

Высший колледж информатики НГУ

А.А. Жирнов

**Методическое пособие по курсу
«Программирование мобильных приложений» в
IDE Android Studio**

НОВОСИБИРСК

2022

Аннотация

Методическое пособие предназначено для студентов специальности 09.02.03 Программирование в компьютерных системах по дисциплине *«Программирование мобильных приложений»* и специальности 09.02.07 Информационные системы и программирование по дисциплине *«Разработка мобильных приложений»*.

В нём содержится краткий теоретический материал и закрепляющий его практический материал. Пособие ориентировано на курс по разработке мобильных приложений в IDE Android Studio, где каждый раздел / подраздел соответствует лабораторной работе и в совокупности соответствует 32-м академически часам.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
ГЛАВА 1 РАБОТА В IDE ANDRIOD STUDIO	5
1.1 Настройка Android SDK. Android Studio и создание первого проекта	5
1.2 Основные элементы управления	10
1.3 Создание многостраничного приложения	13
1.4 Диалоговые окна	15
1.5 Создание дизайна	22
1.6 Работа с мультимедиа	27
1.6.1 Работа с анимацией	27
1.6.2 Работа со звуком	29
1.6.3 Работа с видео	31
1.7 Настройки и состояния приложения	34
1.8 Работа с файловой системой	36
1.9 Работа с базой данных SQLite	43
1.10 Телефония	56
1.10.1 Вызов абонента	57
1.10.2 Отправка СМС-сообщений	60
1.11 Публикация мобильного приложения	62
ГЛАВА 2 ОБЩИЕ ПОДХОДЫ К РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ	72
2.1 Разработка технической документации на программное обеспечение ..	72
2.2 Анализ требований к мобильному ПО	74
ЗАКЛЮЧЕНИЕ	79
Список используемой литературы	80

ВВЕДЕНИЕ

В настоящее время в связи с широким распространением мобильных приложений всё более востребованными становятся средства их разработки. Мобильные приложения охватывают огромный круг решаемых задач в современном обществе: игры; бизнес; образование; утилиты; путешествия; еда и напитки; здоровье и фитнес; общение; новости; книги; магазины и т.д. Смартфоны позволяют решать задачи, которые были невозможны при использовании обычных стационарных компьютеров или ноутбуков.

Сейчас существует ряд мобильных операционных систем, наиболее популярными из них являются Android от компании Google и iOS от компании Apple.

Мобильными приложениями пользуются люди всех возрастов, однако наибольший пик по численности пользователей прослеживается от 15 до 30 лет [1], поэтому основная масса разработчиков ориентируется на эту возрастную группу.

Среди использования всех типов приложений лидируют бесплатные: так, для операционной системы Android в магазине Google Play Store на март 2022 года их доля составляет в среднем 96,3% [2]. Диапазон цен на приложение в России, исходя из политики компании Google, может варьироваться от 4,5 до 29000 рублей [3]. Число загруженных мобильных приложений с каждым годом только растёт.

Распространение мобильных приложений, как правило, осуществляется через Интернет-магазины, наиболее популярными являются: AppStore; Google Play; Amazon Appstore; Huawei AppGallery; Samsung Galaxy Store.

Основной доход (монетизация) разработчиков мобильных приложений распределяется следующим образом: реклама – 65%; покупки в приложениях – 50; виртуальная валюта – 25%; платные приложения – 15%; подписки в приложениях – 5%.

Исходя из вышесказанного, можно сделать вывод, что разработка мобильных приложений является перспективным и востребованным направлением в современной индустрии программного обеспечения, где приложения вступают в большую конкуренцию из-за постоянного пополнения этого рынка.

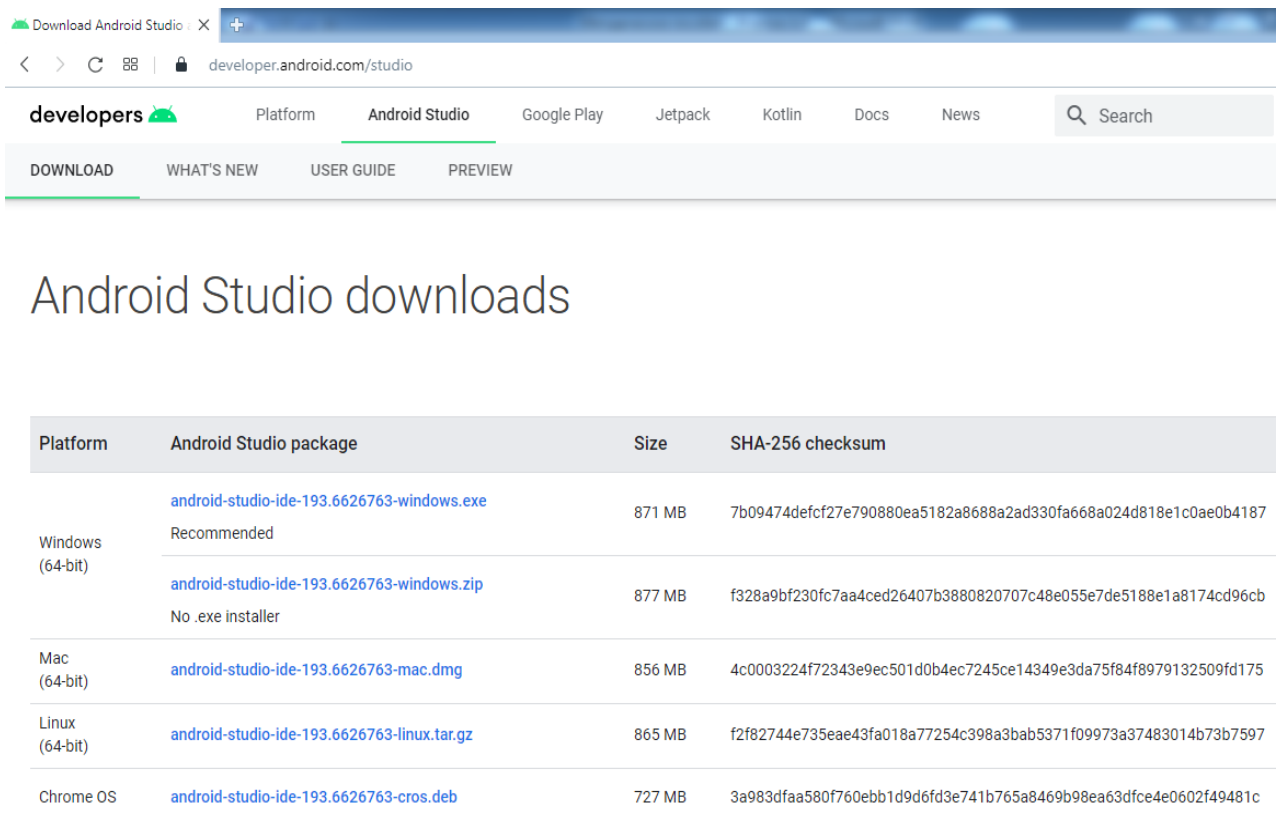
Для разработки мобильных приложений будем использовать программную среду IDE Android Studio, так как она является бесплатной и количество мобильных устройств на операционной системе Android занимает лидирующую позицию.

ГЛАВА 1 РАБОТА В IDE ANDRIOD STUDIO

В данной главе описаны некоторые возможности среды разработки Android Studio, которая позволяет создавать мобильные приложения для операционной системы Android. Сюда входит установка и настройка среды разработки, знакомство с основными элементами и компонентами, работа с диалоговыми окнами и создание дизайна, работа с мультимедиа и базой данных, функциями телефонии, и публикация приложения в Интернет-магазине.

1.1 Настройка Android SDK. Android Studio и создание первого проекта

Работа в IDE Android Studio предполагает предварительное скачивание установочного файла с официального сайта <https://developer.android.com/studio?hl=ru#downloads>, рисунок 1. Здесь можно выбрать варианты для разработки под операционными системами: Windows; Mac; Linux; Chrome.



Platform	Android Studio package	Size	SHA-256 checksum
Windows (64-bit)	android-studio-ide-193.6626763-windows.exe Recommended	871 MB	7b09474defcf27e790880ea5182a8688a2ad330fa668a024d818e1c0ae0b4187
	android-studio-ide-193.6626763-windows.zip No .exe installer	877 MB	f328a9bf230fc7aa4ced26407b3880820707c48e055e7de5188e1a8174cd96cb
Mac (64-bit)	android-studio-ide-193.6626763-mac.dmg	856 MB	4c0003224f72343e9ec501d0b4ec7245ce14349e3da75f84f8979132509fd175
Linux (64-bit)	android-studio-ide-193.6626763-linux.tar.gz	865 MB	f2f82744e735eae43fa018a77254c398a3bab5371f09973a37483014b73b7597
Chrome OS	android-studio-ide-193.6626763-cros.deb	727 MB	3a983dfaa580f760ebb1d9d6fd3e741b765a8469b98ea63dfce4e0602f49481c

See the [Android Studio release notes](#). More downloads are available in the [download archives](#).

Рисунок 1 – Ссылки для установки IDE Android Studio

Установка среды разработки осуществляется с помощью мастера, где предлагаются варианты установки и настройки, рисунок 2.

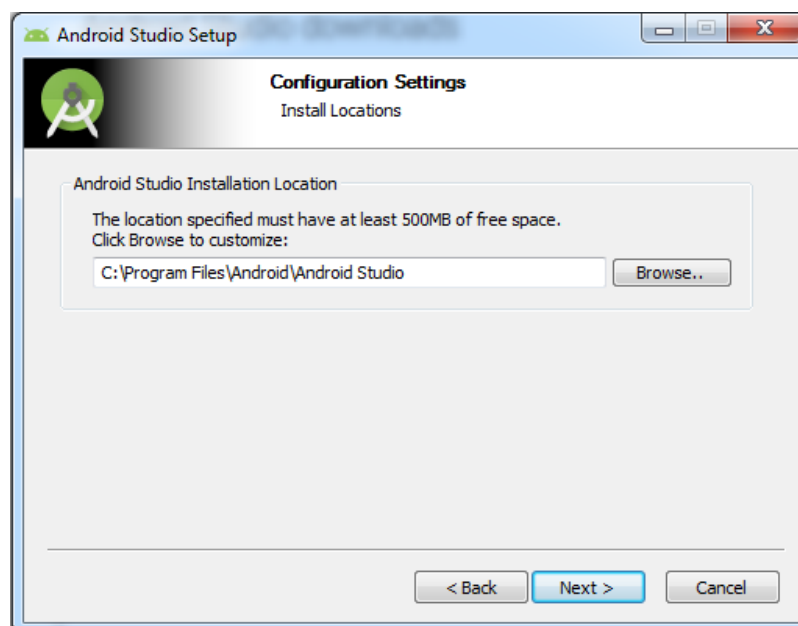


Рисунок 2 – Работа с мастером установки

После установки IDE, при первом запуске необходимо установить и настроить Android SDK [4], рисунок 3.

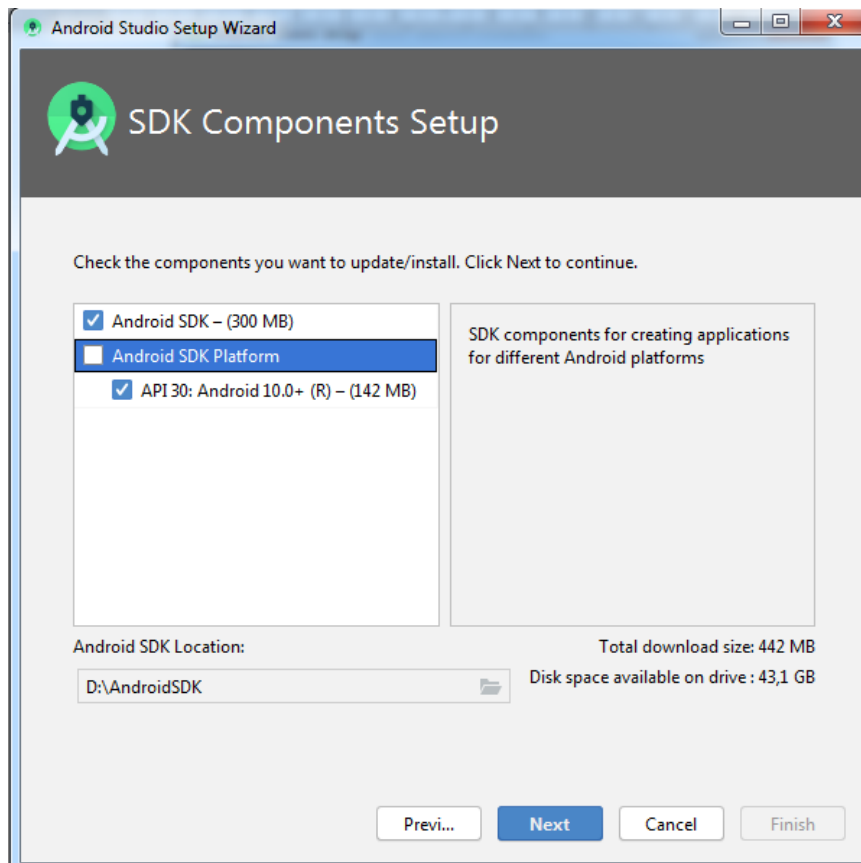


Рисунок 3 – Установка Android SDK

Android SDK содержит компоненты для различных платформ. После установки всех компонентов можно переходить к началу работы в среде Android Studio. После запуска появляется стартовый экран, рисунок 4.

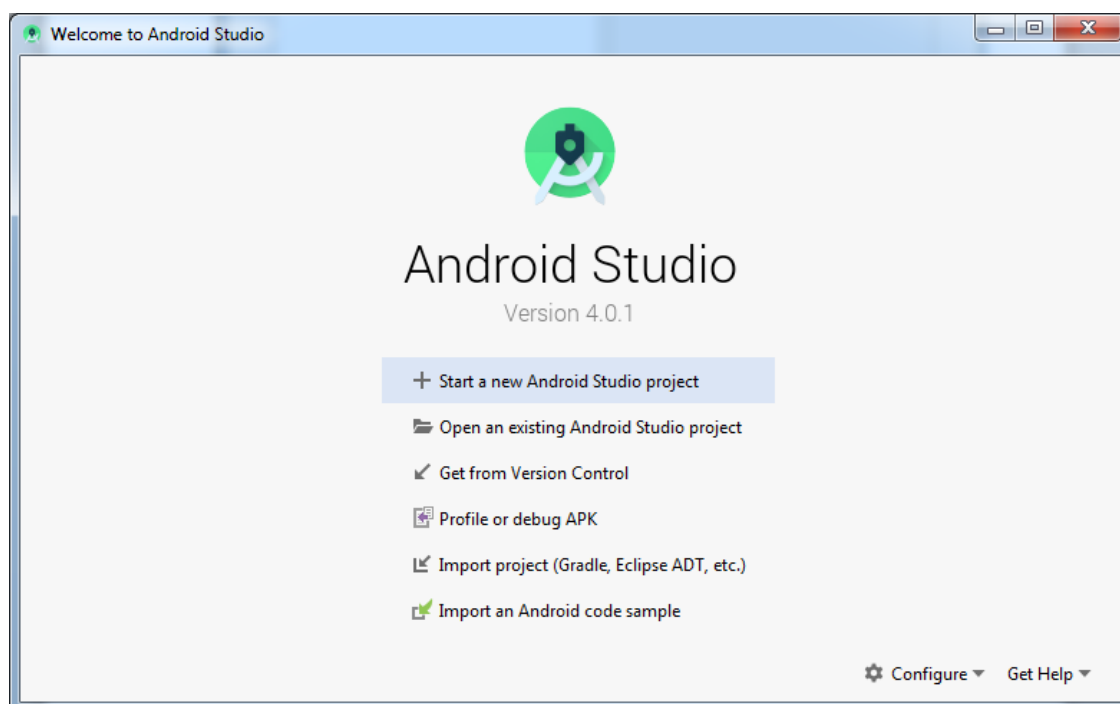


Рисунок 4 – Стартовое окно Android Studio

Для создания своего первого приложения необходимо нажать кнопку «*Start a new Android Studio project*» после чего выбрать варианты шаблонов. В данном случае мы выбираем «*Empty Activity*» (рисунок 5) и задаём имя проекта, рисунок 6.

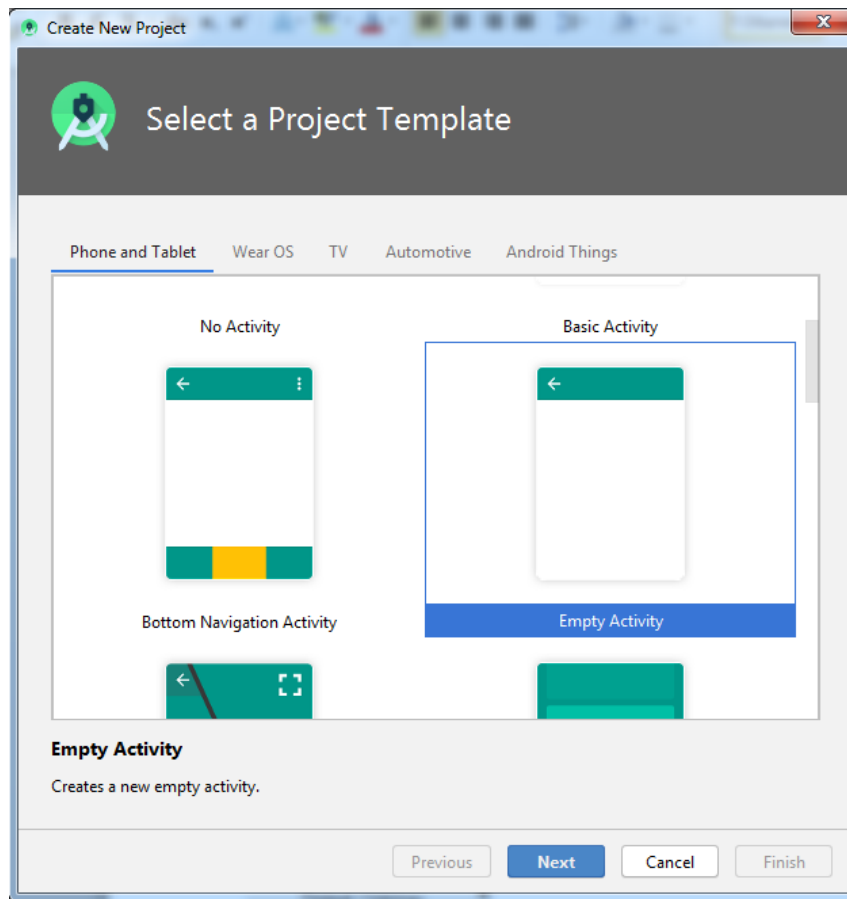


Рисунок 5 – Диалоговое окно выбора готовых шаблонов

В диалоговом окне «*Configure Your Project*» выполняется настройка вашего проекта: задаётся имя проекта; имя пакета; путь сохранения; язык программирования – Java/Kotlin; минимальная версия SDK, которая определяет поддерживаемую версию операционной системы Вашего приложения и примерное количество устройств, в процентах (%), которое её поддерживает.

