опциональное меню добавляется новый пункт, который в зависимости от состояния поля checkbox скрывает или показывает элемент TextView. Однако только этого метода onOptionsItemSelected недостаточно, необходимо добавить в MainActivity.java ещё один метод:

```

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu, menu);
    return super.onCreateOptionsMenu(menu);
}

```

В этом методе опциональное меню menu заполняется содержимым, которое лежит в файле menu.xml, т.е. – нашим единственным пунктом Скрыть текст. Кроме того, без этого метода меню не будет видно в интерфейсе приложения. Запустите проект и проверьте, что всё работает (рисунок 39).

Теперь создадим контекстное меню, для этого необходимо создать ещё один xml-файл в папке menu. Нажмите по папке menu правой кнопкой мыши, затем New -> Menu resource file. Оставьте все поля по умолчанию и назовите файл contextmenu. После создания перейдите в данный файл (закладка Text) и напишите:

```

<item android:id="@+id/color_red" android:title="Красный" />
<item android:id="@+id/color_black" android:title="Чёрный" />

```

Вернитесь в файл MainActivity.java и в методе onCreate напишите:

```
TextView textView = findViewById(R.id.text1);
registerForContextMenu(textView);
```

Таким образом, мы зарегистрировали элемент TextView в контекстном меню, и теперь добавленное контекстное меню будет работать для этого текстового элемента. Затем создайте новый метод onCreateContextMenu после метода onCreate. Начните писать onCreateC и выберите шаблон. В методе onCreateContextMenu первой строчкой перед существующим кодом напишите:

```
getMenuInflater().inflate(R.menu.contextmenu, menu);
```

Далее необходимо создать ещё один метод, по аналогии с меню опций: onContextItemSelected. Начните писать onContextI и выберите предложенный шаблон. В созданном методе onContextItemSelected нужно описать кнопки и действия:

```
int id = item.getItemId();
if (id == R.id.color_red) {
    TextView textView = (TextView) findViewById(R.id.text1);
    textView.setTextColor(Color.parseColor("red"));
}
if (id == R.id.color_black) {
    TextView textView = (TextView) findViewById(R.id.text1);
    textView.setTextColor(Color.parseColor("black"));
}
```

Контекстное меню создано. Запустите приложение, нажмите и держите кнопку мыши / палец на элементе TextView, после чего появится контекстное меню с двумя созданными пунктами, которые меняют цвет текста (рисунок 40).

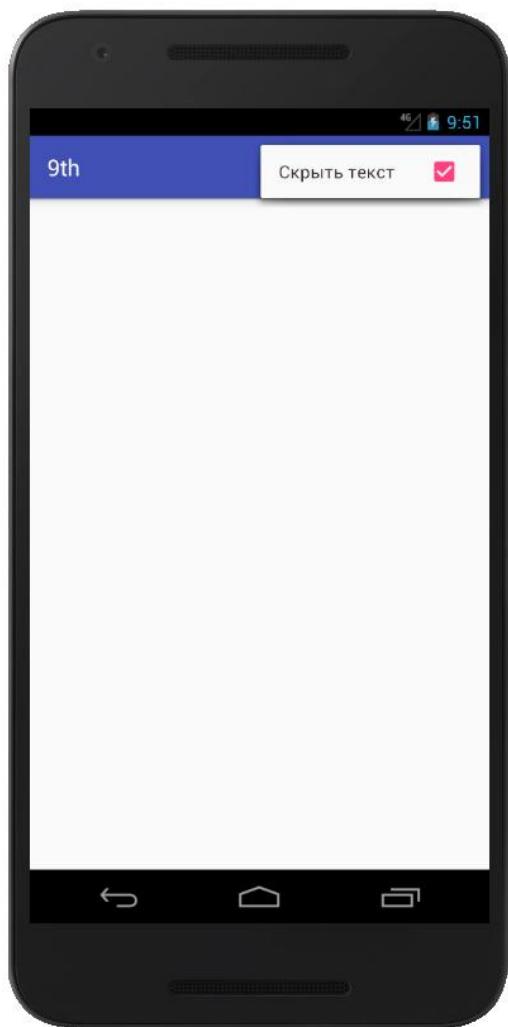


Рисунок 39 – Опциональное меню

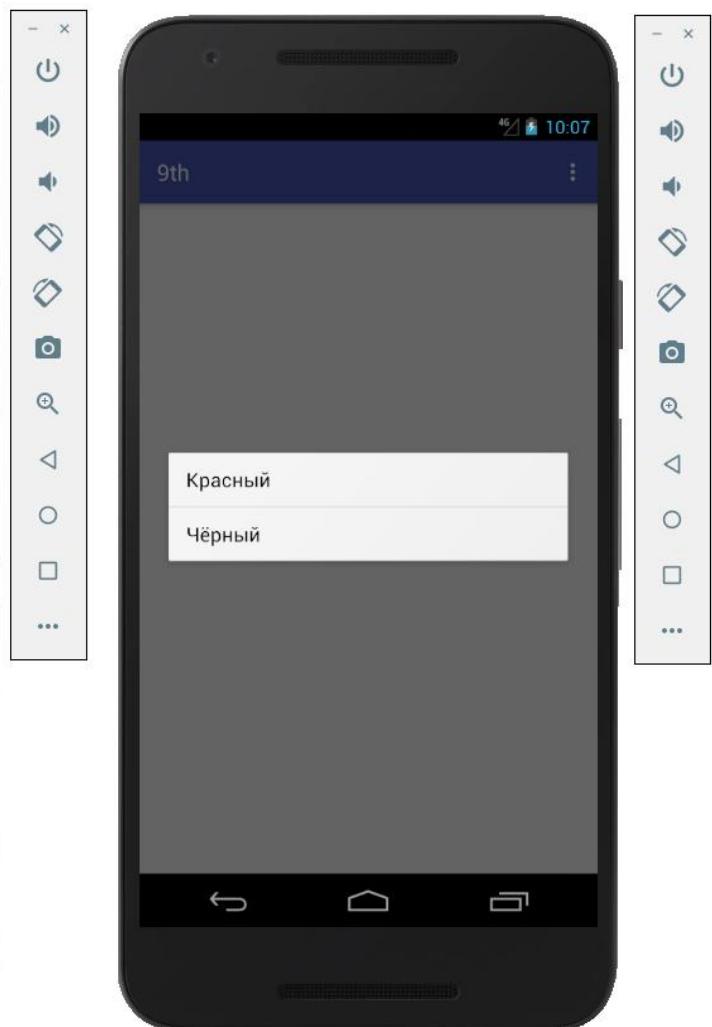


Рисунок 40 – Контекстное меню

Вы наверняка заметили, что в этом задании не были упомянуты ФИО. Сделайте ещё один пункт опционального меню **О программе**, в котором выводите с помощью Toast следующее: Автор – ФИО, гр. (создайте новый строковый ресурс, как обычно), добавив в соответствующее место кода следующее:

```
Toast.makeText(MainActivity.this, R.string.fio,
Toast.LENGTH_LONG).show();
```

В результате появится информация о разработчике программы и новый пункт меню, как показано на рисунках 41 и 42.

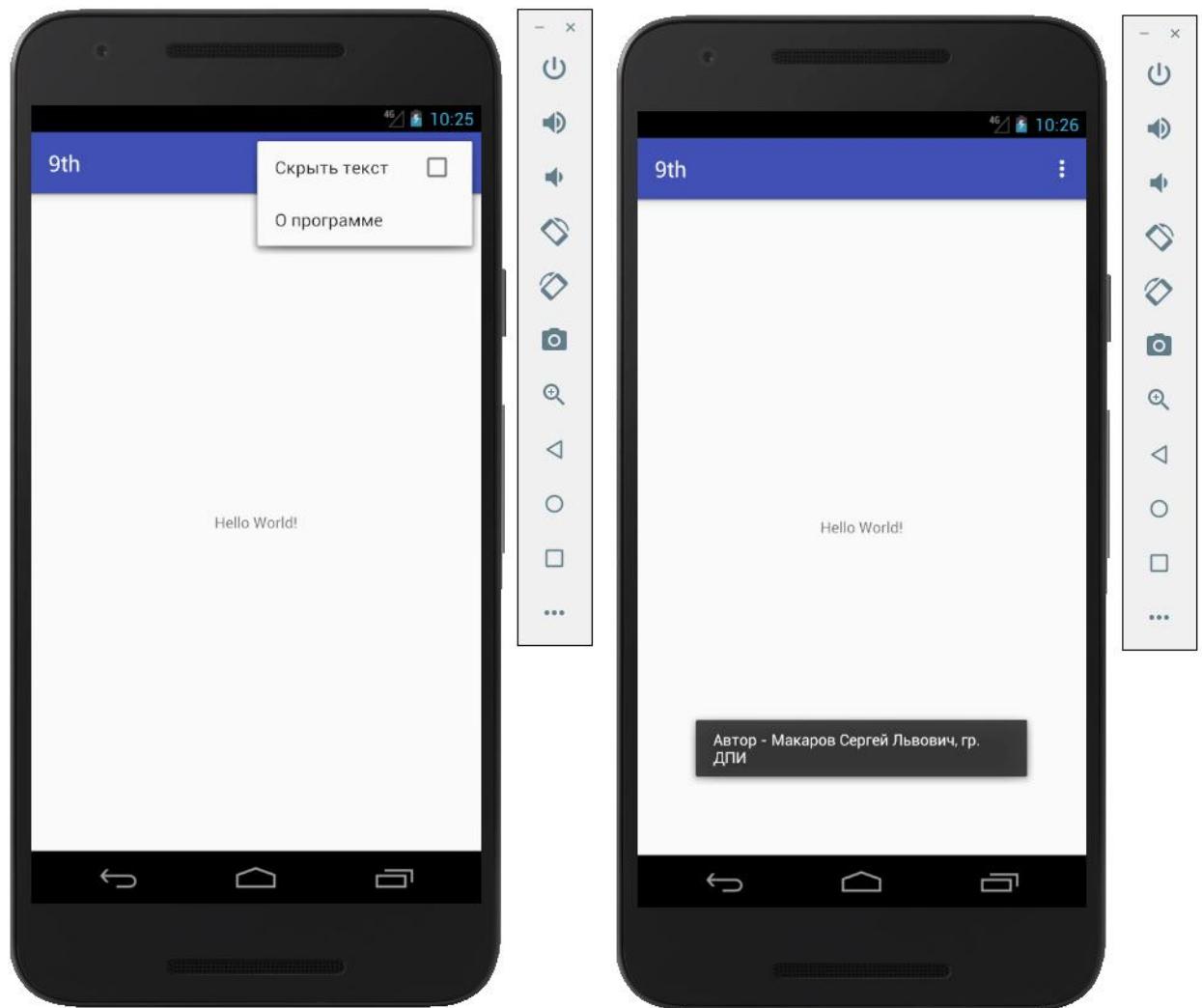


Рисунок 41 – Пункт "О программе"

Рисунок 42 – Toast об авторе

Лабораторная работа №10. Работа с диалоговыми окнами

Задание: создать приложение, отображающее после некоторых действий (нажатия на кнопку, например, или проверки корректности ввода текста в EditText) диалоговое окно, свидетельствующее об ошибке/информирующее/предупреждающее пользователя о чём-то.

