

# Технологии разработки мобильных приложений

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 4

## УЧЕБНЫЕ ВОПРОСЫ

<i>1</i>	<i>ОСНОВЫ ИСПОЛЬЗОВАНИЯ АППАРАТНЫХ ВОЗМОЖНОСТЕЙ МОБИЛЬНЫХ УСТРОЙСТВ</i>	<i>2</i>
1.1	Задание. Список датчиков.	6
1.2	Снимаем показания акселерометра	7
1.3	Камера	10
1.4	Микрофон. MediaRecorder	12
<i>2</i>	<i>Разработка приложения с GoogleMaps/YandexMaps</i>	<i>18</i>
2.1	Google Maps	18
2.2	Задание GoogleMaps	18
	Задание	18
2.3	Yandex Maps	24
2.4	Задание YandexMaps	26
2.4.1	Местоположение пользователя.	26
2.4.2	Маршрут водителя.	28
<i>3</i>	<i>КОНТРОЛЬНОЕ ЗАДАНИЕ</i>	<i>33</i>

## 1 ОСНОВЫ ИСПОЛЬЗОВАНИЯ АППАРАТНЫХ ВОЗМОЖНОСТЕЙ МОБИЛЬНЫХ УСТРОЙСТВ

Наличие в современных телефонах электронных компасов, датчиков равновесия, яркости и близости позволяют реализовывать такие функции как дополненная реальность, ввод данных, основанный на перемещениях в пространстве и многое другое. Датчиковое оборудование делится на несколько категорий: движения, положения и окружающей среды. Доступ к датчикам осуществляется используя класс `SensorManager`, ссылку на который можно получить с помощью стандартного метода `getSystemService`:

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

Чтобы узнать, какие сенсоры есть в смартфоне, следует использовать метод `getSensorList` объекта `SensorManager`:

```
List<Sensor> sensors = sensorManager.getSensorList(Sensor.TYPE_ALL);
```

Полученный список будет включать все поддерживаемые датчики: как аппаратные, так и виртуальные. Более того, некоторые из них будут иметь различные независимые реализации, отличающиеся количеством потребляемой энергии, задержкой, рабочим диапазоном и точностью.

Также требуется интерфейс `android.hardware.SensorListener`. Интерфейс реализован с помощью класса, который используется для ввода значений датчиков по мере их изменения в режиме реального времени. Приложение реализует этот интерфейс для мониторинга одного или нескольких имеющихся аппаратных датчиков.

Интерфейс включает в себя два необходимых метода:

- Метод `onSensorChanged(int sensor, float values[])` вызывается всякий раз, когда изменяется значение датчика. Этот метод вызывается только для датчиков, контролируемых данным приложением. В число аргументов метода входит целое, которое указывает, что значение датчика изменилось, и массив значений с плавающей запятой, отражающих собственно значение датчика. Некоторые датчики выдают только одно значение

данных, тогда как другие предоставляют три значения с плавающей запятой. Датчики ориентации и акселерометр дают по три значения данных каждый.

- Метод `onAccuracyChanged(int sensor,int accuracy)` вызывается при изменении точности показаний датчика. Аргументами служат два целых числа: одно указывает датчик, а другое соответствует новому значению точности этого датчика.

Чтобы получать события, генерируемые датчиками, требуется зарегистрировать свою реализацию интерфейса `SensorEventListener` с помощью `SensorManager`, указать объект `Sensor`, за которым требуется наблюдать, и частоту, с которой необходимо получать обновления:

```
Sensor defPressureSensor  
    =sensorManager.getDefaultSensor(Sensor.TYPE_PRESSURE);  
sensorManager.registerListener(workingSensorEventListener, defPressureSensor,  
    SensorManager.SENSOR_DELAY_NORMAL);
```

В первой строчке указывается, что будет использоваться датчик – барометр. Далее вызывается метод `registerListener` объекта класса `SensorManager` для установки параметров датчика.

В классе `SensorManager` определены четыре статические константы, определяющие частоту обновления:

- `SensorManager.SENSOR_DELAY_FASTEST` — максимальная частота обновления данных;
- `SensorManager.SENSOR_DELAY_GAME` — частота, обычно используемая в играх, поддерживающих гироскоп;
- `SensorManager.SENSOR_DELAY_NORMAL` — частота обновления по умолчанию;
- `SensorManager.SENSOR_DELAY_UI` — частота, подходящая для обновления пользовательского интерфейса.

В таблице 1.1 указаны типы датчиков и описание возвращаемых значений в метод `onSensorChanged(int sensor, float values[])`.

Таблица 1-1 Значения, возвращаемые датчиками

Тип датчика	Кол-во значений	Содержание значений	Примечание
TYPE_ACCELEROMETER	3	value[0]:ось X (поперечная) value[1]: ось Y (продольная) value[2]:ось Y (вертикальная)	Ускорение ( $\text{м/с}^2$ ) по трём осям. Константы SensorManager.GRAVITY_*
TYPE_GRAVITY	3	value[0]:ось X (поперечная) value[1]: ось Y (продольная) value[2]:ось Y (вертикальная)	Сила тяжести ( $\text{м/с}^2$ ) по трём осям. Константы SensorManager.GRAVITY_*
TYPE_RELATIVE_HUMIDITY	1	value[0]:относительная влажность	Относительная влажность в процентах (%)
TYPE_LINEAR_ACCELERATION	3	value[0]:ось X (поперечная) value[1]: ось Y (продольная) value[2]:ось Y (вертикальная)	Линейное ускорение ( $\text{м/с}^2$ ) по трём осям без учёта силы тяжести
TYPE_GYROSCOPE	3	value[0]:ось X value[1]:ось Y value[2]:ось Z	Скорость вращения (рад/с) по трём осям
TYPE_ROTATION_VECTOR	4	values[0]: $x \cdot \sin(q/2)$ values[1]: $y \cdot \sin(q/2)$ values[2]: $z \cdot \sin(q/2)$ values[3]: $\cos(q/2)$	Положение устройства в пространстве. Описывается в виде угла поворота относительно оси в градусах
TYPE_MAGNETIC_FIELD	3	value[0]:ось X (поперечная) value[1]: ось Y (продольная)	Внешнее магнитное поле (мкТл)

		value[2]:ось Y (вертикальная)	
TYPE_LIGHT	1	value[0]:освещённость	Внешняя освещённость (лк). Константы SensorManager.LIGHT_*
TYPE_PRESSURE	1	value[0]:атм.давление	Атмосферное давление (мбар)
TYPE_PROXIMITY	1	value[0]:расстояние	Расстояние до цели
TYPE_AMBIENT_TEMPERATURE	1	value[0]:температура	Температура воздуха в градусах по Цельсию
TYPE_POSE_6DOF	15	см. документацию	
TYPE_STATIONARY_DETECT	1	value[0]	5 секунд неподвижен
TYPE_MOTION_DETECT	1	value[0]	В движении за последние 5 секунд
TYPE_HEART_BEAT	1	value[0]	

## 1.1 Задание. Список датчиков.

Создать новый проект. В меню File> New> New Project> Phone & Tablet Module> Empty Activity. Приложение назвать **practice\_v4**.

