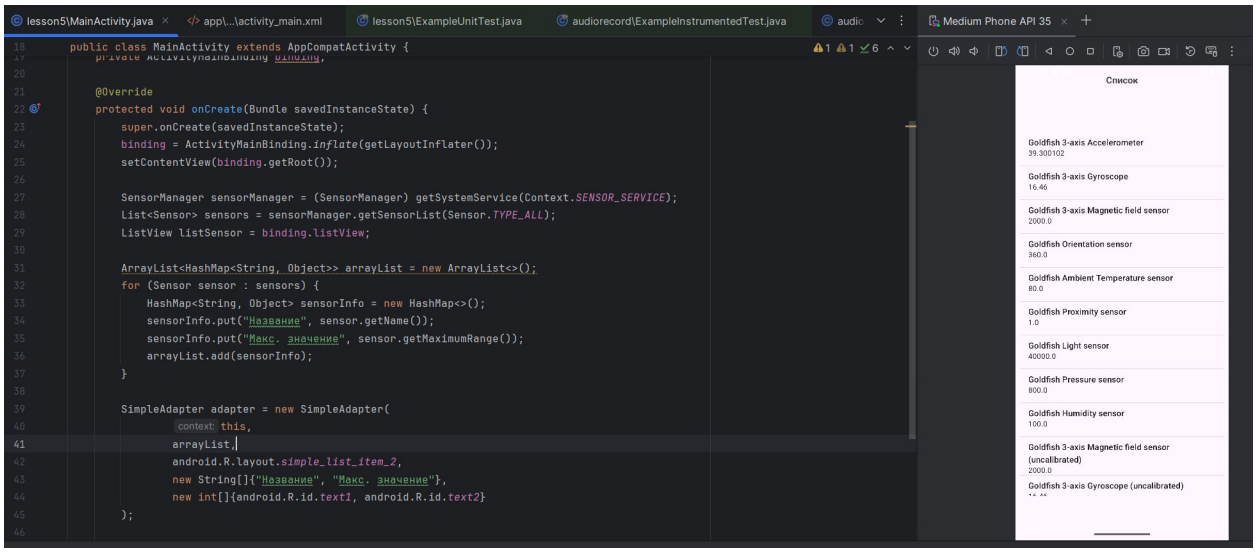
Практическая работа №5

Работа с датчиками устройства, камерой, микрофоном и аппаратной частью Android

Задание 1. Отображение списка сенсоров

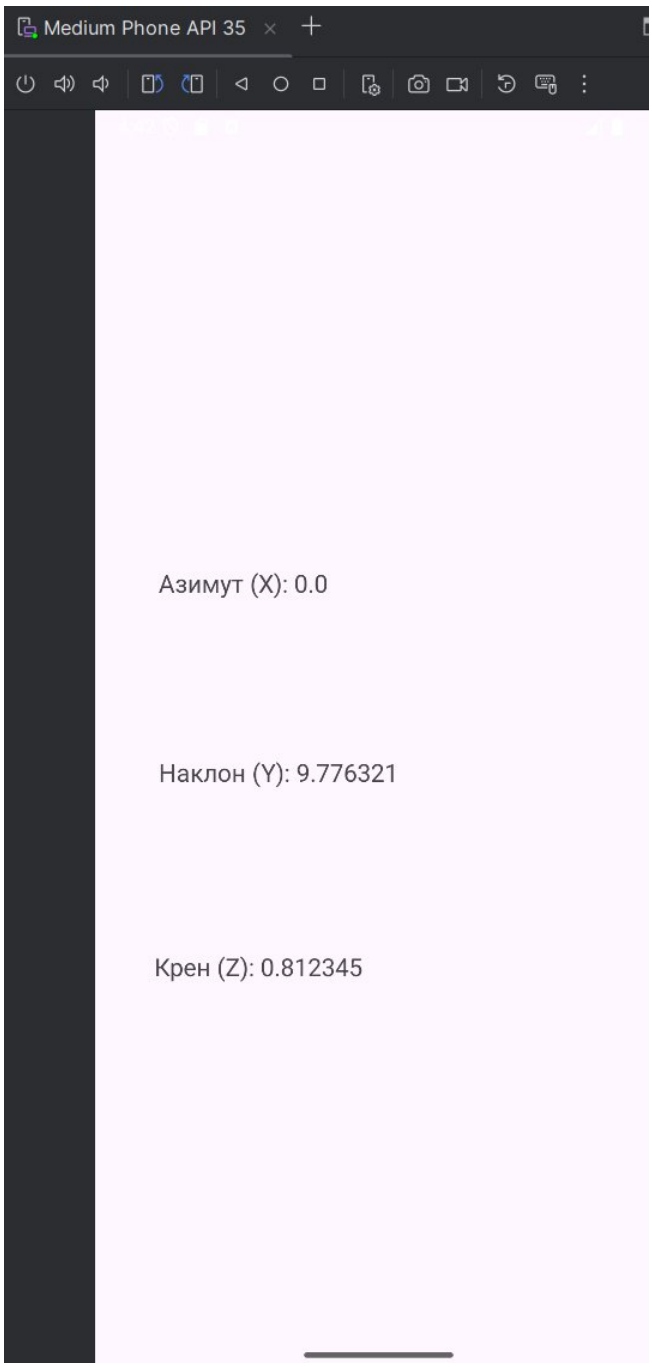
Создан новый проект ru.mirea.sda.lesson5. На главном экране отображается список всех доступных сенсоров устройства. Для этого используется SensorManager и getSensorList(Sensor.TYPE_ALL). Результаты выводятся в TextView через ViewBinding.



Задание 2. Акселерометр

Создан модуль Accelerometer. В интерфейсе размещены три TextView, отображающие значения по осям X, Y, Z. Реализован интерфейс SensorEventListener, зарегистрирован Sensor.TYPE_ACCELEROMETER.

```java
public class MainActivity extends AppCompatActivity implements SensorEventListener
    protected void onCreate(Bundle savedInstanceState) {


        sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        accelerometerSensor = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

        azimuthTextView = findViewById(R.id.textViewAzimuth);
        pitchTextView = findViewById(R.id.textViewPitch);
        rollTextView = findViewById(R.id.textViewRoll);
    }

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

    @Override
    protected void onResume() {
        super.onResume();
        if (accelerometerSensor != null)
        {
            sensorManager.registerListener( listener: this, accelerometerSensor, SensorManager.SENSOR_DELAY_NORMAL);
        }
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER)
        {
            float azimuth = event.values[0];  // Ось X
            float pitch = event.values[1];    // Ось Y
            float roll = event.values[2];     // Ось Z

            azimuthTextView.setText("Азимут (X): " + azimuth);
            pitchTextView.setText("Наклон (Y): " + pitch);
            rollTextView.setText("Крен (Z): " + roll);
        }
```

