|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |
| Институт кибербезопасности и цифровых технологий |
| Кафедра КБ-14 «Цифровые технологии обработки данных» |

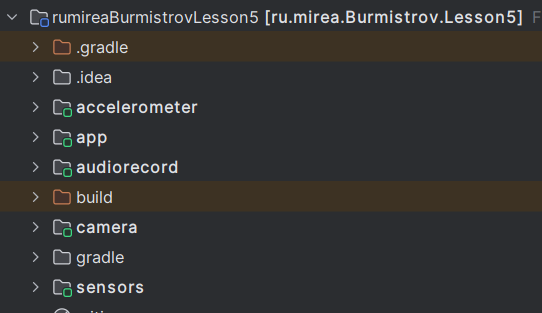
**ОТЧЕТ**

**по практической работе №5**

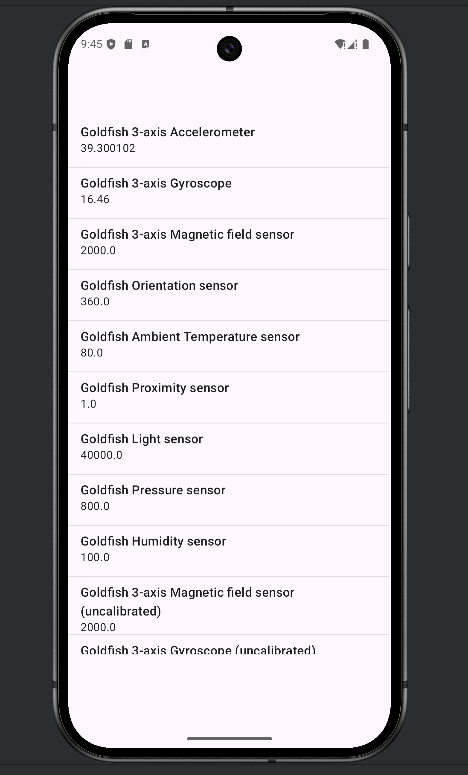
|  |  |  |  |
| --- | --- | --- | --- |
| Выполнил |  |  | Бурмистров И.Г.  *фамилия, имя, отчество* |
| шифр | 22Б0616 | группа | БСБО-07-22 |
|  |  |  |  |
| Проверил |  | *ученая степень, должность* | Изергин Д.А.  *фамилия, имя, отчество* |

**Москва 2025г.**

В ходе данной работы были созданы модули: «accelerometer», «audiorecord», «camera» и «sensors» (см. Рисунок 1).

  
Рисунок 1. Модули проекта

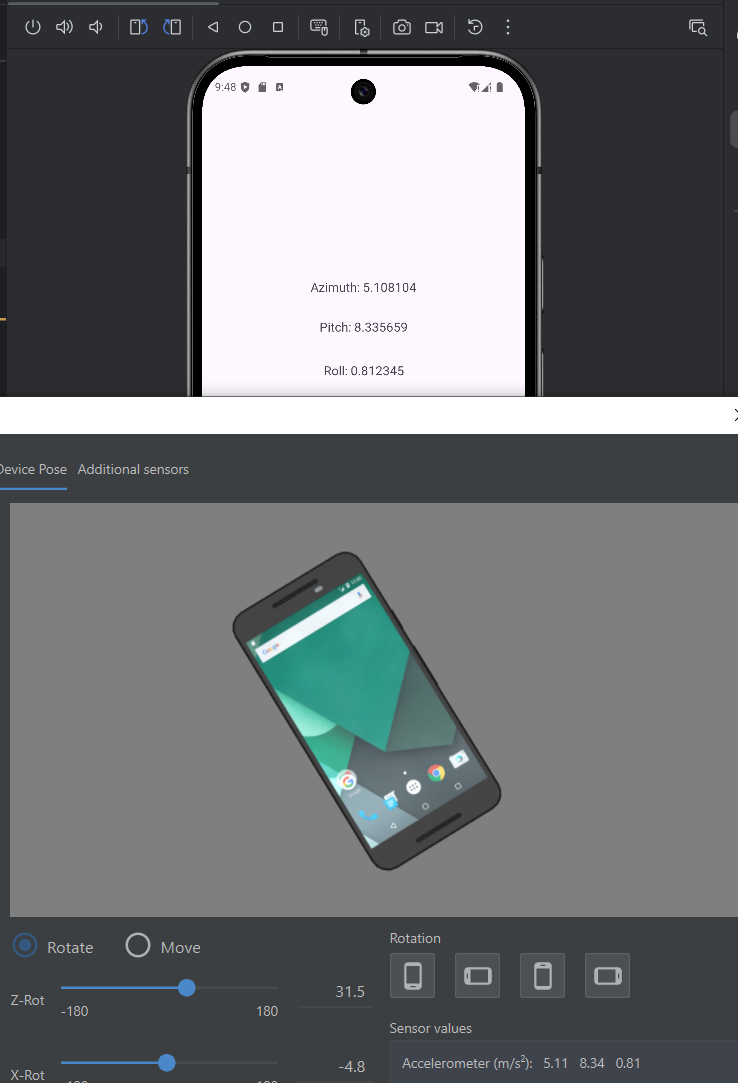
В первом модуле «sensors» был создан экран, в котором показываются все датчики телефона (см. Рисунок 2 и Листинг 1).