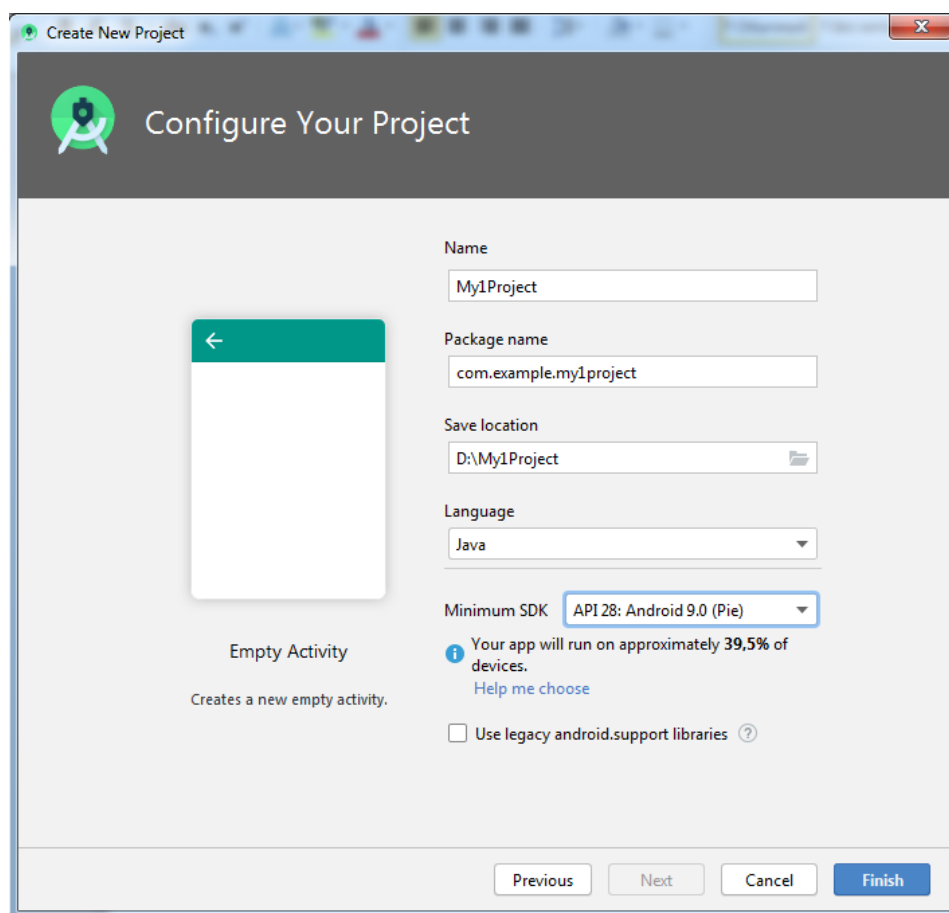


Рисунок 6 – Диалоговое окно конфигурации проекта

После конфигурации проекта открывается окно самой среды разработки. Если предполагается работа на реальном смартфоне, а не на эмуляторе, то необходимо установить драйвер через меню *Tools*→*SDK Manager*→*SDK Tools*→*Google USB Driver*. В результате этого в Диспетчере устройств Windows должны появиться устройства: *Android Device*→*Android Composite ADB Interface*, а также в разделе «Переносные устройства» должно отобразиться используемое Вами устройство.

При работе в режиме эмулятора необходимо его настроить соответствующим образом в меню *Tools*→*ADV Manager*.

Для создания первого проекта нужно в компоненте *TextView* заполнить свойство *text*, *textSize*, *textColor* и запустить приложение, как показано на рисунке 7.

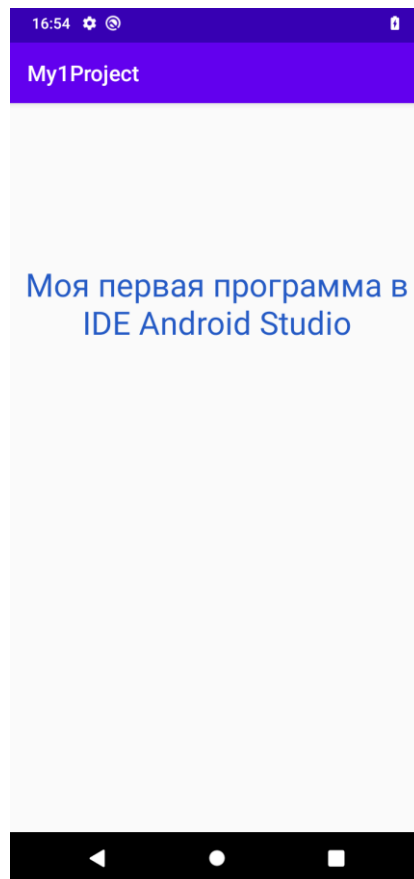


Рисунок 7 – Первая программа в IDE Android Studio

1.2 Основные элементы управления

Данная среда разработки достаточно стандартизована по интерфейсу. Здесь имеется: главное меню, окно структуры проекта (рисунок 8), окно предпросмотра, палитра компонентов (рисунок 9), панель атрибутов компонентов (рисунок 10); имеется панель быстрого доступа, окно хода компиляции и сборки с выводом логов событий.

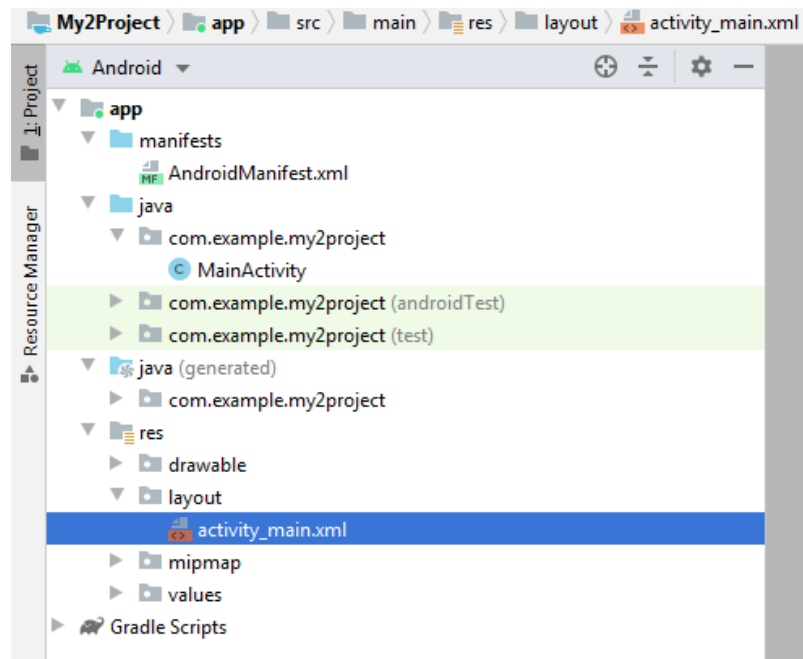


Рисунок 8 – Окно структуры проекта

Окно структуры проекта имеет древовидную структуру, где в папке `manifest` содержится файл *AndroidManifest.xml*, в котором сохранены настройка проекта. Папка `java` содержит файлы исходного программного кода, в данном случае это файл, созданный по умолчанию – *MainActivity.java*. В папке `res` содержатся файлы ресурсов: *drawable* – изображения; *layout* – слои; *mipmap* – множественное отображение; *values* – отдельные значения и переменные. В папке `res` можно создавать свои папки для хранения необходимых файлов.

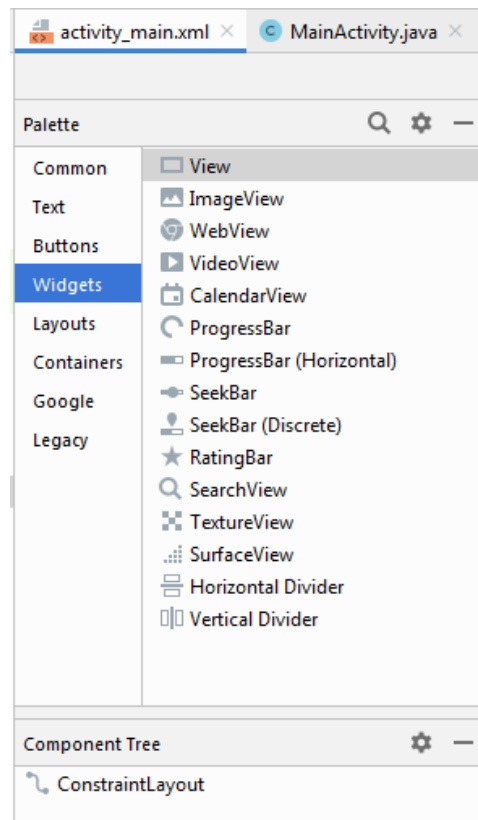


Рисунок 9 – Окно доступных компонентов

Палитра компонентов представлена вкладками: *Common*/Основные, *Text*/Текстовые, *Buttons*/Кнопки, *Layouts*/Слои, *Containers*/Контейнеры, *Google*/Гугл, *Legacy*/Наследники. Каждая вкладка имеет перечень доступных компонентов, который можно расширять. Наносить компоненты на форму можно только при просмотре слоёв, путём переключения соответствующей пиктограммы – *Code* (Просмотр кода)/*Split* (Совмещённый режим: код и форма)/*Design* (Форма).

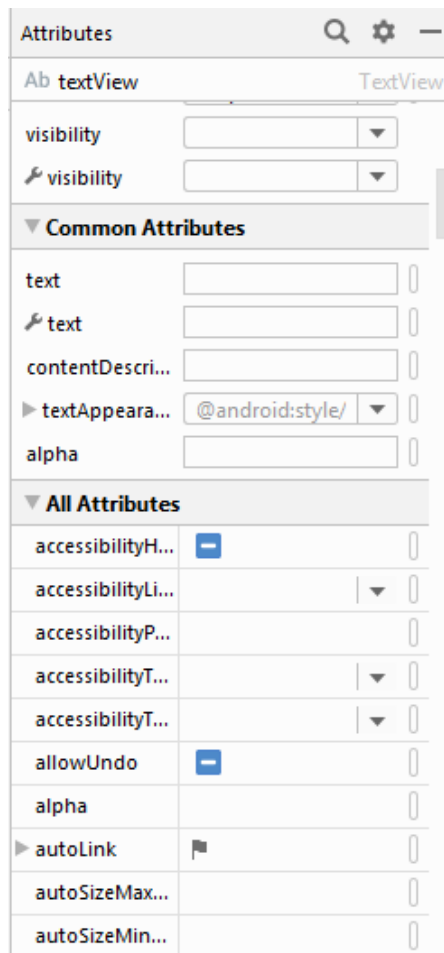


Рисунок 10 – Окно атрибутов компонента *textView*

У каждого компонента есть набор атрибутов (свойств), который доступен в окне *Attributes*. Данный перечень достаточно широкий, однако следует отметить свойство «*onClick*», где можно написать имя обработчика события для этого компонента.

1.3 Создание многостраничного приложения

В данном разделе и далее в главе 1 рассмотрим практическое применение IDE Android Studio.

Часто в современных мобильных приложениях используются перелистывающие экраны, в результате чего создаётся эффект многостраничного приложения. Это достигается различными способами, мы рассмотрим самый простой из них – реализация двух *Activity*.

Для этого в наш проект «*MyApplication*» добавим новую *Activity* с именем «*MainActivity2*» путём выполнения команд: *New*→*Activity*→*Empty*. После этого в файле *AndroidManifest.xml* у нас появится запись, где можно

настроить, какую из *Activity* запускать первой, а также некоторые другие настройки.

В файлах *activity_main.xml* и *activity_main2.xml* необходимо нанести 2 компонента: *textView* и *Button*, после чего настроить их необходимым для себя образом. *Activity_main.xml*:

```
<TextView
    android:id="@+id/textView2"
    android:layout_width="359dp"
    android:layout_height="116dp"
    android:text="Мы находимся на 1-ой Activity"
    android:textAlignment="center"
    android:textColor="#8BC34A"
    android:textSize="36sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.108" />

<Button
    android:id="@+id/button"
    android:layout_width="330dp"
    android:layout_height="194dp"
    android:onClick="onClick"
    android:text="Переход ко 2-й Activity"
    android:textSize="24sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.493"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.735" />
```

Activity_main2.xml:

```
<TextView
    android:id="@+id/textView"
    android:layout_width="359dp"
    android:layout_height="116dp"
    android:text="Мы перешли на 2-ю Activity"
    android:textAlignment="center"
    android:textColor="#8BC34A"
    android:textSize="36sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.211" />

<Button
    android:id="@+id/button2"
    android:layout_width="253dp"
    android:layout_height="158dp"
    android:onClick="onClickButton2"
    android:text="На первую Activity"
    android:textSize="24sp"
    app:layout_constraintBottom_toBottomOf="parent"
```

```

app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/textView"
app:layout_constraintVertical_bias="0.196" />

```

Создадим обработчики событий «onClick» и «onClickButton2» для этих кнопок в файлах *MainActivity.java* и *MainActivity2.java*, которые и будут вызывать сами *Activity*:

```

public void onClick(View view)
{
    Intent intent = new Intent(this, MainActivity2.class);
    startActivity(intent);
}

public void onClickButton2(View view)
{
    Intent intent = new Intent(this, MainActivity.class);
    startActivity(intent);
}

```

После выполненных действий у нас появятся экраны, представленные на рисунках 11, 12.

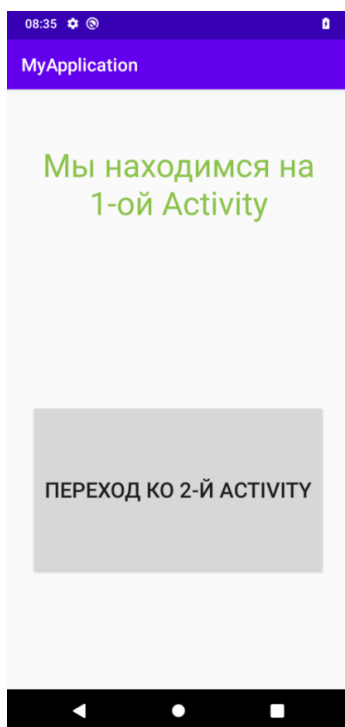


Рисунок 11 – Работа первой *Activity*

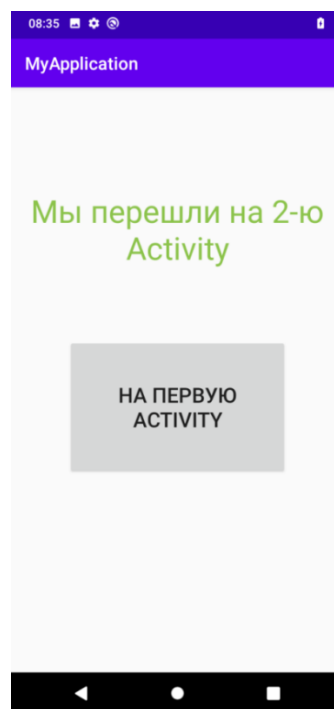


Рисунок 12 – Работа второй *Activity*

1.4 Диалоговые окна

Диалоговые окна являются частыми элементами при работе с графическим интерфейсом приложения. С помощью них можно выводить

ошибки, информацию, предупреждения, а при работе с модальными окнами блокировать доступ к другим элементам приложения, пока пользователь не сделает определённый выбор.

Разработаем дизайн простого мобильного приложения, которое будет выводить окно оценки рейтинга и окно предупреждения. Для этого поместим на форму два компонента *Button* и добавим к ним обработчики «*onShowDialog*» и «*onAlertButton*», рисунок 13.

```
<?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">

    <Button
        android:id="@+id/buttonModal"
        android:layout_width="285dp"
        android:layout_height="89dp"
        android:layout_marginTop="72dp"
        android:background="#67EA5D"
        android:onClick="onShowDialog"
        android:text="Показать диалоговое окно"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.571"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/buttonAlert"
        android:layout_width="286dp"
        android:layout_height="91dp"
        android:layout_marginTop="84dp"
        android:background="#EDD456"
        android:onClick="onAlertButton"
        android:text="Показать окно предупреждения"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.564"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/buttonModal" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

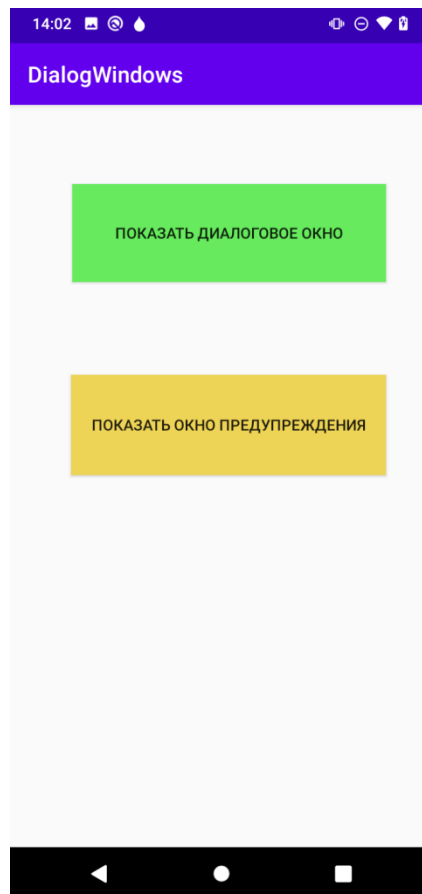
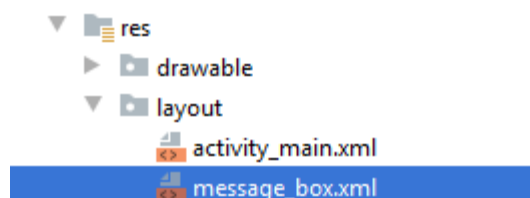



Рисунок 13 – Внешний вид главного окна разрабатываемого приложения

Для реализации кнопки «Показать диалоговое окно» создадим новый слой и назовём его «*message_box*»:



Добавим к нему небольшой интерфейс, компоненты *TextView*, *Button* и *RatingBar*, рисунок 14.

```
<?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">
    <RatingBar
        android:id="@+id/My_rating"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="48dp"
        android:background="#F499A1"
        android:indeterminate="false"
```

```

        android:isIndicator="false"
        android:numStars="5"
        android:stepSize="1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView1" />
<Button
    android:id="@+id/buttonOk"
    android:layout_width="239dp"
    android:layout_height="53dp"
    android:layout_marginTop="48dp"
    android:background="#EFDCE"
    android:text="Заккрыть окно"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/My_rating" />
<TextView
    android:id="@+id/textView1"
    android:layout_width="338dp"
    android:layout_height="44dp"
    android:layout_marginTop="36dp"
    android:background="#7BE8DE"
    android:textAlignment="center"
    android:textSize="24sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>

```

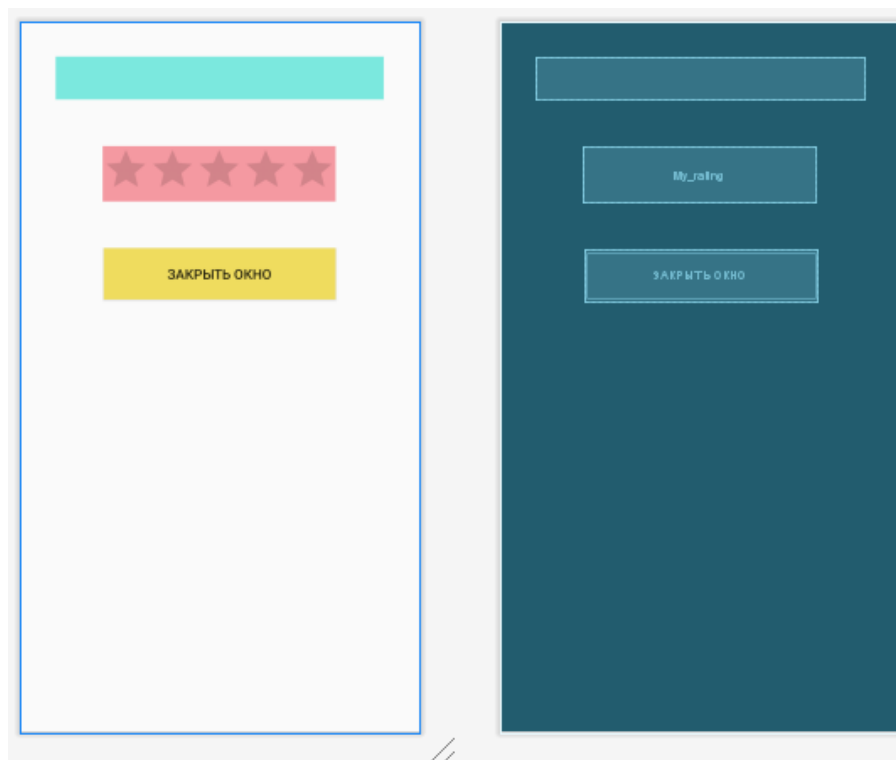


Рисунок 14 – Внешний вид диалогового диалогового окна

Для запуска разработанного окна сообщения из главного приложения добавим код, который включает в себя три функции: «*MessageBox*» – установка параметров; *onClick* – закрытие диалогового окна; и «*onShowDialog*» для вызова окна:

```
public class MainActivity extends AppCompatActivity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    protected void MessageBox (String message)
    {
        final Dialog dialog = new Dialog(this);
        dialog setContentView(R.layout.message_box);
        RatingBar ratingBar = (RatingBar)dialog.findViewById(R.id.My_rating);
        TextView textView = (TextView)dialog.findViewById(R.id.textView1);
        ratingBar.setRating((float)3.0);
        textView.setText(message);
        dialog.show();
        Button btn = (Button)dialog.findViewById(R.id.buttonOk);
        btn.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                dialog.dismiss();
            }
        });
    }

    public void onShowDialog(View view)
    {
        MessageBox("Диалоговое окно");
    }
}
```

Запустим разработанное приложение и нажмём на кнопку «*Показать диалоговое окно*», рисунок 15:

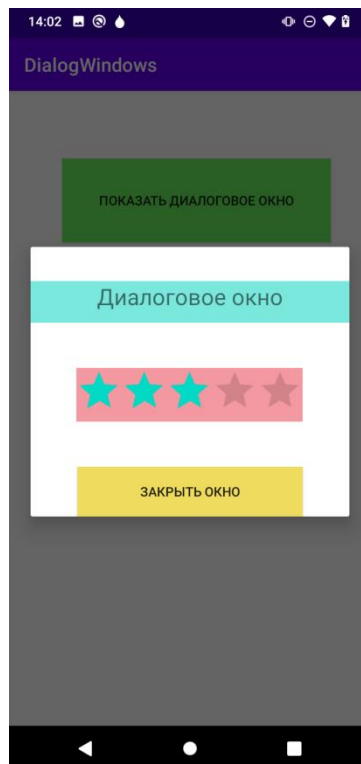


Рисунок 15 – Демонстрация работы кнопки «Показать диалоговое окно»

Реализуем окно предупреждения; для этого добавим новый класс «CustomDialogFragment»:

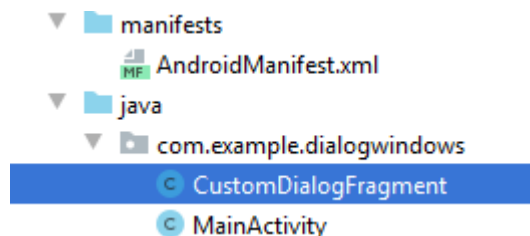


Рисунок 16 – Добавление нового класса в проект

Расширим этот класс и запишем в нём:

```
public class CustomDialogFragment extends DialogFragment
{
    @NonNull
    public Dialog onCreateDialog(Bundle savedInstanceState)
    {
        AlertDialog.Builder builder=new AlertDialog.Builder(getActivity());
        return builder
            .setTitle("Диалоговое окно на удаление")
            .setIcon(android.R.drawable.ic_delete)
            .setMessage("Для закрытия окна нажмите кнопку Готово/Отмена")
            .setPositiveButton("Готово", null)
            .setNegativeButton("Отмена", null)
            .create();
    }
}
```

В данном классе можно определить множество параметров диалогового окна, мы будем использовать лишь некоторые: «*setTitle*», «*setIcon*», «*setMessage*», «*setPositiveButton*», «*setNegativeButton*».

Добавим код запуска в главной Activity:

```
public class MainActivity extends AppCompatActivity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void onAlertButton(View view)
    {
        CustomDialogFragment dialog = new CustomDialogFragment();
        dialog.show(getSupportFragmentManager(), null);
    }
}
```

В результате получим диалоговое окно, запускаемое при щелчке по кнопке «Показать окно предупреждения», рисунок 17. В функции «*onAlertButton*» параметр *null* предназначен для обработки кнопок Отмена/Готово.

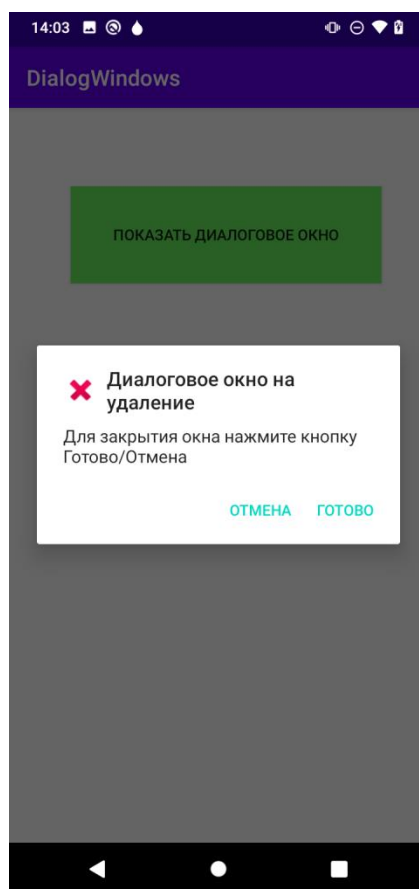


Рисунок 17 – Диалоговое окно предупреждения на удаление

1.5 Создание дизайна

Дизайн, как и функционал приложения, является весомым аргументом при выборе приложения пользователем. Интуитивно понятный интерфейс приложения, а также гармоничное сочетание цветов привлекательно для пользователя.

При разработке дизайна в Android Studio удобно использовать слои, которые находятся на вкладках *Layouts* и *Legacy* [5]. Они являются контейнерами для других элементов и позволяют упростить расположение (выравнивание) остальных элементов на экране смартфона, а также фиксировать их при изменении ориентации. Для упрощения исходного кода разработанные мобильные программы в данном пособии используются без слоёв.

Для настройки слоёв и остальных компонентов используется окно атрибутов (свойств). Корневым слоем всегда является *ConstraintLayout*.

Единицы измерения *dp* являются наиболее предпочтительными, потому что точнее отражают пропорции на различных устройствах независимо от их разрешения.

Разработаем приложение, имитирующее работу калькулятора (без описания обработчиков событий). С помощью него можно наглядно продемонстрировать удобство работы со слоями.

Установим первый слой *RelativeLayout* на который поместим компонент *textView*. Установим следующие настройки:

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="4dp"
    app:layout_constraintTop_toTopOf="parent"
    tools:layout_editor_absoluteX="-6dp">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#FFC107"
        android:text="Разработка и создание дизайна"
        android:textAlignment="center"
        android:textColor="#0A2CE6"
        android:textSize="24sp"
        android:textStyle="italic" />
```

Поместим другой слой – *GridLayout*, в котором установим количество столбцов и строк в сетке, после чего поместим компонент *Button* в каждую ячейку:

```

<androidx.gridlayout.widget.GridLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="20dp"
    android:layout_marginEnd="15dp"
    android:layout_marginBottom="-1dp"
    app:columnCount="4"
    app:rowCount="5">

    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="AC"
        android:textAlignment="center"
        android:textSize="24sp"
        app:layout_column="0"
        app:layout_row="0" />

    <Button
        android:id="@+id/button5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="+/-"
        android:textAlignment="center"
        android:textSize="24sp"
        app:layout_column="1"
        app:layout_row="0" />

    <Button
        android:id="@+id/button6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="%"
        android:textAlignment="center"
        android:textSize="24sp"
        app:layout_column="2"
        app:layout_row="0" />

    <Button
        android:id="@+id/button7"
        android:layout_width="116dp"
        android:layout_height="wrap_content"
        android:text="/"
        android:textAlignment="center"
        android:textSize="24sp"
        app:layout_column="3"
        app:layout_row="0" />

    <Button
        android:id="@+id/button8"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="7"
        android:textAlignment="center"
        android:textSize="24sp"
        app:layout_column="0"
        app:layout_row="1" />

```

```

<Button
    android:id="@+id/button9"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="8"
    android:textAlignment="center"
    android:textSize="24sp"
    app:layout_column="1"
    app:layout_row="1" />

<Button
    android:id="@+id/button10"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="9"
    android:textAlignment="center"
    android:textSize="24sp"
    app:layout_column="2"
    app:layout_row="1" />

<Button
    android:id="@+id/button11"
    android:layout_width="113dp"
    android:layout_height="wrap_content"
    android:text="x"
    android:textAlignment="center"
    android:textSize="24sp"
    app:layout_column="3"
    app:layout_row="1" />

<Button
    android:id="@+id/button12"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="4"
    android:textAlignment="center"
    android:textSize="24sp"
    app:layout_column="0"
    app:layout_row="2" />

<Button
    android:id="@+id/button13"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="5"
    android:textAlignment="center"
    android:textSize="24sp"
    app:layout_column="1"
    app:layout_row="2" />

<Button
    android:id="@+id/button14"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="6"
    android:textAlignment="center"
    android:textSize="24sp"
    app:layout_column="2"
    app:layout_row="2" />

```



```

<Button
    android:id="@+id/button15"
    android:layout_width="115dp"
    android:layout_height="wrap_content"
    android:text="-"
    android:textAlignment="center"
    android:textSize="24sp"
    app:layout_column="3"
    app:layout_row="2" />

<Button
    android:id="@+id/button16"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="1"
    android:textAlignment="center"
    android:textSize="24sp"
    app:layout_column="0"
    app:layout_row="3" />

<Button
    android:id="@+id/button17"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="2"
    android:textAlignment="center"
    android:textSize="24sp"
    app:layout_column="1"
    app:layout_row="3" />

<Button
    android:id="@+id/button18"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="3"
    android:textAlignment="center"
    android:textSize="24sp"
    app:layout_column="2"
    app:layout_row="3" />

<Button
    android:id="@+id/button19"
    android:layout_width="114dp"
    android:layout_height="wrap_content"
    android:text="+"
    android:textAlignment="center"
    android:textSize="24sp"
    app:layout_column="3"
    app:layout_row="3" />

<Button
    android:id="@+id/button20"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="0"
    android:textAlignment="center"
    android:textSize="24sp"
    app:layout_column="0"
    app:layout_row="4" />

<Button

```

```

        android:id="@+id/button22"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=","
        android:textAlignment="center"
        android:textSize="24sp"
        app:layout_column="2"
        app:layout_row="4" />

<Button
    android:id="@+id/button23"
    android:layout_width="114dp"
    android:layout_height="wrap_content"
    android:text="="
    android:textAlignment="center"
    android:textSize="24sp"
    app:layout_column="3"
    app:layout_row="4" />

<Button
    android:id="@+id/button25"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="00"
    android:textAlignment="center"
    android:textSize="24sp"
    app:layout_column="1"
    app:layout_row="4" />

</androidx.gridlayout.widget.GridLayout>

```

В результате получим мобильное приложение, представленное на рисунках 18 и 19.



Рисунок 18 – Расположение элементов при вертикальной ориентации смартфона

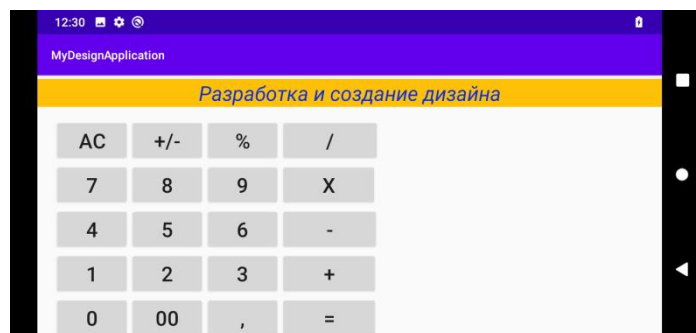


Рисунок 19– Расположение элементов при горизонтальной ориентации смартфона

1.6 Работа с мультимедиа

В настоящее время понятие мультимедиа включает в себя не только работу с аудио и видеофайлами, а также изображения, анимацию. Большое количество современных мобильных приложений позволяют выполнять работу с медиафайлами: редакторы; корректоры. В данном разделе будем разрабатывать простейшие приложения для демонстрации такого функционала.

1.6.1 Работа с анимацией

Существует несколько вариантов анимации. *Cell animation* представляет собой технику анимации, при которой ряд изображений или кадров последовательно сменяют друг друга за короткий промежуток времени. Такая техника довольно широко распространена при создании мультфильмов. Разработаем простое приложение для реализации такой анимации. Например, имеется следующий набор изображений, рисунок 20. Назовём их соответственно *rabbit1.jpg* - *rabbit5.jpg*.

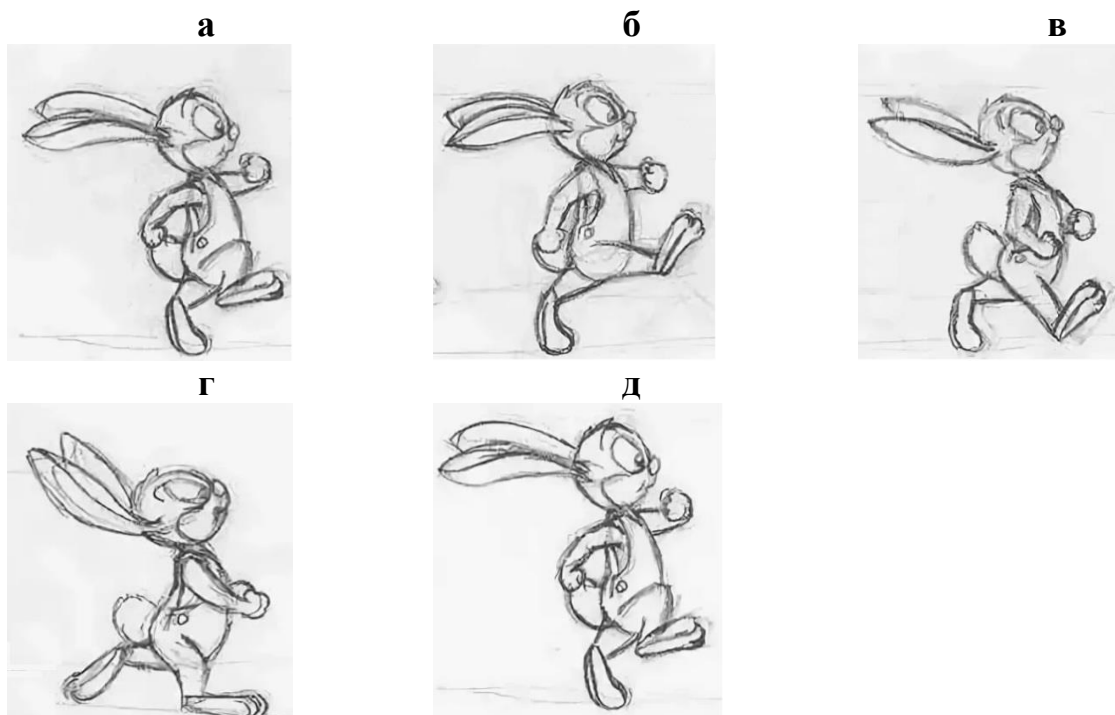


Рисунок 20 а, б, в, г, д – Образцы изображений для создания анимации

Для того чтобы эти изображения «ожили», необходимо выполнить три шага:

1. Надо добавить все эти изображения в проект в папку *res/drawable*. В эту же папку добавим новый xml-файл и назовем его *rabbit_animation.xml*, куда поместим следующее содержимое:

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false" >
    <item android:drawable="@drawable/rabbit1" android:duration="100" />
    <item android:drawable="@drawable/rabbit2" android:duration="100" />
    <item android:drawable="@drawable/rabbit3" android:duration="100" />
    <item android:drawable="@drawable/rabbit4" android:duration="100" />
    <item android:drawable="@drawable/rabbit5" android:duration="100" />
</animation-list>
```

2. Добавить компонент *ImageView* и разметить его соответствующим образом:

```
<ImageView
    android:id="@+id/rabbitFoto"
    android:layout_width="354dp"
    android:layout_height="315dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.492"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.498" />
```

3. В функции «*onCreate*» необходимо добавить код запуска:

```

super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
ImageView img = (ImageView)findViewById(R.id.rabbitFoto);
img.setBackgroundResource(R.drawable.rabbit_animation);
AnimationDrawable frameAnimation = (AnimationDrawable) img.getBackground();
frameAnimation.setOneShot(false);
frameAnimation.start();

```

Этих действий достаточно для простейшего запуска анимации.

1.6.2 Работа со звуком

Рассмотрим работу со звуком на примере простейшего аудиоплеера, который воспроизводит добавленный нами файл формата *.mp3. Добавим файл *music.mp3* в папку *res*→*raw*. Добавим кнопки управления на форму и подпись *textView*, рисунок 21.

```

<TextView
    android:id="@+id/textView"
    android:layout_width="403dp"
    android:layout_height="69dp"
    android:background="#FFEB3B"
    android:text="Аудио проигрыватель"
    android:textAlignment="center"
    android:textColor="#0779D3"
    android:textSize="30sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    tools:layout_editor_absoluteY="3dp" />
<Button
    android:id="@+id/buttonStart"
    android:layout_width="203dp"
    android:layout_height="86dp"
    android:background="#00BCD4"
    android:onClick="onStartClick"
    android:text="Старт"
    android:textSize="30sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.576"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView" />
<Button
    android:id="@+id/buttonPause"
    android:layout_width="187dp"
    android:layout_height="88dp"
    android:background="#CDDC39"
    android:onClick="onPauseClick"
    android:text="Пауза"
    android:textSize="30sp"
    app:layout_constraintBottom_toTopOf="@+id/buttonStop"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.535"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/buttonStart" />
<Button

```

```

android:id="@+id/buttonStop"
android:layout_width="182dp"
android:layout_height="86dp"
android:background="#FFC107"
android:onClick="onStopClick"
android:text="Стоп"
android:textSize="30sp"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.545"
app:layout_constraintStart_toStartOf="parent" />

```

Для кнопок создадим соответствующие обработчики событий «*onStartClick*», «*onPauseClick*», «*onStopClick*» и напомним в них код, а также в самом классе добавим функции:

```

public class MainActivity extends AppCompatActivity implements
MediaPlayer.OnPreparedListener {
    MediaPlayer mediaPlayer;
    Button startButton, pauseButton, stopButton;
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mediaPlayer=MediaPlayer.create(this, R.raw.music);
        mediaPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
            @Override
            public void onCompletion(MediaPlayer mp) {
                stopPlay();
            }
        });
        startButton = (Button) findViewById(R.id.buttonStart);
        pauseButton = (Button) findViewById(R.id.buttonPause);
        stopButton = (Button) findViewById(R.id.buttonStop);
        pauseButton.setEnabled(false);
        stopButton.setEnabled(false);
    }
    private void stopPlay(){
        mediaPlayer.stop();
        pauseButton.setEnabled(false);
        stopButton.setEnabled(false);
        try {
            mediaPlayer.prepare();
            mediaPlayer.seekTo(0);
            startButton.setEnabled(true);
        }
        catch (Throwable t) {
            Toast.makeText(this, t.getMessage(), Toast.LENGTH_SHORT).show();
        }
    }
    public void onStartClick(View view)
    {
        mediaPlayer.start();
        startButton.setEnabled(false);
        pauseButton.setEnabled(true);
        stopButton.setEnabled(true);
    }
    public void onPauseClick(View view)
    {

```

```

        mediaPlayer.pause();
        startButton.setEnabled(true);
        pauseButton.setEnabled(false);
        stopButton.setEnabled(true);
    }
    public void onStopClick(View view)
    {
        stopPlay();
    }
    @Override
    public void onDestroy() {
        super.onDestroy();
        if (mediaPlayer.isPlaying()) {
            stopPlay();
        }
    }
    @Override
    public void onPrepared(MediaPlayer mp) {
    }
}

```

После установки компонентов и внесения программного кода получим результат, представленный на рисунке 21.

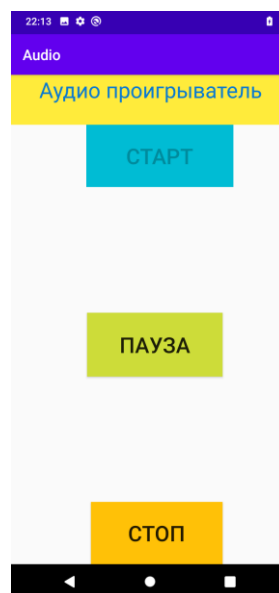


Рисунок 21– Простейший аудиопроигрыватель

1.6.3 Работа с видео

Для создания простейшего проигрывателя в Android Studio можно использовать компонент *VideoView*, в котором задать источник видео файла, а также реализовать кнопки управления.

Добавим видео в папку `res→raw` и назовём его *systemf.mp4*, а также 3 кнопки для управления и поместим компонент *videoView*. В итоге получится следующий файл *activity_main.xml*:

```
<TextView
    android:id="@+id/textView"
    android:layout_width="360dp"
    android:layout_height="43dp"
    android:text="Видео проигрыватель"
    android:textAlignment="center"
    android:textSize="24sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintHorizontal_bias="0.49"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.01" />
<Button
    android:id="@+id/start"
    android:layout_width="167dp"
    android:layout_height="71dp"
    android:onClick="onStartClick"
    android:text="Старт"
    app:layout_constraintBottom_toTopOf="@+id/pause"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView"
    app:layout_constraintVertical_bias="0.542" />
<Button
    android:id="@+id/pause"
    android:layout_width="161dp"
    android:layout_height="70dp"
    android:layout_marginBottom="32dp"
    android:onClick="onPauseClick"
    android:text="Пауза"
    app:layout_constraintBottom_toTopOf="@+id/stop"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.512"
    app:layout_constraintStart_toStartOf="parent" />
<Button
    android:id="@+id/stop"
    android:layout_width="167dp"
    android:layout_height="70dp"
    android:layout_marginBottom="40dp"
    android:onClick="onStopClick"
    android:text="Стоп"
    app:layout_constraintBottom_toTopOf="@+id/videoView"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />
<VideoView
    android:id="@+id/videoView"
    android:layout_width="323dp"
    android:layout_height="273dp"
    android:layout_marginBottom="16dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />
```

Добавим три обработчика событий для этих кнопок (*onStartClick*, *onPauseClick*, *onStopClick*) и укажем ресурс для воспроизведения:


```

public class MainActivity extends AppCompatActivity
{
    VideoView MyvideoPlayer;
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        MyvideoPlayer = (VideoView)findViewById(R.id.videoView);
        Uri myVideoUri= Uri.parse( "android.resource://" + getPackageName() + "/" +
R.raw.systemf);
        MyvideoPlayer.setVideoURI(myVideoUri);
        MediaController mediaController = new MediaController(this);
        MyvideoPlayer.setMediaController(mediaController);
        mediaController.setMediaPlayer(MyvideoPlayer);
    }
    public void onStartClick(View view)
    {
        MyvideoPlayer.start();
    }
    public void onPauseClick(View view)
    {
        MyvideoPlayer.pause();
    }
    public void onStopClick(View view)
    {
        MyvideoPlayer.stopPlayback();
        MyvideoPlayer.resume();
    }
}

```

В результате получим простейший видеопроигрыватель, представленный на рисунке 22.

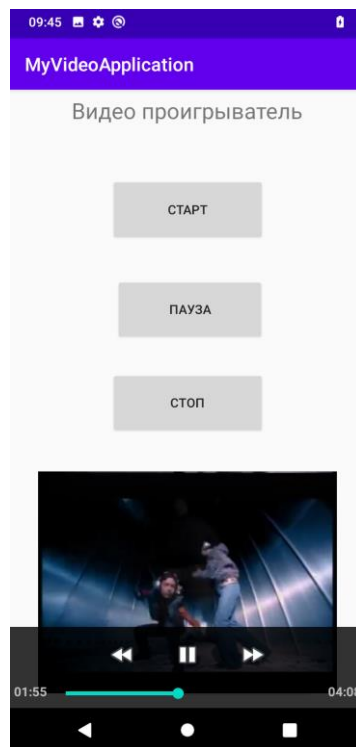


Рисунок 22 – Простейший видеопроигрыватель

1.7 Настройки и состояния приложения

При работе со смартфоном бывают ситуации, когда при вводе текста в поле необходимо его перевернуть, тем самым изменить ориентацию, или свернуть, а через некоторое время развернуть приложение. При таких операциях теряется введенная пользователем информация, что вызывает неудобство и раздражение. Чтобы избежать этого, необходимо запоминать состояния приложения, поэтому создадим простое приложение для реализации данной функции. Поместим на форму два компонента *Button* и два компонента *TextView* со свойством *Multiline*. Код *activity_main.xml* с использованием двух слоёв помимо основного представлен ниже:

```
<EditText
    android:id="@+id/editTextTextMultiLine"
    android:layout_width="match_parent"
    android:layout_height="67dp"
    android:background="#CDDC39"
    android:ems="10"
    android:gravity="start|top"
    android:inputType="textMultiLine"
    android:textSize="24sp"
    tools:layout_editor_absoluteX="-2dp"
    tools:layout_editor_absoluteY="6dp" />

<androidx.gridlayout.widget.GridLayout
    android:id="@+id/gridLayout"
    android:layout_width="392dp"
    android:layout_height="82dp"
    app:columnCount="2"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/editTextTextMultiLine"
    app:rowCount="1">
    <Button
        android:id="@+id/button2"
        android:layout_width="221dp"
        android:layout_height="65dp"
        android:onClick="restoreField"
        android:text="Получить данные"
        android:textSize="18sp"
        app:layout_column="0"
        app:layout_row="0" />
    <Button
        android:id="@+id/button"
        android:layout_width="187dp"
        android:layout_height="72dp"
        android:onClick="saveField"
        android:text="Сохранить"
        android:textSize="18sp"
        app:layout_column="1"
        app:layout_row="0" />
</androidx.gridlayout.widget.GridLayout>

<androidx.gridlayout.widget.GridLayout
    android:layout_width="397dp"
```

```

        android:layout_height="68dp"
        app:columnCount="1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/gridLayout"
        app:rowCount="1">
        <TextView
            android:id="@+id/saveTextView"
            android:layout_width="match_parent"
            android:layout_height="62dp"
            android:background="#00BCD4"
            android:textAlignment="center"
            android:textSize="24sp"
            app:layout_column="0"
            app:layout_gravity="start"
            app:layout_row="0" />
    </androidx.gridlayout.widget.GridLayout>

```

Создадим четыре функции, из которых два обработчика событий по кнопкам *saveField* и *restoreField*.

```

public class MainActivity extends AppCompatActivity {

    String name = "неопределено";
    final static String nameVariableKey = "NAME_VAR";
    final static String textViewTexKey = "TEXT_VIEW";
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    protected void onSaveInstanceState(Bundle outState)
    {
        outState.putString(nameVariableKey, name);
        TextView nameView = (TextView) findViewById(R.id.saveTextView);
        outState.putString(textViewTexKey, nameView.getText().toString());
        super.onSaveInstanceState(outState);
    }
    @Override
    protected void onRestoreInstanceState(Bundle savedInstanceState)
    {
        super.onRestoreInstanceState(savedInstanceState);
        name = savedInstanceState.getString(nameVariableKey);
        String textViewText= savedInstanceState.getString(textViewTexKey);
        TextView nameView = (TextView) findViewById(R.id.saveTextView);
        nameView.setText(textViewText);
    }
    public void saveField(View view)
    {
        TextView nameBox = (TextView) findViewById(R.id.editTextTextMultiLine);
        name = nameBox.getText().toString();
    }
    public void restoreField(View view)
    {
        TextView nameView = (TextView) findViewById(R.id.saveTextView);
        nameView.setText(name);
    }
}

```

Получим мобильное приложение, представленное на рисунках 23, 24.

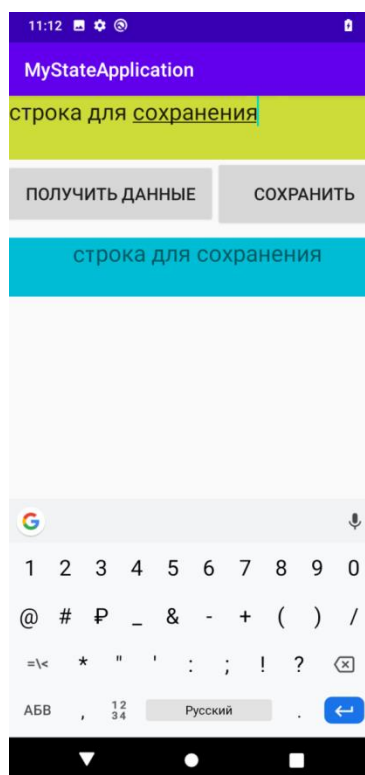


Рисунок 23 – Вертикальная ориентация при вводе

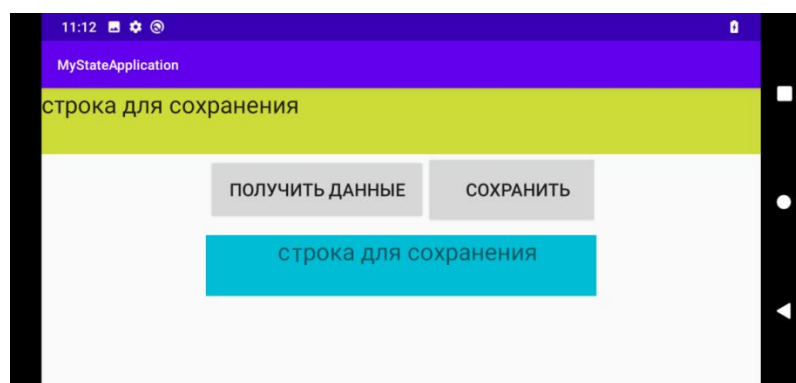


Рисунок 24 – Изменение ориентации смартфона после ввода

1.8 Работа с файловой системой

Изучение среды разработки, как правило, подразумевает под собой освоение работы с файловой системой. Необходимо понимать, где и каким образом хранятся файлы на смартфоне, а также как к ним обращаться. Как правило, на смартфоне с операционной системе Android имеются три вида хранилища: внутреннее хранилище (системные файлы), SD-карта и внутренний общий накопитель.

Для примера напишем приложения для сохранения текстового файла на внутренний общий накопитель. Вначале необходимо установить разрешение на чтение/запись в память смартфона в файле *AndroidManifest.xml*.

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

Создадим интерфейс, поместив два компонента *Button* для сохранения в файл и чтения из него, один компонент *Plain Text* для ввода текста и компонент *TextView* для вывода на экран, рисунок 25.

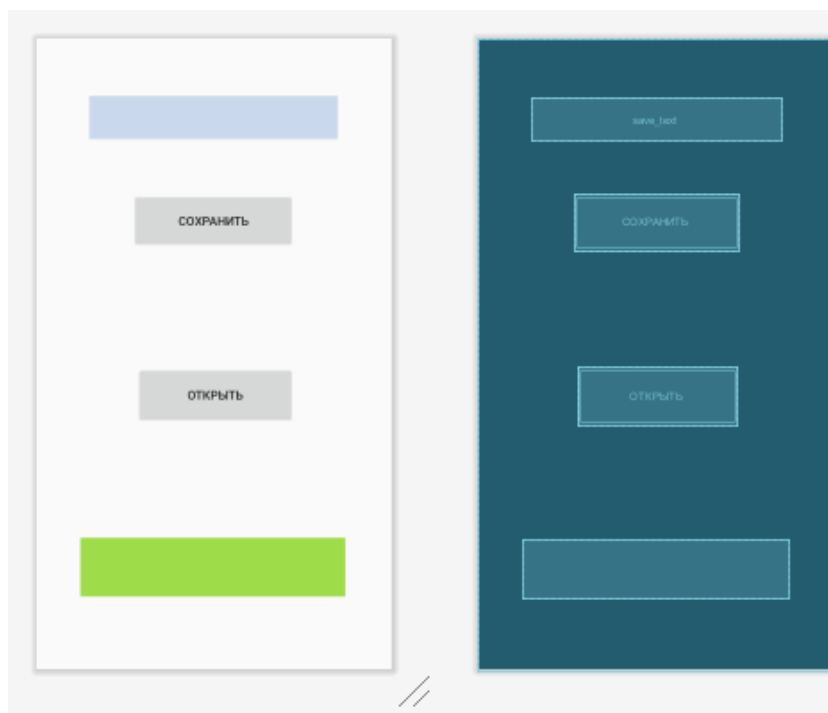


Рисунок 25 – Интерфейс приложения для работы с файловой системой

Опишем свойства установленных нами элементов в файле *activity_main.xml*.

```
<TextView
    android:id="@+id/open_text"
    android:layout_width="308dp"
    android:layout_height="68dp"
    android:layout_marginBottom="84dp"
    android:background="#9FDC49"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.495"
    app:layout_constraintStart_toStartOf="parent" />
<Button
    android:id="@+id/buttonSave"
    android:layout_width="189dp"
    android:layout_height="65dp"
    android:layout_marginBottom="136dp"
    android:onClick="buttonSaveClick"
```

```

        android:text="Сохранить"
        app:layout_constraintBottom_toTopOf="@+id/buttonOpen"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />
<Button
    android:id="@+id/buttonOpen"
    android:layout_width="184dp"
    android:layout_height="67dp"
    android:layout_marginBottom="132dp"
    android:onClick="buttonOpenClick"
    android:text="Открыть"
    app:layout_constraintBottom_toTopOf="@+id/open_text"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.511"
    app:layout_constraintStart_toStartOf="parent" />
<EditText
    android:id="@+id/save_text"
    android:layout_width="289dp"
    android:layout_height="50dp"
    android:layout_marginTop="56dp"
    android:background="#C9D8EC"
    android:ems="10"
    android:inputType="textPersonName"
    app:layout_constraintBottom_toTopOf="@+id/buttonSave"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.148" />

```

Создадим обработчик события для кнопки «Сохранить». Для корректности работы необходимо сделать обработчик исключений *try-catch*. Имя и путь для сохранения файла с именем «*content.txt*» указывается в классе *MainActivity* с расширением *AppCompatActivity*:

```

public class MainActivity extends AppCompatActivity
{
    String fileName = "content.txt";
    File file = new File(Environment.getExternalStorageDirectory().getAbsolutePath(),
fileName);
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void buttonSaveClick(View view)
    {
        try
        {
            EditText textBox = (EditText) findViewById(R.id.save_text);
            String text = textBox.getText().toString();
            FileOutputStream fos = new FileOutputStream(file);
            fos.write(text.getBytes());
            fos.close();
            Toast.makeText(this, "Текстовый файл успешно сохранён!",
Toast.LENGTH_SHORT).show();
        } catch (FileNotFoundException e)
        {
            e.printStackTrace();
            Toast.makeText(this, "Файл не найден!", Toast.LENGTH_SHORT).show();
        }
    }
}

```

```

    } catch (IOException e)
    {
        e.printStackTrace();
        Toast.makeText(this, "Ошибка сохранения файла!",
Toast.LENGTH_SHORT).show();
    }
}
}

```

После окончания записи необходимо закрыть поток командой *fos.close*. На рисунке 26 показан процесс успешного сохранения в файл строки «строка в текстовом файле», полученной из поля *EditText* с *id/save_text*.



Рисунок 26 – Процесс сохранения текстовой строки в файл на смартфоне

Для реализации функции чтения из сохранённого нами файла создадим обработчик события для кнопки «Открыть». Выводить информацию из файла будем в поле *EditText* с *id/open_text*.

```

public void buttonOpenClick(View view)
{
    try
    {
        FileInputStream fin = new FileInputStream(file);
        byte[] bytes = new byte[fin.available()];
        fin.read(bytes);
        String text = new String(bytes);
        TextView textView = (TextView) findViewById(R.id.open_text);
        textView.setText(text);
    }
}

```

