

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**"МИРЭА - Российский технологический университет"**

РТУ МИРЭА

Институт информационных технологий

Кафедра математического обеспечения и стандартизации информационных технологий

# ОТЧЕТ

# ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1

«Создание первого приложения»

# по дисциплине

«Разработка мобильных приложений»

Выполнил студент группы ИКБО-06-21 Гейдаров Э.З.

Принял доцент кафедры МОСИТ Синицын И.В.

|  |  |
| --- | --- |
| Практическая работа выполнена | « » 2022 г. |
| (подпись студента) |
| «Зачтено» | « » 2022 г. |
|  | (подпись руководителя) |

Москва 2023

# Введение

Android Studio — интегрированная среда разработки для работы с платформой Android. С помощью Android Studio можно изучить и получить навыки основы разработки приложения для Android, включая создание простого пользовательского интерфейса и обработку ввода информации пользователем.

# Постановка задачи

1. Создать простой проект в Android Studio.
2. Произвести запуск приложения.
3. Создать простой интерфейс пользователя. Создать линейную разметку (Linear Layout). Добавить текстовые поля. Создать строковый ресурс. Добавить кнопки. Растянуть поле ввода. Запустить другое явление. Реализовать обработку нажатия на кнопку. Создать намерения (Intent).
4. Создать второе явление. Создать новое явление в Android Studio. Обеспечить получение намерения. Обеспечить отображение сообщения.

# Ход выполнения

1. **Создать простой проект в Android Studio**

# Создадим первое приложение в среде Android Studio для операционной системы Android. Откроем Android Studio и на начальном экране выберем пункт New Project:

# 

Рисунок 1 – начальный экран

При создании проекта Android Studio вначале предложит нам выбрать шаблон проекта:

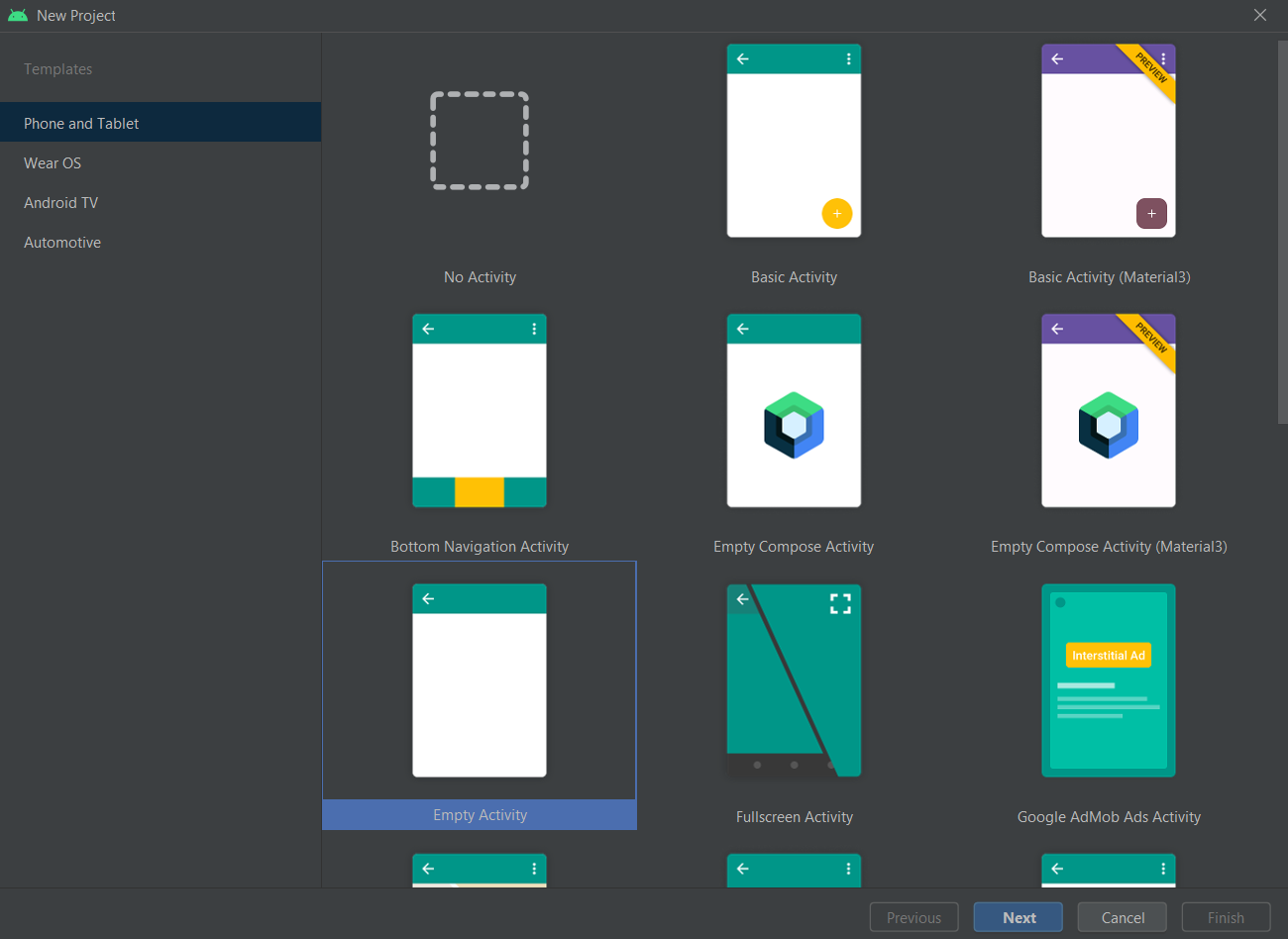


Рисунок 2 – шаблон проекта

Android Studio предоставляет ряд шаблонов для различных ситуаций. Выберем в этом списке шаблон Empty Activity, который предосавляет самый простейший фукционал, необходимый для начала, и нажмем на кнопку Next.