  
Рисунок 2. Список датчиков

public class MainActivity extends AppCompatActivity {  
 private ActivityMainBinding binding;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 EdgeToEdge.*enable*(this);  
 setContentView(R.layout.*activity\_main*);  
 ViewCompat.*setOnApplyWindowInsetsListener*(findViewById(R.id.*main*), (v, insets) -> {  
 Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.*systemBars*());  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);  
 return insets;  
 });  
  
 binding = ActivityMainBinding.*inflate*(getLayoutInflater());  
 setContentView(binding.getRoot());  
  
 SensorManager sensorManager = (SensorManager) getSystemService(Context.*SENSOR\_SERVICE*);  
 List<Sensor> sensors = sensorManager.getSensorList(Sensor.*TYPE\_ALL*);  
 ListView listSensor = binding.listView;  
  
 ArrayList<HashMap<String, Object>> arrayList = new ArrayList<>();  
 for (int i = 0; i <sensors.size(); i++){  
 HashMap<String, Object> sensorTypeList = new HashMap<>();  
 sensorTypeList.put("Name", sensors.get(i).getName());  
 sensorTypeList.put("Value", sensors.get(i).getMaximumRange());  
 arrayList.add(sensorTypeList);  
 }  
  
 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*});  
 listSensor.setAdapter(mHistory);  
  
 }  
}

Листинг 1. Вывод датчиков

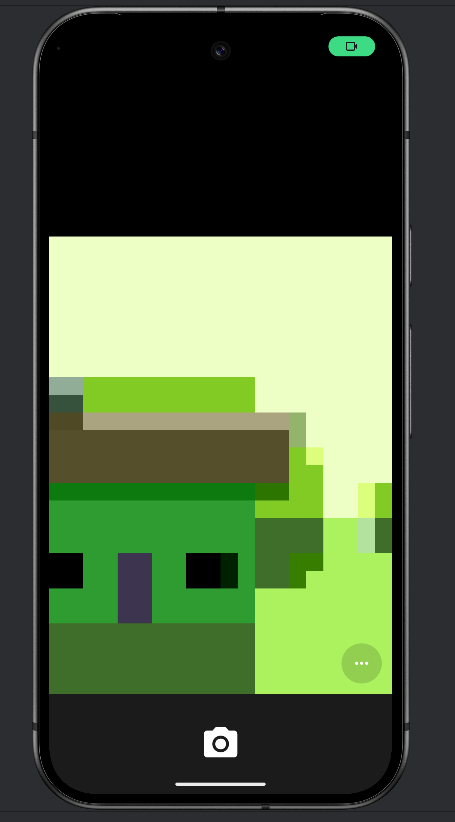
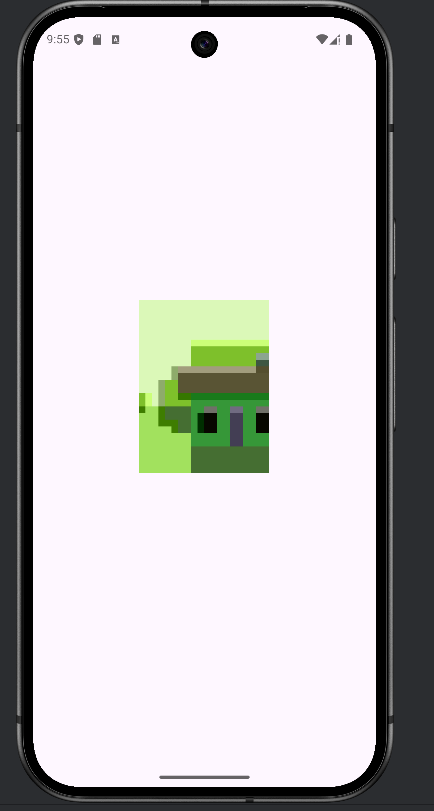
Далее был создан модуль «Accelerometer» в котором было реализовано приложение с тремя текстовыми полями, в которых показываются данные с акселерометра (см. Рисунок 3 и Листинг 2).

  
Рисунок 3. Данные с акселерометра

public class MainActivity extends AppCompatActivity implements SensorEventListener {  
 private ActivityMainBinding binding;  
 private SensorManager sensorManager;  
 private Sensor accelerometer;  
 private TextView azimuthTextView;  
 private TextView pitchTextView;  
 private TextView rollTextView;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 EdgeToEdge.*enable*(this);  
 setContentView(R.layout.*activity\_main*);  
 ViewCompat.*setOnApplyWindowInsetsListener*(findViewById(R.id.*main*), (v, insets) -> {  
 Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.*systemBars*());  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);  
 return insets;  
 });  
  
 binding = ActivityMainBinding.*inflate*(getLayoutInflater());  
 setContentView(binding.getRoot());  
  
 sensorManager = (SensorManager) getSystemService(*SENSOR\_SERVICE*);  
 accelerometer = sensorManager.getDefaultSensor(Sensor.*TYPE\_ACCELEROMETER*);  
 sensorManager.registerListener(this, accelerometer, SensorManager.*SENSOR\_DELAY\_NORMAL*);  
 azimuthTextView = binding.TextAzimuth;  
 pitchTextView = binding.textPitch;  
 rollTextView = binding.textRoll;  
  
 }  
  
 @Override  
 protected void onPause(){  
 super.onPause();  
 sensorManager.unregisterListener(this);  
 }  
  
 @Override  
 protected void onResume(){  
 super.onResume();  
 sensorManager.registerListener(this, accelerometer, SensorManager.*SENSOR\_DELAY\_NORMAL*);  
  