Добавить в файл разметки ListView:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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">

    <ListView
        android:id="@+id/list_view"
        android:layout_width="368dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>

</android.support.constraint.ConstraintLayout>
```

Чтобы узнать, какие сенсоры есть в смартфоне, следует использовать метод `getSensorList` объекта `SensorManager`:

```
List<Sensor> sensors = sensorManager.getSensorList(Sensor.TYPE_ALL);
```

Полученный список будет включать все поддерживаемые датчики: как аппаратные, так и виртуальные. Более того, некоторые из них будут иметь различные независимые реализации, отличающиеся количеством потребляемой энергии, задержкой, рабочим диапазоном и точностью.

Для получения списка всех доступных датчиков конкретного типа необходимо указать соответствующую константу. Например, код:

```
List<Sensor> pressureList =
    sensorManager.getSensorList(Sensor.TYPE_PRESSURE);
```

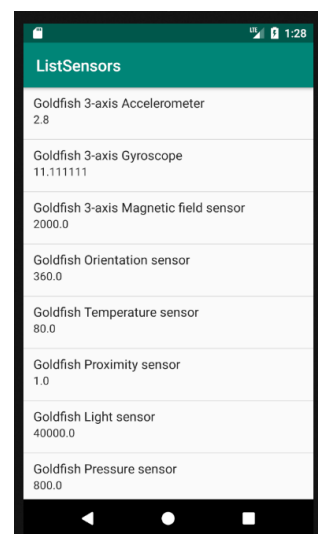


Рисунок 1.1 – Список датчиков

вернет все доступные барометрические датчики. Причем аппаратные реализации окажутся в начале списка, а виртуальные — в конце (правило действует для всех типов датчиков).

```
public class MainActivity extends AppCompatActivity {
    private SensorManager sensorManager;
    private ListView listCountSensor;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        listCountSensor = findViewById(R.id.list_view);

        sensorManager = (SensorManager)
getSystemService(Context.SENSOR_SERVICE);
        // получаем список всех датчиков на устройстве
        List<Sensor> sensorsList =
sensorManager.getSensorList(Sensor.TYPE_ALL);
        // создаем список для отображения в ListView найденных датчиков
        ArrayList<HashMap<String, Object>> arrayList = new ArrayList<>();
        HashMap<String, Object> sensorTypeList;
        for (int i = 0; i < sensorsList.size(); i++) {
            sensorTypeList = new HashMap<>();
            sensorTypeList.put("Name", sensorsList.get(i).getName());
            sensorTypeList.put("Value",
sensorsList.get(i).getMaximumRange());
            arrayList.add(sensorTypeList);
        }
        // создаем адаптер и устанавливаем тип адаптера - отображение двух
полей
        SimpleAdapter mHistory = new SimpleAdapter(this, arrayList,
android.R.layout.simple_list_item_2,
            new String[]{"Name", "Value"},
            new int[]{android.R.id.text1, android.R.id.text2});
        listCountSensor.setAdapter(mHistory);
    }
}
```

## 1.2 Снимаем показания акселерометра

Практически любой современный телефон имеет акселерометр, позволяющий определить положение телефона относительно земли, а также ускорение в пространстве по осям X, Y, Z.

Акселерометр используется для измерения ускорения. Его иногда называют датчиком силы притяжения.

Акселерометры часто выступают в качестве датчиков силы притяжения, так как они не могут определить, чем вызвано ускорение — движением или гравитацией. В результате этого в состоянии покоя акселерометр будет указывать на ускорение по оси Z (вверх/вниз), равное 9,8м/с<sup>2</sup> (это значение доступно в виде константы SensorManager.STANDARD\_GRAVITY).



Ускорение — это производная скорости по времени, поэтому акселерометр определяет, насколько быстро изменяется скорость устройства в заданном направлении. Используя этот датчик, вы можете обнаруживать движение и, что более полезно, изменение его скорости. Акселерометр не измеряет скорость как таковую, поэтому вы не можете получить скорость движения, основываясь на единичном замере. Вместо этого необходимо учитывать изменения ускорения на протяжении какого-то отрезка времени.

Ускорение может быть измерено в трех направлениях: по оси ординат, по оси абсцисс, а также по вертикальной оси. Менеджер `SensorManager` сообщает об изменениях в показаниях акселерометра по всем трём направлениям.

Значения, переданные через свойство `values` объекта `SensorEvent`, отражают боковое, продольное и вертикальное ускорение (именно в таком порядке).

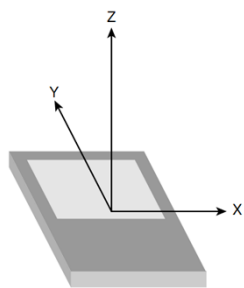


Рисунок 1.3 – Оси устройства

Рисунок 1.3 иллюстрирует нанесение трех направляющих осей на устройство, находящееся в состоянии покоя, которое в интерпретации `SensorManager` наступает тогда, когда устройство лежит на плоской поверхности экраном вверх и находится при этом в портретном режиме.

Ось X (ординаты). Боковое (влево или вправо) ускорение, положительные значения которого свидетельствуют о движении в направлении правой части устройства, а отрицательные — в направлении левой его части. Например, положительное ускорение по оси X будет, если устройство, лежа экраном вверх, повернется вправо (не отрывая крышку от поверхности).

Ось Y (абсциссы). Ускорение вперед или назад. В первом случае показатели будут больше нуля. Разместив устройство таким образом, как в предыдущем пункте, и подвинув в сторону его верхнюю часть, вы создадите положительное продольное ускорение.

Ось Z (аппликаты). Ускорение вверх или вниз. В первом случае при подъеме устройства значения будут положительными. В состоянии покоя вертикальное ускорение равно  $9,8 \text{ м/с}^2$  (вследствие силы тяжести).

Изменения ускорения отслеживаются посредством `SensorEventListener`. Зарегистрируйте реализацию этого интерфейса с помощью `SensorManager`,

используя объект `Sensor` с типом `Sensor.TYPE_ACCELEROMETER`, чтобы запрашивать обновления для акселерометра. В следующем листинге регистрируется акселерометр по умолчанию, используя стандартную частоту обновлений.

