Курс по Android разработке

Яборов Андрей Владимирович

avyaborov@gravity-group.ru

Марквирер Владлена Дмитриевна

vdmarkvirer@hse.ru

НИУ ВШЭ 2022

GRAVITY group

Разработка под Android

Урок 4

- Ресурсы
- Preference
- Permissions
- Датчики
- Base проект

GRAVITY group

Ресурсы

Ресурс в приложении Android представляет собой файл, например, файл разметки интерфейса или некоторое значение, например, простую строку. То есть ресурсы представляют собой и файлы разметки, и отдельные строки, и звуковые файлы, файлы изображений и т.д. Все ресурсы находятся в проекте в каталоге *res*. Для различных типов ресурсов, определенных в проекте, в каталоге *res* создаются подкаталоги. Поддерживаемые подкаталоги:

- **animator/**: xml-файлы, определяющие анимацию свойств
- **anim/**: xml-файлы, определяющие tween-анимацию (анимацию различных свойств объекта)
- **color/**: xml-файлы, определяющие список цветов
- **drawable/**: Графические файлы (*.png*, *.jpg*, *.gif*)
- **mipmap/**: Графические файлы, используемые для иконок приложения под различные разрешения экранов
- **layout/**: xml-файлы, определяющие пользовательский интерфейс приложения
- **menu/**: xml-файлы, определяющие меню приложения
- **raw/**: различные файлы, которые сохраняются в исходном виде
- **values/**: xml-файлы, которые содержат различные используемые в приложении значения, например, ресурсы строк
- **xml/**: Произвольные xml-файлы

Применение ресурсов

Существует два способа доступа к ресурсам: в файле исходного кода и в файле xml.

Ресурсы

Ссылка на ресурсы в коде

Тип ресурса в данной записи ссылается на одно из пространств (внутренних классов), определенных в файле R.java, которые имеют соответствующие им типы в xml:

- R.drawable (ему соответствует тип в xml drawable)
- R.id (id)
- R.layout (layout)
- R.string (string)
- R.attr (attr)
- R.plural (plurals)
- R.array (string-array)

Например, для установки ресурса activitv main.xml в качестве графического интерфейса в коде MainActivity в методе onCreate() есть такая строка:

```
setContentView(R.layout.activity_main);
```

Через выражение R.layout.activity_main мы и ссылаемся на ресурс activity_main.xml, где layout - тип ресурса, а activity_main - имя ресурса. Подобным образом мы можем получать другие ресурсы. Например, в файле *res/values/strings.xml* определен ресурс app_name:

```
<resources>

    <string name="app_name">ViewsApplication</string>

</resources>
```

Этот ресурс ссылается на строку. Чтобы получить ссылку на данный ресурс в коде java, мы можем использовать выражение R.string.app_name

Ресурсы

Доступ в файле xml

Нередко возникает необходимость ссылаться на ресурс в файле xml, например, в файле, который определяет визуальный интерфейс, к примеру, в activity_main.xml. Ссылки на ресурсы в файлах xml имеют следующую формализованную форму: @[имя_пакета:]тип_ресурса/имя_ресурса

- имя_пакета представляет имя пакета, в котором ресурс находится (указывать необязательно, если ресурс находится в том же пакете)
- тип_ресурса представляет подкласс, определенный в классе R для типа ресурса
- имя_ресурса имя файла ресурса без расширения или значение атрибута android:name в XML-элементе (для простых значений).

Например, мы хотим вывести в элемент TextView строку, которая определена в виде ресурса в файле strings.xml:

```
<TextView
        android:id="@+id/welcome"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/app_name" />
```

В данном случае свойство text в качестве значения будет получать значение строкового ресурса app_name.