 }  
  
 @Override  
 public void onSensorChanged(SensorEvent event) {  
 if (event.sensor.getType() == Sensor.*TYPE\_ACCELEROMETER*) {  
 float x = event.values[0];  
 float y = event.values[1];  
 float z = event.values[2];  
 azimuthTextView.setText("Azimuth: " + x);  
 pitchTextView.setText("Pitch: " + y);  
 rollTextView.setText("Roll: " + z);  
 }  
  
 }  
  
 @Override  
 public void onAccuracyChanged(Sensor sensor, int accuracy) {  
  
 }  
}

Листинг 2. Класс экрана с данными

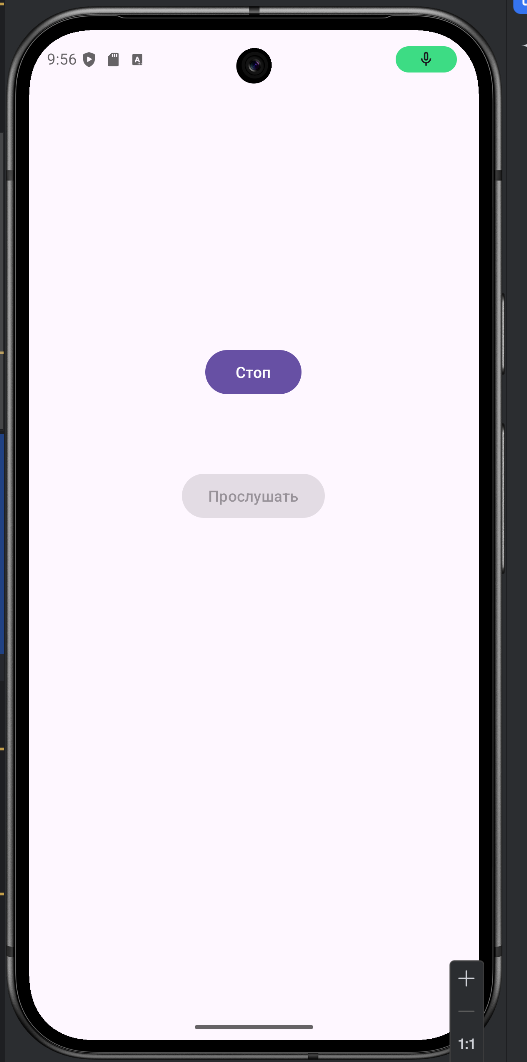
Далее был создан модуль «camera», в котором был реализован вызов системного приложения «камера», сохранение изображения в папку приложения и отображение снимка на экране. Камера вызывается при нажатии на изображение (см. Рисунок 4 и Листинг 3).

  
Рисунок 4. Пример работы модуля

public class MainActivity extends AppCompatActivity {  
 private ActivityMainBinding binding;  
 private static final int *REQUEST\_CODE\_PERMISSION* = 100;  
 private static final int *CAMERA\_REQUEST* = 0;  
 private Uri imageUri;  
 private boolean isWork;  
  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 EdgeToEdge.*enable*(this);  
 setContentView(R.layout.*activity\_main*);  
 ViewCompat.*setOnApplyWindowInsetsListener*(findViewById(R.id.*main*), (v, insets) -> {  
 Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.*systemBars*());  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);  
 return insets;  
 });  
   
 int CameraPermissionStatus = ContextCompat.*checkSelfPermission*(this, android.Manifest.permission.*CAMERA*);  
 int storagePermissionStatus = ContextCompat.*checkSelfPermission*(this, Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*); // на android 10+ не работает 🤷‍♂️  
 if(CameraPermissionStatus == PackageManager.*PERMISSION\_GRANTED*){  
 isWork = true;  
 } else {  
 ActivityCompat.*requestPermissions*(this, new String[] {android.Manifest.permission.*CAMERA*, Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*}, *REQUEST\_CODE\_PERMISSION*);  
 }  
  
  
 binding = ActivityMainBinding.*inflate*(getLayoutInflater());  
 setContentView(binding.getRoot());  
  
 ActivityResultCallback<ActivityResult> callback = new ActivityResultCallback<ActivityResult>() {  
 @Override  
 public void onActivityResult(ActivityResult o) {  
 if(o.getResultCode() == *RESULT\_OK*){  
 Intent data = o.getData();  
 binding.imageView.setImageURI(imageUri);  
  
 }  
 }  
 };  
 ActivityResultLauncher<Intent> cameraActivityResultLauncher = registerForActivityResult(  
 new ActivityResultContracts.StartActivityForResult(),  
 callback);  
 binding.imageView.setOnClickListener(new View.OnClickListener(){  
 @Override  
 public void onClick(View v){  
 Intent cameraIntent = new Intent(MediaStore.*ACTION\_IMAGE\_CAPTURE*);  
 if(isWork) {  
 try {  
 File photoFile = createImageFile();  
 String authorities = getApplicationContext().getPackageName() + ".fileprovider";  
 imageUri = FileProvider.*getUriForFile*(MainActivity.this, authorities, photoFile);  
 cameraIntent.putExtra(MediaStore.*EXTRA\_OUTPUT*, imageUri);  
 cameraActivityResultLauncher.launch(cameraIntent);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 });  
  