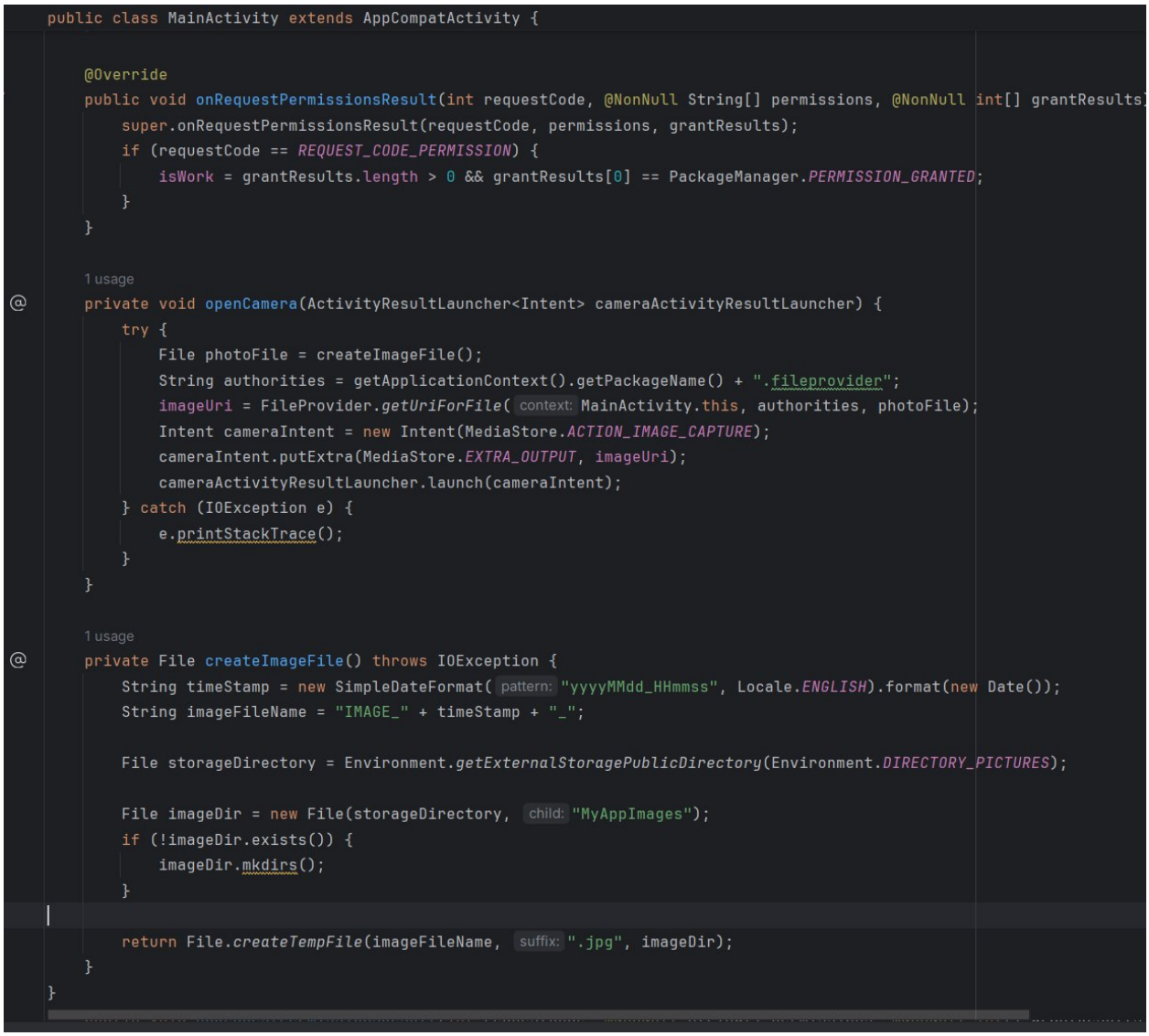


В onResume происходит регистрация слушателя, в onPause — его снятие. Метод onSensorChanged отслеживает изменения и обновляет отображаемые значения. При вращении устройства данные на экране изменяются в реальном времени.