Ресурсы

Ресурсы строк

Ресурсы строк - один из важных компонентов приложения. XML-файлы, представляющие собой ресурсы строк, находятся в проекте в папке *res/values*. По умолчанию ресурсы строк находятся в файле *strings.xml*, который может выглядеть следующим образом:

```
<resources>

    <string name="app_name">ViewsApplication</string>

</resources>
```

В самом простом виде этот файл определяет один ресурс "app_name", который устанавливает название приложения. Но естественно мы можем определить любые строковые ресурсы. Каждый отдельный ресурс определяется с помощью элемента **string**, а его атрибут name содержит название ресурса. Для ресурсов строк в классе R определяется внутренний класс static final class string. Этот класс используется в качестве пространства для хранения идентификаторов ресурсов строк:

```
public static final class string {

    public static final int app_name=0x7f040000;

}
```

Константа app_name имеет тип не String, а int, а ее значение - числовой идентификатор ресурса. Затем в приложении в файлах кода мы можем ссылаться на эти ресурсы: R.string.app_name. А ОС Android сама сопоставит данные числовые идентификаторы с соответствующими ресурсами строк. Например: *String application_name = getResources().getString(R.string.app_name);*

Ресурсы

Ресурсы dimension

Определение размеров должно находиться в папке **res/values** в файле с любым произвольным именем. Общий синтаксис определения ресурса следующий:

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

<resources>

    <dimen name="имя_ресурса">используемый_размер</dimen>

</resources>
```

Как и другие ресурсы, ресурс dimension определяется в корневом элементе <resources>. Тег <dimen> обозначает ресурс и в качестве значения принимает некоторое значение размера в одной из принятых единиц измерения (dp, sp, pt, px, mm, in). Пример файла с данным типом ресурса:

```
<resources>

    <dimen name="activity_horizontal_margin">16dp</dimen>

    <dimen name="activity_vertical_margin">16dp</dimen>

    <dimen name="text_size">16sp</dimen>

</resources>
```

Ресурсы

Ресурсы color

В приложении Android также можно определять ресурсы цветов (Color). Они должны храниться в файле по пути *res/values* и также, как и ресурсы строк, заключены в тег <resources>. Так, по умолчанию при создании самого простого проекта в папку *res/values* добавляется файл **colors.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

Цвет определяется с помощью элемента <color>. Его атрибут name устанавливает название цвета, которое будет использоваться в приложении, а шестнадцатеричное число - значение цвета. Для задания цветовых ресурсов можно использовать следующие форматы:

- #RGB (#F00 - 12-битное значение)
- #ARGB (#8F00 - 12-битное значение с добавлением альфа-канала)
- #RRGGBB (#FF00FF - 24-битное значение)
- #AARRGGBB (#80FF00FF - 24-битное значение с добавлением альфа-канала)

Preference

SharedPreferences — постоянное хранилище на платформе Android, используемое приложениями для хранения своих настроек. Это хранилище является относительно постоянным, пользователь может зайти в настройки приложения и очистить данные приложения, тем самым очистив все данные в хранилище.

Для работы с данными постоянного хранилища нам понадобится экземпляр класса SharedPreferences, который можно получить у любого объекта, унаследованного от класса *android.content.Context* (например, Activity или Service).

У объектов этих классов (унаследованных от Context) есть метод getSharedPreferences, который принимает 2 параметра:

name — выбранный файл настроек. Если файл настроек с таким именем не существует, он будет создан при вызове метода edit() и фиксировании изменений с помощью метода commit() или apply()..

mode — режим работы. Возможные значения:

- MODE_PRIVATE — используется в большинстве случаев для приватного доступа к данным приложением-владельцем
- MODE_WORLD_READABLE — только для чтения
- MODE_WORLD_WRITEABLE — только записи
- MODE_MULTI_PROCESS — несколько процессов совместно используют один файл SharedPreferences.

Preference

Чтобы получить значение необходимой переменной, используйте следующие методы объекта **SharedPreferences**:

- **getBoolean**(String key, boolean defValue)
- **getFloat**(String key, float defValue)
- **getInt**(String key, int defValue)
- **getLong**(String key, long defValue)
- **getString**(String key, String defValue)
- **getStringSet**(String key, Set defValues)

Чтобы записать значение переменной необходимо:

1. получить объект **SharedPreferences.Editor** выполнив метод edit() объекта класса **SharedPreferences**
2. записать значение с помощью методов:
   - **putBoolean**(String key, boolean value),
   - **putFloat**(String key, float value),
   - **putInt**(String key, int value),
   - **putLong**(String key, long value),
   - **putString**(String key, String value),
   - **putStringSet**(String key, Set values)
3. выполнить метод **commit()** или **apply**()

GRAVITY group

Preference