 }  
  
 @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> 1 && grantResults[0] == PackageManager.*PERMISSION\_GRANTED* && grantResults[1] == PackageManager.*PERMISSION\_GRANTED*;  
 if(!isWork){  
 Toast.*makeText*(this, "Permission denied", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 }  
 private File createImageFile() throws IOException{  
 String timeStamp = new SimpleDateFormat("yyyyMMdd\_HHmmss", Locale.*ENGLISH*).format(new Date());  
 String imageFileName = "IMAGE\_" + timeStamp + "\_";  
  
 File storageDirectory = getExternalFilesDir(Environment.*DIRECTORY\_PICTURES*);  
 return File.*createTempFile*(imageFileName, ".jpg", storageDirectory);  
  
 }  
  
}

Листинг 3. Класс для работы с камерой

Далее был создан модуль «AudioRecord» для работы с диктофоном. При нажатии на кнопку «Начать/остановить запись» начинается запись, если она не была включена. Затем при нажатии на «воспроизведение» начинается воспроизведение записи (см. Рисунок 5 и Листинг 4).

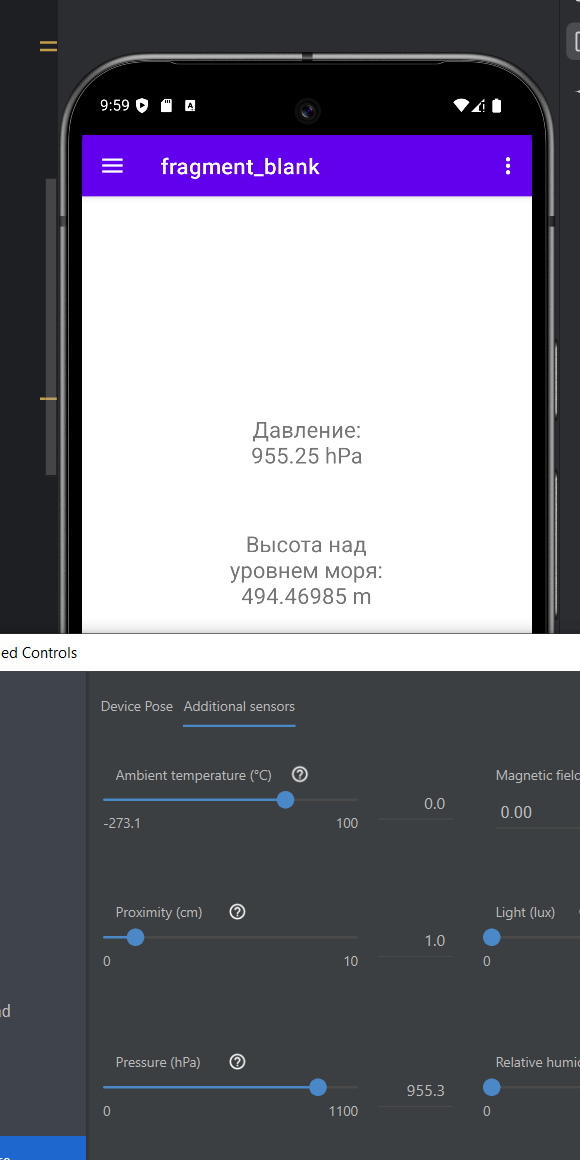
  
Рисунок 5. Пример работы модуля

public class MainActivity extends AppCompatActivity {  
 private ActivityMainBinding binding;  
 private static final int *REQUEST\_CODE\_PERMISSION* = 200;  
 //private final String TAG = MainActivity.class.getSimpleName();  
  
 private boolean isWork;  
 private String fileName = null;  
 private Button recordButton = null;  
 private Button playButton = null;  
 private MediaRecorder recorder = null;  
 private MediaPlayer player = null;  
 boolean isStartRecord = true;  
 boolean isStartPlay = true;  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 EdgeToEdge.*enable*(this);  
 setContentView(R.layout.*activity\_main*);  
 ViewCompat.*setOnApplyWindowInsetsListener*(findViewById(R.id.*main*), (v, insets) -> {  
 Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.*systemBars*());  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);  
 return insets;  
 });  
  
 binding = ActivityMainBinding.*inflate*(getLayoutInflater());  
 setContentView(binding.getRoot());  
  
 recordButton =binding.RecordButton;  
 playButton = binding.PlayButton;  
 playButton.setEnabled(false);  
  
 recordButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if(isStartRecord){  
 recordButton.setText("Стоп");  
 playButton.setEnabled(false);  
 startRecord();  
 } else{  
 recordButton.setText("Запись");  
 playButton.setEnabled(true);  
 stopRecord();  
 }  
 isStartRecord = !isStartRecord;  
  
 }  
 });  
  
 playButton.setOnClickListener(new View.OnClickListener(){  
 @Override  
 public void onClick(View v){  
 if(isStartPlay){  
 playButton.setText("Стоп");  
 recordButton.setEnabled(false);  
 startPlay();  
 } else {  
 playButton.setText("Воспроизведение");  
 recordButton.setEnabled(true);  
 stopPlay();  
 }  
 isStartPlay = !isStartPlay;  
 }  
 });  
  