```
SensorManager sensorManager =
    (SensorManager) getSystemService(Context.SENSOR_SERVICE);
int sensorType = Sensor.TYPE_ACCELEROMETER;
sensorManager.registerListener(this,
    sensorManager.getDefaultSensor(sensorType),
    SensorManager.SENSOR_DELAY_NORMAL);
```

Интерфейс `SensorEventListener` содержит два обязательных метода. Нас интересует обработчик `onSensorChanged()`, который будет срабатывать при измерении ускорения в произвольном направлении.

```
@Override
public void onSensorChanged(SensorEvent sensorEvent) {
    if (sensorEvent.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
        float xAxis_lateralA = sensorEvent.values[0];
        float yAxis_longitudinalA = sensorEvent.values[1];
        float zAxis_verticalA = sensorEvent.values[2];
        // TODO Использовать полученное ускорение в своей программе.
    }
}
```

Метод `onSensorChanged()` получает объект `SensorEvent`, содержащий массив трёх значений типа `float`, представляющий собой показатели ускорения по всем трём осям. Основываясь на состоянии покоя, когда устройство лежит на задней крышке в портретном режиме, первый элемент означает боковое ускорение, второй — продольное, третий — вертикальное. Процесс извлечения этих показателей показан ниже:

*Создать новый проект. В меню `File > New > New Project > Phone & Tablet Module > Empty Activity`. Приложение назвать `accelerometer`.*

1. Добавить 3 текстовых поля:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textViewAzimuth"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/textViewPitch"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/textViewRoll"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

2. Далее следует класс, реализующий интерфейс `SensorEventListener`. Регистрируем датчик в методе `onResume()`, снимаем регистрацию в методе `onPause()` для освобождения ресурсов, и отслеживаем изменения в методе `onSensorChanged()`.

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

    mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
    mAccelerometerSensor = mSensorManager
        .getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

    mAzimuthTextView = findViewById(R.id.textviewAzimuth);
    mPitchTextView = findViewById(R.id.textviewPitch);
    mRollTextView = findViewById(R.id.textviewRoll);
}

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

@Override
protected void onResume() {
    super.onResume();
    mSensorManager.registerListener(this, mAccelerometerSensor,
        SensorManager.SENSOR_DELAY_NORMAL);
}

@Override
public void onAccuracyChanged(Sensor arg0, int arg1) {
    // TODO Auto-generated method stub
    // Не используется
}

@Override
public void onSensorChanged(SensorEvent event) {

    float valueAzimuth = event.values[0];
    float valuePitch = event.values[1];
    float valueRoll = event.values[2];

    mAzimuthTextView.setText("Azimuth: " + String.valueOf(valueAzimuth));
    mPitchTextView.setText("Pitch: " + String.valueOf(valuePitch));
    mRollTextView.setText("Roll: " + String.valueOf(valueRoll));
}
```

3. Запустите проект.

*Задание: Добавить `ImageView` на экран и в зависимости от показаний датчика определять экран вверх смотрит или вниз.*

## 1.3 Камера

Если в приложении необходимо сделать снимок или снять видео необязательно для этого создавать отдельное Activity и работать в нем с объектом `Camera`. Можно использовать уже существующее в системе приложение.

Для этого ваше приложение должно отправить Intent с action = MediaStore.ACTION\_IMAGE\_CAPTURE (фото) или MediaStore.ACTION\_VIDEO\_CAPTURE (видео) и ждать ответ. Т.е. надо использовать методы startActivityForResult и onActivityResult.

Также, в этот Intent вы можете поместить желаемый путь к файлу и туда будет сохранен результат работы камеры. Для этого в Intent необходимо добавить Uri с ключом MediaStore.EXTRA\_OUTPUT.

### **Задание:**

*Создать новый проект. В меню File > New > New Module > Phone & Tablet Module > Empty Activity. Приложение назвать camera.*

Требуется программно запустить приложение "Камера", а полученную фотографию сохранить в папке и на экране.

Используйте статическую константу MediaStore.ACTION\_IMAGE\_CAPTURE для создания намерения, которое потом нужно передать методу startActivityForResult(). Разместите на форме кнопку и ImageView, в который будем помещать полученный снимок. Этот код запускает стандартное приложение камеры. Полученное с камеры изображение можно обработать в методе onActivityResult():

```
public class MainActivity extends AppCompatActivity {
    final String TAG = MainActivity.class.getSimpleName();
    private ImageView imageView;
    private static final int CAMERA_REQUEST = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        imageView = findViewById(R.id.imageView);
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == CAMERA_REQUEST && resultCode == RESULT_OK) {
            // извлекаем изображение
            Bitmap thumbnailBitmap = (Bitmap) data.getExtras().get("data");
            imageView.setImageBitmap(thumbnailBitmap);
        }
    }

    public void imageViewOnClick(View view) {
        Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        startActivityForResult(cameraIntent, CAMERA_REQUEST);
    }
}
```

Данный код запустит приложение, работающее с камерой, позволяя пользователю поменять настройки изображения, что освобождает вас от необходимости создавать своё собственное приложение для этих нужд. Вполне возможно, что у вас будет несколько приложений, умеющих делать фотографии, тогда сначала появится окно выбора программы.

По умолчанию фотография возвращается в виде объекта Bitmap, содержащего миниатюру. Этот объект находится в параметре data, передаваемом в метод onActivityResult(). Чтобы получить миниатюру в виде объекта Bitmap, нужно вызвать метод getParcelableExtra() из намерения, передав ему строковое значение data. В примере использовался упрощённый вариант, ниже показан пример другого варианта кода.

Если вы укажете исходящий путь URI с помощью параметра MediaStore.EXTRA\_OUTPUT в запущенном намерении, полноразмерное изображение, снятое камерой, сохранится в заданном месте. В таком случае в метод onActivityResult() не будет передана миниатюра, а итоговое намерение продемонстрирует значение null.