Создайте новый проект с Empty Activity, перейдите в файл activity_main.xml, удалите TextView и поместите в центр экрана кнопку "Выход". Далее перейдите в файл MainActivity.java, в методе onCreate создайте ссылку на кнопку, а затем создайте для неё listener и метод onClick. В методе onClick напишите следующий код:

```
AlertDialog.Builder dialog = new
AlertDialog.Builder(MainActivity.this);
dialog.setMessage("Вы действительно хотите выйти?");
dialog.setCancelable(false);
dialog.setPositiveButton("Да", new
DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        MainActivity.this.finish();
    }
});
dialog.setNegativeButton("Нет", new
DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        dialog.cancel();
    }
});
AlertDialog alertDialog = dialog.create();
alertDialog.show();
```

Здесь происходит построение и отрисовка диалогового окна, в котором указывается сообщение и устанавливаются две кнопки. При нажатии на кнопку "Да" закроется приложение, а при нажатии на кнопку "Нет" закроется диалоговое окно. Пример выполнения представлен на рисунках 43 и 44.



Рисунок 43 – Стартовое окно

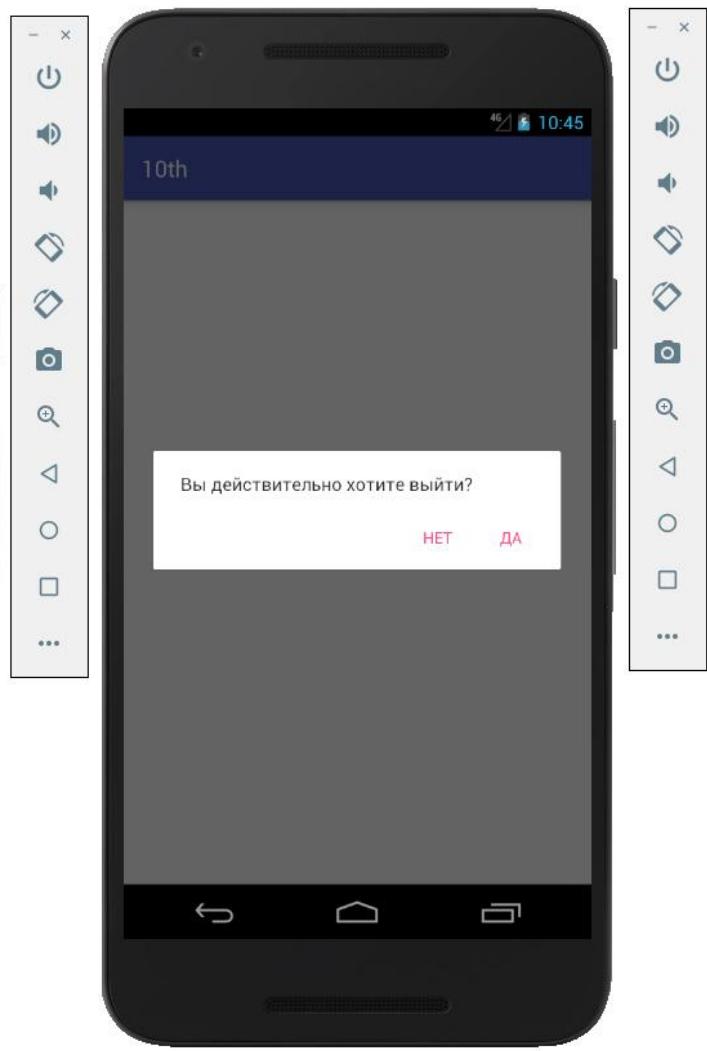


Рисунок 44 – Диалоговое окно

Теперь было бы логично совместить это задание с предыдущим заданием, так как выводить информацию о программе в Toast непривычно и не очень красиво. Добавьте в приложение меню опций, в него добавьте пункт "О программе" и сформируйте по нажатию на этот пункт диалоговое окно с информацией о программе и её авторе, например, так:

```
AlertDialog.Builder dialog = new
AlertDialog.Builder(MainActivity.this);
try {
    dialog.setMessage(getTitle().toString() + " версия " +
getPackageManager().getPackageInfo(getPackageName(), 0).versionNa
me + "\r\n\r\nПрограмма с примером выполнения диалогового окна
\r\n\r\n Автор - Макаров Сергей Львович, гр. ДПИ");
} catch (PackageManager.NameNotFoundException e) {
    e.printStackTrace();
```

```
}

dialog.setTitle("О программе");
dialog.setNeutralButton("OK", new
DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        dialog.dismiss();
    }
});
dialog.setIcon(R.mipmap.ic_launcher_round);
AlertDialog alertDialog = dialog.create();
alertDialog.show();
```

Результат добавления пункта меню "О программе" с диалоговым окном можно увидеть на рисунках 45 и 46. Теперь нет необходимости писать свои ФИО и группу в элементе TextView – у нас есть хороший способ показать сведения об авторе. В последующих заданиях можно копировать этот код, чтобы пункт меню "О программе" всегда присутствовал в ваших приложениях.



Рисунок 45 – Пункт меню "О
программе"

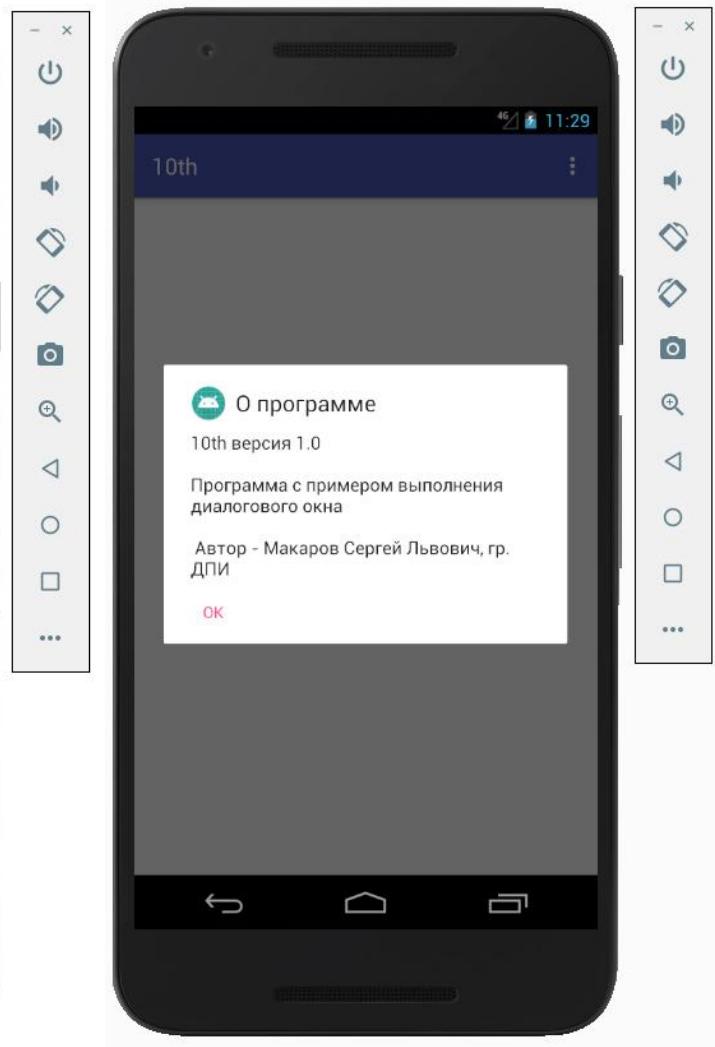


Рисунок 46 – Диалоговое окно "О
программе"

Лабораторная работа №11. Работа с уведомлениями

Задание: создать приложение, помещающее по нажатию на кнопку какое-то сообщение со звуком в панель уведомлений/статус-панель на эмуляторе.

Создайте стандартный проект с Empty Activity и поместите кнопку в интерфейс приложения. Уведомление будет приходить после нажатия на данную кнопку. Перейдите в файл MainActivity.java и в методе onCreate создайте ссылку на кнопку, а затем создайте для неё listener и метод onClick. Далее в методе onClick настройте уведомление (данный код актуален только для версий Android ниже Oreo, современный код см. после скриншотов):

```
Context context = MainActivity.this;
Notification notification = new Notification.Builder(context)
    .setContentTitle("Hi from Alex")
    .setContentText("Hey, listen!")
    .setTicker("new notification!")
    .setSmallIcon(android.R.drawable.ic_dialog_alert)

    .setSound(Uri.parse("android.resource://" + getPackageName() + "/" + R
    .raw.al_heylisten))
    .build();
NotificationManager notificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
notificationManager.notify(0, notification);
```

Здесь мы устанавливаем заголовок (setTitle) и текст (setContentText) для уведомления, а также небольшой текст, выводящийся в небольшую панель уведомлений ("шторку") рядом с иконкой (setTicker, см. рисунок 47), иконку и звуковой файл, который будет воспроизводиться при получении уведомления. Звуковой файл нужно поместить в папку res/raw, однако для начала её стоит создать. Нажмите правой кнопкой мыши по папке res и создайте Android resource directory. Resource type выберите raw и нажмите OK. Обратите внимание, что имя звукового файла не может начинаться с цифры. Не забудьте добавить в приложение опциональное

меню с пунктом о программе, который открывает диалоговое окно с вашими ФИО, см. предыдущие 2 задания. Теперь всё готово, можно запускать приложение и нажимать на кнопку, после чего в панели уведомлений появится уведомление от приложения, как показано на рисунках 47-48.

