

Практическая работа №6

Работа с хранилищем данных, безопасными настройками, внешними файлами и базой данных Room

Задание 1. Внутреннее хранилище

Создан модуль `internalfilestorage`. На экране размещены поле ввода (`EditText`) и кнопка (`Button`). Реализовано сохранение и загрузка пользовательского текста во внутреннем хранилище устройства с использованием методов `openFileOutput` и `openFileInput`.

```
package ru.mirea.sda.internalfilestorage;

import ...

public class MainActivity extends AppCompatActivity {

    2 usages
    private static final String LOG_TAG = MainActivity.class.getSimpleName();
    2 usages
    private static final String FILE_NAME = "data.txt";

    3 usages
    private EditText text;
    2 usages
    private Button save;

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

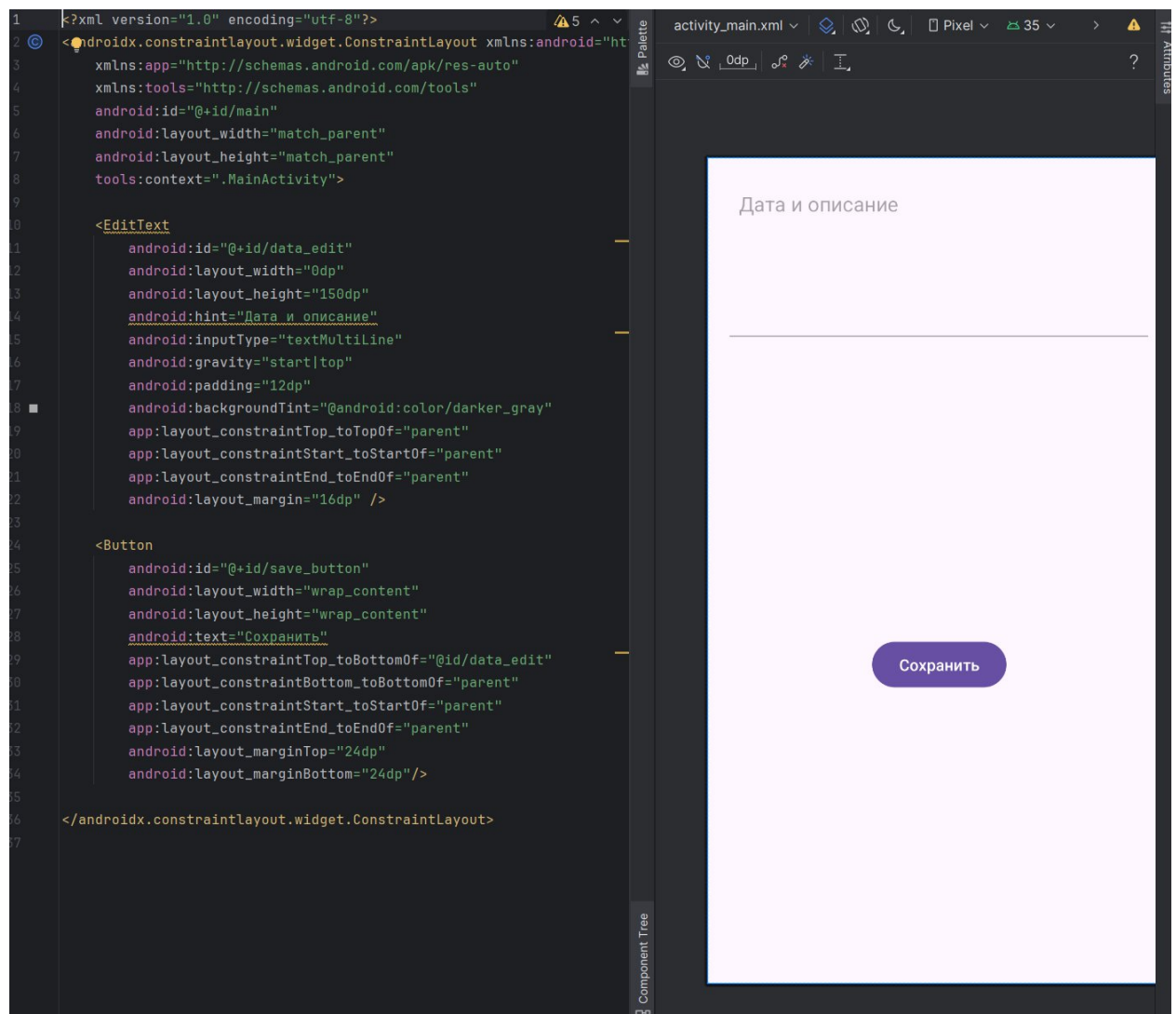
        text = findViewById(R.id.data_edit);
        save = findViewById(R.id.save_button);

        try (FileInputStream inputStream = openFileInput(FILE_NAME)) {
            byte[] bytes = new byte[inputStream.available()];
            inputStream.read(bytes);
            text.setText(new String(bytes));
        } catch (Exception e) {
            Log.e(LOG_TAG, msg: "Ошибка при чтении файла", e);
        }

        save.setOnClickListener( View v -> {
            String data = text.getText().toString();
            try (FileOutputStream outputStream = openFileOutput(FILE_NAME, Context.MODE_PRIVATE)) {
                outputStream.write(data.getBytes());
                Toast.makeText( context: MainActivity.this, text: "Данные сохранены", Toast.LENGTH_LONG).show();
            } catch (Exception e) {
                Log.e(LOG_TAG, msg: "Ошибка при записи файла", e);
                Toast.makeText( context: MainActivity.this, text: "Ошибка при сохранении данных", Toast.LENGTH_LONG).show();
            }
        });
    }
}
```