```java
public class PreferenceManager {

    private final static String PREFERENCE_FILE = "org.hse.android.file";

    private final SharedPreferences sharedPref;

    public PreferenceManager(Context context) {
        sharedPref = context.getSharedPreferences(PREFERENCE_FILE, Context.MODE_PRIVATE);
    }

    ////////////////////////////////

    private void saveValue(String key, String value) {
        SharedPreferences.Editor editor = sharedPref.edit();
        editor.putString(key, value);
        editor.apply();
    }

    private String getValue(String key, String defaultValue) {
        return sharedPref.getString(key, defaultValue);
    }
```

GRAVITY group

Permissions

Каждое приложение Android, установленное на устройстве, работает в собственной "песочнице" (изолированной программной среде):

- операционная система Android представляет собой многопользовательскую систему Linux, в которой каждое приложение является отдельным пользователем;
- по умолчанию система назначает каждому приложению уникальный идентификатор пользователя Linux (этот идентификатор используется только системой и неизвестен приложению); система устанавливает полномочия для всех файлов в приложении, с тем чтобы доступ к ним был разрешен только пользователю с идентификатором, назначенным этому приложению;
- у каждого процесса имеется собственная виртуальная машина (ВМ), так что код приложения выполняется изолированно от других приложений;
- по умолчанию каждое приложение выполняется в собственном процессе Linux. Android запускает процесс, когда требуется выполнить какой-либо компонент приложения, а затем завершает процесс, когда он больше не нужен либо когда системе требуется освободить память для других приложений.

Таким образом система Android реализует *принцип предоставления минимальных прав*. То есть каждое приложение по умолчанию имеет доступ только к тем компонентам, которые ему необходимы для работы, и ни к каким другим. Благодаря этому формируется исключительно безопасная среда, в которой приложение не имеет доступа к недозволенным областям системы.

Permissions

В Android есть очень хороший защитный механизм — система разрешений для приложений. По сути, это набор действий, которые система разрешает приложению выполнять. Дело в том, что по умолчанию все приложения в Android работают в изолированной среде — так называемой «песочнице». И для того чтобы сделать что-либо с чем-то, так сказать, общественным, им надо получить разрешение.

Разрешения эти разделены на несколько категорий, но нас интересуют только две из них — «Обычные» и «Опасные». В группу «Обычные» входят такие вещи, как доступ в Интернет, создание ярлыков, подключение по Bluetooth и так далее. Эти разрешения выдаются приложениям без обязательного согласия пользователя, то есть система вас ни о чем не спрашивает.

А вот для того, чтобы получить одно из «опасных» разрешений, приложение обязательно должно спросить владельца устройства, согласен ли он его выдать.

Permissions

Опасные разрешения

В категорию «Опасные» входят девять групп разрешений, которые так или иначе связаны с безопасностью данных пользователя. В свою очередь, каждая из групп содержит несколько разрешений, которые может запрашивать приложение.

Если одно из разрешений в данной группе пользователь уже одобрил, все остальные разрешения из той же группы приложение получит автоматически — без нового запроса пользователю. Например, если приложение уже успело запросить и получить разрешение на чтение SMS, то впоследствии оно автоматически получит разрешение и на отправку SMS, и на прием MMS, и на все остальные разрешения из данной группы.

Для указания того, что приложение использует какое-либо разрешение, его необходимо указать в манифесте при помощи тега **<uses-permission>.** Например, для использования BlueTooth могут понадобиться следующие разрешения:

Base проект

Permissions

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.example.cool_app">

  <uses-permission android:name="android.permission.BLUETOOTH"/>
  <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>

  <application ...>
        ...
  </application>
</manifest>
```

GRAVITY group

Permissions

Дальнейшая работа с разрешениями зависит от того, к какой категории опасности они относятся. Например, все стандартные разрешения выдаются автоматически и нет необходимости в дальнейшем запрашивать эти разрешения в процессе работы, либо проверять их наличие.