**Домашнее задание: сохранить полученное изображение на карте памяти.**

**Внимание! Требуется проверка возможности записи в память устройства, поэтому либо реализуйте запрос разрешений, либо зайдите в:**

**Настройки телефона -> Программы -> "Название Вашей программы" -> Разрешения, установите требуемые разрешения!**

## 1.4 Микрофон. MediaRecorder

*Задание:*

*Напишем простейший диктофон для записи звука на микрофон с сохранением в память устройства.*

*Создать новый проект. В меню File> New> New Module> Phone & Tablet Module> Empty Activity. Приложение назвать microphone.*

В файл activity\_main.xml добавить две кнопки с идентификаторами для начала записи и остановки:

```
android:id="@+id/btnStart";  
android:id="@+id/btnStop"
```

Класс android.media.MediaRecorder позволяет делать записи медиафрагментов, включая аудио и видео. MediaRecorder действует как

конечный автомат. Вы задаёте различные параметры, такие как устройство-источник и формат. После установки запись может выполняться как угодно долго, пока не будет остановлена.

```

public class MainActivity extends AppCompatActivity {
    private static final String TAG = "MainActivity";

    private Button mStartRecordingButton;
    private Button mStopRecordingButton;
    private MediaRecorder mMediaRecorder;
    private File mAudioFile;

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

        mMediaRecorder = new MediaRecorder();
        mStartRecordingButton = findViewById(R.id.btnStart);
        mStopRecordingButton = findViewById(R.id.btnStop);

        mStartRecordingButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
                try {
                    mStartRecordingButton.setEnabled(false);
                    mStopRecordingButton.setEnabled(true);
                    mStopRecordingButton.requestFocus();

                    startRecording();
                } catch (Exception e) {
                    Log.e(TAG, "Caught io exception " + e.getMessage());
                }
            }
        });

        mStopRecordingButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
                mStartRecordingButton.setEnabled(true);
                mStopRecordingButton.setEnabled(false);
                mStartRecordingButton.requestFocus();
                stopRecording();
                processAudioFile();
            }
        });

        mStopRecordingButton.setEnabled(false);
        mStartRecordingButton.setEnabled(true);
    }

    private void processAudioFile() {
        ContentValues values = new ContentValues(4);
        long current = System.currentTimeMillis();

        values.put(MediaStore.Audio.Media.TITLE, "audio" +
mAudioFile.getName());
        values.put(MediaStore.Audio.Media.DATE_ADDED, (int) (current /
1000));
        values.put(MediaStore.Audio.Media.MIME_TYPE, "audio/3gpp");
        values.put(MediaStore.Audio.Media.DATA,
mAudioFile.getAbsolutePath());
        ContentResolver contentResolver = getContentResolver();

        Uri baseUrl = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
        Uri newUri = contentResolver.insert(baseUrl, values);

        sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE,
newUri));
    }
}

```

*Алгоритм записи:*

Метод `processAudioFile`. Сначала определимся с источником (свойство `MediaRecorder.AudioSource`):

- MIC - встроенный микрофон
- VOICE\_UPLINK - исходящий голосовой поток при телефонном звонке (вы говорите)
- VOICE\_DOWNLINK - входящий голосовой поток при телефонном звонке (вам говорят)
- VOICE\_CALL - запись телефонного звонка
- CAMCORDER - микрофон, связанный с камерой (если доступен)
- VOICE\_RECOGNITION - микрофон, используемый для распознавания голоса (если доступен)
- VOICE\_COMMUNICATION – аудио поток с микрофона будет "заточен" под VoIP (если доступен)

Если указанный источник не поддерживается текущим устройством, то будет использован микрофон по умолчанию.

Далее определимся с форматом записываемого звука (свойство `MediaRecorder.OutputFormat`):

- THREE\_GPP - формат 3GPP
- MPEG\_4 - формат MPEG4
- AMR\_NB - формат AMR\_NB (подходит для записи речи)
- AMR\_WB
- RAW\_AMR

Определимся с сжатием звука (свойство `MediaRecorder.AudioEncoder`).

- AAC
- AMR\_NB
- AMR\_WB

Указываем путь к файлу, в котором будут сохранены аудиоданные (метод `setOutputFile()`)

В методе `startRecording()` создается и инициализируется экземпляр `MediaRecorder`. В качестве источника данных выбирается микрофон (MIC). Выходной формат устанавливается в 3GPP (файлы \*.3gp) - медиа формат, ориентированный на мобильные устройства. Кодек настроен на AMR\_NB - аудиоформат с частотой дискретизации 8 кГц. NB означает узкую полосу частот.



Аудиофайл сохраняется на внешней карте. Затем этот файл связывается с экземпляром `MediaRecorder`, обращаясь к методу `setOutputFile`. Аудиоданные будут храниться в этом файле.

Вызов метода `prepare()` завершает инициализацию `MediaRecorder`. Когда нужно начать процесс записи, вызывается метод `start()`. Запись в файл на карте памяти ведется до тех пор, пока не будет вызван метод `stop()`, который освобождает ресурсы, выделенные экземпляру `MediaRecorder`.

Когда аудиофрагмент записан, можно выполнить несколько действий:

- добавить аудиозапись в медиатеку на устройстве;
- выполнить шаги по распознаванию звука;
- автоматически загрузить звуковой файл в сетевую папку для обработки.

В этом примере метод `processAudioFile()` добавляет аудиозапись в медиатеку. Для уведомления встроенного приложения о том, что доступна новая информация, используется `Intent`.

Для работы с записью звуков и использования внешней SD-карты нужно добавить пару разрешений через `Runtime Permission`.

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

**Внимание! Требуется проверка возможности использования микрофона и записи в память устройства, поэтому либо реализуйте запрос разрешений, либо зайдите в:**

Настройки телефона -> Программы -> "Название Вашей программы"  
->Разрешения, установите требуемые разрешения!

Подготавливаем и запускаем запись (методы `prepare()` и `start()`). Остановить запись можно при помощи метода `stop()`.

- `setAudioChannels` – задаёт режим записи: 1 (моно) или 2 (стерео).
- `setAudioEncodingBitRate` - задаёт битрейт (качество записи звука).
- `setAudioSamplingRate` - задаёт сэмплрейт (как часто считываются данные с микрофона).

• `setMaxDuration` позволяет указать максимальную длительность записи. По достижении этого времени (в миллисекундах), запись остановится, а слушатель, указанный в `MediaRecorder.OnInfoListener`, получит код `what = MEDIA_RECORDER_INFO_MAX_DURATION_REACHED`.

- `setMaxFileSize` позволяет указать максимальный размер файла. По достижении указанного размера (в байтах), запись остановится, а слушатель, указанный в `MediaRecorder.OnInfoListener`, получит код `what = MEDIA_RECORDER_INFO_MAX_FILESIZE_REACHED`.

Методы надо вызывать перед вызовом метода `prepare`.

В качестве дополнительной защиты требуется добавить проверку наличия микрофона на устройстве, ведь программу можно запустить не только на телефоне, но и на читалке, холодильнике и швейной машинке под управлением Android.

## 2 Разработка приложения с GoogleMaps/YandexMaps

### 2.1 Google Maps

Google Maps - это лидер среди современных картографических сервисов, предоставляющих спутниковые интерактивные карты онлайн. По крайней мере лидер в области спутниковых снимков и по количеству разнообразных дополнительных сервисов и инструментов (Google Earth, Google Mars, разнообразные погодные и транспортные сервисы, одно из самых мощных API). Популярность Карт Google остается одной из самых высоких из всех других картографических сервисов. Отчасти причина в том, что именно в Google Maps возможно найти самые детализированные спутниковые фотографии для самых обширных регионов любых стран.