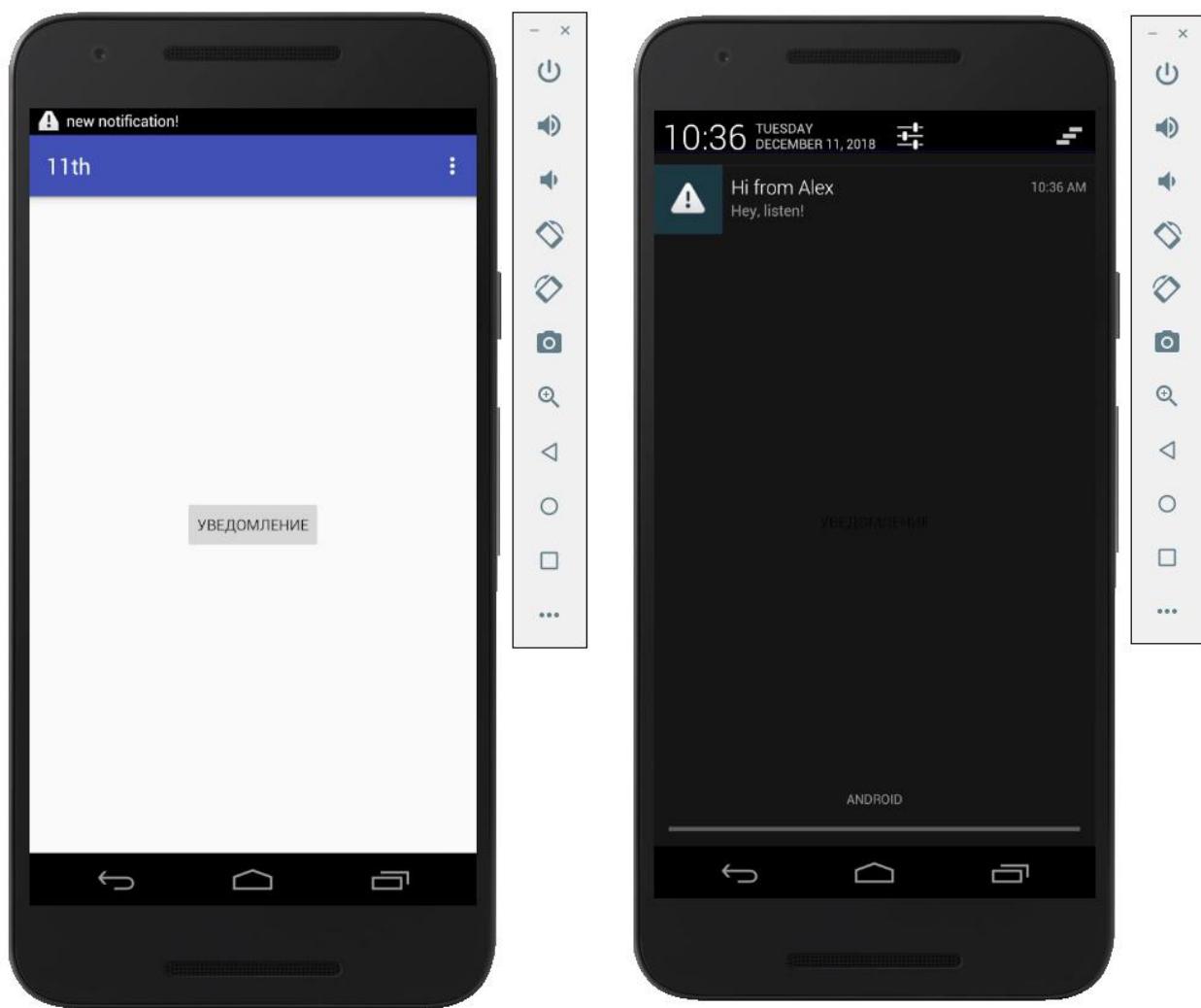


Рисунок 47 – Отображение краткого уведомления в панели уведомлений

Рисунок 48 – Отображение уведомления в панели уведомлений

Для версий Android, начиная с Oreo (Android 8, или API 26) нужен другой код. Дело в том, что начиная с Oreo появились так называемые каналы, которые пользователь приложения может настроить самостоятельно, перейдя в настройки приложения и открыв пункт уведомлений (Notifications). Поэтому, если у Вас приложение только для Oreo и выше, в коде нужно создавать и настраивать канал уведомлений NotificationChannel. Если в

gradle-файле приложения заданы minSdkVersion и targetSdkVersion больше либо равные 26 (Oreo), предусматривать проверку текущей версии Android на устройстве не надо – просто надо взять новый код с поддержкой каналов. Однако часто бывает, что приложение должно поддерживать и старые версии, и, например, minSdkVersion=16 при targetSdkVersion=28. Тогда в коде нужно делать проверку версии Android на устройстве, и для версии, начиная с Oreo (O), работать с каналами, как показано в коде ниже:

```
Context context = MainActivity.this;
NotificationChannel newnotchan= null;
if (android.os.Build.VERSION.SDK_INT >=
    android.os.Build.VERSION_CODES.O) {
    newnotchan= new
    NotificationChannel("mychannel1","mychannel",NotificationManager
        .IMPORTANCE_HIGH);
    AudioAttributes audioAttributes = new
    AudioAttributes.Builder()
        .setContentType(AudioAttributes.CONTENT_TYPE_MUSIC)
        .setUsage(AudioAttributes.USAGE_MEDIA)
        .build();

    newnotchan.setSound(Uri.parse(ContentResolver.SCHEME_ANDROID_RESOURCE+"://"+getPackageName()+"."+R.raw.al_heylisten),
        audioAttributes);
}
NotificationManager notificationManager = (NotificationManager)
getApplicationContext().getSystemService(Context.NOTIFICATION_SERVICE);
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    notificationManager.createNotificationChannel(newnotchan);
}
Notification notification = null;

if (android.os.Build.VERSION.SDK_INT >=
    android.os.Build.VERSION_CODES.O) {
    notification = new
    Notification.Builder(context,"mychannel1")
        .setContentTitle("Hi from Alex")
        .setContentText("Hey, listen!")
        .setTicker("new notification!")
        .setChannelId("mychannel1")
        .setSmallIcon(android.R.drawable.ic_dialog_alert)
        .setOngoing(true)
        .build();
}
```

```
notificationManager.notify(0, notification);
```

Обратите внимание – для того, чтобы слышать именно тот звук, который расположен в папке raw, нужно сконфигурировать для канала так называемые AudioAttributes, а метод setSound, в котором нужны путь к файлу и эти аудиоатрибуты, затем использовать для канала, а не для уведомления. В приведённом выше коде отсутствует часть для версий Android ниже Oreo в конструкторе else, но она есть в начале описания этой лабораторной работы, и её в этот конструктор не составляют труда добавить, таким образом, сделав это приложение универсальным.

Лабораторная работа №12. Работа с аудио

Задание: создать приложение, по нажатию кнопки в котором проигрывается какой-то звук.

Создайте стандартный проект, но на этот раз – с Basic Activity. У нас уже есть кнопка на экране. Теперь необходимо в папке res создать папку raw, как это делалось в прошлой лабораторной работе, и добавить туда какой-то звуковой файл. После этого перейдите в файл MainActivity.java и в методе onCreate в методе onClick кнопки FloatingActionButton добавьте всего две строчки:

```
MediaPlayer mediaPlayer = MediaPlayer.create(MainActivity.this,
R.raw.dimon);
mediaPlayer.start();
```

Также можно изменить строчку с методом Snackbar.make():

```
Snackbar.make(view, "Your should hear a sound",
Snackbar.LENGTH_LONG)
.setAction("Action", null).show();
```

Не забудьте добавить в приложение опциональное меню с пунктом о программе, который открывает диалоговое окно с вашими ФИО. В данном случае мы использовали Basic Activity, поэтому все необходимые методы в коде уже есть, а пункт меню Settings можно поменять на "О программе". Запустите проект. После нажатия на кнопку должен проигрываться звуковой файл. Бывает так, что приложение начинает работать не с первого раза, тогда нужно выполнить Build -> Clean Project и заново загрузить его в эмулятор.

Лабораторная работа №13. Работа с видео

Задание: создать приложение, при запуске которого проигрывается какое-то видео.

Создайте стандартный проект с Empty Activity, перейдите в файл activity_main.java, удалите стандартное текстовое поле, включите магнит и перетащите на экран элемент VideoView из категории Widgets. Теперь создайте папку raw (правой кнопкой мыши по папке res -> New -> Directory -> raw) и поместите в неё видео. Приложение сможет воспроизвести только видео в формате *.mp4 и *.3gp.

Перейдите в файл MainActivity.java и в методе onCreate создайте ссылку на VideoView:

```
VideoView videoView = findViewById(R.id.videoView);
```

Далее укажите путь к видеофайлу:

```
videoView.setVideoURI(Uri.parse("android.resource://" +  
getPackageName() + "/" + R.raw.video));
```

И инициируйте показ видео с помощью класса MediaController, помещающего элементы управления видео в виджет videoView:

```
videoView.setMediaController(new MediaController(this));  
videoView.start();  
videoView.requestFocus();
```

Не забудьте добавить в приложение опциональное меню с пунктом о программе, который открывает диалоговое окно с вашими ФИО.