После этого отобразится окно настроек нового проекта:

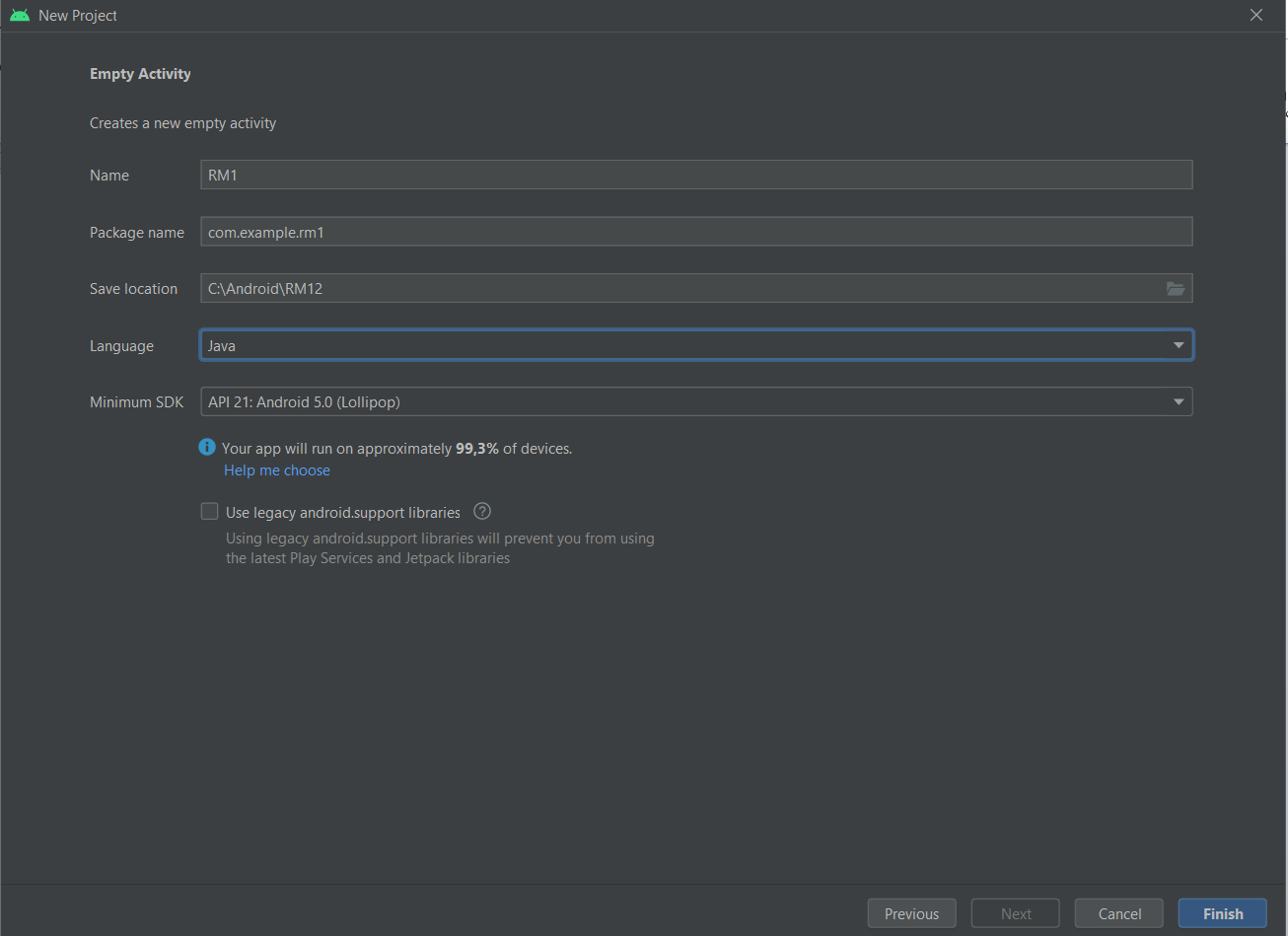


Рисунок 3 – окно настроек нового проекта

В окне создания нового проекта мы можем установить его начальные настройки:

* В поле Name вводится название приложения. Укажем в качестве имени название RM1;
* В поле Package Name указывается имя пакета, где будет размещаться главный класс приложения. В данном случае для тестовых проектов это значение не играет ольшого значения, поэтому установим com.example.rm1;
* В поле Save Location установливается расположение файлов проекта на жестком диске. Можно оставить значение по умолчанию;
* В поле Language в качестве языка программирования укажем Java;
* В поле Minimum SDK указывается самая минимальная поддерживаемая версия SDK. Оставим значение по умолчанию - API 21: Android 5.0 (Lollipop), которая означает, что наше приложение можно будет запустить начиная с Android 5.0, это 99,3% устройств. На более старых устройствах запустить будет нельзя.

Стоит учитывать, что чем выше версия SDK, тем меньше диапазон поддерживаемых устройств.