С версией Google Maps Android API v2 имеется возможность использовать следующие возможности:

- поддержка фрагментов
- больше слоёв Google Maps, включая спутниковый, гибридный, пробки, а также новый слой — внутренние карты для основных аэропортов и торговых центров
- уменьшение строк кода, чтобы создавать маркеры и информационные окна (infowindow)

*Минимальная требуемая версия — Android 2.2 с OpenGL ES версии 2.*

### 2.2 Задание GoogleMaps

#### Задание:

Создать новый проект. В меню File> New> New Project> Phone & Tablet Module> Google maps Activity (рис. 2.1). Приложение назвать google\_maps.

1. Откройте файл `res/values/google_maps_api.xml`. В нём вы найдёте ссылку на получение специального ключа, чтобы карты заработали. Переходим по ссылке, выбираем из выпадающего списка нужный проект и нажимаем кнопку (рис. 2.2).

⚠ - Требуется личный кабинет в gmail.com

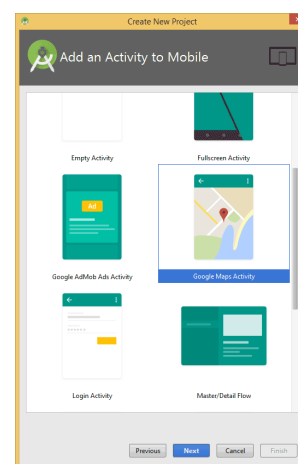


Рисунок 2.1 – Создание проекта

<resources>

<!--

**TODO: Before you run your application, you need a Google Maps API key.**

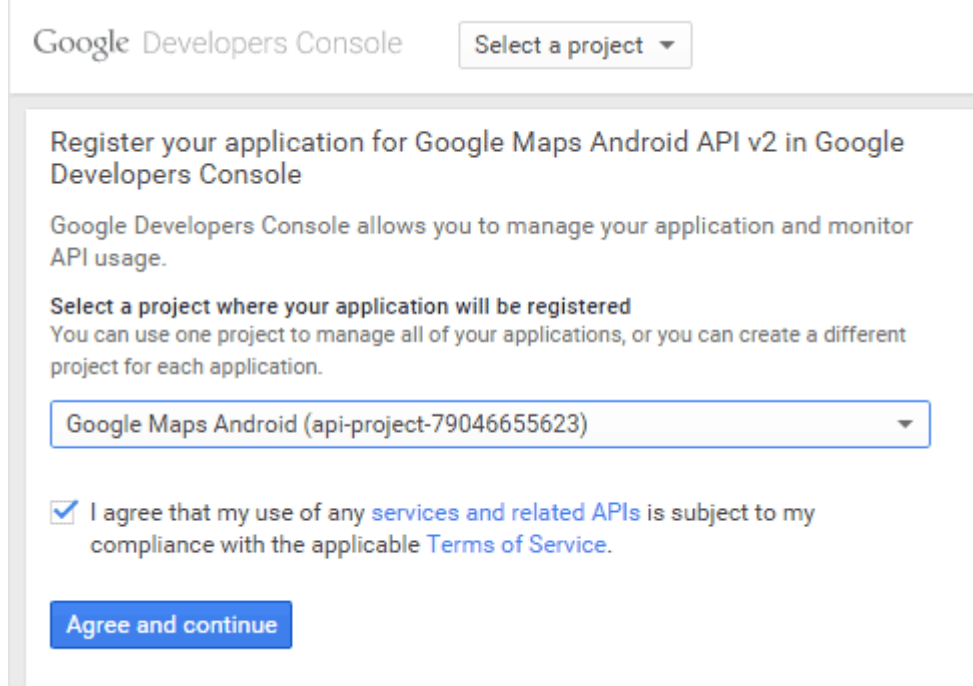
To get one, follow this link, follow the directions and press "Create" at the end:

[https://console.developers.google.com/flows/enableapi?apiid=maps\\_android\\_backend&keyType=CLIENT\\_SIDE\\_ANDROID&r=09:F3:92:AA:70:26:AD:29:7E:17:F7:C1:A6:71:BD:XXXX](https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=09:F3:92:AA:70:26:AD:29:7E:17:F7:C1:A6:71:BD:XXXX)

You can also add your credentials to an existing key, using these values:

Рисунок 2.2 – Пример сгенерированной ссылке

Выберите предложенный проект, либо создайте новый



Google Developers Console Select a project ▾

**Register your application for Google Maps Android API v2 in Google Developers Console**

Google Developers Console allows you to manage your application and monitor API usage.

**Select a project where your application will be registered**  
You can use one project to manage all of your applications, or you can create a different project for each application.

Google Maps Android (api-project-79046655623) ▾

☒ I agree that my use of any [services and related APIs](#) is subject to my compliance with the applicable [Terms of Service](#).

Agree and continue

Далее создайте новый ключ

[Учетные данные](#) [Окно запроса доступа OAuth](#) [Подтверждение прав на домен](#)

Создать учетные данные ▾ Удалить

Чтобы получить доступ к включенным API, создайте учетные данные. Изучить документацию по API можно [здесь](#).

Ключи API

| <input type="checkbox"/> Название   | Дата создания ▾  | Ограничения        | Ключ                  |
|-------------------------------------|------------------|--------------------|-----------------------|
| <input type="checkbox"/> Ключ API 1 | 12 нояб. 2018 г. | Приложения Android | AlzaSyBwQQv4lZTd0drB5 |

Внесите сгенерированный ключ в файл res/values/google\_maps\_api.xml.

2. Изучите файл `AndroidManifest.xml`, `activity_maps.xml` и `MapsActivity.java`.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_maps);
    // Obtain the SupportMapFragment and get notified when the map is ready
    to be used.
    SupportMapFragment mapFragment = (SupportMapFragment)
    getSupportFragmentManager()
        .findFragmentById(R.id.map);
    mapFragment.getMapAsync(this);
}

@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;