До 6 версии андроид приложения не требовали подтверждения прав дополнительно, т.е. На момент установки проверялись права и дальше приложение имело доступ ко всем запрошенным правам. Начиная с 6 версии это момент изменился. Теперь с опасными разрешениями ситуация обстоит иначе: для работы с функциями Android, которые требуют наличие опасного разрешения, необходимо:

- Запрашивать разрешение при первом использовании опасного функционала;
- Проверять наличие разрешения при каждом использовании опасного функционала, поскольку опасные разрешения можно в любой момент отозвать в настройках системы.

Перед каждым выполнением опасной операции необходимо проверять наличие выданных опасных разрешений, так как пользователь мог их отозвать после установки приложения

Permissions

Основные опасные разрешения:

- Календарь
- Камера
- Контакты
- Местоположение
- Микрофон
- Телефон
- Сенсоры
- SMS
- Память

Особые права

Помимо разрешений, которые входят в категорию «Опасные», в Android есть еще несколько прав приложений, о которых стоит знать:

Приложение для работы с SMS по умолчанию

Permissions

Специальные возможности (Accessibility)

Наличие этих прав в приложении позволяет ему упростить использование приложения или устройства для пользователей с ограничениями, такими как слабое зрение или проблемы со слухом.

Права на отображение своего окна поверх других приложений

Права администратора устройства

В Android 8 таких настроек стало гораздо больше. Добавилось

Разрешения, которые настраиваются в списке «Разрешения приложений» (App permissions)

В этот список входят разрешения, позволяющие приложениям получить доступ к хранящимся в смартфоне личным данным его владельца — контактам, истории звонков, коротким сообщениям, фотографиям и так далее, а также тем встроенным устройствам, которые позволяют личные данные получить и записать — камере, микрофону, телефону и GPS-приемнику.

Разрешения, которые настраиваются в списке «Специальный доступ» (Special app access):

Экономия заряда батареи, Приложения администратора устройства, Установка неизвестных приложений, Доступ к функции «Не беспокоить» и др.

Base проект

Permissions

```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.hse.android">

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

    <uses-feature
        android:name="android.hardware.camera"
        android:required="true" />
```

```java
public void checkPermission() {
    int permissionCheck = ActivityCompat.checkSelfPermission(
            context: this, PERMISSION);
    if (permissionCheck != PackageManager.PERMISSION_GRANTED) {
        if (ActivityCompat.shouldShowRequestPermissionRationale( activity: this,
                PERMISSION)) {
            showExplanation("Нужно предоставить права",
                    "Для снятия фото нужно предоставить права на фото", PERMISSION, REQUEST_PERMISSION_CODE);
        } else {
            requestPermission(PERMISSION, REQUEST_PERMISSION_CODE);
        }
    } else {
        dispatchTakePictureIntent();
    }
}
```

GRAVITY group

Base проект

Permissions

```
private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        //Create a file to store the image
        File photoFile = null;
        try {
            photoFile = createImageFile();
        } catch (IOException ex) {
            Log.e(TAG,  msg: "Create file", ex);
        }
        if (photoFile != null) {
            Uri photoURI = FileProvider.getUriForFile( context: this,  authority: BuildConfig.APPLICATION_ID + ".provider", photoFile);
            takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT,
                    photoURI);
            try {
                startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
            } catch (ActivityNotFoundException e) {
                Log.e(TAG,  msg: "Start activity", e);
            }
        }
    }
}
```

```
<provider
    android:name="androidx.core.content.FileProvider"
    android:authorities="${applicationId}.provider"
    android:exported="false"
    android:grantUriPermissions="true">
    <meta-data
        android:name="android.support.FILE_PROVIDER_PATHS"
        android:resource="@xml/file_paths" />
</provider>
```