```

        fin.close();
    } catch (IOException ex)
    {
        Toast.makeText(this, ex.getMessage(), Toast.LENGTH_SHORT).show();
    }
}

```

После нажатия на кнопку «Открыть» получим результат, представленный на рисунке 27.



Рисунок 27 – Процесс чтения из файла на смартфоне

Найдем созданный файл на смартфоне. Для этого перейдём в «Хранилище», рисунок 28.

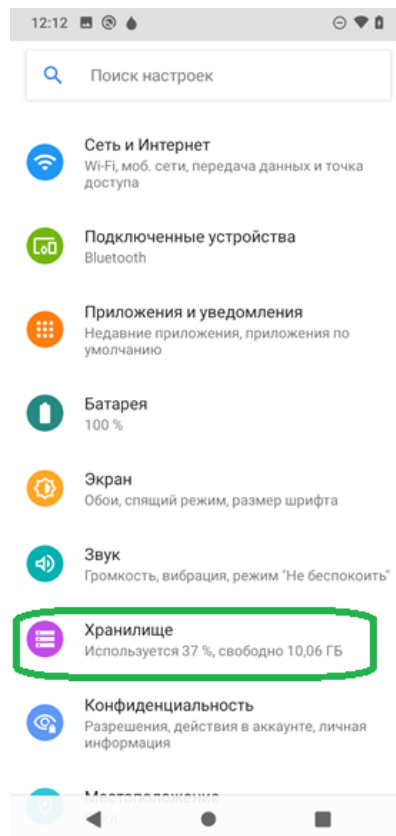


Рисунок 28 – Поиск файла на смартфоне

Далее выберем пункт «*Файлы*», рисунок 29.

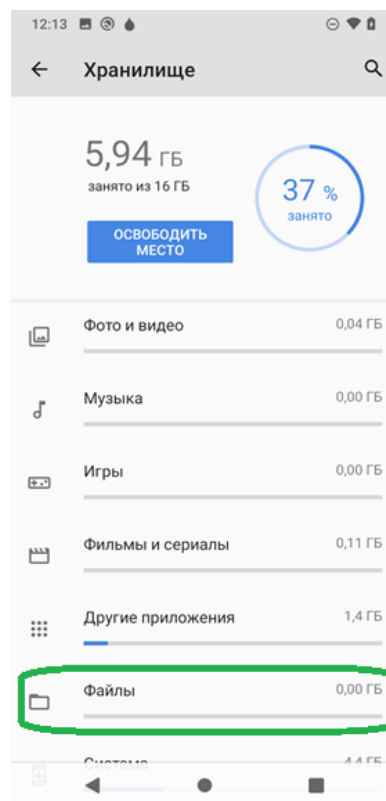


Рисунок 29 – Поиск файла на смартфоне через меню «*Хранилище*»

После открытия меню «*Файлы*» мы видим созданный нами файл «*content.txt*».

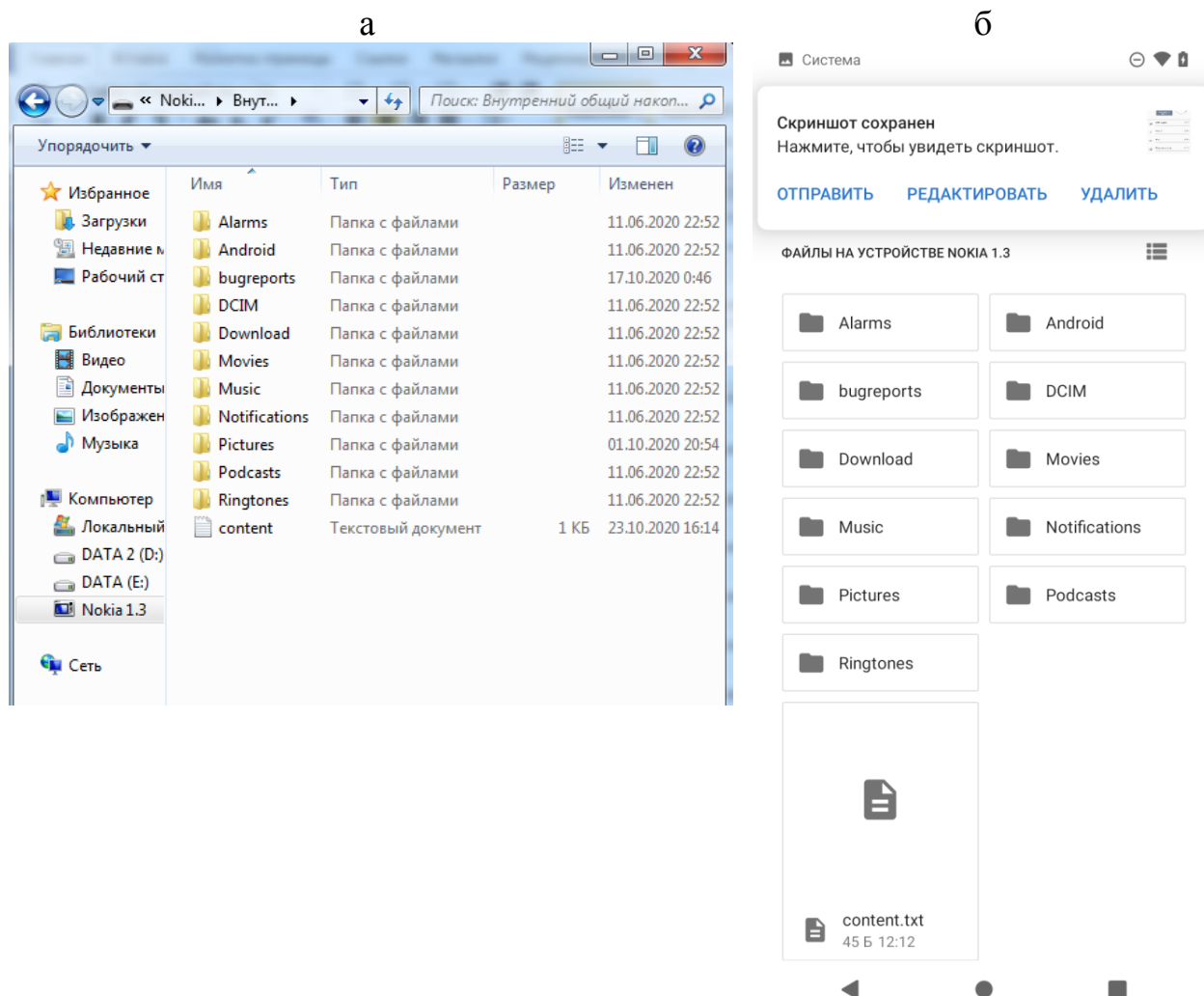


Рисунок 30 – Поиск файла на смартфоне (а) и через меню «*Файлы*» (б)

Откроем файл на смартфоне, после чего увидим сохранённую нами строку «*строка в текстовом файле*», рисунок 31.



Рисунок 31 – Запись в сохранённом текстовом файле

1.9 Работа с базой данных SQLite

В IDE Android Studio можно работать с большинством современных баз данных. Разработчик должен для себя решить две основные задачи: какой будет интерфейс взаимодействия между базой данных и приложением; достаточно ли будет функционала его приложения с используемой базой данных для реализации поставленной задачи [6].

SQLite является встраиваемой системой управления базами данных (СУБД), и для демонстрации работы мобильного приложения её вполне достаточно. Для работы с любой базой данных необходимо знать типы данных, с которыми предстоит работа, таблица 1.

Таблица 1 – Типы данных в СУБД *SQLite*

Номер п/п	Тип	Описание
1	NULL	Пустое значение
2	INTEGER	Целочисленное значение
3	REAL	Значение с плавающей точкой
4	TEXT	Строки или символы в кодировке UTF-8, UTF-16BE или UTF-16LE
5	BLOB	Бинарные данные

Разработаем интерфейс мобильного приложения для работы с базой данных, рисунок 32, в которой будет всего одна таблица.

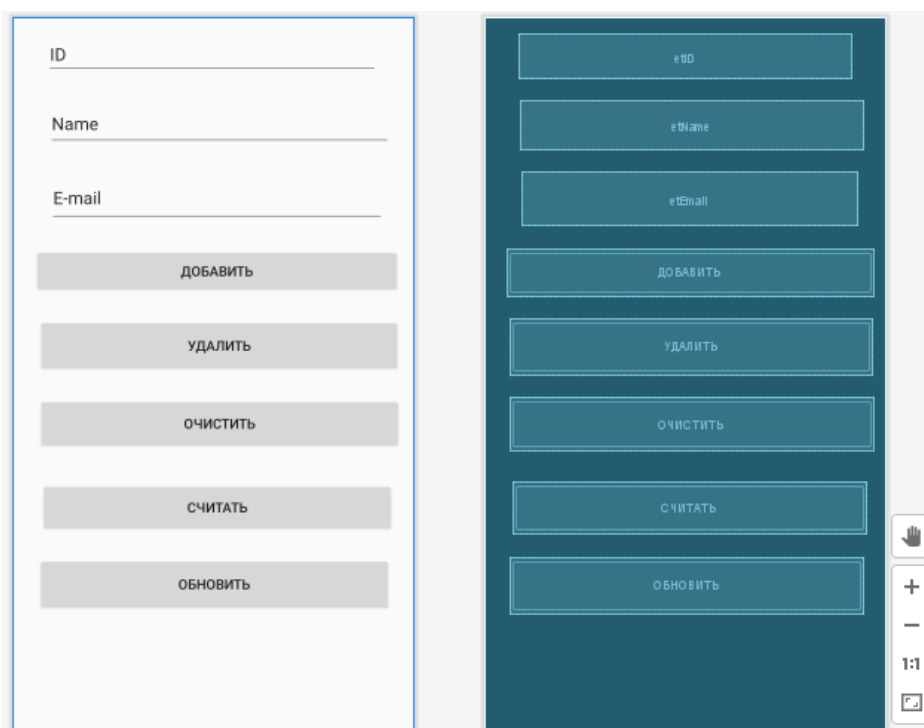


Рисунок 32 – Интерфейс мобильного приложения для реализации основных команд при работе с базой данных

Здесь поле *ID* является первичным ключом. Поля *Name* и *E-mail* являются текстовыми и предназначены для ввода имени и электронной почты пользователя. Кнопка «Добавить» предназначена для внесения новой записи в базу данных из полей *Name* и *E-mail*. Кнопка «Удалить» предназначена для удаления одной записи из базы данных по известному *ID*. Кнопка «Очистить» предназначена для удаления всех записей сразу из таблицы в базе данных. Кнопка «Считать» предназначена считывания всех записей из таблицы в базе данных. Кнопка «Обновить» предназначена для изменения полей *Name* и/или *E-mail* при известном *ID*.

Опишем кнопки и поля следующим образом:

```

<Button
    android:id="@+id/buttonRead"
    android:layout_width="363dp"
    android:layout_height="53dp"
    android:layout_marginTop="32dp"
    android:text="Считать"
    app:layout_constraintTop_toBottomOf="@+id/buttonClear"
    tools:layout_editor_absoluteX="29dp" />

<Button
    android:id="@+id/buttonAdd"
    android:layout_width="376dp"
    android:layout_height="48dp"
    android:layout_marginTop="24dp"
    android:text="Добавить"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.628"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/etEmail" />

<Button
    android:id="@+id/buttonDelete"
    android:layout_width="372dp"
    android:layout_height="57dp"
    android:layout_marginTop="24dp"
    android:text="Удалить"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.666"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/buttonAdd" />

<Button
    android:id="@+id/buttonClear"
    android:layout_width="373dp"
    android:layout_height="55dp"
    android:layout_marginTop="24dp"
    android:text="Очистить"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.684"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/buttonDelete" />

<EditText
    android:id="@+id/etName"
    android:layout_width="352dp"
    android:layout_height="50dp"
    android:layout_marginTop="24dp"
    android:ems="10"
    android:inputType="textPersonName"
    android:text="Name"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.615"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/etID" />

<EditText
    android:id="@+id/etEmail"
    android:layout_width="344dp"
    android:layout_height="54dp"
    android:layout_marginTop="24dp"
    android:ems="10"

```

```

android:inputType="textPersonName"
android:text="E-mail"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.567"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/etName" />

```

<EditText

```

android:id="@+id/etID"
android:layout_width="341dp"
android:layout_height="45dp"
android:layout_marginTop="16dp"
android:ems="10"
android:inputType="textPersonName"
android:text="ID"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.494"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />

```

<Button

```

android:id="@+id/buttonUpdate"
android:layout_width="363dp"
android:layout_height="57dp"
android:layout_marginTop="24dp"
android:text="Обновить"
app:layout_constraintTop_toBottomOf="@+id/buttonRead"
tools:layout_editor_absoluteX="26dp" />

```

Добавим новый класс *DBHelper.java* для работы с базой данных, рисунок 33.

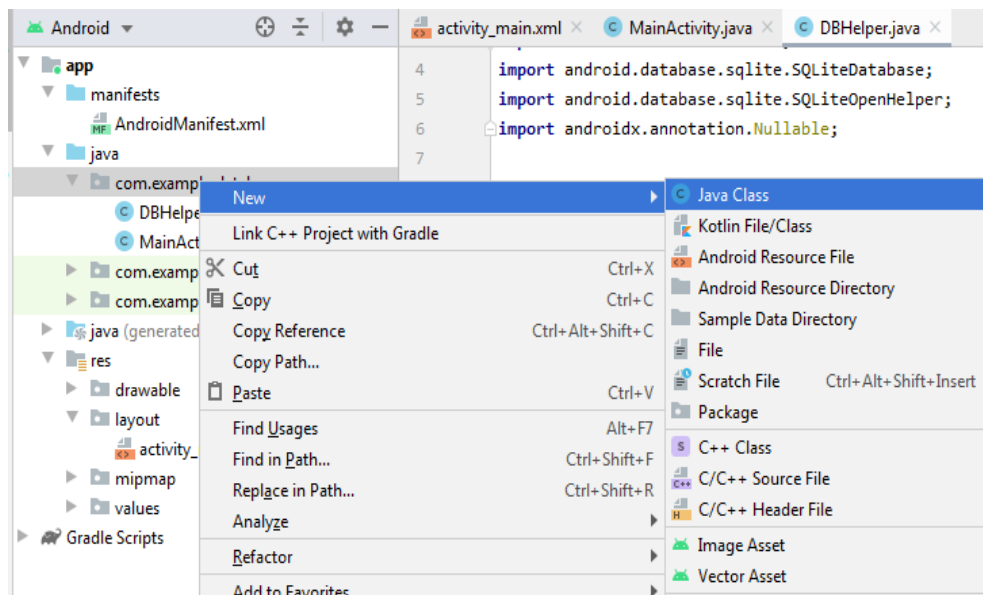


Рисунок 33 – Добавление нового класса для работы с базой данных

Сделаем расширение класса и включим методы *onCreate* и *onUpdate*, рисунок 34.

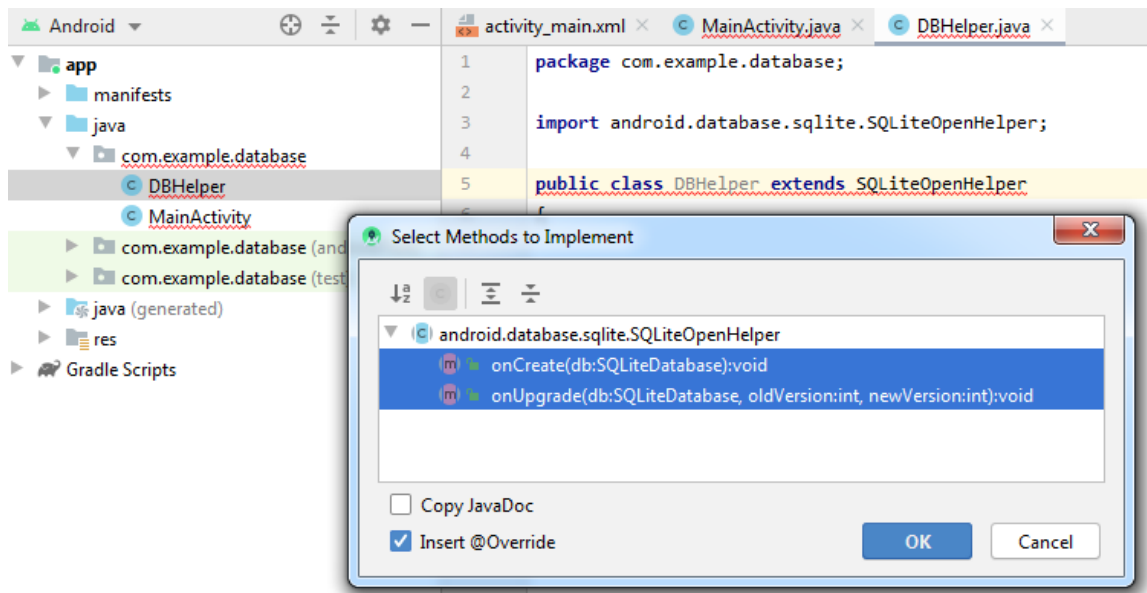


Рисунок 34 – Добавление необходимых методов в класс *DBHelper.java*

С помощью конструктора добавляем супер класс *SQLiteOpenHelper* с параметрами *context:Context, name:String, factory:CursorFactory, version:int*.

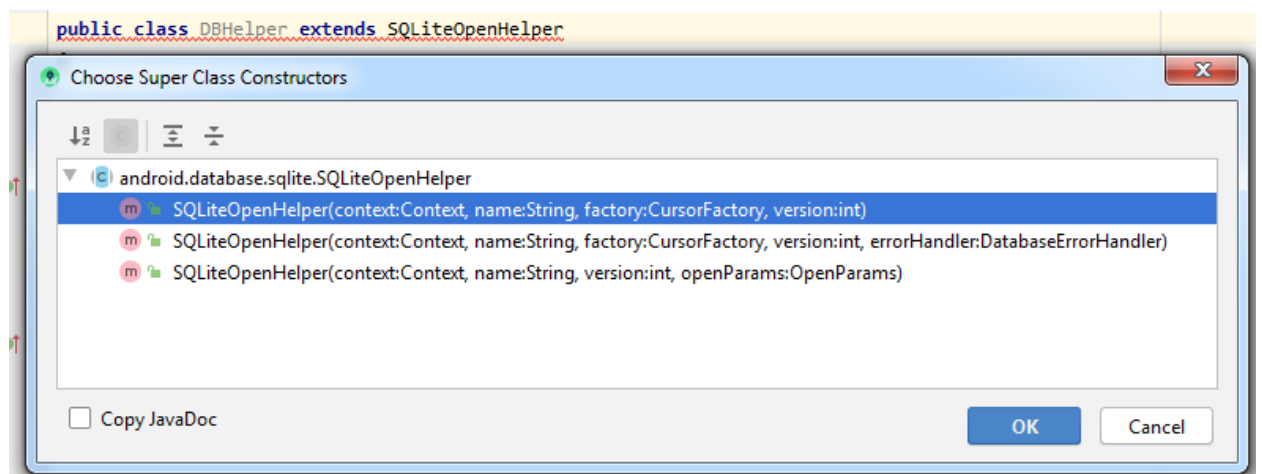


Рисунок 35 – Интерфейс мобильного приложения для реализации основных команд при работе с базой данных

Удаляем не используемый параметр и получаем:

```

public DBHelper(@Nullable Context context)
{
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
}

```

Код файла *DBHelper.java* выглядит следующим образом:

```

public class DBHelper extends SQLiteOpenHelper
{
    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "myBase";
    public static final String TABLE_PERSONS = "persons";
    public static final String KEY_ID = "_id";
    public static final String KEY_NAME = "name";

```

```

public static final String KEY_MAIL = "mail";

public DBHelper(@Nullable Context context)
{
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
}

@Override
public void onCreate(SQLiteDatabase db)
{
    db.execSQL("create table " + TABLE_PERSONS + "(" + KEY_ID + " integer primary key," + KEY_NAME + " text," + KEY_MAIL + " text" + ")");
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
{
    db.execSQL("drop table if exists " + TABLE_PERSONS);
    onCreate(db);
}
}

```

Расширяем класс *MainActivity* до *AppCompatActivity* и делаем включение *View.OnClickListener*:

```

public class MainActivity extends AppCompatActivity implements View.OnClickListener
{
    Button buttonAdd, buttonDelete, buttonClear, buttonRead, buttonUpdate;
    EditText etName, etEmail, etID;
    DBHelper dbHelper;
}

```

Создаём обработчик событий для кнопки «Добавить запись»:

```

case R.id.buttonAdd:
    contentValues.put(DBHelper.KEY_NAME, name);
    contentValues.put(DBHelper.KEY_MAIL, email);
    database.insert(DBHelper.TABLE_PERSONS, null, contentValues);
    break;

```

Создаём обработчик событий для кнопки «Считать»:

```

case R.id.buttonRead:
    Cursor cursor = database.query(DBHelper.TABLE_PERSONS, null, null, null,
        null, null, null); // все поля без сортировки и группировки

    if (cursor.moveToFirst())
    {
        int idIndex = cursor.getColumnIndex(DBHelper.KEY_ID);
        int nameIndex = cursor.getColumnIndex(DBHelper.KEY_NAME);
        int emailIndex = cursor.getColumnIndex(DBHelper.KEY_MAIL);
        do {
            Log.d("mLog", "ID = " + cursor.getInt(idIndex) +
                ", name = " + cursor.getString(nameIndex) +
                ", email = " + cursor.getString(emailIndex));
        } while (cursor.moveToNext());
    } else
        Log.d("mLog", "0 rows");

    cursor.close(); // освобождение памяти
    break;

```



```
}
```

Создаём обработчик событий для кнопки «Очистить» для удаления всех записей из таблицы «*persons*»:

```
case R.id.buttonClear:
    database.delete(DBHelper.TABLE_PERSONS, null, null);
    break;
```

Создаём обработчик событий для кнопки «Удалить» для удаления всех записей из таблицы «*persons*»:

```
case R.id.buttonDelete:
    if (ID.equalsIgnoreCase(""))
    {
        break;
    }
    int delCount = database.delete(DBHelper.TABLE_PERSONS, DBHelper.KEY_ID + "= " +
ID, null);
    Log.d("mLog", "Удалено строк = " + delCount);
```

Создаём обработчик событий для кнопки «Обновить» для обновления строки с выбранным *ID* из таблицы «*persons*»:

```
case R.id.buttonUpdate:
    if (ID.equalsIgnoreCase(""))
    {
        break;
    }
    contentValues.put(DBHelper.KEY_MAIL, email);
    contentValues.put(DBHelper.KEY_NAME, name);
    int updCount = database.update(DBHelper.TABLE_PERSONS, contentValues,
DBHelper.KEY_ID + "= ?", new String[] {ID});
    Log.d("mLog", "Обновлено строк = " + updCount);
```

Изменим код в файле *MainActivity.java*:

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener
{
    Button buttonAdd, buttonDelete, buttonClear, buttonRead, buttonUpdate;
    EditText etName, etEmail, etID;
    DBHelper dbHelper;

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

        buttonAdd = (Button) findViewById(R.id.buttonAdd);
        buttonAdd.setOnClickListener(this);

        buttonRead = (Button) findViewById(R.id.buttonRead);
        buttonRead.setOnClickListener(this);

        buttonClear = (Button) findViewById(R.id.buttonClear);
        buttonClear.setOnClickListener(this);

        buttonUpdate = (Button) findViewById(R.id.buttonUpdate);
        buttonUpdate.setOnClickListener(this);
    }
}
```

```

        buttonDelete = (Button) findViewById(R.id.buttonDelete);
        buttonDelete.setOnClickListener(this);

        etID = (EditText) findViewById(R.id.etID);
        etName = (EditText) findViewById(R.id.etName);
        etEmail = (EditText) findViewById(R.id.etEmail);

        dbHelper = new DBHelper(this);
    }

    @Override
    public void onClick(View v)
    {
        String ID = etID.getText().toString();
        String name = etName.getText().toString();
        String email = etEmail.getText().toString();

        SQLiteDatabase database = dbHelper.getWritableDatabase();
        ContentValues contentValues = new ContentValues(); // класс для добавления
новых строк в таблицу

        switch (v.getId())
        {
            case R.id.buttonAdd:
                contentValues.put(DBHelper.KEY_NAME, name);
                contentValues.put(DBHelper.KEY_MAIL, email);
                database.insert(DBHelper.TABLE_PERSONS, null, contentValues);
                break;

            case R.id.buttonRead:
                Cursor cursor = database.query(DBHelper.TABLE_PERSONS, null, null,
null,
                null, null, null); // все поля без сортировки и группировки

                if (cursor.moveToFirst())
                {
                    int idIndex = cursor.getColumnIndex(DBHelper.KEY_ID);
                    int nameIndex = cursor.getColumnIndex(DBHelper.KEY_NAME);
                    int emailIndex = cursor.getColumnIndex(DBHelper.KEY_MAIL);
                    do {
                        Log.d("mLog", "ID =" + cursor.getInt(idIndex) +
                            ", name =" + cursor.getString(nameIndex) +
                            ", email =" + cursor.getString(emailIndex));
                    } while (cursor.moveToNext());
                } else
                    Log.d("mLog", "0 rows");

                cursor.close(); // освобождение памяти
                break;

            case R.id.buttonClear:
                database.delete(DBHelper.TABLE_PERSONS, null, null);
                break;

            case R.id.buttonDelete:
                if (ID.equalsIgnoreCase(""))
                {
                    break;
                }
                int delCount = database.delete(DBHelper.TABLE_PERSONS,

```

```

DBHelper.KEY_ID + "= " + ID, null);
    Log.d("mLog", "Удалено строк = " + delCount);

    case R.id.buttonUpdate:
        if (ID.equalsIgnoreCase(""))
        {
            break;
        }
        contentValues.put(DBHelper.KEY_MAIL, email);
        contentValues.put(DBHelper.KEY_NAME, name);
        int updCount = database.update(DBHelper.TABLE_PERSONS, contentValues,
DBHelper.KEY_ID + "= ?", new String[] {ID});
        Log.d("mLog", "Обновлено строк = " + updCount);
    }
    dbHelper.close(); // закрываем соединение с БД
}
}
}

```

Вывод информации о работе с базой данных осуществляется в информационное поле Android Studio с целью предотвращения избыточности информации на главном экране мобильного приложения.