Запустите приложение, и сразу же после запуска начнёт проигрываться видео. Если нажать на область воспроизведения, то снизу появятся элементы управления видео, как показано на рисунке 49.

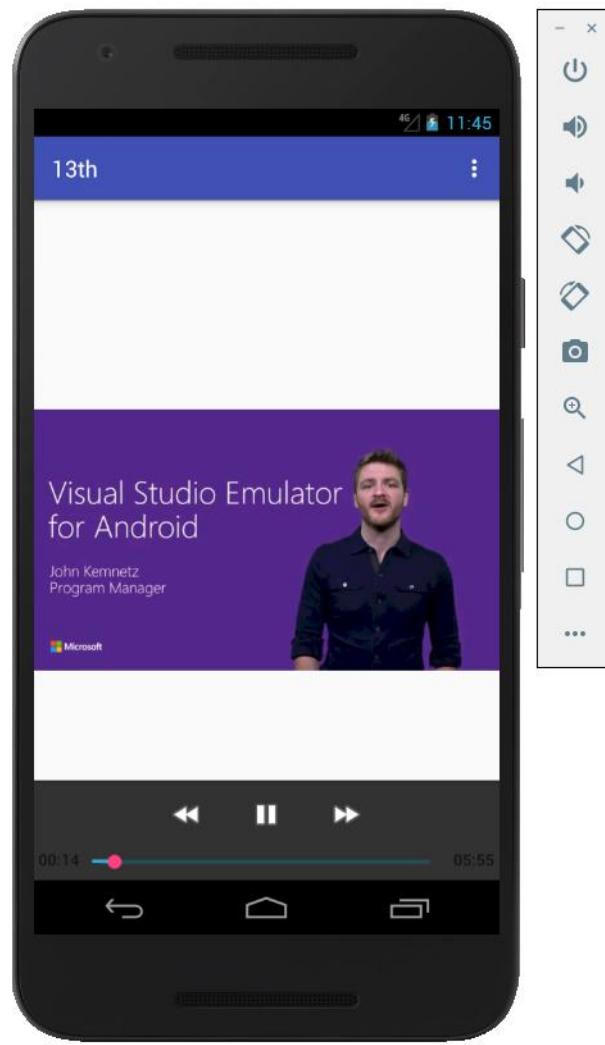


Рисунок 49 – Запущенное приложение с элементами управления видео

Обратите внимание, что видео может и не проигрываться на эмуляторе, но прекрасно воспроизводиться на реальном устройстве.

Для того чтобы проигрывать видео из интернета по ссылке, нужно воспользоваться методом setVideoPath объекта videoView, например:

```
videoView.setVideoPath("http://video.ch9.ms/some.mp4");
```

и добавить соответствующее разрешение в AndroidManifest.xml перед тэгом <application>:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Лабораторная работа №14. Работа с камерой

Задание: создать приложение, при запуске которого активируется фотокамера телефона, производится снимок, и этот снимок помещается в ImageView интерфейса приложения.

Создайте проект с Empty Activity, перейдите в файл activity_main.xml, удалите стандартное текстовое поле и расположите на экране приложения элементы ImageView и Button. Расставьте элементы так, чтобы элемент ImageView занимал практически весь экран (у него в ширине и высоте должно быть установлено значение match_parent).

Далее перейдите в файл MainActivity.java и объявите перед методом onCreate переменную imageView:

```
ImageView imageView;
```

Затем внутри метода onCreate свяжите переменные (объекты) с интерфейсом, как обычно:

```
imageView = findViewById(R.id.imageView);
Button button = findViewById(R.id.button);
```

Далее создайте для кнопки listener и метод onClick. В методе onClick напишите:

```
Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(intent, 0);
```

То есть после нажатия на кнопку загружается экран с камерой телефона, где можно сделать снимок. Теперь необходимо создать метод onActivityResult, который будет срабатывать после возвращения на начальное activity:

```
@Override  
protected void onActivityResult(int requestCode, int resultCode,  
Intent data) {  
    super.onActivityResult(requestCode, resultCode, data);  
    Bitmap bitmap = (Bitmap) data.getExtras().get("data");  
    imageView.setImageBitmap(bitmap);  
}
```

Приложение готово. Не забудьте добавить в приложение опциональное меню с пунктом о программе, который открывает диалоговое окно с вашими ФИО и краткой информацией о приложении. Запустите приложение, нажмите на кнопку, и должно появиться окно, где эмулируется работа камеры (рисунок 50). Сделайте снимок и вернитесь в activity (при необходимости нажмите на галочку), где в элементе ImageView уже будет находиться сделанный снимок (рисунок 51).

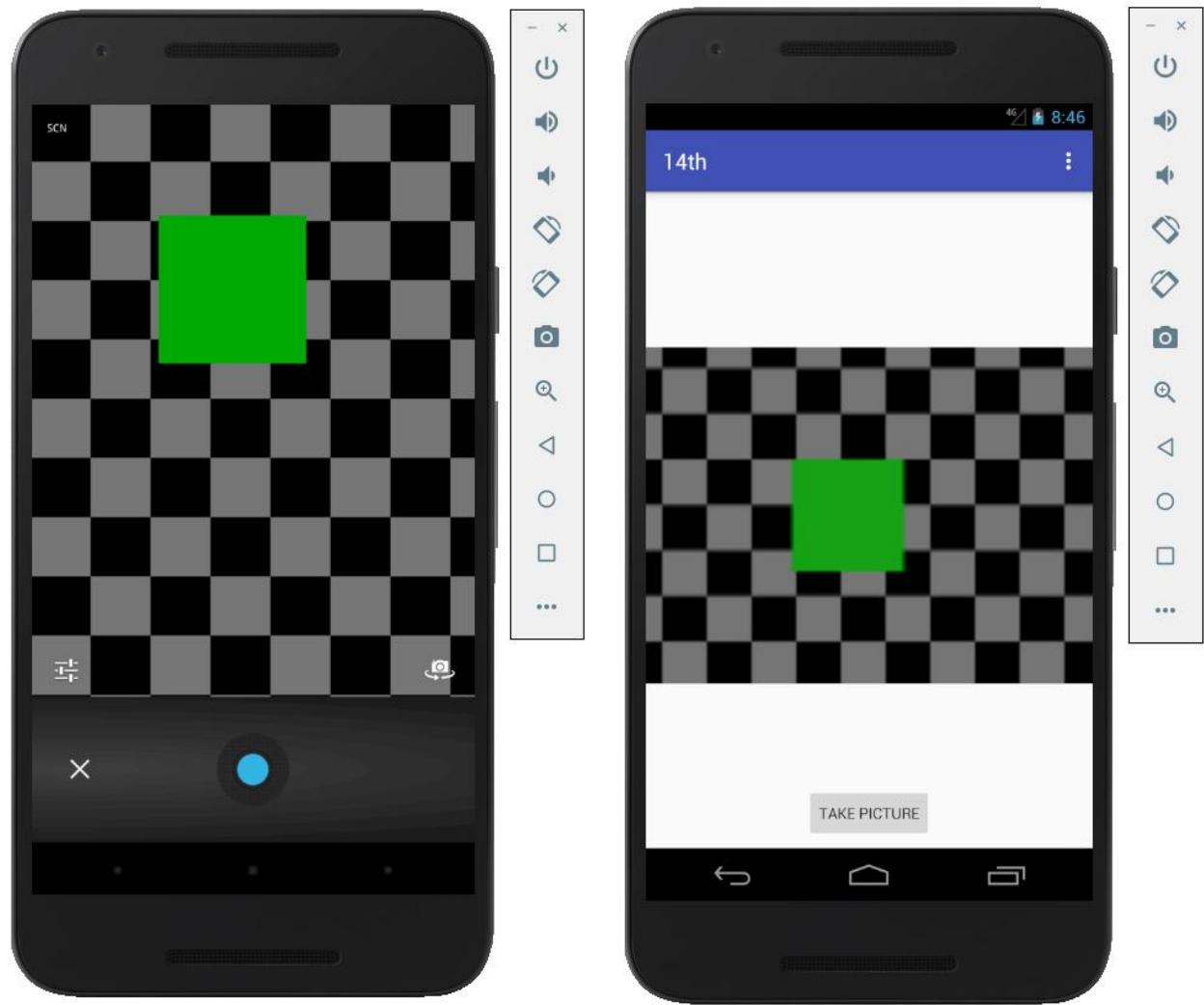


Рисунок 50 – Окно камеры

Рисунок 51 – Activity со снимком

Если к компьютеру подключена веб-камера, или она есть у Вас на ноутбуке, можно попытаться вывести изображение с неё. Для этого зайдите в список симуляторов (Tools -> Android -> AVD Manager), нажмите на значок в виде карандаша (справа), затем Show Advanced Settings и поменяйте в разделе Camera пункт Emulated на Webcam0. Сохраните настройки и попробуйте заново запустить приложение. Не забудьте, что если Вы выбрали Webcam0 для Front Camera, надо нажать на соответствующий значок в правом нижнем углу интерфейса захвата изображения (см. рис. 50), чтобы переключиться с основной камеры на фронтальную (сэлфи) камеру.

Лабораторная работа №15. Работа с настройками и БД

Задание: создать приложение, работающее с SharedPreferences и сохраняющее настройки, а также работающее с БД SQLite - заполняющее БД по нажатию кнопки 1 с помощью EditText, и выводящее все записи этой БД в какой-нибудь интерфейсный элемент с помощью кнопки 2 (в виде списка, datagrid или просто правильно настроенного TextView).

Практически во всех приложениях есть какие-то данные, которые надо сохранять при перезапуске приложения, например настройки. Чтобы приложение имело возможность сохранить данные, обычно используют Shared Preferences.

Создайте проект с Settings Activity, перейдите в файл SettingsActivity.java. В последней версии Android Studio (4.1.2) в этом файле используется уже готовый фрагмент с некоторыми настройками, поэтому менять его не надо.