GRAVITY group

Датчики

Большинство устройств на базе Android имеют встроенные датчики, которые измеряют движение, ориентацию и различные условия окружающей среды. Эти датчики могут предоставлять необработанные данные с высокой точностью и полезны, если вы хотите отслеживать трехмерное перемещение или позиционирование устройства, или если вы хотите отслеживать изменения в окружающей среде рядом с устройством.

Например, игра может отслеживать показания датчика силы тяжести устройства, чтобы делать выводы о сложных жестах и движениях пользователя, таких как наклон, встряхивание, вращение или качание. Аналогичным образом погодное приложение может использовать датчик температуры устройства и датчик влажности для расчета и сообщения о точке росы, или приложение для путешествий может использовать датчик геомагнитного поля и акселерометр для определения направления по компасу.

Датчики

Платформа Android поддерживает три категории датчиков:

- **Датчики движения**
  Эти датчики измеряют силы ускорения и вращательные силы по трем осям. В эту категорию входят акселерометры, датчики силы тяжести, гироскопы и датчики вектора вращения.
- **Датчики окружающей среды**
  Эти датчики измеряют различные параметры окружающей среды, такие как температуру и давление окружающего воздуха, освещенность и влажность. В эту категорию входят барометры, фотометры и термометры.
- **Датчики положения**
  Эти датчики измеряют физическое положение устройства. В эту категорию входят датчики ориентации и магнитометры.

Table 1. Sensor types supported by the Android platform.

Sensor	Type	Description	Common Uses
`TYPE_ACCELEROMETER`	Hardware	Measures the acceleration force in m/s^2 that is applied to a device on all three physical axes (x, y, and z), including the force of gravity.	Motion detection (shake, tilt, etc.).
`TYPE_AMBIENT_TEMPERATURE`	Hardware	Measures the ambient room temperature in degrees Celsius (°C). See note below.	Monitoring air temperatures.
`TYPE_GRAVITY`	Software or Hardware	Measures the force of gravity in m/s^2 that is applied to a device on all three physical axes (x, y, z).	Motion detection (shake, tilt, etc.).
`TYPE_GYROSCOPE`	Hardware	Measures a device's rate of rotation in rad/s around each of the three physical axes (x, y, and z).	Rotation detection (spin, turn, etc.).
`TYPE_LIGHT`	Hardware	Measures the ambient light level (illumination) in lx.	Controlling screen brightness.
`TYPE_LINEAR_ACCELERATION`	Software or Hardware	Measures the acceleration force in m/s^2 that is applied to a device on all three physical axes (x, y, and z), excluding the force of gravity.	Monitoring acceleration along a single axis.
`TYPE_MAGNETIC_FIELD`	Hardware	Measures the ambient geomagnetic field for all three physical axes (x, y, z) in μT.	Creating a compass.
`TYPE_ORIENTATION`	Software	Measures degrees of rotation that a device makes around all three physical axes (x, y, z). As of API level 3 you can obtain the inclination matrix and rotation matrix for a device by using the gravity sensor and the geomagnetic field sensor in conjunction with the `getRotationMatrix()` method.	Determining device position.
`TYPE_PRESSURE`	Hardware	Measures the ambient air pressure in hPa or mbar.	Monitoring air pressure changes.
`TYPE_PROXIMITY`	Hardware	Measures the proximity of an object in cm relative to the view screen of a device. This sensor is typically used to determine whether a handset is being held up to a person's ear.	Phone position during a call.
`TYPE_`	Hardware	Measures the relative ambient humidity in percent (%)	Monitoring

https://developer.android.com/guide/topics/sensors/sensors_overview

GRAVITY group

Датчики

За работу с сенсорами отвечает класс SensorManager, содержащий несколько констант, которые характеризуют различные аспекты системы датчиков Android, в том числе:

- **Тип датчика**
  Ориентация, акселерометр, свет, магнитное поле, близость, температура и т.д.
- **Частота измерений**
  Максимальная, для игр, обычная, для пользовательского интерфейса. Когда приложение запрашивает конкретное значение частоты отсчётов, с точки зрения сенсорной подсистемы это лишь рекомендация. Никакой гарантии, что измерения будут производиться с указанной частотой, нет.
- **Точность**
  Высокая, низкая, средняя, ненадёжные данные.

```kotlin
// Kotlin
private lateinit var sensorManager: SensorManager

sensorManager = getSystemService(SENSOR_SERVICE) as SensorManager
```

```java
// Java
private SensorManager sensorManager;

sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
```

Устройство может включать в себя несколько реализаций одного и того же типа датчиков. Чтобы найти реализацию, используемую по умолчанию, вызовите метод getDefaultSensor()

https://developer.android.com/reference/android/hardware/SensorManager

Датчики

Устройство может включать в себя несколько реализаций одного и того же типа датчиков. Чтобы найти реализацию, используемую по умолчанию, вызовите метод **getDefaultSensor()**

Динамические датчики

В Android 7.0 Nougat (API 24) появилось понятие динамических датчиков, рассчитанных на платформу Android Things. Датчики могут присоединяться и отсоединяться от платы в любое время.

Для определения доступных динамических датчиков используются методы **isDynamicSensorDiscoverySupported()**, **isDynamicSensor()**, **getDynamicSensorList()**.

Момент присоединения или отсоединения датчика от платы можно отслеживать через класс **SensorManager.DynamicSensorCallback**.

Base проект

Датчики

```
public class SettingsActivity extends AppCompatActivity implements SensorEventListener {
```