Далее нажмем на кнопку Finish, и Android Studio создаст новый проект:

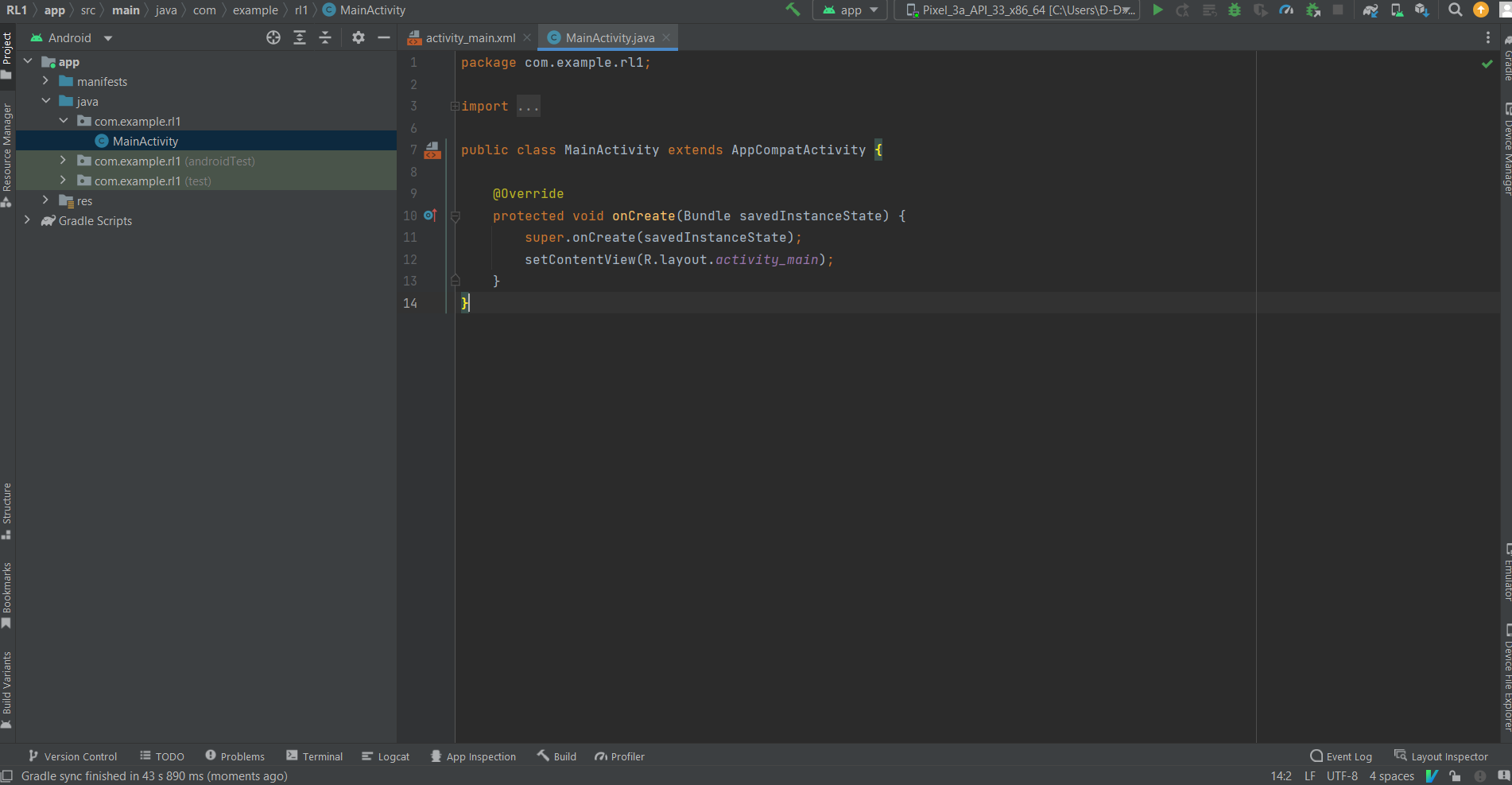


Рисунок 4 – создание нового проекта

1. **Произвести запуск приложения**

Созданный выше проект уже содержит некоторый примитивный функционал. Правда, этот функционал почти ничего не делает, только выводит на экран строку "Hello world!". Тем не менее, это уже фактически приложение, которое мы можем запустить.

Для запуска и тестирования приложения мы можем использовать эмуляторы или реальные устройства.

Для того чтобы протестировать приложение на эмуляторе надо выбрать соответствующий ему модуль в выпадающем списке сверху:

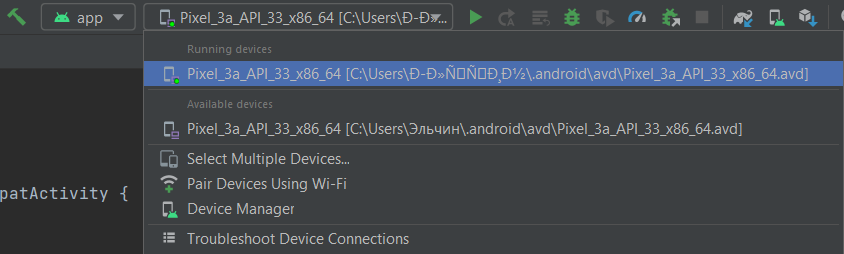


Рисунок 5 – выбор модуля

В Android Studio щелкните по кнопке Run на панели инструментов.