Для начала добавим в приложение класс MainActivity.java и файл интерфейса main_activity.xml (правый клик мыши по названию упаковки класса -> New -> Activity -> Empty Activity). Затем добавим в MainActivity.java опциональное меню с пунктом о программе, который открывает диалоговое окно с вашими ФИО и краткой информацией о приложении. Затем добавим новый пункт в файл menu.xml и в код метода onOptionsItemSelected - кнопку Settings для перехода на экран SettingsActivity, соответственно:

```
<item android:id="@+id/settings" android:title="Настройки" />

if (item.getItemId() == R.id.settings) {
```

Для этого в операторе if для пункта settings осуществим переход:

```
Intent intent = new Intent(MainActivity.this,  
SettingsActivity.class);  
startActivity(intent);
```

Последнее, что нужно сделать – это сообщить системе, что главным в нашем приложении является MainActivity, а не SettingsActivity. Для этого перейдём в файл манифеста приложения AndroidManifest.xml и перенесём следующие строки из тэга activity для SettingsActivity в тэг activity для класса MainActivity:

```
android:label="@string/app_name">  
<intent-filter>  
    <action android:name="android.intent.action.MAIN" />  
  
    <category android:name="android.intent.category.LAUNCHER" />  
</intent-filter>
```

Запустите приложение, откроется пустое окно, содержащее только опциональное меню, см. рис. 52. Перейдите в настройки, измените параметры настроек и выйдите из приложения. Затем снова запустите приложение и вернитесь в настройки, параметры остались такими же, какими они были до выхода из приложения. Пример стандартных настроек приведён на рисунке 53.

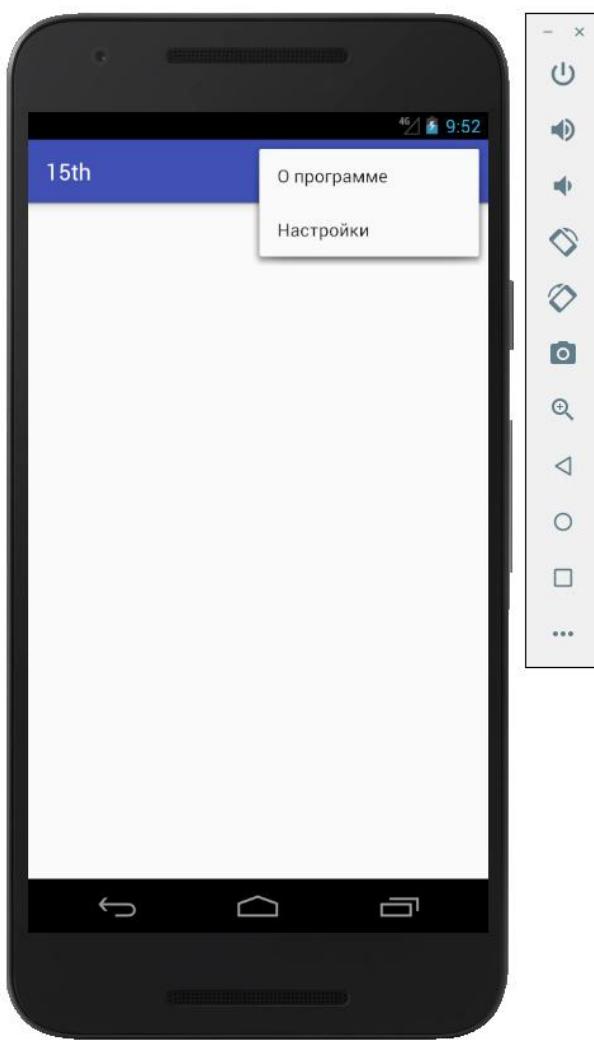


Рисунок 52 – Интерфейс приложения

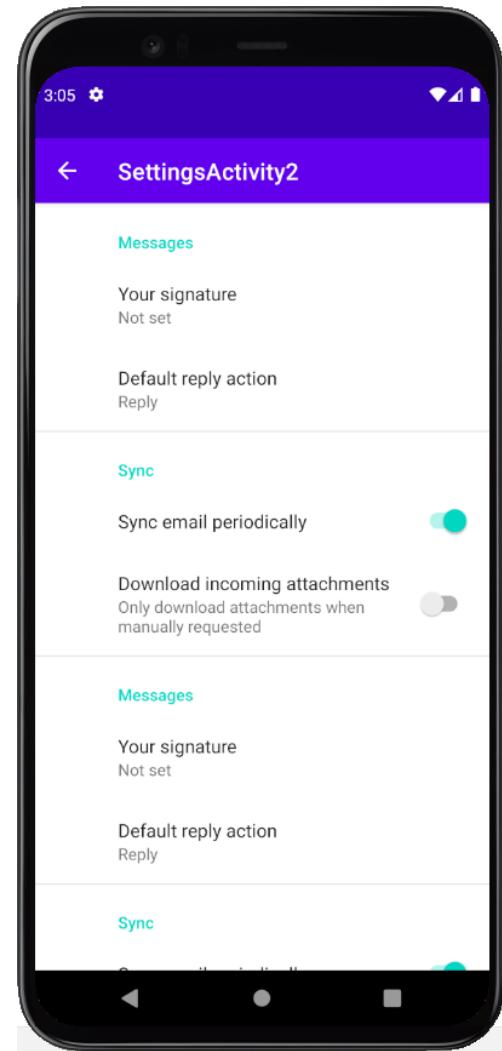


Рисунок 53 – Настройки программы

Однако в этом проекте всё сделано за нас, ни о чём думать не надо, всё уже готово с помощью класса `SettingsActivity`. Поэтому, чтобы лучше прочувствовать механизм `SharedPreferences`, сделаем следующее. Создадим элемент `EditText` в интерфейсе приложения. Затем создадим переменную в классе `MainActivity.java`:

```
private EditText edit1;
```

В методе `onCreate` класса `MainActivity.java` добавим ссылку на переменную и создадим настройки с помощью класса `SharedPreferences`, после этого тексту в `EditText` присвоим значение, сохранённое ранее:

```
edit1 = findViewById(R.id.editText);
SharedPreferences save = getSharedPreferences("SAVE", 0);
edit1.setText(save.getString("text", ""));
```

Текст мы будем сохранять по закрытию приложения, поэтому создадим с помощью конструктора метод onStop и добавим туда следующее:

```
SharedPreferences save = getSharedPreferences("SAVE", 0);
SharedPreferences.Editor editor = save.edit(); //создаём
редактор shared preferences
editor.putString("text",edit1.getText().toString()); //сохраняем
текст из edit1
editor.commit(); //применение редактирования shared preferences
```

Проверим работу приложения – запускаем, набираем текст, выходим из приложения, затем запускаем снова и видим, что текст сохранился.

Теперь рассмотрим работу приложения с базой данных. Для этого понадобятся базовые знания SQL.

Перейдите в файл MainActivity.java и в методе onCreate создайте базу данных с названием «DBName»:

```
SQLiteDatabase db = openOrCreateDatabase("DBname", MODE_PRIVATE,
null);
```

Далее в базе данных нужно создать таблицу:

```
db.execSQL("CREATE TABLE IF NOT EXISTS MyTable5 (_id INTEGER
PRIMARY KEY AUTOINCREMENT, Name VARCHAR);");
```

И теперь необходимо заполнить созданную таблицу:

```
db.execSQL("INSERT INTO MyTable5 VALUES ('1','some text');");
```

После работы с базой данных её обязательно необходимо закрыть:

```
db.close();
```

Запустите приложение, и если не возникло никаких ошибок, значит создалась база данных, затем в ней создалась таблица и в таблице создалась запись.

Так как не было создано никакого взаимодействия с интерфейсом, то для проверки содержимого базы данных необходимо закомментировать последнюю строку кода, отвечающую за заполнение базы данных (создание первой записи). Далее перед кодом закрытия базы данных необходимо создать указатель:

```
Cursor cursor = db.rawQuery("SELECT * FROM Mytable5", null);
```

И передвинуть его на первую запись:

```
cursor.moveToFirst();
```

Теперь нужно вывести в лог поле Name из первой записи:

```
Log.d("ME", cursor.getString(cursor.getColumnIndex("Name")));
```

После этого запустите приложение и в Android Studio на нижней панели перейдите в Logcat. Далее в поисковой строке данного раздела нужно написать «ME» и нажать Enter, после чего по заданному фильтру найдётся нужный нам лог:

```
02-12 10:54:36.838 1523-1523/com.example.me.a15thdb D/ME: some
text
```

Данный лог свидетельствует о том, что была корректно создана база данных, таблица и запись в таблице, а также корректно составлен запрос на извлечение нужных данных.

Теперь необходимо добавить в приложение работу с базой данных через интерфейсные элементы. Интерфейс нашего приложения пустой, этим надо воспользоваться. Например, можно добавить поле EditText и кнопку Button для добавления записи в таблицу, а также ListView и кнопку Button для отображения всех данных в таблице. Можно также добавить кнопки для редактирования и удаления записей из БД – таким образом, мы охватим все наиболее используемые команды языка SQL: SELECT, INSERT, UPDATE, DELETE. Пример интерфейса показан на рисунке 54.

Сформулируем логику приложения. Как показано на рисунке 54, мы хотим, чтобы в приложении можно было добавлять запись в таблицу базы данных с помощью кнопки Add и поля EditText, чтобы можно было просматривать список всех записей в этой таблице, а также - чтобы можно было редактировать запись в таблице и удалять запись. Допустим, что редактировать запись мы будем с помощью диалогового окна, а для отдельной записи в списке ListView предусмотрим элементы TextView и 2 Button для отображения записи и редактирования + удаления записи из таблицы. Таким образом, у нас будет кастомный список ListView и кастомный диалог AlertDialog в приложении.

Первое, что нужно сделать, это – нарисовать соответствующий интерфейс для элемента списка и диалога. Создадим в папке Layout 2 файла: list_item.xml и dialog.xml. Файл dialog.xml будет содержать только один элемент – EditText для ввода нового значения записи таблицы базы данных, больше ничего не нужно. Его можно расположить по центру экрана с помощью ConstraintLayout. Пример расположения элементов в list_item.xml с помощью того же ConstraintLayout приведён на рисунке 55.