  
  
  
 int audioRecordPermissionStatus = ContextCompat.*checkSelfPermission*(this, Manifest.permission.*RECORD\_AUDIO*);  
 if(audioRecordPermissionStatus == PackageManager.*PERMISSION\_GRANTED*){  
 isWork = true;  
 } else {  
 ActivityCompat.*requestPermissions*(this, new String[] {Manifest.permission.*RECORD\_AUDIO*}, *REQUEST\_CODE\_PERMISSION*);  
 }  
 }  
 @Override  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults){  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 switch(requestCode){  
 case *REQUEST\_CODE\_PERMISSION*:  
 isWork = grantResults.length > 0 && grantResults[0] == PackageManager.*PERMISSION\_GRANTED*;  
 break;  
 }  
 if(!isWork){  
 finish();  
 }  
 }  
  
 private void startRecord(){  
 fileName = getExternalFilesDir(null).getAbsolutePath() + "/audiorecord.3gp";  
 recorder = new MediaRecorder();  
 recorder.setAudioSource(MediaRecorder.AudioSource.*MIC*);  
 recorder.setOutputFormat(MediaRecorder.OutputFormat.*THREE\_GPP*);  
 recorder.setAudioEncoder(MediaRecorder.AudioEncoder.*AMR\_NB*);  
 recorder.setOutputFile(fileName);  
 try{  
 recorder.prepare();  
 recorder.start();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
  
  
 }  
  
 private void stopRecord(){  
 recorder.stop();  
 recorder.release();  
 recorder = null;  
 }  
  
 private void startPlay() {  
 player = new MediaPlayer();  
 try {  
 player.setDataSource(fileName);  
 player.prepare();  
 player.start();  
 } catch (Exception e) {  
 }  
 }  
  
 private void stopPlay() {  
 player.release();  
 player = null;  
 }  
  
}

Листинг 4. Класс для работы с диктофоном

Далее в проекте «MireaProject» **(В Lesson3)** были добавлены 3 фрагмента, а также был реализован механизм запроса разрешений.

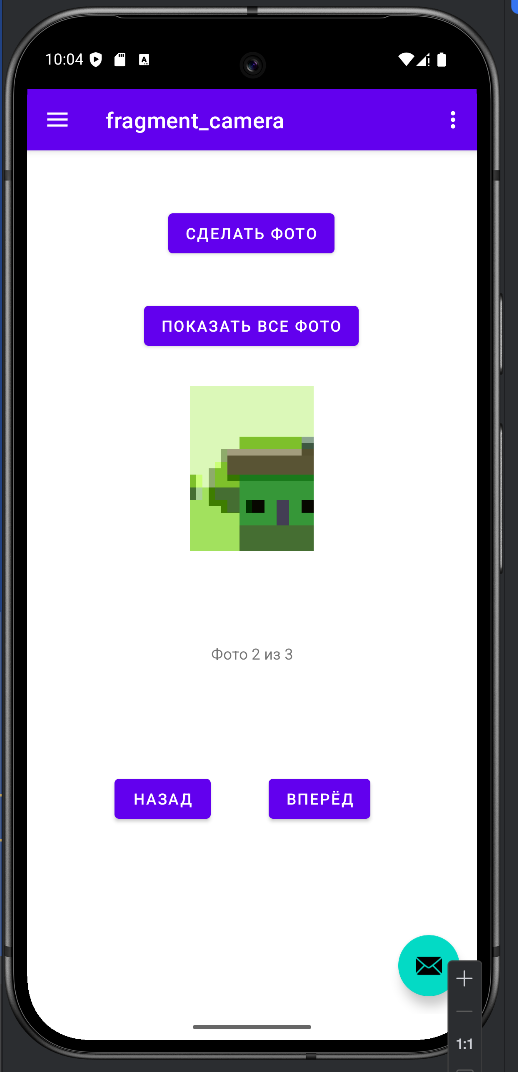
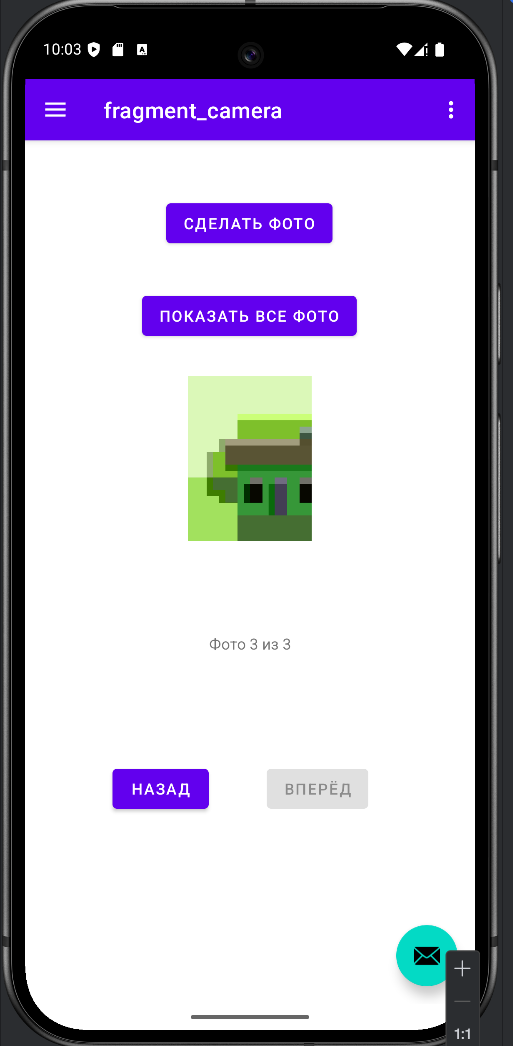
В первом фрагменте были использованы показания барометра для вычисления высоты над уровнем моря (см. Рисунок 6 и Листинг 5).