И после запуска мы увидим наше приложение на экране устройства:

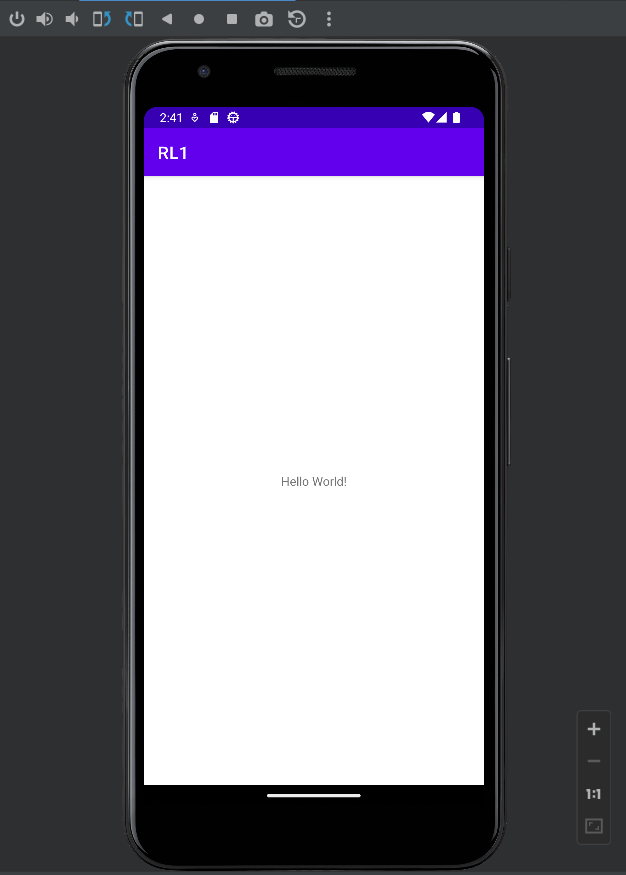


Рисунок 6 – приложение на устройстве

# Создать простой интерфейс пользователя

# Создание линейного макета.

# Необходимо перейти в layout/activity\_main.xml и вначале удалить [<TextView>](http://developer.android.com/reference/android/widget/TextView.html) элемент. Затем добавьте [android:orientation](http://developer.android.com/reference/android/widget/LinearLayout.html" \l "attr_android:orientation) атрибут и установите его в "horizontal".

Результат выглядит следующим образом:

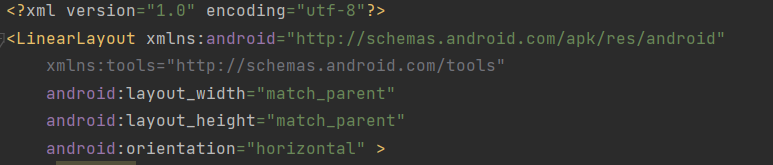


Рисунок 7 – фрагмент кода

[LinearLayout](http://developer.android.com/reference/android/widget/LinearLayout.html) представляет собой группу представлений (подкласс [ViewGroup](http://developer.android.com/reference/android/view/ViewGroup.html)) , который раскладывает дочерние представления в вертикальном или горизонтальном положении, как указано в [android:orientation](http://developer.android.com/reference/android/widget/LinearLayout.html" \l "attr_android:orientation) атрибуте. Каждый дочерний элемент [LinearLayout](http://developer.android.com/reference/android/widget/LinearLayout.html) появляется на экране в том порядке, в котором он появляется в XML.

Два других атрибута, [android:layout\_width](http://developer.android.com/reference/android/view/View.html" \l "attr_android:layout_width) и [android:layout\_height](http://developer.android.com/reference/android/view/View.html" \l "attr_android:layout_height), требуются для всех представлений для того, чтобы указать их размер.

Поскольку [LinearLayout](http://developer.android.com/reference/android/widget/LinearLayout.html) является корнем в макете, он должен заполнить всю область экрана, которая доступна для приложения, установив ширину и высоту в "match\_parent". Это значение указывает, что представление должно расширить свою ширину или высоту до соответствия ширине или высоте родительского представления.

Чтобы создать текстовое поле, редактируемое пользователем, добавьте [<EditText>](http://developer.android.com/reference/android/widget/EditText.html) элемент внутри [<LinearLayout>](http://developer.android.com/reference/android/widget/LinearLayout.html).

Как и любому [View](http://developer.android.com/reference/android/view/View.html) объекту, необходимо задать определенные XML атрибуты для указания[EditText](http://developer.android.com/reference/android/widget/EditText.html) свойств объекта. Вот как вы должны объявить его в [<LinearLayout>](http://developer.android.com/reference/android/widget/LinearLayout.html) элементе:

    <EditText android:id="@+id/edit\_message"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:hint="@string/edit\_message" />

1. [android:id](http://developer.android.com/reference/android/view/View.html#attr_android:id):

Он обеспечивает уникальный идентификатор для представления, который можно использовать для ссылки на объект из кода приложения для чтения и манипулирования объектом (вы увидите это в следующем уроке).Знак «собачки»(@) требуется, когда вы ссылаетесь на любой объект ресурса из XML. За ним следует тип ресурса(id в данном случае), косая черта, а затем имя ресурса(edit\_message).Знак плюс(+) перед типом ресурса требуется только тогда, когда вы определяете идентификатор ресурса впервые. При компиляции приложения, инструменты SDK используют этот имя идентификатора ресурса для создания нового идентификатора ресурса в вашем проекте вgen/R.java файле, который относится к [EditText](http://developer.android.com/reference/android/widget/EditText.html)элементу. После того, как идентификатор ресурса объявляется один раз таким образом, другие ссылки на него не требуют знака плюс. Использовать знак плюс необходимо только при указании нового идентификатора ресурса, но не требуется для конкретных ресурсов, таких как строки или макеты.