Задание 3. Работа с камерой

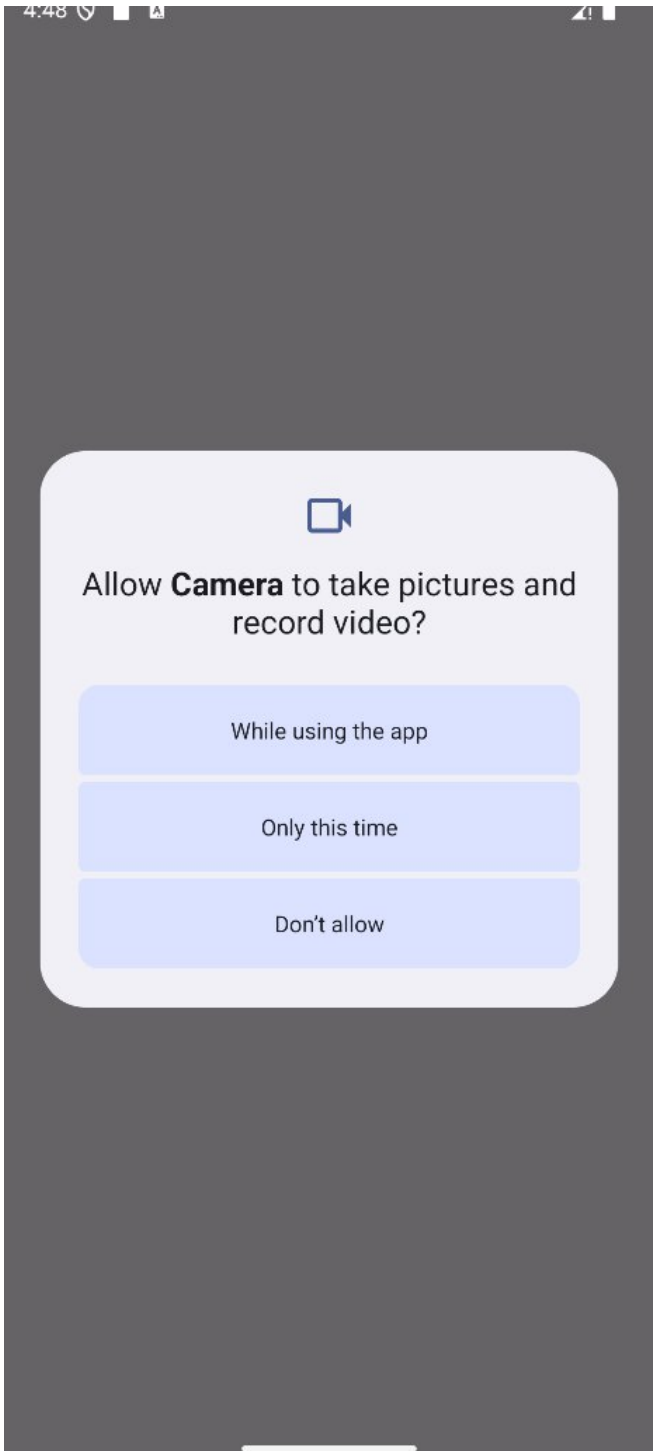
Создан модуль Camera. Интерфейс содержит кнопку и ImageView.

Реализован вызов системного приложения камеры с последующим сохранением изображения в директорию приложения. Создан файл paths.xml, добавлены разрешения на использование камеры и запись.