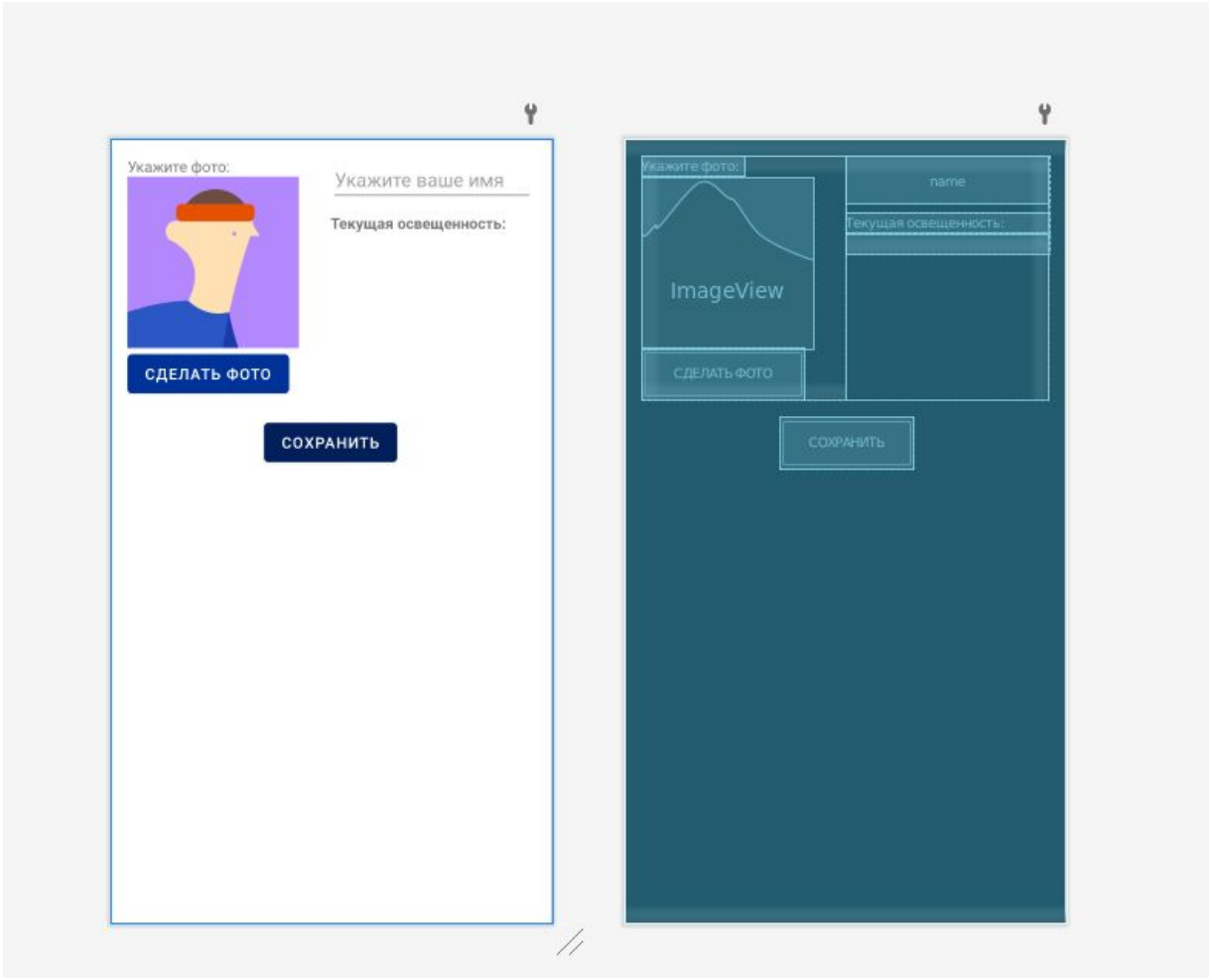
```
private SensorManager sensorManager;
private Sensor light;
private TextView sensorLight;

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

    sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    light = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
```

```
@Override
public final void onAccuracyChanged(Sensor sensor, int accuracy) {
    // Do something here if sensor accuracy changes.
}

@Override
public final void onSensorChanged(SensorEvent event) {
    // The light sensor returns a single value.
    // Many sensors return 3 values, one for each axis.
    float lux = event.values[0];
    sensorLight.setText("{lux} lux");
}

@Override
protected void onResume() {
    super.onResume();
    sensorManager.registerListener( listener: this, light, SensorManager.SENSOR_DELAY_NORMAL);
}

@Override
protected void onPause() {
    super.onPause();
    sensorManager.unregisterListener(this);
}
```

GRAVITY group

Base проект


HIGHER SCHOOL OF ECONOMICS
NATIONAL RESEARCH UNIVERSITY
РАСПИСАНИЕ ДЛЯ СТУДЕНТОВ
РАСПИСАНИЕ ДЛЯ ПРЕПОДАВАТЕЛЯ
НАСТРОЙКИ


Укажите фото:
Укажите ваше имя
Текущая освещенность:
СДЕЛАТЬ ФОТО
СОХРАНИТЬ
Укажите фото:
name
Текущая освещенность:
ImageView
СДЕЛАТЬ ФОТО
СОХРАНИТЬ

Base проект

Задания

1. Использовать разрешения для доступа к камере, чтобы добавить аватар пользователя. Если ранее картинку сохранили, то показывать ее при открытии экрана. Картинку хранить в файле.
2. Использовать параметры датчика освещённости и вывести в настройках динамическое значение освещённости.
3. Доработать работу с полем name. Сохранять его в преференсы и считывать при открытии экрана. Работу с преференсами сделать через PreferenceManager.
4. Вывести в настройках список всех доступных датчиков.

Литература

https://developer.android.com/guide/topics/sensors

https://developer.android.com/reference/android/hardware/SensorManager

https://developer.android.com/guide/topics/permissions/overview

https://developer.android.com/reference/android/content/SharedPreferences

GRAVITY group