1. [android:layout\_width](http://developer.android.com/reference/android/view/View.html#attr_android:layout_width) и [android:layout\_height](http://developer.android.com/reference/android/view/View.html" \l "attr_android:layout_height):

Вместо того чтобы использовать конкретные размеры по ширине и высоте, "wrap\_content"указывает, что представление должно быть большим на столько, чтобы вместить содержимое представления. Если бы вы использовали вместо него "match\_parent", то [EditText](http://developer.android.com/reference/android/widget/EditText.html) элемент заполнил бы весь экран, потому что это бы соответствовало размеру родительскому[LinearLayout](http://developer.android.com/reference/android/widget/LinearLayout.html).

1. [android:hint](http://developer.android.com/reference/android/widget/TextView.html#attr_android:hint):

Это строка по умолчанию для отображения, когда текстовое поле пустое. Вместо использования жестко запрограммированной строки в качестве значения,"@string/edit\_message" значение ссылается на ресурс строки, определенный в отдельном файле. Поскольку это относится к конкретному ресурсу (это не просто идентификатор), ему не нужно указывать знак плюс. Тем не менее, т.к. вы еще не определили строковый ресурс, вы увидите ошибку компилятора на этом этапе. Мы исправим это, определив строку. Результат выглядит следующим образом:

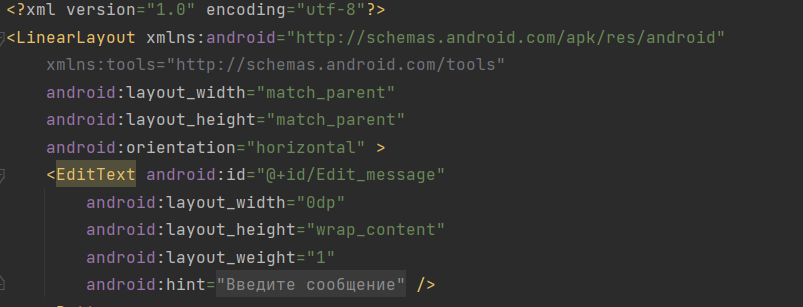


Рисунок 8 – результат изменения

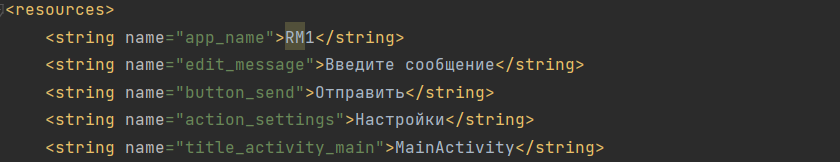
**Добавление строковых ресурсов.**

Когда вам нужно добавить текст в пользовательский интерфейс, вы всегда должны указывать каждую строку как ресурс. Строковые ресурсы позволяют управлять всем текстом пользовательского интерфейса в одном месте, что позволяет его легче найти и обновить текст. Использование внешних ресурсов для строк также позволяет локализовать ваше приложение под разные языки, предоставляя альтернативные определения для каждого строкового ресурса.

По умолчанию, ваш Android проект включает в себя файл строковых ресурсов res/values/strings.xml. Добавьте новую строку с именем "edit\_message" и задайте ей значение «Введите сообщение».

Пока вы находитесь в этом файле, также добавьте строку «Отправить» для кнопки, которую вы скоро добавите, называемую "button\_send".

В результате strings.xml выглядит следующим образом:

 Рисунок 9 – демонстрация strings.xml

## Добавление кнопки.

Теперь добавьте [<Button>](http://developer.android.com/reference/android/widget/Button.html) в макет, сразу после [<EditText>](http://developer.android.com/reference/android/widget/EditText.html) элемента:

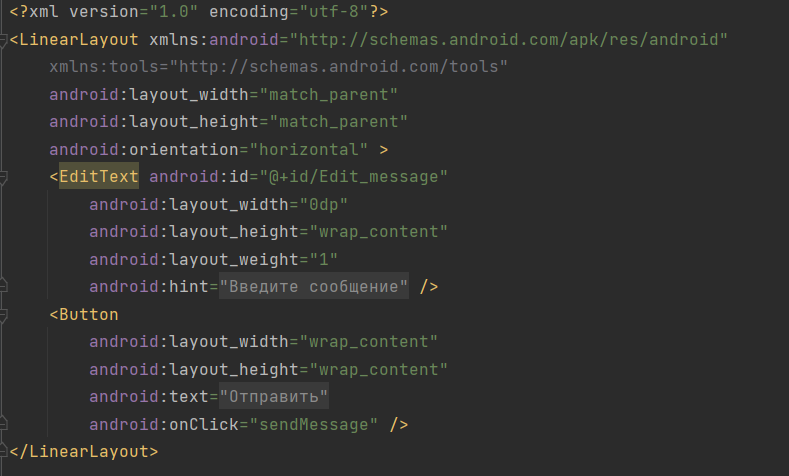


Рисунок 10 – добавление <Button>

Высота и ширина установлены в "wrap\_content" , поэтому кнопка будет такое размера, чтобы вместить текст кнопки. Этой кнопке не нужен [android:id](http://developer.android.com/reference/android/view/View.html" \l "attr_android:id) атрибут, потому что на него не будут ссылаться из кода деятельности.