    // Add a marker in Sydney and move the camera
    LatLng sydney = new LatLng(-34, 151);
    mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in
    Sydney"));
    mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
}
```

Метод `onMapReady()` является реализацией интерфейса `OnMapReadyCallback`. В этом методе можно писать код, когда карта готова к использованию. В примере добавляется маркер (метод `addMarker()`) в указанной точке (через объект `LatLng`) с указанием текста (метод `title()`). Метод `moveCamera()` перемещает карту в указанную позицию и мы можем видеть сразу нужное место.

Сейчас там только пример добавления нового маркера в указанную позицию и заголовка, который появится при нажатии на маркер.

### *Запустите проект*

3. По умолчанию карты выводятся без настроек. Добавьте функциональность. Чтобы на карте выводился кнопка местоположения, вызывайте метод `setMyLocationButtonEnabled()` (выводится правом верхнем углу). С его помощью можно быстро оказаться в том месте, где вы сейчас находитесь, при условии, что вы разрешили использовать определение своего местоположения. Этот код устанавливаем в метод обратного вызова `onMapReady()`

Обратите внимание, что недостаточно вызвать метод для вывода кнопки, надо также разрешить приложению



Рисунок 2.3 –  
Google Maps

определять ваше местоположение через метод `setMyLocationEnabled(true)`. Но если вы вставите эту строчку в код, то увидите сообщение об ошибке с красной волнистой чертой и проект не запустится. Подсказка будет говорить, что требуется разрешение. Подведите курсор к методу и используя комбинацию `Alt+Enter`, выберите `Add permission check`. Получим готовый шаблон кода.

Данный метод проверяет возможность определять местоположение, но не запрашивает, поэтому либо реализуйте запрос разрешений, либо зайдите в:

Настройки телефона -> Программы -> "Название Вашей программы"  
->Разрешения, установите требуемые разрешения!

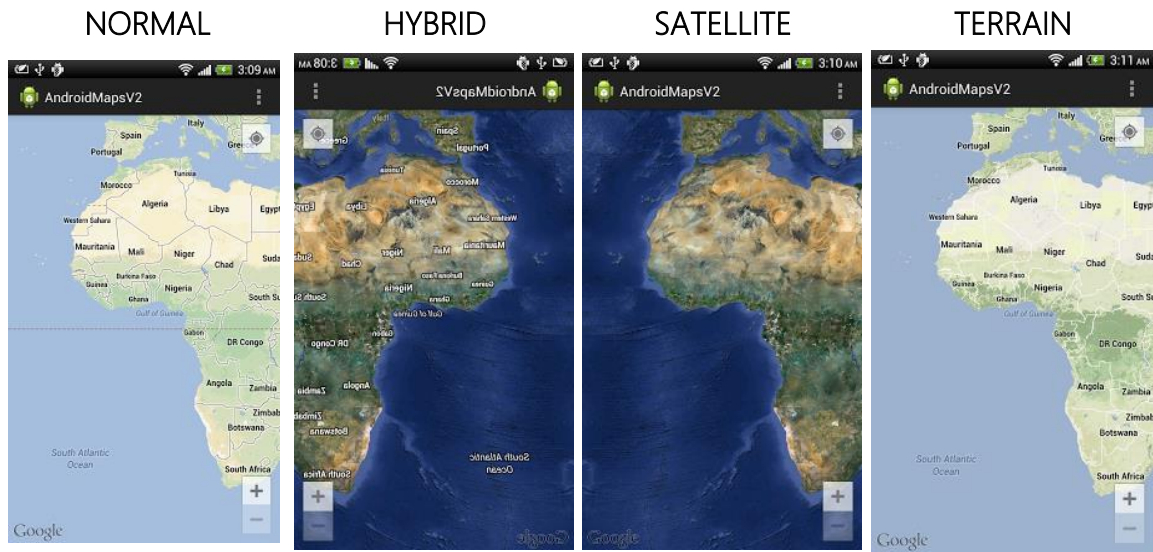
Также можно включить дополнительные элементы: компас (в верхнем левом углу при вращении карты) и зум (в нижнем правом углу).

```
mMap.getUiSettings().setZoomControlsEnabled(true);  
mMap.getUiSettings().setCompassEnabled(true);  
mMap.getUiSettings().setMyLocationButtonEnabled(true);
```

Компас появится на экране, если вы повернёте карту жестом под каким-нибудь углом. Таким образом должен получиться следующий код:

```
@Override  
public void onMapReady(GoogleMap googleMap) {  
    mMap = googleMap;  
    if (ActivityCompat.checkSelfPermission(this,  
Manifest.permission.ACCESS_FINE_LOCATION) !=  
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,  
Manifest.permission.ACCESS_COARSE_LOCATION) !=  
PackageManager.PERMISSION_GRANTED) {  
        // TODO: Consider calling  
        //     ActivityCompat#requestPermissions  
        // here to request the missing permissions, and then overriding  
        //     public void onRequestPermissionsResult(int requestCode,  
String[] permissions,  
        //                                     int[] grantResults)  
        // to handle the case where the user grants the permission. See  
the documentation  
        // for ActivityCompat#requestPermissions for more details.  
        return;  
    }  
    mMap.setMyLocationEnabled(true);  
    mMap.getUiSettings().setZoomControlsEnabled(true);  
    mMap.getUiSettings().setCompassEnabled(true);  
    mMap.getUiSettings().setMyLocationButtonEnabled(true);  
    // Add a marker in Sydney and move the camera  
    LatLng sydney = new LatLng(-34, 151);  
    mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in  
Sydney"));  
    mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));  
}
```

#### 4. Установите типы карт:



```
// выбираем один вариант
//mMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
//mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
//mMap.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
//mMap.setMapType(GoogleMap.MAP_TYPE_NONE);
```

#### 5. Включите отображение пробок

```
mMap.setTrafficEnabled(true);
```

#### 6. Добавьте метку на карту

```
private void setUpMap() {
    // выбираем один вариант
    mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);

    LatLng mirea = new LatLng(55.670005, 37.479894);
    CameraPosition cameraPosition = new CameraPosition.Builder().target(
        mirea).zoom(12).build();