Добавление новой записи в таблицу осуществляется путём ввода информации в поля *Name* и *E-mail*, рисунок 36. В данном случае нет необходимости вводить поле *ID*, так как оно является уникальным полем со свойством автоинкремент.

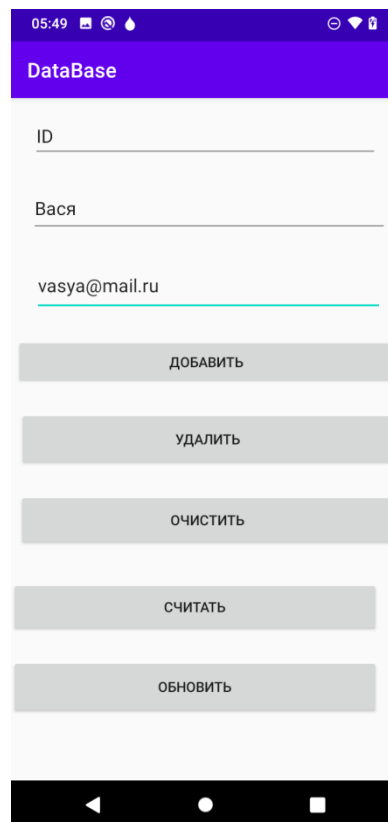


Рисунок 36 – Добавление новой записи в таблицу базы данных

Результат работы виден в информационном сообщении на рисунке 37 в первой, второй и третьей строках.

Считать все записи:

```
2020-11-03 08:51:01.282 2384-2384/com.example.database D/mLog: ID =1, name = Вася, email = vasya@mail.ru
2020-11-03 08:51:01.283 2384-2384/com.example.database D/mLog: ID =2, name Коля, email = kolya@rambler.ru
ll successfully finished in 375 ms.
restart successful without requiring a re-install. m.example.database D/mLog: ID =3, name = Маша, email = mascha@rambler.ru
```

Рисунок 37 – Вывод информации, хранящейся в таблице базы данных

Кнопка «Обновить» подразумевает изменение записи/записей, уже имеющихся в таблице. Для примера изменим запись с *ID* = 2 для поля *Name* = «Коля», рисунок 37. Изменим поле *Name* = «Коля» на «Николай». Для этого необходимо ввести *ID* изменяемого поля, которое в данном случае равно 2 и внести соответствующую корректировку, рисунок 38.

Рисунок 38 – Внесение изменений в поле *Name* для *ID* = 2

Результат работы кнопки «Обновить» продемонстрирован на рисунках 37 и 39.

```
2020-11-03 08:54:47.620 2384-2384/com.example.database D/mLog: ID =1, name = Вася, email = vasya@mail.ru
2020-11-03 08:54:47.620 2384-2384/com.example.database D/mLog: ID =2, name Николай, email = kolya@rambler.ru
l successfully finished in 375 ms.
restart successful without requiring a re-install. m.example.database D/mLog: ID =3, name = Маша, email = mascha@rambler.ru
```

Рисунок 39 – Вывод информации, хранящейся в таблице базы данных

Удаление записи осуществляется по *ID*, поэтому введём известное значение и нажмём кнопку «Удалить», рисунок 40.

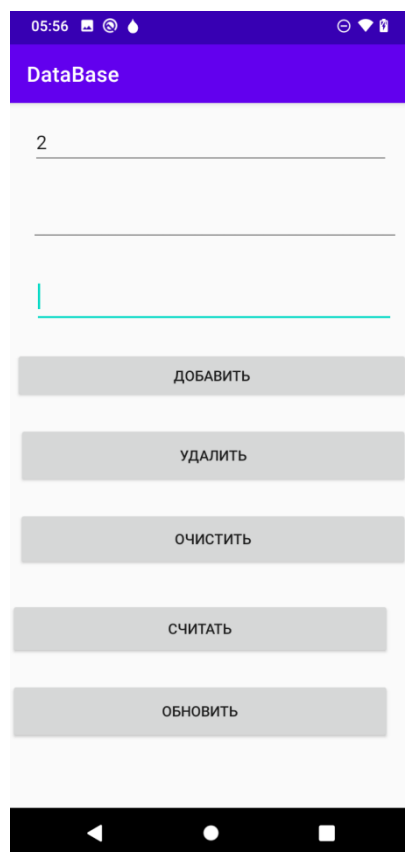


Рисунок 40 – Удаление записи по *ID* = 2

Работоспособность кнопки «Удалить» продемонстрирована на рисунках 39 и 41.

```
2020-11-03 08:57:04.025 2384-2384/com.example.database D/mLog: Удалено строк = 1
2020-11-03 08:57:04.028 2384-2384/com.example.database D/mLog: Обновлено строк = 0
2020-11-03 08:57:07.691 2384-2384/com.example.database D/mLog: ID =1, name = Вася, email = vasya@mail.ru
ll successfully finished in 375 ms. m.example.database D/mLog: ID =3, name = Маша, email = mascha@rambler.ru
restart successful without requiring a re-install
```

Рисунок 41– Вывод информации, хранящейся в таблице базы данных

Чтобы очистить всю таблицу «*persons*» целиком необходимо нажать кнопку «Очистить», рисунок 42.

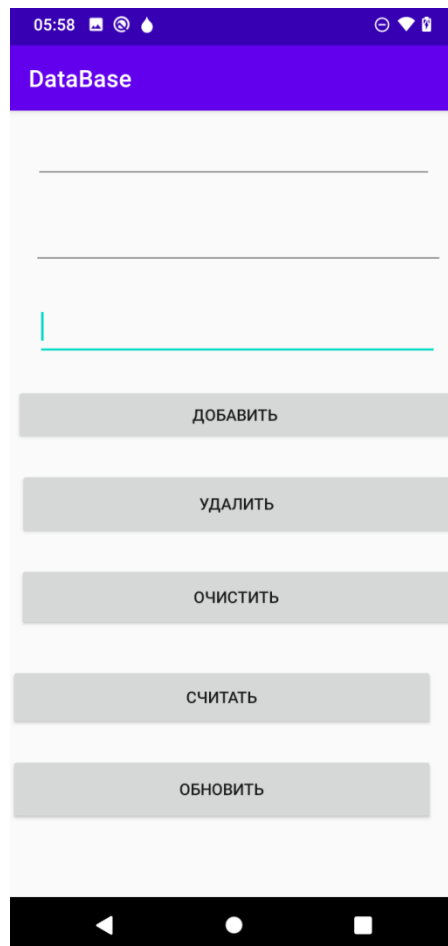


Рисунок 42 – Процесс очистки всей таблицы

Работоспособность кнопки «*Очистить*» продемонстрирована на рисунках 41 и 43.

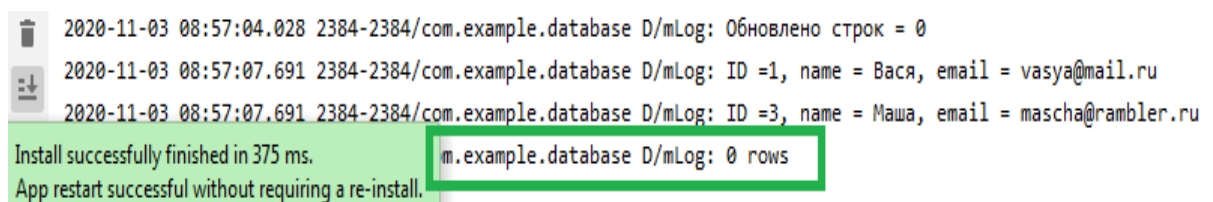


Рисунок 43 – Вывод информации, хранящейся в таблице базы данных

Просмотр базы данных осуществляется с помощью обозревателя файлов устройства, рисунок 44.

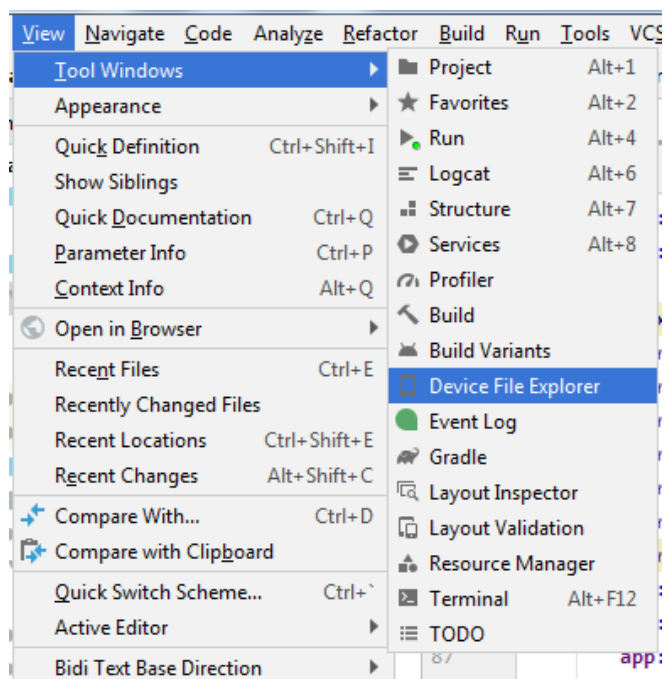


Рисунок 44 – Обозреватель файлов на смартфоне

В папке *com.example.database* находится созданная нами база данных с именем *myBase*.

HMD Global Nokia 1.3 Android 10, API 29			
Name	Permissions	Date	Size
▶ com.android.wallpaperbackup	drwxrwx--x	1970-01-02 01:11	4 KB
▶ com.android.wallpaperpicker	drwxrwx--x	1970-01-02 01:11	4 KB
▶ com.dsi.ant.server	drwxrwx--x	1970-01-02 01:11	4 KB
▶ com.example.animation	drwxrwx--x	1970-01-02 01:11	4 KB
▶ com.example.audio	drwxrwx--x	1970-01-02 01:11	4 KB
▼ com.example.database	drwxrwx--x	1970-01-02 01:11	4 KB
▶ cache	drwxrws--x	2020-10-26 11:04	3,4 KB
▶ code_cache	drwxrws--x	2020-10-26 11:04	3,4 KB
▼ databases	drwxrwx--x	2020-10-26 18:11	3,4 KB
myBase	-rw-rw----	2020-11-03 05:58	16 KB

Рисунок 45 – Очистка всей таблицы

В процессе программирования функционала приложения можно было бы создать отдельные обработчики событий для каждой из кнопок, но в данной ситуации реализация была осуществлена через операторы *switch-case-break*.

1.10 Телефония

Техническое развитие и модернизация сотовых телефонов дало толчок появлению смартфонов. Смартфоны умеют всё то же, что делали их предшественники: отправлять СМС-сообщения и звонить другому абоненту. Функции СМС-сообщений и звонков в настоящее время уже замещаются, так как могут осуществляться с помощью сторонних приложений, например *WhatsApp*, *ВКонтакте (VK)*, *Skype*, *Viber* и др.

В данном разделе будут представлены реализации данных функций для ознакомления. Разработаем графический интерфейс приложения, в который будут входить: поле ввода номера телефона; кнопка «Позвонить»; поле ввода текста СМС-сообщения и кнопка «Отправить СМС», рисунок 46.

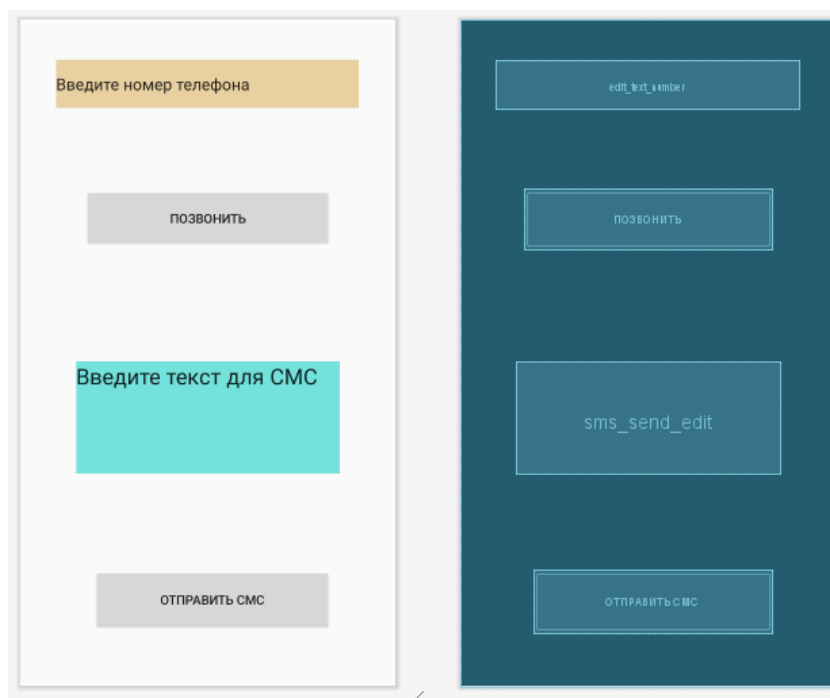


Рисунок 46 – Графический интерфейс приложения для отправки СМС и звонков

Чтобы смартфон имел возможность отправлять СМС-сообщения и делать звонить на другой номер телефона необходимо установить разрешение в файле *AndroidManifest.xml*:

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

Настройки компонентов графического интерфейса представлены в файле *ActivityMain.xml*:

```
<EditText
    android:id="@+id/edit_text_number"
    android:layout_width="332dp"
```



```

        android:layout_height="53dp"
        android:layout_marginTop="44dp"
        android:background="#E8D0A0"
        android:ems="10"
        android:inputType="phone"
        android:text="Введите номер телефона"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.493"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

<Button
    android:id="@+id/call_button"
    android:layout_width="271dp"
    android:layout_height="66dp"
    android:layout_marginTop="88dp"
    android:text="Позвонить"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/edit_text_number" />

<EditText
    android:id="@+id/sms_send_edit"
    android:layout_width="289dp"
    android:layout_height="123dp"
    android:layout_marginTop="124dp"
    android:background="#73E1DC"
    android:ems="10"
    android:gravity="start|top"
    android:inputType="textMultiLine"
    android:text="Введите текст для СМС"
    android:textSize="24sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/call_button" />

<Button
    android:id="@+id/sms_send_button"
    android:layout_width="261dp"
    android:layout_height="69dp"
    android:layout_marginBottom="60dp"
    android:onClick="onSms"
    android:text="Отправить СМС"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.533"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/sms_send_edit"
    app:layout_constraintVertical_bias="1.0" />

```

1.10.1 Вызов абонента

В данном разделе описана часть программного кода для функции отправки сообщения.

Чтобы осуществить вызов другого абонента необходимо использовать функцию *ACTION_CALL*, а также отправить номер телефона в функцию *Uri.parse*. Ниже приведён изменённый код функции *onCreate*:

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

    Button mDialButton = (Button) findViewById(R.id.call_button);
    final EditText mPhoneNoEt = (EditText) findViewById(R.id.edit_text_number);
    final EditText smsEdit = (EditText) findViewById(R.id.sms_send_edit);

    mDialButton.setOnClickListener(new View.OnClickListener()
    {
        @Override
        public void onClick(View view)
        {
            String phoneNo = mPhoneNoEt.getText().toString();
            if(!TextUtils.isEmpty(phoneNo))
            {
                String dial = "tel:" + phoneNo;
                startActivity(new Intent(Intent.ACTION_CALL, Uri.parse(dial)));
            }
            else {
                Toast.makeText(MainActivity.this, "Введите номер телефона",
                    Toast.LENGTH_SHORT).show();
            }
        }
    });
}
```

Для установленного мобильного приложения необходимо дать разрешение на осуществления вызова в настройках самого приложения, рисунок 47, 48.

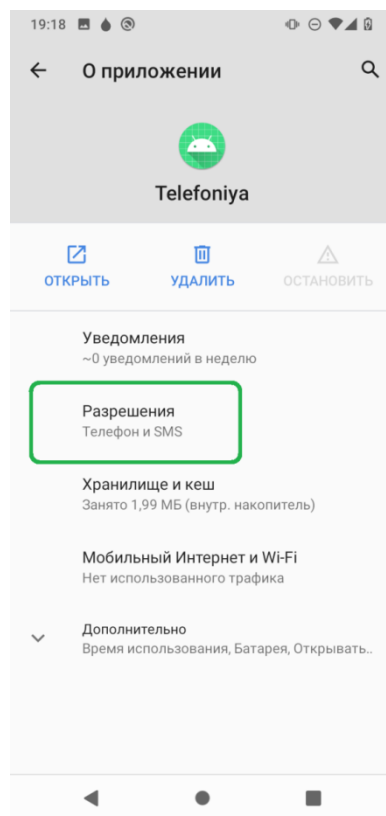


Рисунок 47 – Установка разрешений для приложения

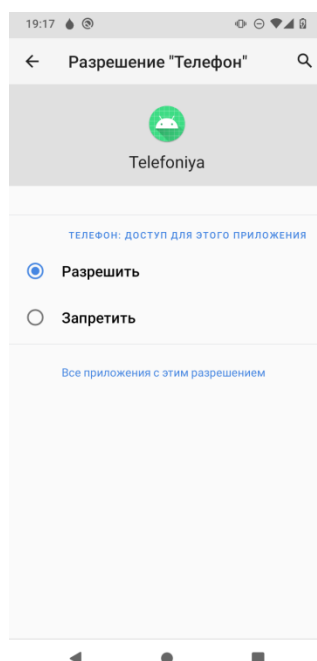


Рисунок 48 – Проверка установленного разрешения для приложения

Чтобы проверить работоспособность приложения достаточно позвонить самому себе (рисунок 49) и записать короткое сообщение, которое можно будет потом прослушать по ссылке в СМС, рисунок 50.

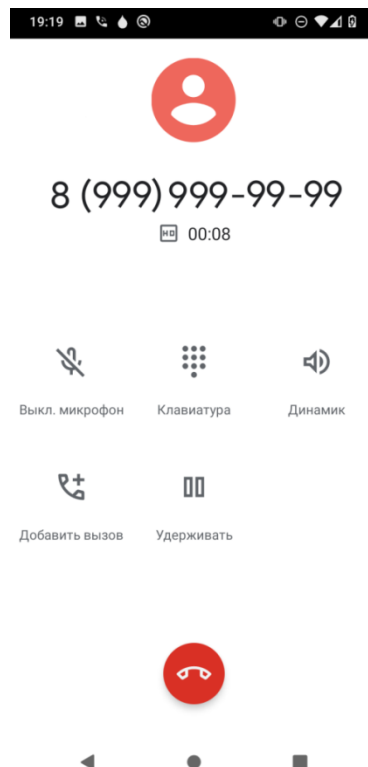


Рисунок 49 – Вызов на свой собственный номер и запись голосового сообщения

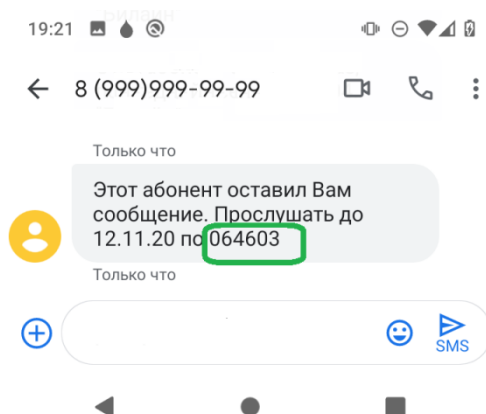


Рисунок 50 – Ссылка для прослушивания своего голосового сообщения

1.10.2 Отправка СМС-сообщений

Используя интерфейс, разработанный в начале раздела 10.1 программируем обработчик кнопки «*Отправить СМС*»:

```

public void onSms(View view)
{
    EditText edit_Number=(EditText)findViewById(R.id.edit_text_number);
    String phoneNo = edit_Number.getText().toString();
    EditText sms_edit=(EditText)findViewById(R.id.sms_send_edit);
    String toSms="smsto:"+edit_Number.getText().toString();
    String messageText= sms_edit.getText().toString();
    Intent sms=new Intent(Intent.ACTION_SENDTO, Uri.parse(toSms));

    sms.putExtra("sms_body", messageText);
    startActivity(sms);
    SmsManager.getDefault().sendTextMessage(phoneNo, null, messageText.toString(),
    null, null);
}

```

Устанавливаем разрешение на отправку СМС по аналогу на осуществление вызова, рисунок 51, 52.

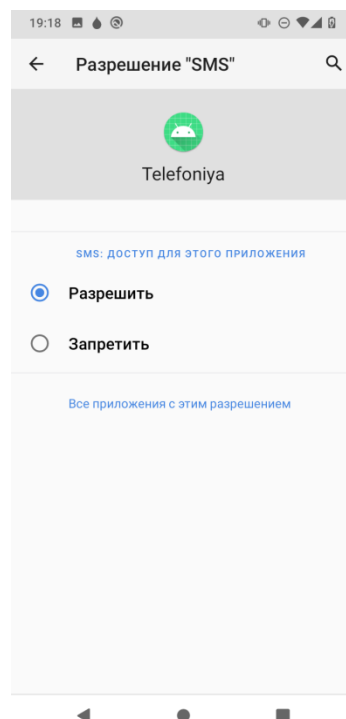


Рисунок 51 – Проверка установленного разрешения для приложения

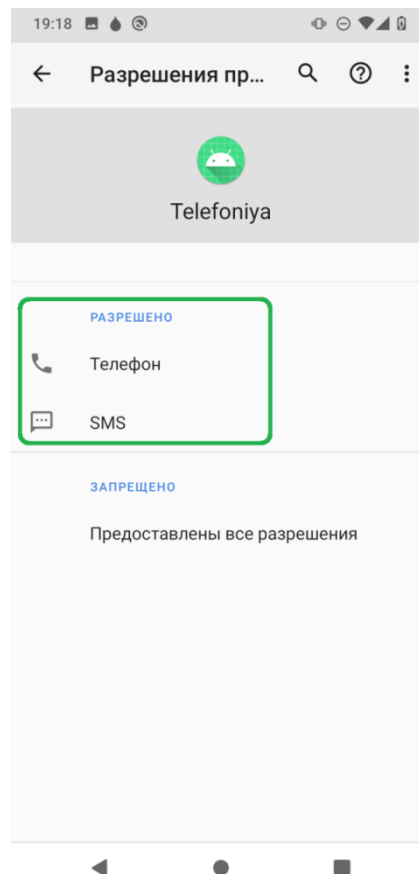


Рисунок 52 – Установка разрешений для приложения

Заполняем поле ввода номера и текст СМС сообщения, нажимаем на кнопку «*Отправить СМС*» и проверяем результат, рисунок 53.

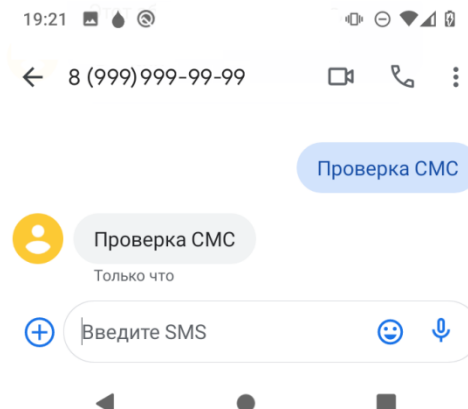


Рисунок 53 – Установка разрешений для приложения

1.11 Публикация мобильного приложения

В настоящее время существует большое количество Интернет-магазинов для публикации мобильных приложений. Некоторые из них: *Google Play*, *Samsung Galaxy Store*, *Amazon Appstore* и др. Основная часть из

этих магазинов требует денежное вознаграждение за публикацию мобильного приложения разработчика.

Многие Интернет-магазины используют *apk*-файлы для публикации, поэтому начнём публикацию именно с этого. В Android Studio выбираем пункт *Build->Generate Signed Bundle / APK* и выбираем следующий пункт *APK*, рисунок 54.



Рисунок 54 – Окно выбора пункта *APK*

На следующем диалоговом окне вводим имя ключа **.jks* и выбираем место для его сохранения на компьютере, рисунок 55. Для примера будем использовать приложение для работы с диалоговыми окнами из главы 1.4.

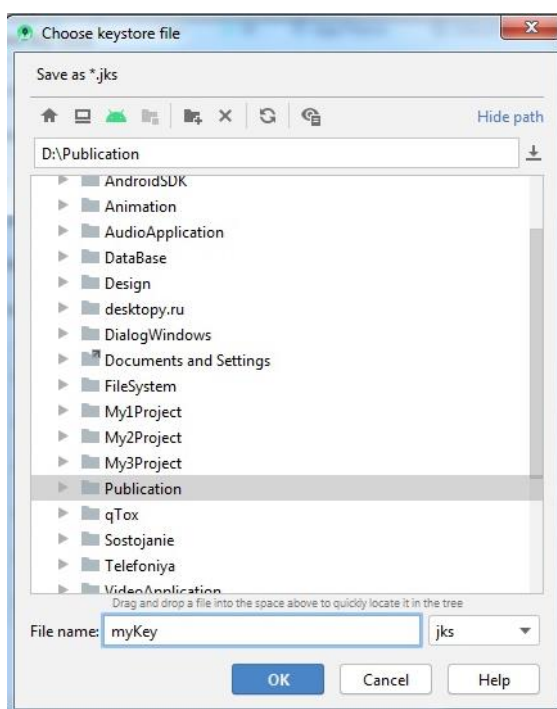


Рисунок 55 – Установка пути и имени файла ключа

В следующем диалоговом окне заполняем поля по своему усмотрению, однако обязательными являются *Password*, которые необходимо ввести повторно, рисунок 56.

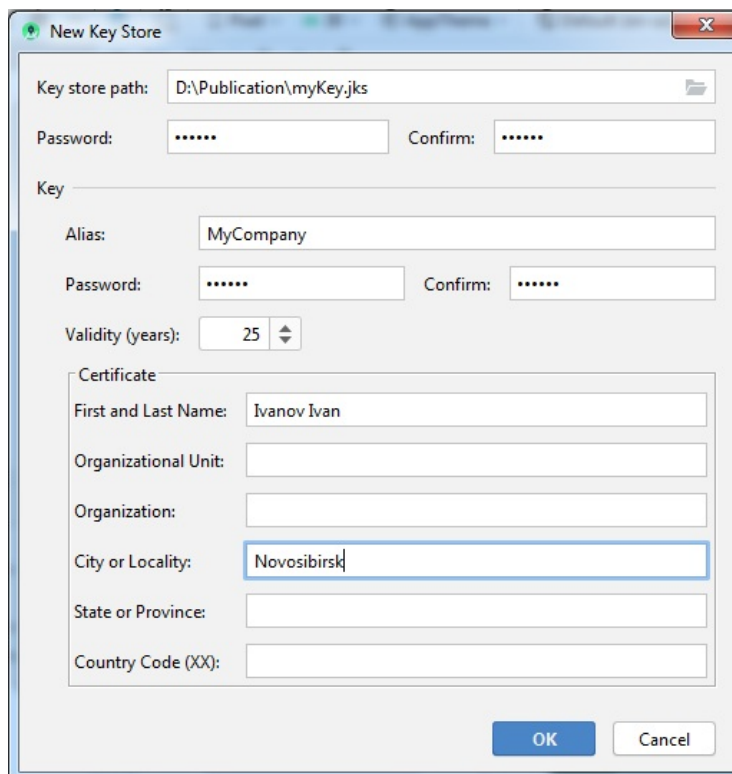


Рисунок 56 – Установка дополнительной информации

Далее осуществляется проверка введенных данных, где требуется нажать на кнопку *Next*, рисунок 57.

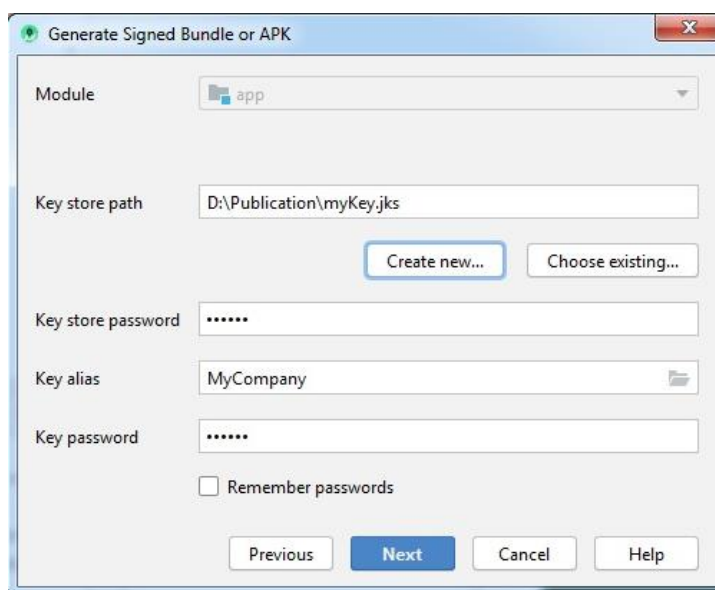


Рисунок 57 – Проверка установленных полей

Публикация приложения возможна только в версии *release*, рисунок 58.

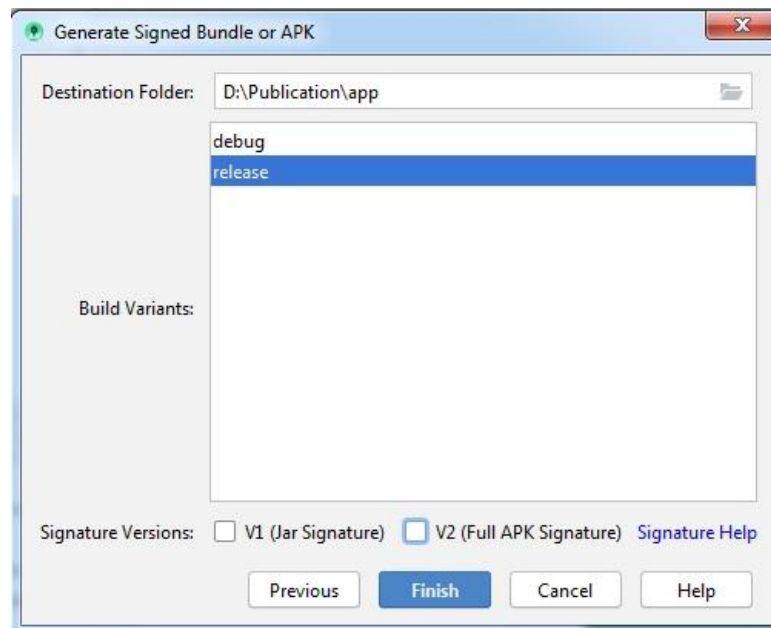


Рисунок 58 – Выбор версии *apk*-файла

Готовый *apk*-файл создаётся в папке *release*, рисунок 59.

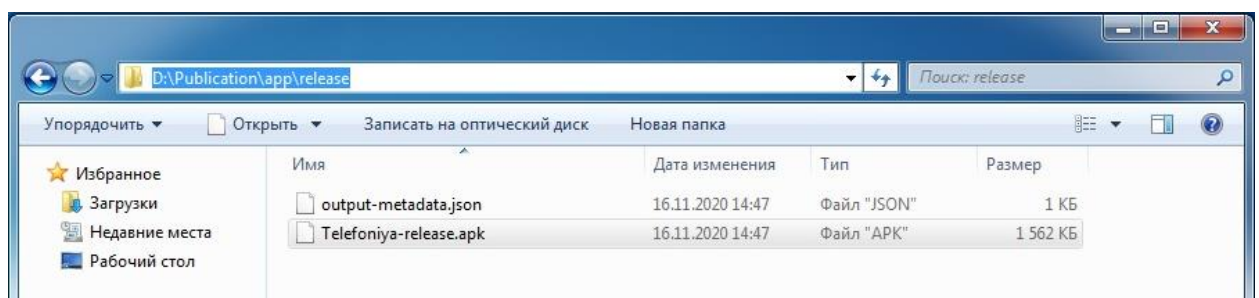


Рисунок 59 – Готовый *apk*-файл, сформированный Android Studio

На этом процесс создания *apk*-файла завершён, далее требуется выбрать Интернет-магазин для его размещения. Мы будем использовать бесплатный и простой в публикации *AmazonAppstore* [7]. Данный магазин доступен по ссылке developer.amazon.com/settings/console/home.

Выбираем пункт *Add a New App* → *Android*, рисунок 60.



Add a New App ▼



Android



Mobile Web

Рисунок 60 – Добавление своего приложения в AmazonAppstore

В следующем диалоговом окне устанавливаются: *App title/заголовок приложения*; *App SKU/Артикул приложения*; *App category/Категория приложения*; *Communication Features/Коммуникационные возможности*; *Customer support contact/Контактное лицо службы поддержки клиентов*, рисунок 61.

New App Submission

App title *	Telefoniya
App SKU ⓘ	Telefoniya-release
App category * ⓘ	Communication ▼
Category Refinements This will improve your app's searchability	
Communication Features	
<input type="checkbox"/> Group Calls	<input type="checkbox"/> Messaging
<input type="checkbox"/> Picture Messaging	<input type="checkbox"/> Video Calling
<input type="checkbox"/> Voicemail	<input checked="" type="checkbox"/> Phone Line Calling
	<input checked="" type="checkbox"/> Phone Line SMS
	<input type="checkbox"/> Video Messaging
	<input type="checkbox"/> Voice Calling
App will be listed in:	
Category Communication	Category Refinements Communication Features - Phone Line Calling, Phone Line SMS
Customer support contact	<input type="checkbox"/> Use my default support information
<div> Your default customer support information can not be shown as it was not provided at the time of registration. Update customer support information in settings.</div>	
Customer support email address *	example@mail.ru
Customer support phone	null

Рисунок 61 – Окно установки главных параметров мобильного приложения

На следующем этапе необходимо заполнить шесть вкладок, первая из которых *General Information/Главная информация*, рисунок 62.

The screenshot shows the 'General Information' tab of the Amazon App Store submission process. At the top, it indicates 'App version in progress' and '1/6 complete'. A 'Submit App' button is in the top right. Below the tab bar, the 'App title' is 'Telefoniya' and the 'App SKU' is 'Telefoniya-release'. The 'App Submission API Keys' section shows the 'App ID' and 'Release ID' with 'Copy' buttons. The 'App category' is 'Communication', with refinements 'Communication Features - Phone Line Calling, Phone Line SMS'. The 'Customer support contact' section includes fields for email address, phone, and website.

App version in progress
1/6 complete

App Submission Help

Submit App

General Information Availability & Pricing Description Images & Multimedia Content Rating APK Files

App title* Telefoniya

App SKU Telefoniya-release

App Submission API Keys

App ID amzn1.devportal.mobileapp.7137caea30d747699a35106a64bf0b35 Release ID amzn1.devportal.apprelease.2cf33f95a991483888c01d4b846e8340

App category* Communication

App will be listed in: Category Communication

Category Refinements Communication Features - Phone Line Calling, Phone Line SMS

Customer support contact

Customer support email address* example@mail.ru

Customer support phone 89619999999

Customer support website —

Edit

Рисунок 62 – Вкладка «Главная информация»

После заполнения первой вкладки необходимо заполнить вторую – *Availability & Pricing/Доступность и цена*, рисунок 63, где нужно выбрать страны, на территории которых планируется использовать мобильное приложение; платное/бесплатное и время начала публикации приложения.

App version in progress 1/6 complete [App Submission Help](#) [Submit App](#)

General Information Availability & Pricing Description Images & Multimedia Content Rating APK Files

Where would you like this app to be available? *

☐ In all countries and regions where Amazon sells apps ☒ Only in selected countries and regions

☐ Africa (0) + [Select countries/regions](#)

☐ Antarctica (0) + [Select countries/regions](#)

☐ Asia (0) + [Select countries/regions](#)

☒ Europe (1) + [Select countries/regions](#)

☐ North America (0) + [Select countries/regions](#)

☐ Oceania (0) + [Select countries/regions](#)

☐ South America (0) + [Select countries/regions](#)

Is your app free or paid? *

☒ Free ☐ Paid

When would you like publishing to start? Leave this field blank to start the publishing process as soon as your app has been approved. To defer the start of the publishing process to a future time and date, select it here. Your app will be available in the Appstore a few hours after the publishing process starts.

11/16/2020, 00:00

UTC +07:00 Asia/Novosibirsk [Clear All](#)

Рисунок 63 – Вкладка *Availability & Pricing*/Доступность и цена

На следующей вкладке *Description/Описание* необходимо заполнить поля: *Display title/Имя на экране*; *Short description/Краткое описание*; *Product feature bullets/Маркеры характеристик продукта*; *Keywords/Ключевые слова*, рисунок 64.

English (U.S.) *

Display title *
English (U.S.)

Telefoniya

Characters remaining: 190/200

Short description *
English (U.S.)
A brief description of your app, shown on mobile devices.

My test publication

Characters remaining: 1181/1200

Long description *
English (U.S.)
A lengthier description of your app, for the Appstore website.

My first trial publication of the app

Characters remaining: 3963/4000

Product feature bullets *
English (U.S.)
Up to 10 key features of your app, one feature per line. These features will appear on the Appstore website, formatted as a bulleted list.

- sms
- call

Keywords
English (U.S.)
Search terms used to increase the discoverability of your app. Use a comma or white space to separate your terms.

telefoniya, sms

English (U.S.) description will be displayed in all marketplaces if translations are not provided. [Add localized descriptions](#)

Рисунок 64 – Вкладка «*Description*»/Описание

На вкладке *Images & Multimedia/Изображения и мультимедиа* устанавливаются иконки приложения, а также скриншоты экрана, рисунок 65.

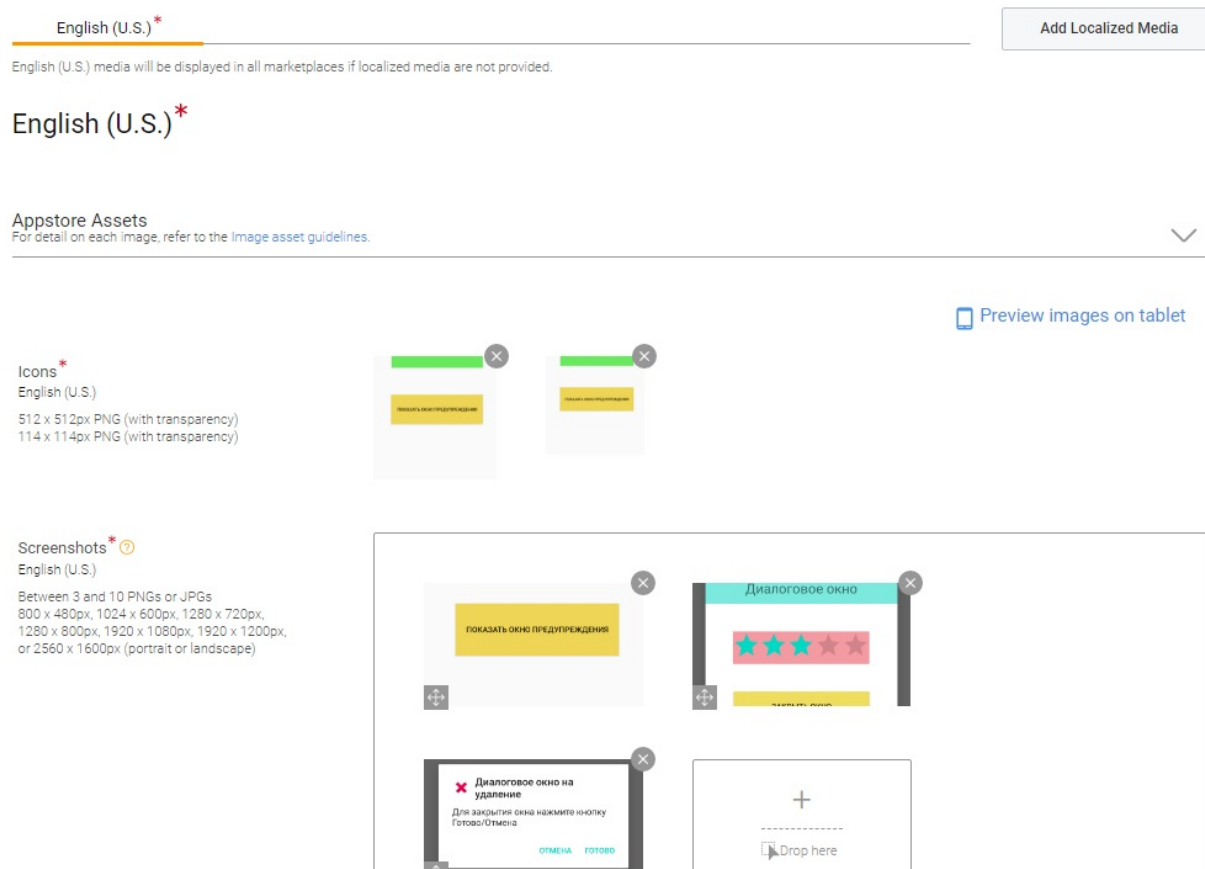


Рисунок 65 – Вкладка «*Images & Multimedia*» / *Изображения и мультимедиа*

На вкладке *Content Rating/Рейтинг контента* устанавливается рейтинг содержимого в мобильном приложении, рисунок 66.

General Information Availability & Pricing Description Images & Multimedia **Content Rating** APK Files

Content Rating

Moderate: Occurs once or rarely and is not fundamental to the overall purpose and/or intent of the app.
Strong: Occurs regularly and is fundamental to the overall purpose and/or intent of the app.
None: Does not occur in the app.

Subject Matter	None	Moderate	Strong
1. Violence* Realistic Violence	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
2. Cartoon Violence* Cartoon or Fantasy violence	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
3. Drugs* Alcohol, Tobacco, or Drug Use or References	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
4. Nudity* Nudity	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
5. Sex* Sexual and suggestive content	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
6. Intolerance* Any disparagement of race, creed, culture, or religion.	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
7. Profanity* Profanity or crude humor	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
8. Academic* This application is for educational purposes	<input type="radio"/> No	<input checked="" type="radio"/> Yes	

Рисунок 66 – Установка рейтинга контента публикуемого приложения

На последней вкладке *APK Files/APK-файлы* необходимо загрузить *APK-файл* и установить *Language Support/Язык поддержки*, рисунок 67.

App version in progress 5/6 complete App Submission Help [Submit App](#)

General Information Availability & Pricing Description Images & Multimedia **Content Rating** **APK Files**

Add APK

Appstore Certificate Hashes

Apply Amazon DRM? ☐ Yes ☒ No

APK File*
Upload your APK files one by one.
As part of the ingestion process, Amazon removes your developer signature and applies an Amazon signature. This signature is unique to you, does not change, and is the same for all apps in your account.

+

Drop APK here

Uploaded APKs

Please select appropriate DRM option for replaces/uploaded APK's from above

APK1 Edit	Ver Code : 1	0+ Supported Devices Edit Manifest	Replace APK
Telefoniya-release.apk			

Add Other Details

Language Support*

<input type="checkbox"/> Arabic	<input type="checkbox"/> Chinese	<input type="checkbox"/> Cornish	<input type="checkbox"/> Czech	<input type="checkbox"/> Dutch
<input type="checkbox"/> English	<input type="checkbox"/> French	<input type="checkbox"/> German	<input type="checkbox"/> Greek	<input type="checkbox"/> Hebrew
<input type="checkbox"/> Hindi	<input type="checkbox"/> Italian	<input type="checkbox"/> Japanese	<input type="checkbox"/> Kazakh	<input type="checkbox"/> Korean
<input type="checkbox"/> Polish	<input type="checkbox"/> Portuguese	<input checked="" type="checkbox"/> Russian	<input type="checkbox"/> Spanish	<input type="checkbox"/> Tagalog
<input type="checkbox"/> Vietnamese				

Рисунок 67 – Настройка вкладки APK-файлы