**Сделать поле ввода по ширине экрана.**

Макет в настоящее время разработан, так что оба [EditText](http://developer.android.com/reference/android/widget/EditText.html) и [Button](http://developer.android.com/reference/android/widget/Button.html) виджета размера, необходимого для размещения их содержимого.

[EditText](http://developer.android.com/reference/android/widget/EditText.html) и [Button](http://developer.android.com/reference/android/widget/Button.html) виджеты с шириной установленной в "wrap\_content".

Это прекрасно работает для кнопки, но не так хорошо для текстового поля, потому что пользователь может ввести что-то больше. Так что было бы неплохо заполнить неиспользованную ширину экрана текстовым полем. Вы можете сделать это внутри [LinearLayout](http://developer.android.com/reference/android/widget/LinearLayout.html) с помощью weight свойства, которое можно указать с помощью [android:layout\_weight](http://developer.android.com/reference/android/widget/LinearLayout.LayoutParams.html" \l "weight) атрибута.

Значение weight является числом, указывающим количество оставшегося пространства, которое каждое представление должно занимать, по отношению к количеству занимаемого пространства соседними представлениями. Это работает примерно как соотношение ингредиентов в рецепте напитка: «2 части водки, 1 часть кофейного ликера» означает, что две трети напитка является водка. Например, если вы дадите одному из представлений вес 2, а другому вес 1, сумма будет 3, так что первое представление заполняет 2/3 оставшегося пространства, а второе представление заполняет все остальное. Если добавить третье представление и придать ему вес 1, то первые представление (с весом 2) теперь получит 1/2 оставшегося пространства, в то время как остальные два получат по 1/4 каждый.

Вес по умолчанию для всех представлений равно 0, так что если вы укажите любое значение веса больше 0 только одному представлению, то это представление заполнит все пространство, оставшееся после заполнения пространства всеми другими представлениями. Таким образом, чтобы заполнить оставшееся пространство в макете [EditText](http://developer.android.com/reference/android/widget/EditText.html) элементом, дайте ему вес 1 и оставьте кнопку без указания веса.

    <EditText

        android:layout\_weight="1"

        ... />

В целях повышения эффективности макета при указании веса, вы должны изменить ширину [EditText](http://developer.android.com/reference/android/widget/EditText.html) до нуля (0dp). Установка ширины в нуль повышает производительность макета, потому что использование "wrap\_content" в качестве ширины требует от системы вычисления ширины, которая в конечном счете не имеет значения, так как значение веса требует другого расчета ширины для заполнения оставшееся пространства.

    <EditText

        android:layout\_weight="1"

        android:layout\_width="0dp"

        ... />

Вот как ваш файл макета должен выглядеть:

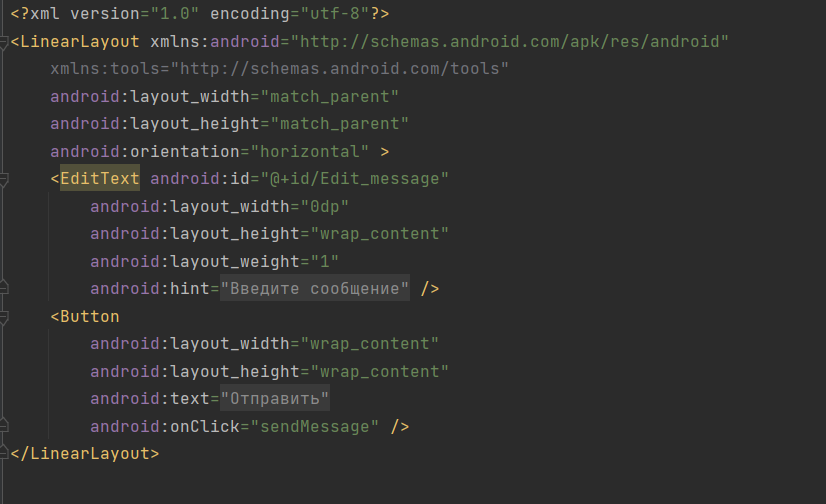


Рисунок 11 – файл макета

Этот макет применяется по умолчанию к [Activity](http://developer.android.com/reference/android/app/Activity.html) классу, который инструменты SDK сгенерировали при создании проекта, так что теперь вы можете запустить приложение, чтобы увидеть результаты:

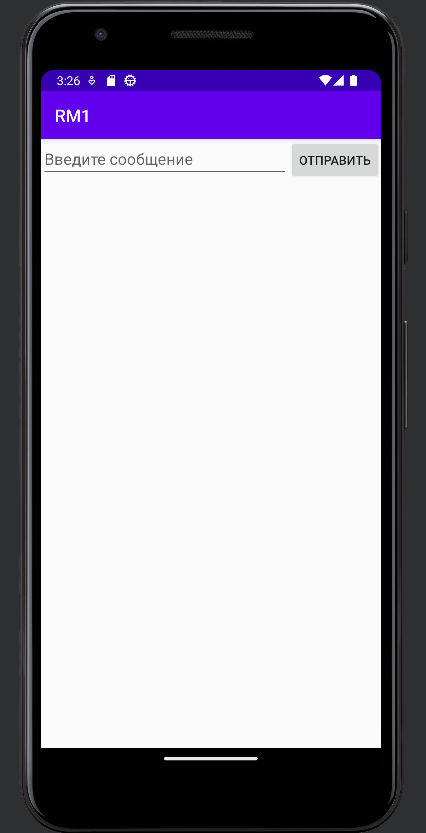


Рисунок 12 – результат запуска приложения

# Обеспечить отображение сообщения

# Ответить на кнопку «Отправить».

Чтобы ответить на событие нажатия кнопки, откройте fragment\_main.xml файла макета и добавьте [android:onClick](http://developer.android.com/reference/android/view/View.html" \l "attr_android:onClick" \t "_blank) атрибут к [<Button>](http://developer.android.com/reference/android/widget/Button.html) элементу:

<Button

    android:layout\_width="wrap\_content"

    android:layout\_height="wrap\_content"

    android:text="@string/button\_send"

    android:onClick="sendMessage" />

android:onClick значение атрибута, "sendMessage", это имя метода в вашей активити, который система вызывает когда пользователь нажимает кнопку.