    mMap.animateCamera(CameraUpdateFactory
        .newCameraPosition(cameraPosition));
    mMap.addMarker(new MarkerOptions().title("МИРЭА")
        .snippet("Крупнейший политехнический ВУЗ").position(mirea));
}
```

В объекте `LatLng` указываются координаты института. Объект `CameraPosition` позволяет задать цель (`target`), зум (`zoom`), направление (`bearing`) и крен (`tilt`). Уровень зума может быть различным для разных участков карты, поэтому используйте методы `getMinZoomLevel()` и `getMaxZoomLevel()` класса

GoogleMap, чтобы узнать минимальное и максимальное значения зума. Чем больше значение зума, тем ближе камера к поверхности земли. Крен зависит от уровня зума. По умолчанию, камера смотрит вертикально вниз. При достаточно низкой высоте камеры можно немного наклонить, чтобы смотреть на землю под углом. Затем добавляем маркер с помощью метода `addMarker()` и даём пояснения к нему.

Исходный код: <https://github.com/MobileMirea/GoogleMaps.git>

*Домашнее задание: разместите карту в фрагмент!!!  
Изучите Google Maps Directions API.*



## 2.3 Yandex Maps

Для создания картографических сервисов, основанных на yandex maps, используется MapKit. MapKit — это кроссплатформенная библиотека, позволяющая использовать картографические данные и технологии Яндекса в мобильных приложениях для iOS и Android. MapKit позволяет создать приложения с картами Яндекс для операционных систем iOS и Android.

### *Возможности MapKit:*

- получить доступ к картам Яндекса с последними обновлениями;
- проложить автомобильный маршрут с учетом дорожной ситуации;
- проложить пешеходный маршрут или маршрут с использованием общественного транспорта;
- отобразить пробки на карте;
- получить данные о топонимах и организациях.



### *Модули MapKit*

Помимо основной библиотеки MapKit доступны дополнительные библиотеки, расширяющие возможности вашего приложения:

- MapKit Directions — используется для построения маршрутов для автомобилей.
- MapKit Places — используется для работы с панорамами.
- MapKit Search — используется для поиска по карте и геокодирования.
- MapKit Transport — используется для построения маршрутов для пешеходов, велосипедов, а также маршрутов с использованием общественного транспорта.

*Используйте эти библиотеки, когда вам нужен предоставляемый ими функционал.*

Использование MapKit возможно бесплатно при условии выполнения следующих условий.

- mapKit используется только в бесплатных приложениях, которые может скачать любой человек;
- запрещается использовать MapKit для ведения по маршруту и в проектах по мониторингу и диспетчеризации;
- все данные, полученные через MapKit, должны отображаться на карте. Сохранять или изменять информацию нельзя;
- общее число запросов к сервисам поиска, маршрутизации и просмотра панорам в сутки не должно превышать 25 тысяч.

⚠ Также есть другие ограничения на бесплатное использование. Подробнее о них можно [прочитать в документации](#). Чтобы снять ограничения требуется [приобрести лицензию](#).

Библиотека MapKit для платформы Android 4.0.3 и выше доступна в репозиториях Maven Central и Gradle. Чтобы создать приложение с картой Яндекса:

1. Получите ключ для работы с MapKit.
2. Установите библиотеку MapKit.
3. Настройте библиотеку.
4. Соберите и запустите приложение.

Шаг 1. Получите ключ для работы с MapKit

1. Перейдите в [Кабинет Разработчика](#).
2. Авторизуйтесь с учетной записью Яндекса или зарегистрируйте новый аккаунт.
3. Нажмите Получить ключ и выберите пакет MapKit SDK.
4. Заполните информацию о проекте, укажите нужный тариф и нажмите Отправить.

Дальнейшие инструкции придут на почту.

Полученный ключ можно использовать для работы с MapKit в нескольких приложениях.

## 2.4 Задание YandexMaps

### 2.4.1 Местоположение пользователя.

Требуется отобразить на YandexMaps местоположение пользователя. Создать новый проект. В меню File> New> New Project> Phone & Tablet Module> Empty Activity. Приложение назвать yandex\_maps.

1. Откройте файл *build.gradle* проекта (рис. 2.1). В секции *repositories* добавьте репозиторий *Maven Central*:

```
repositories {  
    google()  
    jcenter()  
    maven {  
        url "http://maven.google.com/"  
    }  
}
```

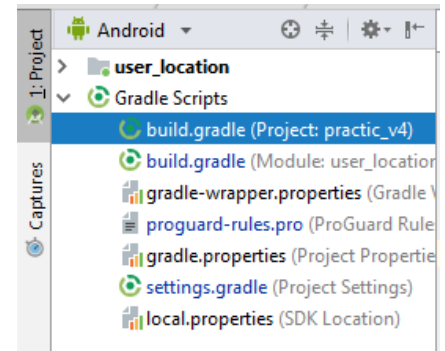


Рисунок 2.4 – Структура проекта

2. Откройте файл *build.gradle* приложения (модуля). Укажите адрес репозитория и добавьте зависимость

```
// Основная библиотека MapKit.  
implementation 'com.yandex.android:mapkit:3.1.2'
```

3. Синхронизируйте проект, чтобы применить изменения. В Android Studio требуется нажать *Sync Now* или выбрать в меню *File* → *Synchronize*. После этого библиотека загрузится в хранилище и будет доступна в проекте.

⚠ Если синхронизация завершилась успешно, при компиляции библиотека будет добавлена в проект автоматически. При ошибке компиляции, убедитесь, что вы правильно указали репозиторий и зависимость и синхронизируйте проект снова.

4. Добавление карты в разметку:

```
<?xml version="1.0" encoding="utf-8"?>  
<android.support.constraint.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">  
  
    <com.yandex.mapkit.mapview.MapView  
        android:id="@+id/mapview"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"/>  
  
</android.support.constraint.ConstraintLayout>
```

5. Инициализируйте карту в методе `onCreate` Activity и настройте слой для отображения местоположения пользователя:

```
private MapView mapView;  
private UserLocationLayer userLocationLayer;  
private final String MARKIT_API_KEY = "Ваш ключ разработчика";  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    /**  
     * Задайте API-ключ перед инициализацией MapKitFactory.  
     * Рекомендуется устанавливать ключ в методе Application.onCreate,  
     * но в данном примере он устанавливается в activity.  
     */  
    MapKitFactory.setApiKey(MARKIT_API_KEY);  
    /**  
     * Инициализация библиотеки для загрузки необходимых нативных библиотек.  
     * Рекомендуется инициализировать библиотеку MapKit в методе  
     * Activity.onCreate.  
     * Инициализация в методе Application.onCreate может привести к лишним  
     * вызовам и увеличенному использованию батареи.  
     */  
    MapKitFactory.initialize(this);  
    setContentView(R.layout.activity_main);  
    // Укажите имя activity  
    mapView = findViewById(R.id.mapview);  
    // Устанавливаем начальную точку и масштаб  
    mapView.getMap().move(new CameraPosition(new Point(0, 0), 14, 0, 0));  
    // Установка слоя для отрисовки пользовательского местоположения  
    userLocationLayer = mapView.getMap().getUserLocationLayer();  
    userLocationLayer.setEnabled(true);  
    userLocationLayer.setAutoZoomEnabled(true);  
    userLocationLayer.setHeadingEnabled(true);  
    userLocationLayer.setObjectListener(this);  
}
```

6. Передайте события `onStart` и `onStop` в `MapKitFactory` и `mapView`. Иначе `MapKit` не сможет отобразить карту и остановить обработку карты, когда Activity с картой становится невидимым для пользователя:

```
@Override  
protected void onStart() {  
    super.onStart();  
    MapKitFactory.getInstance().onStart();  
    mapView.onStart();  
}  
  
@Override  
protected void onStop() {  
    super.onStop();  
    MapKitFactory.getInstance().onStop();  
    mapView.onStop();  
}
```

7. Добавьте интерфейс для получения данных об изменении местоположения пользователя:

```
public class MainActivity extends AppCompatActivity
    implements UserLocationObjectListener
```

8. Добавьте следующую реализацию интерфейса:

```
@Override
public void onObjectAdded(@NonNull UserLocationView userLocationView) {
    userLocationLayer.setAnchor(
        new PointF((float) (mapView.getWidth() * 0.5), (float)
(mapView.getHeight() * 0.5)),
        new PointF((float) (mapView.getWidth() * 0.5), (float)
(mapView.getHeight() * 0.83)));

    // При определении направления движения устанавливается следующая иконка
    userLocationView.getArrow().setIcon(ImageProvider.fromResource(
        this, R.drawable.user_arrow));

    // При получении координат местоположения устанавливается следующая
иконка
    userLocationView.getPin().setIcon(ImageProvider.fromResource(
        this, R.drawable.user_arrow));

    // Обозначается точность определения местоположения с помощью окружности
    userLocationView.getAccuracyCircle().setFillColor(Color.BLUE);
}

@Override
public void onObjectRemoved(@NonNull UserLocationView userLocationView) {
}

@Override
public void onObjectUpdated(@NonNull UserLocationView userLocationView,
@NonNull ObjectEvent objectEvent) {
}
}
```

9. Соберите и запустите приложение.

⚠ Исходный код: <https://github.com/MobileMirea/YandexMaps.git>

## 2.4.2 Маршрут водителя.

Создать новый модуль. В меню File> New> New Module> Phone & Tablet Module> Empty Activity. Приложение назвать ya\_driving\_location.

1. Откройте файл *build.gradle* приложения (модуля). Укажите адрес репозитория и добавьте зависимость

```
// Основная библиотека MapKit.
implementation 'com.yandex.android:mapkit:3.1.2'
implementation 'com.yandex.android:directions:3.1
```

2. Синхронизируйте проект, чтобы применить изменения. В Android Studio требуется нажать Sync Now или выбрать в меню File → Synchronize.

После этого библиотека загрузится в хранилище и будет доступна в проекте.

### 3. Добавление карты в разметку:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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">

    <com.yandex.mapkit.mapview.MapView
        android:id="@+id/mapview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</android.support.constraint.ConstraintLayout>
```

### 4. Инициализация карты в методе onCreate Activity и настройка слоя для отображения местоположения пользователя:

```
private final String MAPKIT_API_KEY = "xxx";
private final Point ROUTE_START_LOCATION = new Point(55.670005, 37.479894);
private final Point ROUTE_END_LOCATION = new Point(55.794229, 37.700772);
private final Point SCREEN_CENTER = new Point(
    (ROUTE_START_LOCATION.getLatitude() +
    ROUTE_END_LOCATION.getLatitude()) / 2,
    (ROUTE_START_LOCATION.getLongitude() +
    ROUTE_END_LOCATION.getLongitude()) / 2);

private MapView mapView;
private MapObjectCollection mapObjects;
private DrivingRouter drivingRouter;
private DrivingSession drivingSession;
private int[] colors = {0xFFFF0000, 0xFF00FF00, 0x00FFBBBB, 0xFF0000FF};

@Override
protected void onCreate(Bundle savedInstanceState) {
    // установка ключа разработчика
    MapKitFactory.setApiKey(MAPKIT_API_KEY);
    MapKitFactory.initialize(this);
    DirectionsFactory.initialize(this);
    // Укажите имя activity
    setContentView(R.layout.activity_main);
    super.onCreate(savedInstanceState);

    mapView = (MapView) findViewById(R.id.mapview);
    // Устанавливаем начальную точку и масштаб
    mapView.getMap().move(new CameraPosition(
        SCREEN_CENTER, 10, 0, 0));
    // Инициализируем объект для создания маршрута водителя
    drivingRouter = DirectionsFactory.getInstance().createDrivingRouter();
    mapObjects = mapView.getMap().getMapObjects().addCollection();

    submitRequest();
}
```

5. Добавление интерфейса для получения предложенных маршрутов движения:

```
public class MainActivity extends AppCompatActivity
    implements DrivingSession.DrivingRouteListener {
```

6. Передача события *onStart* и *onStop* в *MapKitFactory* и *mapView*.  
Иначе *MapKit* не сможет отобразить карту и остановить обработку карты, когда *Activity* с картой становится невидимым для пользователя:

```
@Override
protected void onStart() {
    super.onStart();
    MapKitFactory.getInstance().onStart();
    mapView.onStart();
}
@Override
protected void onStop() {
    super.onStop();
    MapKitFactory.getInstance().onStop();
    mapView.onStop();
}
```

7. Установка критериев к маршруту

```
private void submitRequest() {
    DrivingOptions options = new DrivingOptions();
    // Кол-во альтернативных путей
    options.setAlternativeCount(3);
    ArrayList<RequestPoint> requestPoints = new ArrayList<>();
    // Устанавливаем точки маршрута
    requestPoints.add(new RequestPoint(
        ROUTE_START_LOCATION,
        new ArrayList<Point>(),
        new ArrayList<DrivingArrivalPoint>(),
        RequestPointType.WAYPOINT));
    requestPoints.add(new RequestPoint(
        ROUTE_END_LOCATION,
        new ArrayList<Point>(),
        new ArrayList<DrivingArrivalPoint>(),
        RequestPointType.WAYPOINT));
    // Делаем запрос к серверу
    drivingSession = drivingRouter.requestRoutes(requestPoints, options,
    this);
}
```

*DrivingSession.DrivingRouteListener*

```
@Override
public void onDrivingRoutes(@NonNull List<DrivingRoute> list) {
    int color;
    for (int i = 0; i < list.size(); i++) {
        // настраиваем цвета для каждого маршрута
        color = colors[i];
        // добавляем маршрут на карту
        mapObjects.addPolyline(list.get(i).getGeometry()).setStroke-
Color(color);
    }
}

@Override
public void onDrivingRoutesError(@NonNull Error error) {
    String errorMessage = getString(R.string.unknown_error_message);
    if (error instanceof RemoteError) {
        errorMessage = getString(R.string.remote_error_message);
    } else if (error instanceof NetworkError) {
        errorMessage = getString(R.string.network_error_message);
    }

    Toast.makeText(this, errorMessage, Toast.LENGTH_SHORT).show();
}
```

## 9. Соберите и запустите приложение

⚠ Исходный код: <https://github.com/MobileMirea/YandexMaps.git>





### 3 КОНТРОЛЬНОЕ ЗАДАНИЕ

Открыть контрольное задание, построенное на базе NavigationDrawer. Добавить к ранее созданному проекту с Fragment калькулятором и браузером карту. На карте отметить местоположение института. При нажатии на иконку отображается название, год создания, адрес и координаты заведения.