После регистрации приложения на *AmazonAppstore* можно просмотреть сообщение о модерации размещаемого разработчиком приложения, рисунок 68.

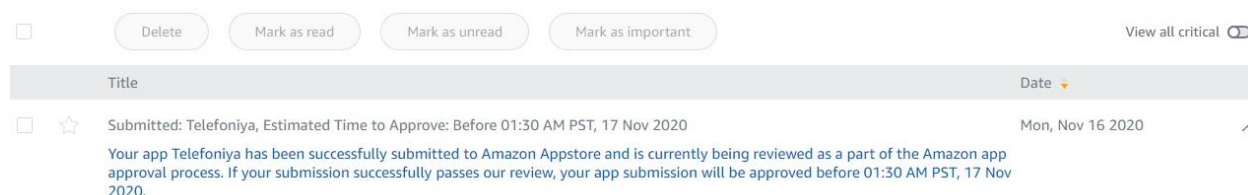


Рисунок 68 – Модерация приложения на сайте *AmazonAppstore*

Через некоторое время на почту и аккаунт *Amazon* придёт сообщение о публикации или отказе в размещении в Интернет-магазине. На рисунке 69 не пройден третий пункт – *Functionality Validation/Валидация функциональности*, что является очевидным для нашего приложения из главы 1.4.

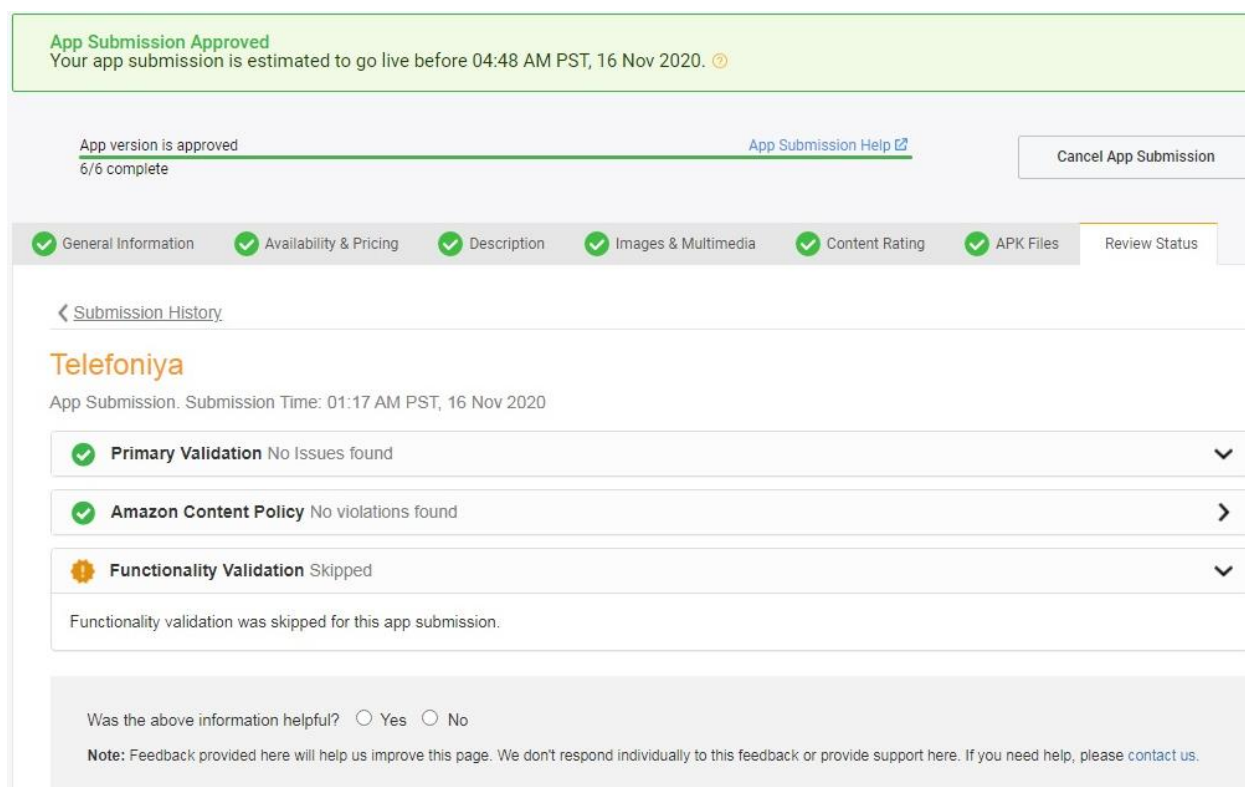


Рисунок 69 – Статус подачи заявки

ГЛАВА 2 ОБЩИЕ ПОДХОДЫ К РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 Разработка технической документации на программное обеспечение

Разработка программной документации на программные продукты является неотъемлемой частью в создании конечного программного продукта. С помощью неё описываются ответы на такие вопросы как: что это за программное обеспечение; для кого оно предназначено; каковы его функции; какие требования к аппаратному и программному обеспечению имеются; сообщения об ошибках ПО; каковы требования к квалификации пользователя. Отдельно создаются руководства для программиста, пользователя, системного администратора и т.д. В России используются государственные стандарты, которые включены в единую систему программной документации.

Единая система программной документации (ЕСПД) — комплекс государственных стандартов Российской Федерации, устанавливающих взаимосвязанные правила разработки, оформления и обращения программ и программной документации.

В стандартах ЕСПД устанавливают требования, регламентирующие разработку, сопровождение, изготовление и эксплуатацию программ, что обеспечивает возможность:

1. унификации программных изделий для взаимного обмена программами и применения ранее разработанных программ в новых разработках;
2. снижения трудоемкости и повышения эффективности разработки, сопровождения, изготовления и эксплуатации программных изделий;
3. сопровождение программы включает анализ функционирования, развитие и совершенствование программы, а также внесение изменений в неё с целью устранения ошибок.
4. автоматизации изготовления и хранения программной документации.

ЕСПД представляет собой набор ГОСТов, в настоящее время её применение на территории РФ носит только рекомендательный характер, поэтому ЕСПД применяется на добровольной основе (если иное не предусмотрено). В таблице 2 приведены действующие ГОСТы на программную документацию в Российской Федерации.

Таблица 2 – Действующие ГОСТы на ЕСПД в России

Номер п/п	ГОСТ	Название
1	19.001-77	Общие положения
2	19.005-85	Р-схемы алгоритмов и программ
3	19.101.77	Виды программ и программных документов
4	19.102.77	Стадии разработки
5	19.103.77	Обозначение программ и программных документов
6	19.104.78	Основные надписи
7	19.105.78	Общие требования к программным документам
8	19.106.78	Требования к программным документам, выполненным печатным способом
9	19.201.78	Техническое задание. Требования к содержанию и оформлению
10	19.202.78	Спецификация. Требования к содержанию и оформлению
11	19.301.79	Программа и методика испытаний. Требования к содержанию и оформлению
12	19.401.78	Текст программы. Требования к содержанию и оформлению
13	19.402.78	Описание программы
14	19.403.79	Ведомость держателей подлинников
15	19.404.79	Пояснительная записка. Требования к содержанию и оформлению
16	19.501.78	Формуляр. Требования к содержанию и оформлению
17	19.502.78	Описание применения. Требования к содержанию и оформлению
18	19.503.79	Руководство системного программиста. Требования к содержанию и оформлению
19	19.504.79	Руководство программиста. Требования к содержанию и оформлению
20	19.505.79	Руководство оператора. Требования к содержанию и оформлению
21	19.506.79	Описание языка. Требования к содержанию и оформлению
22	19.507.79	Ведомость эксплуатационных документов
23	19.508.79	Руководство по техническому обслуживанию. Требования к содержанию и оформлению
24	19.601.78	Общие правила дублирования, учёта и хранения
25	19.602.78	Правила дублирования, учёта и хранения программных документов, выполненных печатным способом
26	19.603.78	Общие правила внесения изменений

27	19.604.78	Правила внесения изменений в программные документы, выполненные печатным способом
28	19.701.90	Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения
29	19.781.90	Термины и определения

2.2 Анализ требований к мобильному ПО

Требования к программному обеспечению – совокупность утверждений относительно атрибутов, функций, свойств или качество программной системы, подлежащей реализации. Требования могут выражаться в виде текстовых утверждений и графических моделей, например, диаграмм (*UML*, *DFD*, *ER* и др).

В классическом техническом подходе совокупность требований используется на стадии проектирования ПО. Требования также используются в процессе проверки ПО, так как тесты основываются на определённых требованиях.

Перед этапом разработки требований может быть технико-экономическое обоснование или концептуальная фаза анализа проекта. Фаза разработки требований может быть разбита на выявление требований (сбор, понимание, рассмотрение и выяснение потребностей заинтересованных лиц), анализ (проверка целостности и законченности), спецификация (документирование требований) и проверка правильности.

Основными средствами анализа требований и методами их выявления являются:

1. интервью, опросы, анкетирование;
2. мозговой штурм, семинар, конференция;
3. наблюдение за производственной деятельностью;
4. анализ нормативной документации;
5. анализ моделей деятельности;
6. анализ конкурентных продуктов на рынке;
7. анализ статистики использования предыдущих версий системы.

Чтобы более правильно составить функционал и определить весь объём работы следует обратиться к ГОСТу 19.102-77 [8], основные положения которого вынесены в таблицу 3.

Таблица 3 – Стадии разработки, этапы и содержание работ

Стадии разработки	Этапы работ	Содержание работ
1. Техническое задание.	Обоснование необходимости разработки программы.	Постановка задачи. Сбор исходных материалов. Выбор и обоснование критериев эффективности и качества разрабатываемой программы. Обоснование необходимости проведения научно-исследовательских работ.
	Научно-исследовательские работы.	Определение структуры входных и выходных данных. Предварительный выбор методов решения задач. Обоснование целесообразности применения ранее разработанных программ. Определение требований к техническим средствам. Обоснование принципиальной возможности решения поставленной задачи.
	Разработка и утверждение технического задания.	Определение требований к программе. Разработка технико-экономического обоснования разработки программы. Определение стадий, этапов и сроков разработки программы

		и документации на неё. Выбор языков программирования. Определение необходимости проведения научно-исследовательских работ на последующих этапах. Согласование и утверждение технического задания.
2. Эскизный проект.	Разработка эскизного проекта.	Предварительная разработка структуры входных и выходных данных. Уточнение методов решения задачи. Разработка общего описания алгоритма решения задачи. Разработка технико-экономического обоснования
	Утверждение эскизного проекта.	Разработка пояснительной записки. Согласование и утверждение эскизного проекта.
3. Технический проект.	Разработка технического проекта.	Уточнение структуры входных и выходных данных. Разработка алгоритма решения задачи. Определение формы представления входных и выходных данных. Определение семантики и синтаксиса языка. Разработка структуры программы. Окончательное определение конфигурации

		технических средств.
	Утверждение технического проекта.	Разработка плана мероприятий по разработке и внедрению программ. Разработка пояснительной записки. Согласование и утверждение технического проекта.
4. Рабочий проект.	Разработка программы.	Программирование и отладка программы.
	Разработка программной документации.	Разработка программных документов в соответствии с ГОСТ 19.101-77.
	Испытания программы.	Разработка, согласование и утверждение порядка и методики испытаний. Проведение предварительных государственных, межведомственных, приёмо-сдаточных и других видов испытаний. Корректировка программы и программной документации по результатам испытаний.
5. Внедрение.	Подготовка и передача программы.	Подготовка и передача программы и программной документации для сопровождения и (или) изготовления. Оформление и утверждение акта о передаче программы на сопровождение и (или) изготовление.

		Передача программы в фонд алгоритмов и программ.
--	--	--

В данной таблице показаны все этапы при разработке ПО, а также расписано содержание каждого из этапа.

Техническое задание поможет найти точки соприкосновения между разработчиком и заказчиком приложения, поэтому в соответствии с ГОСТом 19.201-78 [9] техническое задание должно содержать следующие разделы:

- введение;
- основания для разработки;
- назначение разработки;
- требования к программе или программному изделию;
- требования к программной документации;
- технико-экономические показатели;
- стадии и этапы разработки;
- порядок контроля и приёмки;
- в техническое задание допускается включать приложения.

ГОСТы в п. 2.1 относятся в целом к разработке приложений, как для мобильных устройств, так и для персональных компьютеров, однако требования для мобильных устройств имеют некоторую специфику по сравнению с ПО для персональных компьютеров. Ниже перечислены некоторые основные требования к мобильному приложению:

1. кроссплатформенность;
2. автономность;
3. совместимость, требования к оборудованию;
4. совместимость, требования к ПО;
5. дружелюбный интерфейс;
6. продуманный дизайн;
7. обратная связь, поддержка;
8. обновление;
9. отказоустойчивость;
10. безопасность и надёжность;
11. защита от случайных нажатий;
12. требования к ресурсам и их использованию (скорость работы);
13. требования к эксплуатации и персоналу;

Данный перечень не является исчерпывающим и может изменяться из-за специфики конкретного мобильного приложения.

ЗАКЛЮЧЕНИЕ

В методическом пособии были раскрыты реализации некоторых основных функций при создании современных мобильных приложений. К ним относятся: принципы разработки графического интерфейса и создания успешного приложения; реализация функции телефонии и использование баз данных, а также публикация в Интернет-магазине. Во второй главе особый упор делался на то, чтобы студенты осознали ценность и важность технической документации, которая может позволить избежать недопонимания между заказчиком и разработчиком, кем они будут являться. На практике это позволит избежать конфликтной ситуации, траты времени на переработку программного обеспечения и увеличению сроков ввода в эксплуатацию.

Список используемой литературы

1. URL: <https://www.statista.com> (дата обращения 18.01.2022).
2. URL: <https://www.statista.com/statistics/266211/distribution-of-free-and-paid-android-apps/> (дата обращения 18.01.2022).
3. URL: <https://support.google.com/googleplay/android-developer/table/3541286> (дата обращения 21.01.2022).
4. Филлипс Б., Стюарт К., Марсикано К. Android. Программирование для профессионалов. 3-е изд. – СПб.: Питер, 2017. – 688 с.
5. Колисниченко Д.Н. Программирование для Android 5. Самоучитель. – СПб.: БХВ-Петербург, 2015. – 303 с.
6. URL: <https://metanit.com/java/android/> (дата обращения 21.01.2022).
7. URL: <https://developer.amazon.com/apps-and-games/console/app/new.html> (дата обращения 01.02.2022).
8. ГОСТ 19.102-77 Стадии разработки
9. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению

Для студентов специальности 09.02.03 Программирование в компьютерных системах по дисциплине *«Программирование мобильных приложений»* и специальности 09.02.07 Информационные системы и программирование по дисциплине *«Разработка мобильных приложений»*

Составитель: к.т.н., Жирнов Анатолий Алексеевич, ассистент кафедры информатики ВКИ НГУ. 05.04.2022г.