Откройте MainActivity класс (расположенный в каталоге проекта src/) и добавьте соответствующий метод:

/\*\* Вызывается, когда пользователь нажимает кнопку «Отправить».

\*/

public void sendMessage(View view) {

    // Сделайте что-нибудь в ответ на кнопку

}

# Далее, заполним этот метод, чтобы прочитать содержимое текстового поля и доставить этот текст в другую активити.

# Создать Intent.

[Intent](http://developer.android.com/reference/android/content/Intent.html" \t "_blank) это объект, который обеспечивает связывание отдельных компонент во время выполнения (например, двух активити). Intent представляет «намерение что-то сделать». Вы можете использовать интент для широкого круга задач, но чаще всего они используются, чтобы начать другую активити .

Внутри sendMessage() метода, создайте Intent для запуска активити под названиемDisplayMessageActivity:

Intent intent = new Intent(this, DisplayMessageActivity.class;

Для этого требуется импортировать Intent класс:

import android.content.Intent;

### Отправка intent в другие приложения

Intent, созданный в этом уроке считается явным, поскольку Intent указывает точный компонент приложения, в которое интент следует отправить. Тем не менее, интенты также могут быть неявными, в этом случае Intent не указывает нужный компонент, позволяя любой программе установленной ​​на устройстве отвечать на интенты, пока она удовлетворяет спецификациям мета-данных действия, задаваемыми в различных Intent параметрах. Для получения дополнительной информации читайте курс [Взаимодействие с другими приложениями](https://www.fandroid.info/kurs-dlya-razrabotchikov-android-prilozhenij-chast-1-pristupaem-k-rabote/#8).

**Примечание:** Ссылка на DisplayMessageActivityвызовет ошибку, если вы используете интегрированную среду разработки, такую как Eclipse например, потому что класс еще не существует. Игнорируйте ошибку на данный момент; вы скоро создадите класс.

Intent не только позволяет начать другую Activity, но также может выполнять связь данных в Activity. В sendMessage() методе, используйте [findViewById()](http://developer.android.com/reference/android/app/Activity.html" \l "findViewById(int)" \t "_blank) для получения [EditText](http://developer.android.com/reference/android/widget/EditText.html" \t "_blank) элемента и добавьте его текстовое значение в Intent:

Intent intent = new Intent(this, DisplayMessageActivity.class);

EditText editText = (EditText) findViewById(R.id.edit\_message);

String message = editText.getText().toString();

intent.putExtra(EXTRA\_MESSAGE, message);

**Примечание:**  Необходим оператор импорта для android.widget.EditText. Мы определим EXTRA\_MESSAGE константу буквально сейчас.

Intent может нести коллекцию различных типов данных в виде пары ключ-значение, называемых Extras. Метод [putExtra()](http://developer.android.com/reference/android/content/Intent.html" \l "putExtra(java.lang.String,%20android.os.Bundle)" \t "_blank)  принимает имя ключа в первом параметре и значение во втором параметре.

Для того, чтобы Activity смогла запросить дополнительные данные, вы должны определить ключ для дополнений вашего интента, используя общедоступную константу. Так что добавьте EXTRA\_MESSAGE определение в начало MainActivity класса:

public class MainActivity extends ActionBarActivity {

    public final static String EXTRA\_MESSAGE = "com.example.myfirstapp.MESSAGE";

    ...

}

Хорошая практика, определять ключи для Intent Extras используя имя пакета вашего приложения в качестве префикса. Это гарантирует, что они уникальны, в случае когда ваше приложение взаимодействует с другими приложениями.

**Запуск второй Activity.**

Для запуска активити, вызовите [startActivity()](http://developer.android.com/reference/android/app/Activity.html" \l "startActivity(android.content.Intent)" \t "_blank) и передайте в него ваш Intent. Система получает этот вызов и запускает экземпляр Activity указанный в Intent.

С помощью этого нового кода, полный sendMessage() метод, который вызывается кнопкой Отправить теперь выглядит следующим образом:

/\*\* Вызывается, когда пользователь нажимает кнопку «Отправить».

\*/

public void sendMessage(View view) {

    Intent intent = new Intent(this, DisplayMessageActivity.class);

    EditText editText = (EditText) findViewById(R.id.edit\_message);

    String message = editText.getText().toString();

    intent.putExtra(EXTRA\_MESSAGE, message);

    startActivity(intent);

}

**Создание второй Activity.**

### Для создания новой Activity в **Android Studio:**

В главном меню выберите **File>New>Activity>Basic Activity.**

Заполните поля в окне мастера создания активити:

* **Activity Name**: DisplayMessageActivity;
* **Layout Name**: activity\_display\_message.

Остальные поля оставьте по умолчанию. Нажмите **Finish**.

Удаляем ненужные вкладки и оставляем только необходимое для нашего задания, то есть MainActivity и только что созданный DisplayMessageActivity.