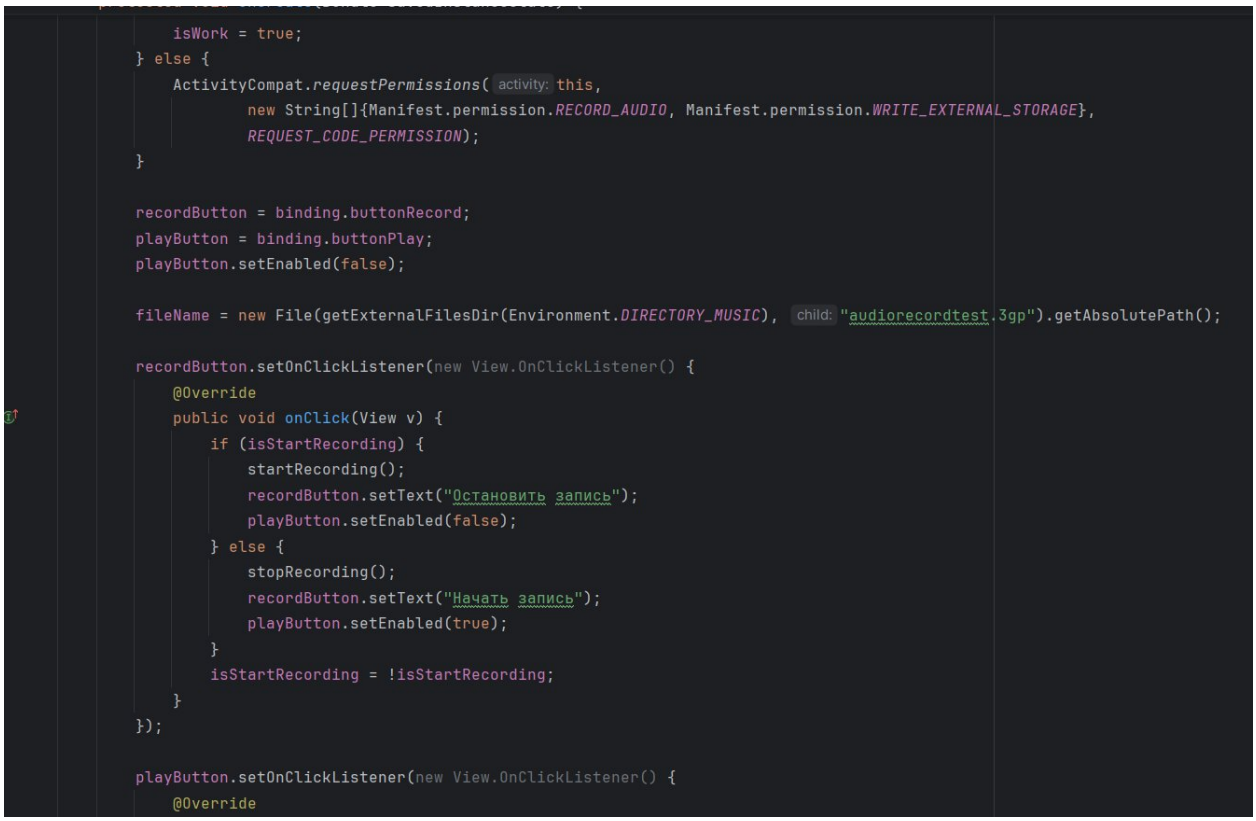
```
public class MainActivity extends AppCompatActivity {

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults)
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (requestCode == REQUEST_CODE_PERMISSION) {
            isWork = grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED;
        }
    }

    1 usage
    private void openCamera(ActivityResultLauncher<Intent> cameraActivityResultLauncher) {
        try {
            File photoFile = createImageFile();
            String authorities = getApplicationContext().getPackageName() + ".fileprovider";
            imageUri = FileProvider.getUriForFile( context: MainActivity.this, authorities, photoFile);
            Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
            cameraIntent.putExtra(MediaStore.EXTRA_OUTPUT, imageUri);
            cameraActivityResultLauncher.launch(cameraIntent);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    1 usage
    private File createImageFile() throws IOException {
        String timeStamp = new SimpleDateFormat( pattern: "yyyyMMdd_HHmmss", Locale.ENGLISH).format(new Date());
        String imageFileName = "IMAGE_" + timeStamp + "_";

        File storageDirectory = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES);

        File imageDir = new File(storageDirectory, child: "MyAppImages");
        if (!imageDir.exists()) {
            imageDir.mkdirs();
        }

        return File.createTempFile(imageFileName, suffix: ".jpg", imageDir);
    }
}
```

Изображение, сделанное пользователем, загружается и отображается в ImageView. Все операции выполнены через Intent, FileProvider и BitmapFactory.