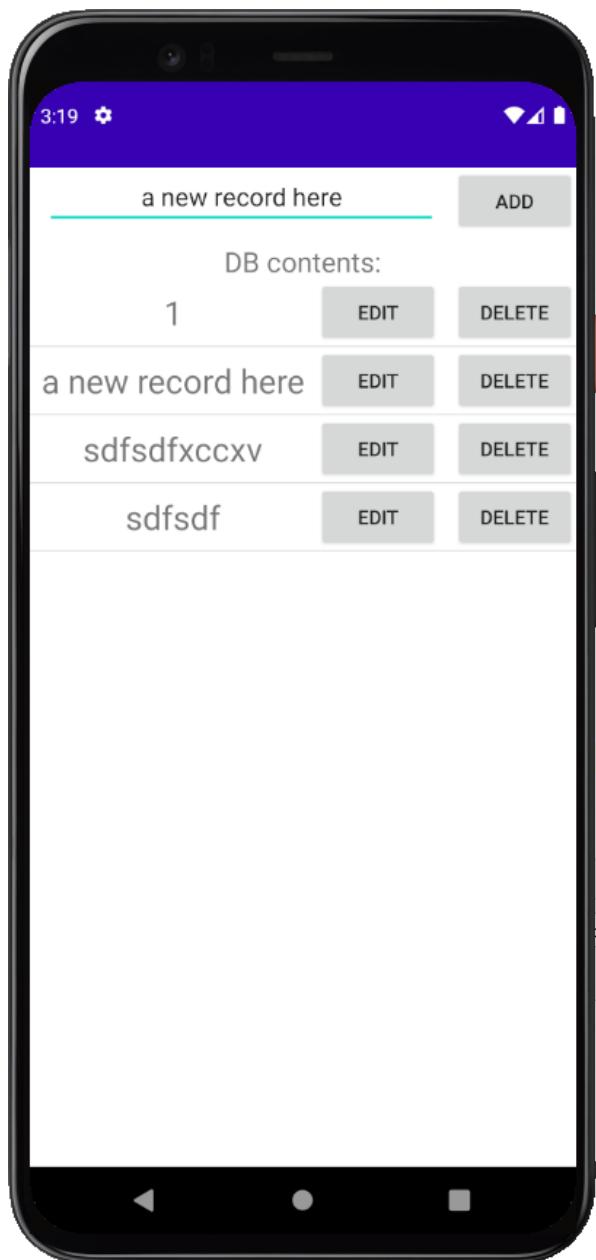


Рисунок 54 – Пример интерфейса
для приложения с БД

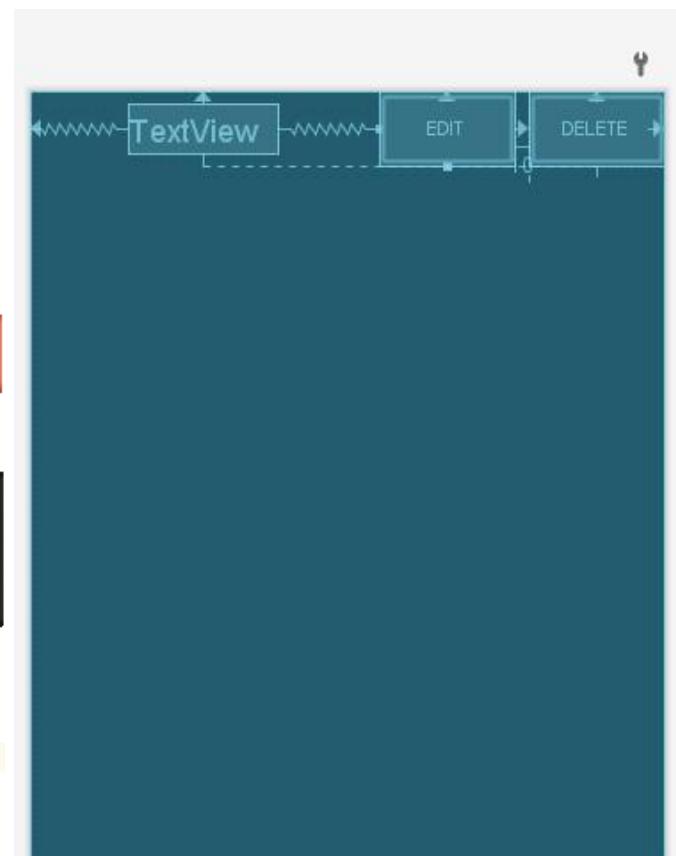


Рисунок 55 – Пример расположения
элементов в list_item.xml

Далее в MainActivity создадим некоторые переменные, которые нам понадобятся, и напишем код для метода onCreate, в котором мы будем загружать данные из БД при загрузке приложения и добавлять новую запись в БД:

```
public class MainActivity extends ListActivity {
    Integer i;
    String[] from;
```

```
int[] to;
static ListView listView;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    from = new String[]{"Name"};
    to =new int[] {R.id.textView};
    Button btnadd = findViewById(R.id.buttonAdd);
    final EditText editadd =
findViewById(R.id.editTextAddingARecord);
    SQLiteDatabase db =
openOrCreateDatabase("DBname", MODE_PRIVATE, null);
    db.execSQL("CREATE TABLE IF NOT EXISTS MyTable5 (_id
INTEGER PRIMARY KEY AUTOINCREMENT, Name VARCHAR);");
    Cursor cursor = db.rawQuery("SELECT * FROM Mytable5",
null);
    i=cursor.getCount()+1;
    if (cursor.getCount()>0) {
        MyCursorAdapter scAdapter = new
MyCursorAdapter(MainActivity.this,R.layout.list_item,cursor,from
,to);
        listView = getListView();
        listView.setAdapter(scAdapter);
    }
    db.close();

    btnadd.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            //TODO
        }
    });
}
}
```

Код метода onClick кнопки ADD будет следующим:

```
db.execSQL("INSERT INTO MyTable5 VALUES  
('"+i+"', '"+editadd.getText().toString()+"')");  
//i++;  
Cursor cursor = db.rawQuery("SELECT * FROM Mytable5", null);  
MyCursorAdapter scAdapter = new  
MyCursorAdapter(MainActivity.this,R.layout.list_item,cursor,from  
,to);  
listView = getListView();  
listView.setAdapter(scAdapter);  
db.close();
```

```
Toast.makeText(getApplicationContext(), "a row added to the
table", Toast.LENGTH_LONG).show();
```

Однако, чтобы избежать метаморфоз с PRIMARY KEY при добавлении новой записи после удаления некоторых старых нужно написать дополнительный код в начале этого метода, чтобы найти подходящее значение для id очередной записи, которое удовлетворяло бы ограничению на уникальность. Иначе после удаления 2 записи при наличии 1 и 3 записи в таблице при попытке добавления очередной записи добавится запись с id=2, после чего при очередном добавлении записи приложение вылетит с ошибкой, т.к. запись с id=3 в таблице уже есть. Этот код выглядит следующим образом:

```
SQLiteDatabase db =
openOrCreateDatabase("DBName", MODE_PRIVATE, null);
Cursor cursor2 = db.rawQuery("SELECT * FROM Mytable5", null);
i=cursor2.getCount()+1;
//цикл для того, чтобы подбирать значения _id и не допускать
повторения одинаковых значений (primary key как-никак)
for (int k=1;k<=i;k++) {
    Cursor cursor3 = db.rawQuery("SELECT * FROM Mytable5 WHERE
_id='"+k+"'", null);
    if (cursor3.getCount()==0) {
        i=k;
        break;
    }
}
```

И, как Вы уже заметили, для того, чтобы приведённый выше код работал, нужно создать дополнительный класс MyCursorAdapter (чтобы не пихать весь код в MainActivity.java), занимающийся обработкой нажатий кнопок EDIT и DELETE в интерфейсе приложения для каждой отдельной записи:

```
public class MyCursorAdapter extends SimpleCursorAdapter {
    private int layout_;
    private Cursor cursor;
    String[] from;
    int[] to;
```

```

ListView listView;
EditText edit2;

    public MyCursorAdapter(Context context, int layout, Cursor c, String[] from, int[] to) {
        super(context, layout, c, from, to);
        layout_=layout;
        cursor=c;
    }

    @Override
    public void bindView(View view, Context _context, Cursor cursor) {
        String data =
cursor.getString(cursor.getColumnIndex("Name"));
        int id = cursor.getInt(cursor.getColumnIndex("_id"));
        TextView text =
view.findViewById(R.id.textViewListItemText);
        text.setText(data);
        Button butdel = view.findViewById(R.id.buttonDelete);
        Button butedit = view.findViewById(R.id.buttonEdit);
        listView = MainActivity.listView;
        butdel.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                SQLiteDatabase db =
_context.openOrCreateDatabase("DBName", MODE_PRIVATE, null);
                db.execSQL("DELETE FROM MyTable5 WHERE
_id='"+id+"'");
                Cursor cursor = db.rawQuery("SELECT * FROM
Mytable5", null);
                from = new String[]{"Name"};
                to =new int[] {R.id.textView};
                MyCursorAdapter scAdapter = new
MyCursorAdapter(_context,R.layout.list_item,cursor,from,to);
                listView.setAdapter(scAdapter);
                db.close();
                Toast.makeText(_context,"row deleted from the db
id='"+id,Toast.LENGTH_LONG).show();
            }
        });
        butedit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                AlertDialog.Builder dialog = new
AlertDialog.Builder(_context);
                dialog.setMessage("Enter new value:");
                dialog.setTitle("Changing the item");
                LayoutInflator inflater = new
LayoutInflater(_context) {
                    @Override
                    public LayoutInflator cloneInContext(Context
context) {