  
Рисунок 6. Барометр

public class Barometr extends Fragment implements SensorEventListener {  
 private FragmentBlankBinding binding;  
 private SensorManager sensorManager;  
 private Sensor barometer;  
 private TextView pressureTextView;  
 private TextView altitudeTextView;  
  
 private static final float *STANDARD\_PRESSURE* = 1013.25f; // Стандартное атмосферное давление  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 binding = FragmentBlankBinding.*inflate*(inflater, container, false);  
 return binding.getRoot();  
 }  
 @Override  
 public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {  
 super.onViewCreated(view, savedInstanceState);  
  
 sensorManager = (SensorManager) requireContext().getSystemService(*SENSOR\_SERVICE*);  
 barometer = sensorManager.getDefaultSensor(Sensor.*TYPE\_PRESSURE*);  
  
 if (barometer != null) {  
 sensorManager.registerListener(this, barometer, SensorManager.*SENSOR\_DELAY\_UI*);  
 }  
  
 pressureTextView = binding.TextViewPressure;  
 altitudeTextView = binding.TextViewAltitude;   
  
 }  
  
 @Override  
 public void onPause() {  
 super.onPause();  
 if (barometer != null) {  
 sensorManager.unregisterListener(this);  
 }  
 }  
  
 @Override  
 public void onResume() {  
 super.onResume();  
 if (barometer != null) {  
 sensorManager.registerListener(this, barometer, SensorManager.*SENSOR\_DELAY\_UI*);  
 }  
 }  
  
 @Override  
 public void onSensorChanged(SensorEvent event) {  
 if (event.sensor.getType() == Sensor.*TYPE\_PRESSURE*) {  
 float pressure = event.values[0];  
 pressureTextView.setText("Давление: " + pressure + " hPa");  
  
 // Вычисление высоты на основе давления  
 float altitude = calculateAltitude(pressure);  
 altitudeTextView.setText("Высота над уровнем моря: " + altitude + " m");  
 }  
 }  
 @Override  
 public void onAccuracyChanged(Sensor sensor, int accuracy) {  
   
 }  
  
 */\*\*  
 \* Вычисление высоты над уровнем моря на основе давления  
 \* @param pressure текущее атмосферное давление в гПа  
 \* @return высота в метрах  
 \*/* private float calculateAltitude(float pressure) {  
 return 44330 \* (1 - (float) Math.*pow*(pressure / *STANDARD\_PRESSURE*, 1 / 5.255));  
 }  
}

Листинг 5. Класс для барометра

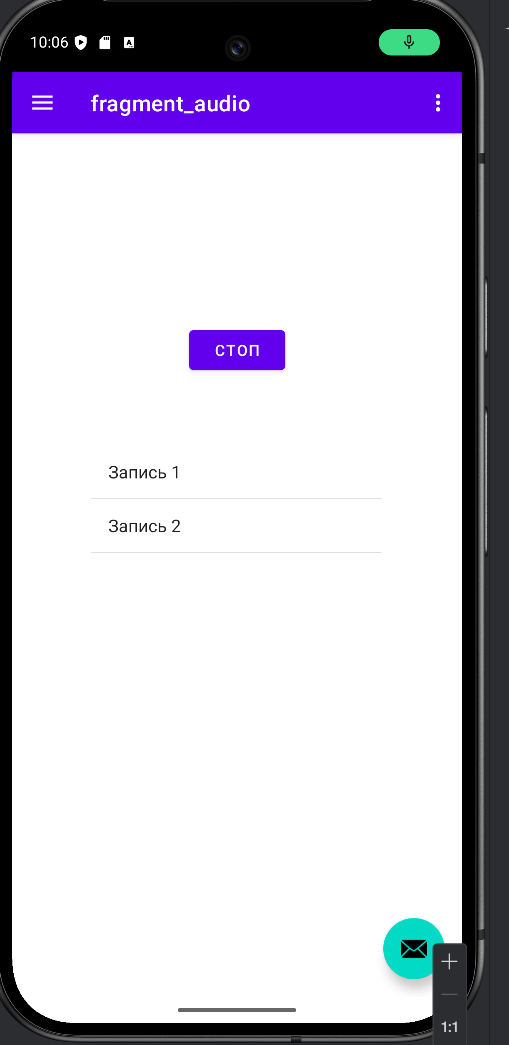
Во втором фрагменте была реализована простая галерея, пользователь по кнопке может вызвать камеру, сделать фото, а потом посмотреть сделанные фотографии, листая по одной (см. Рисунок 7 и Листинг 6).

  
Рисунок 7. Пример работы заметки

public class Camera extends Fragment {  
  
 private FragmentCameraBinding binding;  
 private Uri imageUri;  
 private boolean isWork = false;  
 private final ArrayList<Uri> photoUris = new ArrayList<>();  
 private int currentPhotoIndex = -1; // Индекс текущего отображаемого фото  
  
 private ActivityResultLauncher<String[]> permissionLauncher;  
 private ActivityResultLauncher<Intent> cameraActivityResultLauncher;  
  
 @Override  
 public void onCreate(@Nullable Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 permissionLauncher = registerForActivityResult(  
 new ActivityResultContracts.RequestMultiplePermissions(),  
 result -> {  
 Boolean cameraGranted = result.getOrDefault(Manifest.permission.*CAMERA*, false);  
 isWork = cameraGranted != null && cameraGranted;  
 if (!isWork) {  
 Toast.*makeText*(requireContext(), "Camera permission denied", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
  
 cameraActivityResultLauncher = registerForActivityResult(  
 new ActivityResultContracts.StartActivityForResult(),  
 result -> {  
 if (result.getResultCode() == getActivity().*RESULT\_OK*) {  
 photoUris.add(imageUri);  
 currentPhotoIndex = photoUris.size() - 1;  
 updatePhotoDisplay();  
 }  
 });  
 }  
  