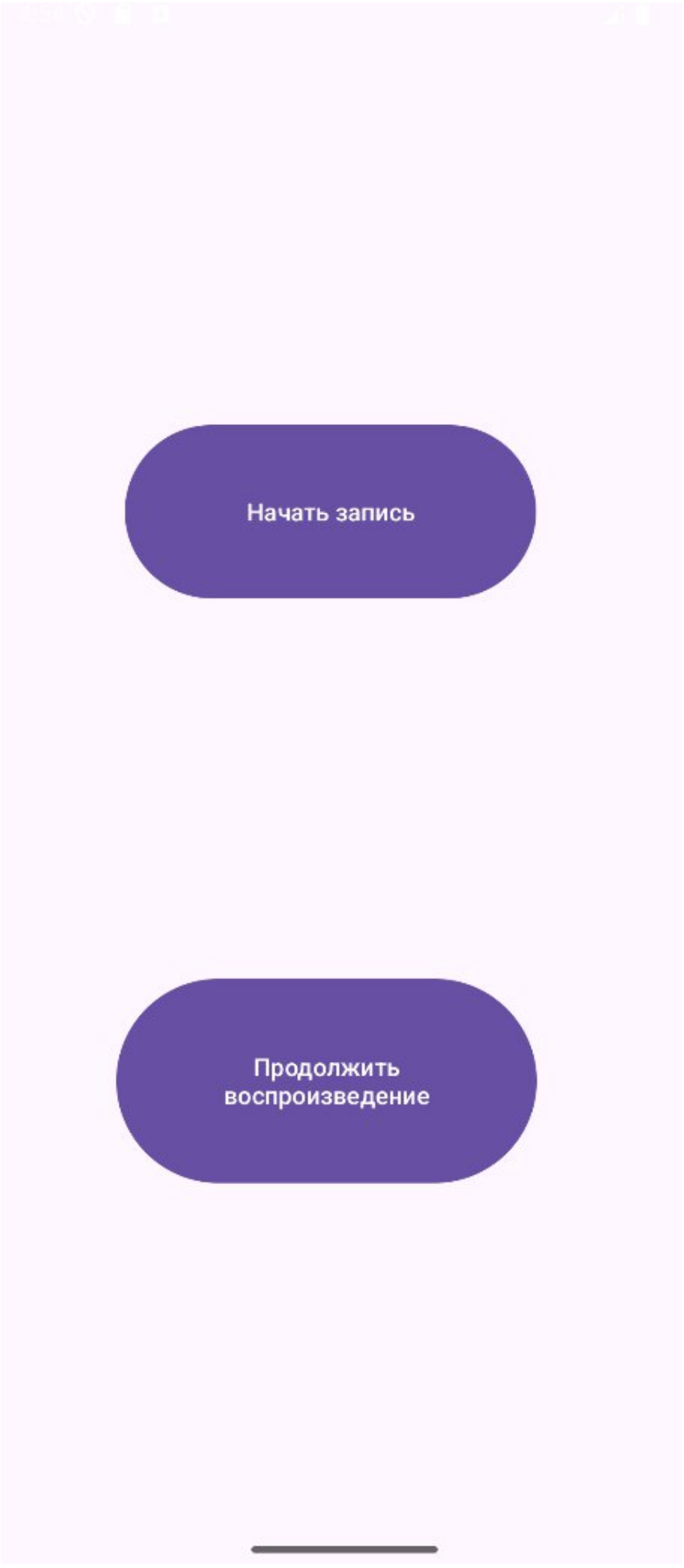


Задание 4. Аудиозапись и воспроизведение

Создан модуль AudioRecord. В activity_main.xml размещены кнопки "Записать" и "Воспроизвести". Добавлены разрешения для работы с микрофоном и файловой системой.



```
            isWork = true;
        } else {
            ActivityCompat.requestPermissions( activity: this,
                    new String[]{Manifest.permission.RECORD_AUDIO, Manifest.permission.WRITE_EXTERNAL_STORAGE},
                    REQUEST_CODE_PERMISSION);
        }

        recordButton = binding.buttonRecord;
        playButton = binding.buttonPlay;
        playButton.setEnabled(false);

        fileName = new File(getExternalFilesDir(Environment.DIRECTORY_MUSIC),  child: "audiorecordtest.3gp").getAbsolutePath();

        recordButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (isStartRecording) {
                    startRecording();
                    recordButton.setText("Остановить запись");
                    playButton.setEnabled(false);
                } else {
                    stopRecording();
                    recordButton.setText("Начать запись");
                    playButton.setEnabled(true);
                }
                isStartRecording = !isStartRecording;
            }
        });

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


Начать запись
Продолжить
воспроизведение

Запись осуществляется с использованием MediaRecorder, воспроизведение — через MediaPlayer. Интерфейс оформлен вручную, управление реализовано с использованием ViewBinding. При повторном запуске можно прослушать записанный фрагмент.