```

```
        return null;
    }
}
View dialogview =
inflater.inflate(R.layout.dialog,null);
dialog.setView(dialogview);
edit2 =
dialogview.findViewById(R.id.editTextCnahgeDBRecord);
edit2.setText(text.getText().toString());
dialog.setNeutralButton("OK", new
DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog,
int i) {
        SQLiteDatabase db =
_context.openOrCreateDatabase("DBName", MODE_PRIVATE,null);
db.execSQL("UPDATE MyTable5 SET
Name='"+edit2.getText().toString()+"' WHERE _id="+id+"");
Cursor cursor = db.rawQuery("SELECT *
FROM Mytable5", null);
from = new String[]{"Name"};
to =new int[] {R.id.textView};
MyCursorAdapter scAdapter = new
MyCursorAdapter(_context,R.layout.list_item,cursor,from,to);
listView.setAdapter(scAdapter);
db.close();
Toast.makeText(_context,"row edited from
the db row id="+id,Toast.LENGTH_LONG).show();
dialog.dismiss();
}
});
dialog.setIcon(R.mipmap.ic_launcher_round);
AlertDialog alertDialog = dialog.create();
alertDialog.show();
}
}
@Override
public View newView(Context context, Cursor cursor,
 ViewGroup parent) {
    LayoutInflater inflater = (LayoutInflater)
context.getSystemService(context.LAYOUT_INFLATER_SERVICE);
    View view = inflater.inflate(layout_,parent,false);
    return view;
}
}
```

И всё - приложение, работающее с базой данных и включающее основные операции SQL завершено. Единственное, что нужно помнить – это

приложение не идеально; все операции с записями хорошо бы перенести в отдельный класс (как и SELECT, и INSERT), причём – в отдельные методы этого класса, называющиеся соответствующим образом; затем – интерфейс, конечно же, нужно загружать в отдельном асинхронном потоке, т.к. – представьте, что у Вас в таблице 1000 строк, а не 10-15; что тогда будет с работой приложения?

Лабораторная работа №16. Облачное мобильное приложение

Задание: Создать простейшее облачное мобильное приложение, используя Firebase и Android Studio.

Для выполнения этого задания можно выбрать любой облачный сервис, будь то push-уведомления, база данных реального времени, аналитика мобильного приложения (не в Google Play, а с помощью сторонних средств, например, Flurry, MixPanel, Crashlytics и т.д.), использование облачных сервисов, связанных с тестированием, распознаванием изображений и т.д. Рассмотрим один из простых примеров – создание push-уведомлений с помощью Firebase.

Первое, что нужно сделать – это создать новый проект в консоли Firebase (<https://console.firebaseio.google.com/>). Добавить проект – назовите как-то проект, выберите местоположение и нажмите на чекбокс "Я принимаю", а затем – на кнопку Создать проект, см. рисунок 46. Затем нажмите на Продолжить.

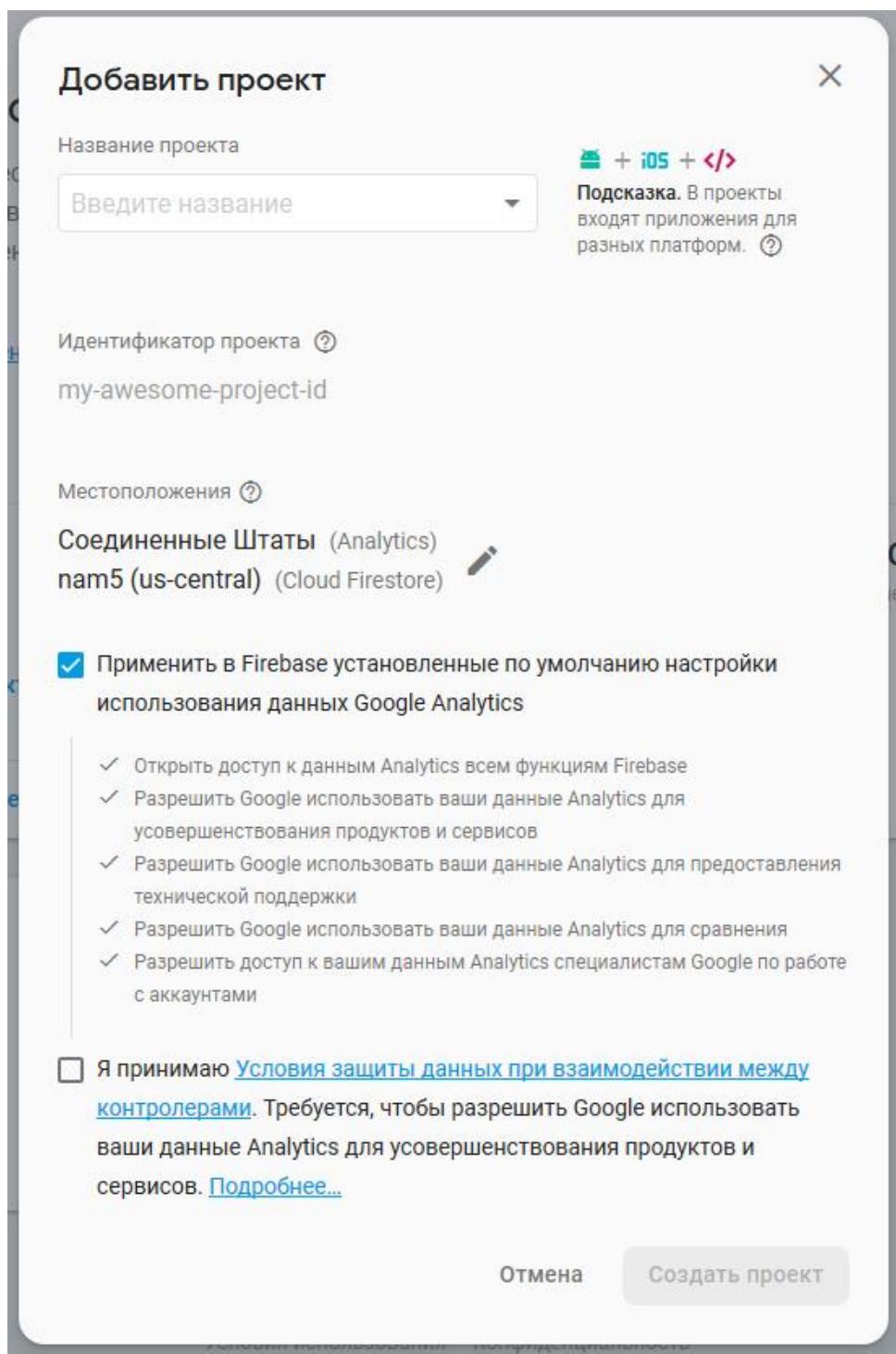


Рисунок 46 – Создание проекта в Firebase

Откроется окно управления проектом, в котором нам нужно нажать на кнопку с иконкой ОС Андроид над надписью Добавьте приложение. После этого нам потребуется ввести название пакета, поэтому перейдём в Android Studio и создадим новый проект с Empty Activity. Добавим в проект

опциональное меню с пунктом вызова окна О программе, как обычно. Затем перейдём в файл манифеста и скопируем значение переменной package, которое вставим в окно Firebase в строку Название пакета Android. Дополнительно можно заполнить поля псевдонима и хэша сертификата, но это необязательно. Нажмём на кнопку Зарегистрировать приложение.

Далее скачаем файл конфигурации google-services.json и поместим его в корень нашего проекта. Для этого переключим вид в Project Explorer (слева вверху) с Android на Project, щёлкнем правой кнопкой мыши по папке app и нажмём Paste, и в открывшемся окне – OK. В режиме Project мы увидим файл json, а в режиме Android его не видно. Затем переключитесь обратно в режим Android.

В окне Firebase браузера нажмём Далее.

Теперь надо изменить файлы gradle. Следуйте инструкциям в окне браузера, так как там может быть более новая версия, чем здесь. Итак, в файл build.gradle уровня проекта (Project: название) в dependencies надо добавить строку

```
classpath 'com.google.gms:google-services:4.0.1'
```

В файл build.gradle уровня приложения (Module: app) добавьте следующее в секцию dependencies:

```
implementation 'com.google.firebaseio:firebase-core:16.0.1'  
implementation 'com.google.firebaseio:firebase-messaging:17.4.0'
```

и в самый конце этого файла:

```
apply plugin: 'com.google.gms.google-services'
```

После этого синхронизируйте проект (Sync Now). В окне браузера нажмите кнопку Далее. После этого обязательно запустите приложение! При этом эмулятор или реальное устройство должны быть подключены к интернету. Если всё в порядке, в окне проверки связи с приложением браузера будет выведена соответствующая зелёная надпись (Поздравляем! Вы добавили сервис Firebase в свое приложение). Если связи с приложением нет, повторите предыдущие шаги.

Теперь удалим TextView "Hello World!", добавим в интерфейс на его место кнопку, и напишем внутри обработчика нажатия на неё следующий код:

```
String tkn = FirebaseInstanceId.getInstance().getToken();
Toast.makeText(MainActivity.this, "Current token ["+tkn+""]",
    Toast.LENGTH_LONG).show();
Log.d("App", "Token ["+tkn+"]");
```

Добавим в папку с MainActivity.java следующий класс в отдельном java-файле FireIDService.java:

```
public class FireIDService extends FirebaseInstanceIdService {
    @Override
    public void onTokenRefresh() {
        String tkn =
        FirebaseInstanceId.getInstance().getToken();
        Log.d("Not", "Token ["+tkn+"]");
    }
}
```

А затем добавим следующий класс, отвечающий за сервис, обрабатывающий входящее push-уведомление:

```
public class FireMsgService extends FirebaseMessagingService {

    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {
        super.onMessageReceived(remoteMessage);
```

```

Log.d("Msg", "Message received ["+remoteMessage+"]");

// Create Notification
Intent intent = new Intent(this, MainActivity.class);
intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
PendingIntent pendingIntent =
PendingIntent.getActivity(this, 1410,
                           intent, PendingIntent.FLAG_ONE_SHOT);

NotificationCompat.Builder notificationBuilder = new
NotificationCompat.Builder(this)

.setSmallIcon(R.drawable.common_full_open_on_phone)
.setContentView("Message")

.setContentText(remoteMessage.getNotification().getBody())
.setAutoCancel(true)
.setContentIntent(pendingIntent);

NotificationManager notificationManager =
(NotificationManager)

getSystemService(Context.NOTIFICATION_SERVICE);

notificationManager.notify(1410,
notificationBuilder.build());
}
}

```

И последнее, что нужно сделать – добавить в файл манифеста соответствующий сервис внутри тэга application:

```

<service android:name=".FireIDSService">
    <intent-filter>
        <action
android:name="com.google.firebaseio.INSTANCE_ID_EVENT"/>
    </intent-filter>
</service>

```

Теперь постройте apk-файл и установите приложение на эмуляторе или на реальном устройстве. После этого нажмите в браузере на кнопку Открыть консоль, нажмите на название своего проекта и выберите кнопку настроек справа от названия. Откроется окно настроек проекта, в котором на закладке Cloud Messaging можно увидеть ключ сервера для аутентификации. Зная этот

ключ, можно посыпать сообщения внутри приложения, с помощью другого приложения и с помощью самой консоли Firebase, при этом планируя их на определённое время, например, и для определённой аудитории.