 @Nullable  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater,  
 ViewGroup container,  
 Bundle savedInstanceState) {  
 binding = FragmentCameraBinding.*inflate*(inflater, container, false);  
 return binding.getRoot();  
 }  
  
 @Override  
 public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {  
 super.onViewCreated(view, savedInstanceState);  
  
 checkPermissions();  
  
 // Кнопка сделать фото  
 binding.buttonTakePhoto.setOnClickListener(v -> {  
 if (isWork) {  
 openCamera();  
 } else {  
 Toast.*makeText*(requireContext(), "Permissions not granted", Toast.*LENGTH\_SHORT*).show();  
 checkPermissions();  
 }  
 });  
  
 binding.buttonPrevPhoto.setOnClickListener(v -> showPreviousPhoto());  
 binding.buttonNextPhoto.setOnClickListener(v -> showNextPhoto());  
  
 updatePhotoDisplay();  
 }  
  
 private void checkPermissions() {  
 boolean cameraPermission = ContextCompat.*checkSelfPermission*(requireContext(), Manifest.permission.*CAMERA*) == PackageManager.*PERMISSION\_GRANTED*;  
 if (cameraPermission) {  
 isWork = true;  
 } else {  
 permissionLauncher.launch(new String[]{Manifest.permission.*CAMERA*});  
 }  
 }  
 private void openCamera() {  
 try {  
 File photoFile = createImageFile();  
 String authorities = requireContext().getPackageName() + ".fileprovider";  
 imageUri = FileProvider.*getUriForFile*(requireContext(), authorities, photoFile);  
 Intent cameraIntent = new Intent(MediaStore.*ACTION\_IMAGE\_CAPTURE*);  
 cameraIntent.putExtra(MediaStore.*EXTRA\_OUTPUT*, imageUri);  
 cameraActivityResultLauncher.launch(cameraIntent);  
 } catch (IOException e) {  
 e.printStackTrace();  
 Toast.*makeText*(requireContext(), "Error creating file", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 private File createImageFile() throws IOException {  
 String timeStamp = new SimpleDateFormat("yyyyMMdd\_HHmmss", Locale.*ENGLISH*).format(new Date());  
 String imageFileName = "IMAGE\_" + timeStamp + "\_";  
 File storageDirectory = requireContext().getExternalFilesDir(Environment.*DIRECTORY\_PICTURES*);  
 return File.*createTempFile*(imageFileName, ".jpg", storageDirectory);  
 }  
  
 private void updatePhotoDisplay() {  
 if (photoUris.isEmpty()) {  
 binding.imageView.setImageResource(android.R.drawable.*ic\_menu\_camera*); // дефолтное изображение  
 binding.textViewPhotoCount.setText("Фото 0 из 0");  
 binding.buttonPrevPhoto.setEnabled(false);  
 binding.buttonNextPhoto.setEnabled(false);  
 } else {  
 binding.imageView.setImageURI(photoUris.get(currentPhotoIndex));  
 binding.textViewPhotoCount.setText("Фото " + (currentPhotoIndex + 1) + " из " + photoUris.size());  
 binding.buttonPrevPhoto.setEnabled(currentPhotoIndex > 0);  
 binding.buttonNextPhoto.setEnabled(currentPhotoIndex < photoUris.size() - 1);  
 }  
 }  
  
 private void showPreviousPhoto() {  
 if (currentPhotoIndex > 0) {  
 currentPhotoIndex--;  
 updatePhotoDisplay();  
 }  
 }  
  
 private void showNextPhoto() {  
 if (currentPhotoIndex < photoUris.size() - 1) {  
 currentPhotoIndex++;  
 updatePhotoDisplay();  
 }  
 }  
  
 @Override  
 public void onDestroyView() {  
 super.onDestroyView();  
 binding = null;  
 }  
}

Листинг 6. Код для работы камеры

Затем был сделан третий фрагмент, в котором был реализован функционал диктофона, пользователь может записать аудио, а потом прослушать нужное из списка (см. Рисунок 8 и Листинг 7).

  
Рисунок 8. Диктофон

public class Audio extends Fragment {  
  
 private FragmentAudioBinding binding;  
 private MediaRecorder recorder;  
 private MediaPlayer player;  
 private boolean isRecording = false;  
 private boolean isPlaying = false;  
 private String currentFilePath;  
 private final ArrayList<String> recordingsPaths = new ArrayList<>();  
 private final ArrayList<String> recordingsNames = new ArrayList<>();  
 private ArrayAdapter<String> adapter;  
  
 private ActivityResultLauncher<String> permissionLauncher;  
 private boolean hasRecordAudioPermission = false;  
  