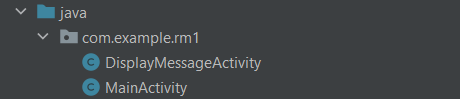


Рисунок 13 – демонстрация каталогов

DisplayMessageActivity класс должен выглядеть следующим образом:

public class DisplayMessageActivity extends ActionBarActivity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_display\_message);

        if (savedInstanceState == null) {

            getSupportFragmentManager().beginTransaction()

                .add(R.id.container, new PlaceholderFragment()).commit();

        }

    }

    @Override

    public boolean onOptionsItemSelected(MenuItem item) {

        // Handle action bar item clicks here. The action bar will

        // automatically handle clicks on the Home/Up button, so long

        // as you specify a parent activity in AndroidManifest.xml.

        int id = item.getItemId();

        if (id == R.id.action\_settings) {

            return true;

        }

        return super.onOptionsItemSelected(item);

    }

    /\*\*

     \* A placeholder fragment containing a simple view.

     \*/

    public static class PlaceholderFragment extends Fragment {

        public PlaceholderFragment() { }

        @Override

        public View onCreateView(LayoutInflater inflater, ViewGroup container,

                  Bundle savedInstanceState) {

              View rootView = inflater.inflate(R.layout.fragment\_display\_message,

                      container, false);

              return rootView;

        }

    }

}

Все подклассы Activity должны реализовать onCreate() метод. Система вызывает его при создании нового экземпляра активити. В этом методе должны определить макет активити с помощью [setContentView()](http://developer.android.com/reference/android/app/Activity.html" \l "setContentView(android.view.View)" \t "_blank) метода, и здесь должны выполнить начальную настройку компонентов активити.

**Получение Интента.** Каждый [Activity](http://developer.android.com/reference/android/app/Activity.html) вызывается с помощью механизма [Intent](http://developer.android.com/reference/android/content/Intent.html), независимо от того, как пользователь между ними переходит. Вы можете получить [Intent](http://developer.android.com/reference/android/content/Intent.html) , который запустил вашу активити, вызвав [getIntent()](http://developer.android.com/reference/android/app/Activity.html" \l "getIntent()) и получить данные, содержащиеся в нем.

В DisplayMessageActivity класса [onCreate()](http://developer.android.com/reference/android/app/Activity.html" \l "onCreate(android.os.Bundle)) метод, получает интент и извлекает сообщение доставленное MainActivity:

Intent intent = getIntent();

String message = intent.getStringExtra(MainActivity.EXTRA\_MESSAGE);

**Отображение сообщения**

Для отображения сообщения на экране, создайте [TextView](http://developer.android.com/reference/android/widget/TextView.html) виджет и установите текст с помощью [setText()](http://developer.android.com/reference/android/widget/TextView.html#setText(char[],%20int,%20int)). Затем добавьте [TextView](http://developer.android.com/reference/android/widget/TextView.html) в качестве корневого представления макета активити, передав его в [setContentView()](http://developer.android.com/reference/android/app/Activity.html#setContentView(android.view.View)).

Полный [onCreate()](http://developer.android.com/reference/android/app/Activity.html" \l "onCreate(android.os.Bundle)) метод для DisplayMessageActivity теперь выглядит следующим образом:

@Override

public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    // Get the message from the intent

    Intent intent = getIntent();

    String message = intent.getStringExtra(MainActivity.EXTRA\_MESSAGE);

    // Create the text view

    TextView textView = new TextView(this);

    textView.setTextSize(40);

    textView.setText(message);

    // Set the text view as the activity layout

    setContentView(textView);

}

Теперь можно запустить приложение. Когда оно откроется, ввести сообщение в текстовое поле, нажмите «Отправить», и сообщение появится во второй активити.

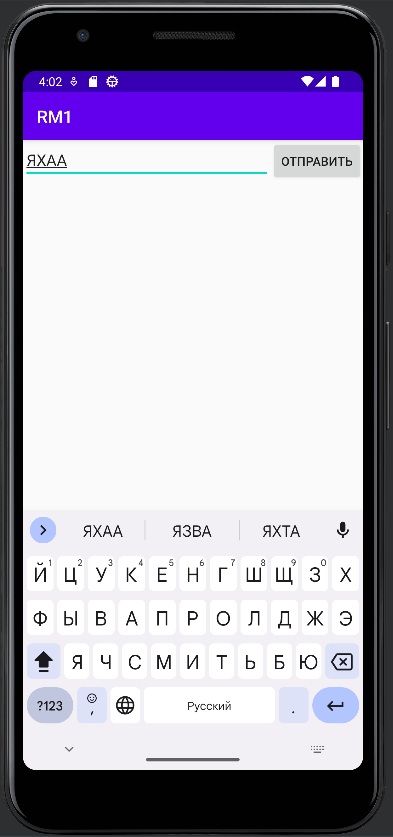


Рисунок 14 – успешная работа первого активити



Рисунок 15 – успешная работа второго активити

# ВЫВОД

Был создан простой проект в Android Studio. Произведён запуск приложения. Создан простой интерфейс пользователя. Создана линейная разметка (Linear Layout). Добавлены текстовые поля. Создан строковый ресурс. Добавлены кнопки. Растянуто поле ввода. Запущено другое явление. Реализована обработка нажатия на кнопку. Создано намерения (Intent). Создано второе явление. Создано новое явление в Android Studio. Создано явления без Android Studio. Обеспечено получение намерения. Обеспечено отображение сообщения.