Чтобы отправить сообщение, перейдите в категорию Grow -> Cloud Messaging и нажмите на кнопку Send your first message. Заполните поля (текст сообщения), нажмите кнопку Далее. Выберите приложение на следующем шаге (Аудитория) – при этом Вы можете выбирать сегмент пользователей – либо все пользователи, у которых установлено приложение, либо конкретного пользователя, либо – пользователей, подписанных на определённую тему. Нажмите кнопку Далее. В секции Планирование можно настроить расписание, когда пользователи получат сообщение, либо просто оставить Now для немедленной отправки сообщения. Нажмите Далее. В следующей секции можно задавать события-конверсии, чтобы понимать, откуда приходят пользователи – это больше связано с аналитикой приложения. Если перейти с помощью кнопки Далее к следующей секции, можно настроить дополнительные параметры, например – ключ конкретного пользователя, который получит уведомление. Для завершения нажмите кнопку Проверить. Вы увидите подтверждающее окно с уведомлением, в котором надо нажать кнопку опубликовать (рис. 47).

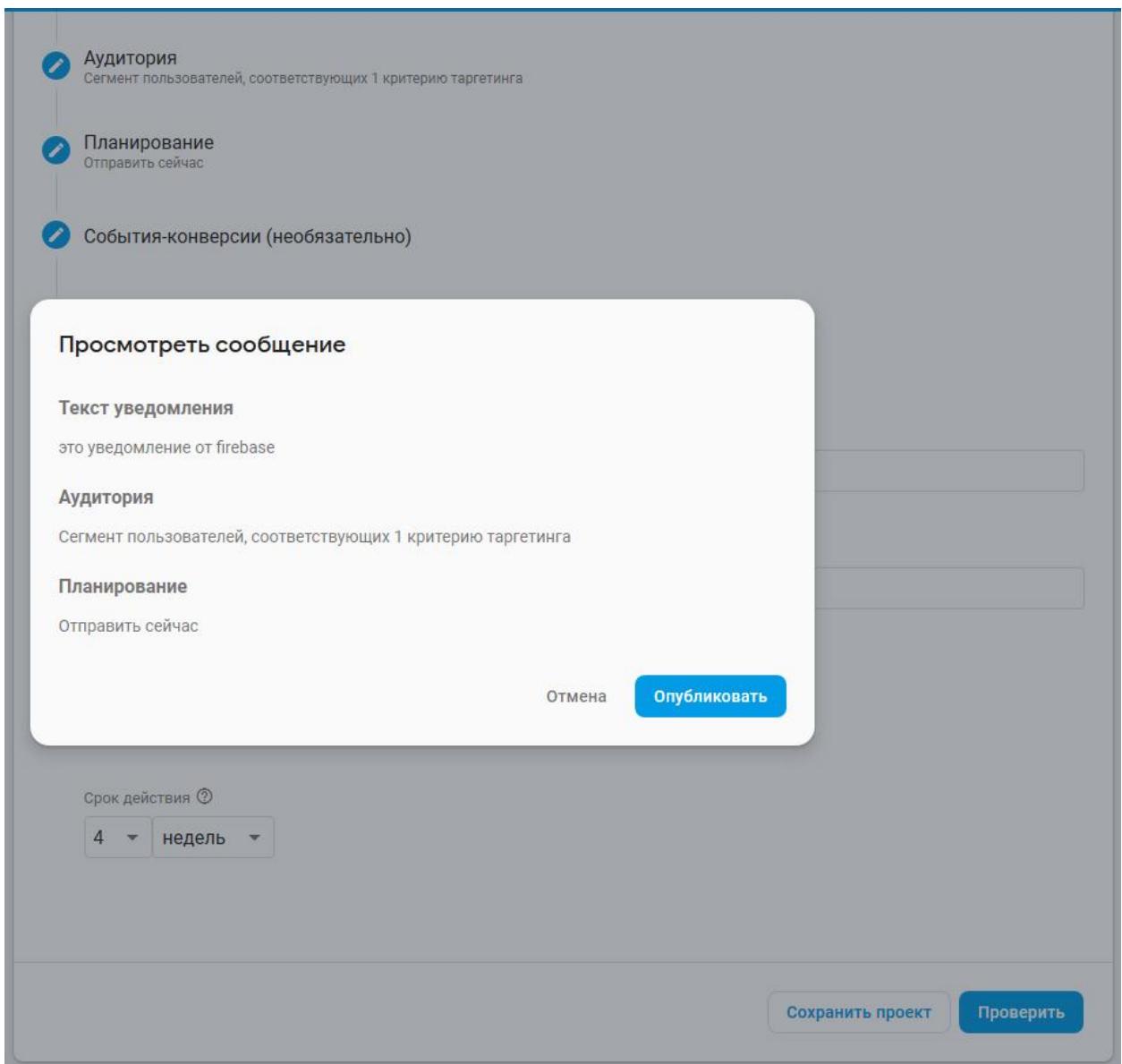
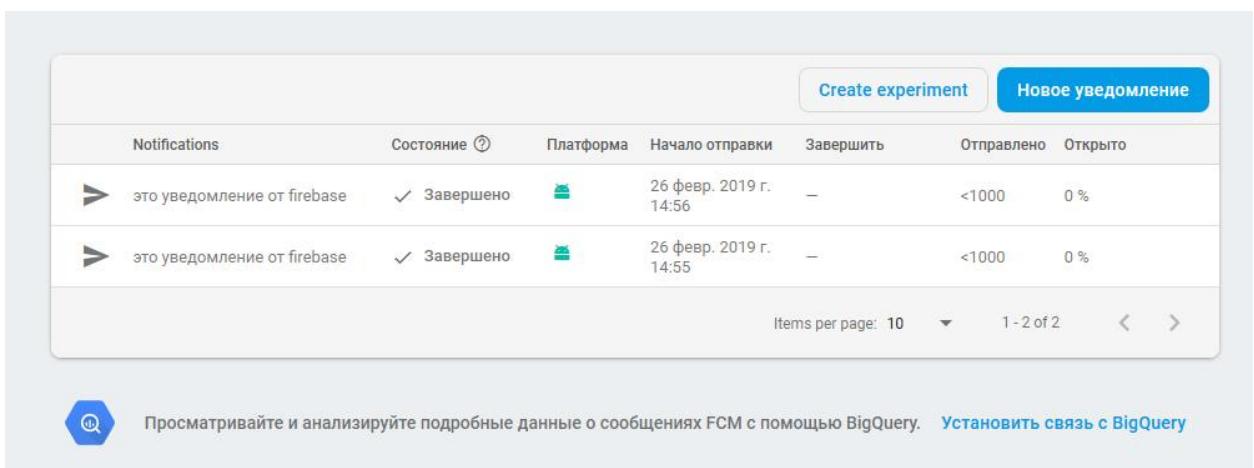


Рисунок 47 – Публикация push-уведомления с помощью Firebase

После нажатия на кнопку Опубликовать, даже если у Вас приложение не запущено на мобильном устройстве, появится уведомление с небольшой картинкой и стандартным звуком уведомления для Вашего устройства. В панели уведомлений Вы увидите тему уведомления и текст. При нажатии на уведомление запустится Ваше приложение. При выключенном интернете Вы увидите уведомление сразу, как только подключитесь к интернету. При этом консоль Firebase переключится на страницу с уведомлениями, где отобразится созданное уведомление с указанием некоторых подробностей, и

где можно будет его продублировать, создать новое и создать эксперимент с определённой целевой аудиторией (рис. 48).



The screenshot shows the Firebase Notifications dashboard. At the top right, there are two buttons: "Create experiment" and a blue button labeled "Новое уведомление" (New notification). Below this is a table with the following columns: Notifications, Состояние (Status), Платформа (Platform), Начало отправки (Start time), Завершить (End), Отправлено (Sent), and Открыто (Opened). Two rows of data are listed:

Notifications	Состояние	Платформа	Начало отправки	Завершить	Отправлено	Открыто
► это уведомление от firebase	✓ Завершено	tv	26 февр. 2019 г. 14:56	—	<1000	0 %
► это уведомление от firebase	✓ Завершено	tv	26 февр. 2019 г. 14:55	—	<1000	0 %

At the bottom of the table, there are controls for "Items per page: 10" and a page indicator "1 - 2 of 2". Below the table, there is a note: "Просматривайте и анализируйте подробные данные о сообщениях FCM с помощью BigQuery." followed by a link "Установить связь с BigQuery".

Рисунок 48 – Результат публикации push-уведомления

Можно написать приложение, в котором будут 2 кнопки: одна будет подписываться на уведомления, а другая – формировать и присыпать уведомление. Можно написать 2 приложения – одно отсылает уведомления, другое принимает. А можно использовать какой-то другой механизм работы с облаком, например, выводить в приложение аналитику по его крашам у пользователей с помощью Firebase.