 @Override  
 public void onCreate(@Nullable Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 permissionLauncher = registerForActivityResult(  
 new ActivityResultContracts.RequestPermission(),  
 granted -> {  
 hasRecordAudioPermission = granted;  
 if (!granted) {  
 Toast.*makeText*(requireContext(), "Permission denied. Closing fragment.", Toast.*LENGTH\_SHORT*).show();  
 requireActivity().getSupportFragmentManager().popBackStack();  
 }  
 });  
  
 checkPermission();  
 }  
 private void checkPermission() {  
 hasRecordAudioPermission = ContextCompat.*checkSelfPermission*(requireContext(), Manifest.permission.*RECORD\_AUDIO*)  
 == android.content.pm.PackageManager.*PERMISSION\_GRANTED*;  
  
 if (!hasRecordAudioPermission) {  
 permissionLauncher.launch(Manifest.permission.*RECORD\_AUDIO*);  
 }  
 }  
 @Nullable  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater,  
 @Nullable ViewGroup container,  
 @Nullable Bundle savedInstanceState) {  
 binding = FragmentAudioBinding.*inflate*(inflater, container, false);  
 return binding.getRoot();  
 }  
  
 @Override  
 public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {  
 super.onViewCreated(view, savedInstanceState);  
  
 setupListView();  
 setupButtons();  
 loadExistingRecordings();  
 }  
 private void setupListView() {  
 adapter = new ArrayAdapter<>(requireContext(),  
 android.R.layout.*simple\_list\_item\_1*,  
 recordingsNames);  
 binding.listViewRecordings.setAdapter(adapter);  
  
 binding.listViewRecordings.setOnItemClickListener((AdapterView<?> parent, View view, int position, long id) -> {  
 if (isPlaying) {  
 stopPlaying();  
 }  
 String filePath = recordingsPaths.get(position);  
 startPlaying(filePath);  
 });  
 }  
 private void setupButtons() {  
 binding.recordButton.setText("Запись");  
 binding.recordButton.setOnClickListener(v -> {  
 if (!hasRecordAudioPermission) {  
 Toast.*makeText*(requireContext(), "Нет разрешения на запись аудио", Toast.*LENGTH\_SHORT*).show();  
 checkPermission();  
 return;  
 }  
 if (!isRecording) {  
 startRecording();  
 } else {  
 stopRecording();  
 }  
 });  
 }  
  
 private void loadExistingRecordings() {  
 recordingsPaths.clear();  
 recordingsNames.clear();  
  
 File dir = requireContext().getExternalFilesDir(Environment.*DIRECTORY\_MUSIC*);  
 if (dir != null && dir.exists()) {  
 File[] files = dir.listFiles();  
 if (files != null) {  
 for (int i = 0; i < files.length; i++) {  
 recordingsPaths.add(files[i].getAbsolutePath());  
 recordingsNames.add("Запись " + (i + 1));  
 }  
 }  
 }  
 adapter.notifyDataSetChanged();  
 }  
  
 private void startRecording() {  
 try {  
 File dir = requireContext().getExternalFilesDir(Environment.*DIRECTORY\_MUSIC*);  
 if (dir == null) {  
 Toast.*makeText*(requireContext(), "Ошибка доступа к хранилищу", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
 String timeStamp = new SimpleDateFormat("yyyyMMdd\_HHmmss", Locale.*getDefault*()).format(new Date());  
 currentFilePath = dir.getAbsolutePath() + "/record\_" + timeStamp + ".3gp";  
  
 recorder = new MediaRecorder();  
 recorder.setAudioSource(MediaRecorder.AudioSource.*MIC*);  
 recorder.setOutputFormat(MediaRecorder.OutputFormat.*THREE\_GPP*);  
 recorder.setAudioEncoder(MediaRecorder.AudioEncoder.*AMR\_NB*);  
 recorder.setOutputFile(currentFilePath);  
  
 recorder.prepare();  
 recorder.start();  
  
 isRecording = true;  
 binding.recordButton.setText("Стоп");  
 binding.listViewRecordings.setEnabled(false);  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 Toast.*makeText*(requireContext(), "Ошибка при старте записи", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 private void stopRecording() {  
 try {  
 recorder.stop();  
 recorder.release();  
 recorder = null;  
  
 isRecording = false;  
 binding.recordButton.setText("Запись");  
 binding.listViewRecordings.setEnabled(true);  
  
 // Добавляем запись в список и обновляем ListView  
 recordingsPaths.add(currentFilePath);  
 recordingsNames.add("Запись " + recordingsPaths.size());  
 adapter.notifyDataSetChanged();  
  
 } catch (RuntimeException e) {  
 e.printStackTrace();  
 Toast.*makeText*(requireContext(), "Ошибка при остановке записи", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 private void startPlaying(String filePath) {  
 try {  
 player = new MediaPlayer();  
 player.setDataSource(filePath);  
 player.prepare();  
 player.start();  
 isPlaying = true;  
  
 binding.recordButton.setEnabled(false);  
  
 player.setOnCompletionListener(mp -> {  
 stopPlaying();  
 });  
  
 Toast.*makeText*(requireContext(), "Воспроизведение записи", Toast.*LENGTH\_SHORT*).show();  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 Toast.*makeText*(requireContext(), "Ошибка воспроизведения", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 private void stopPlaying() {  
 if (player != null) {  
 player.release();  
 player = null;  
 isPlaying = false;  
 binding.recordButton.setEnabled(true);  
 Toast.*makeText*(requireContext(), "Воспроизведение остановлено", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 @Override  
 public void onDestroyView() {  
 super.onDestroyView();  
 if (recorder != null) {  
 recorder.release();  
 recorder = null;  
 }  
 if (player != null) {  
 player.release();  
 player = null;  
 }  
 binding = null;  
 }  
}

Листинг 7. Код для работы диктофона