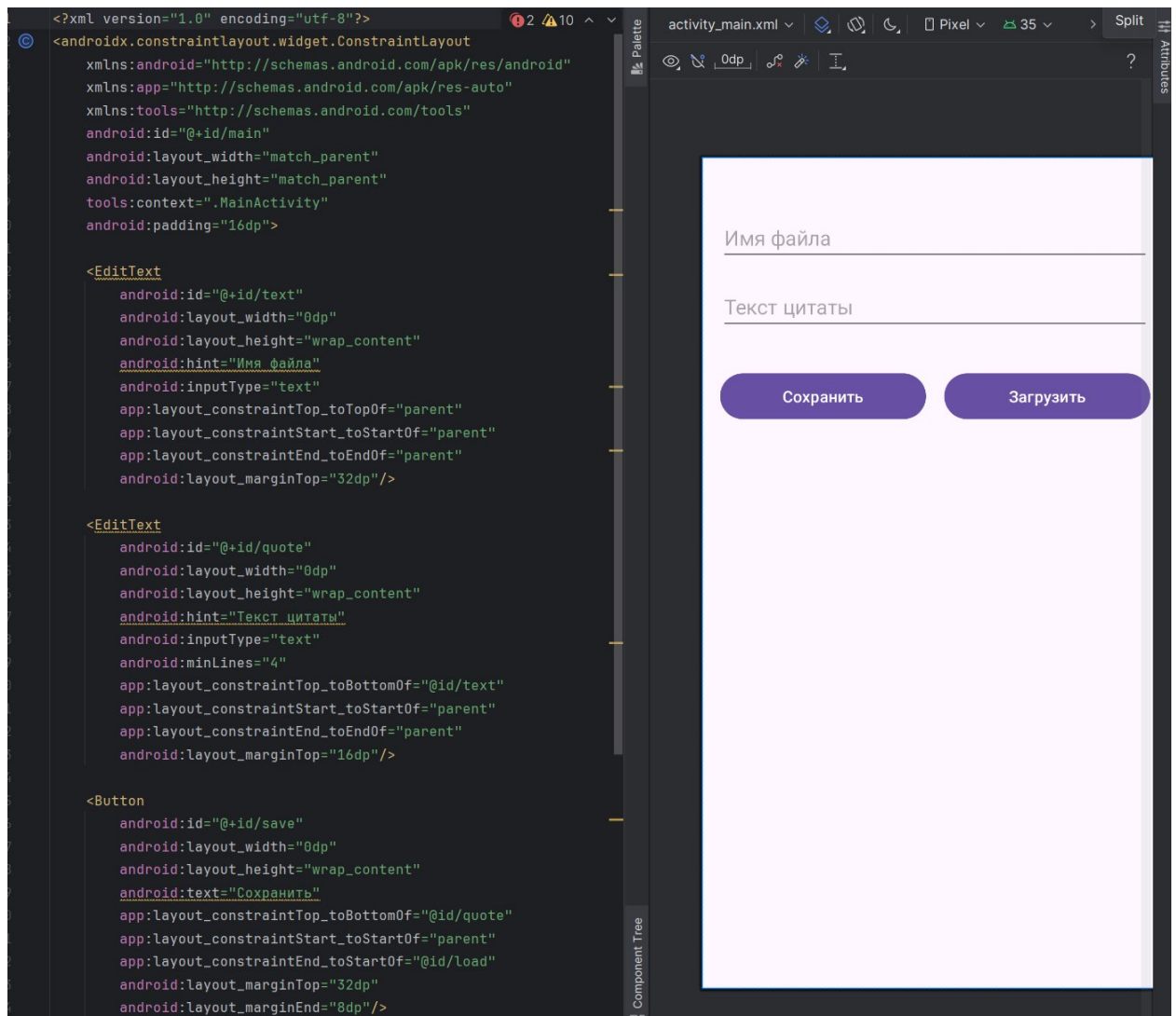
Файл data.txt создаётся в директории приложения
/data/data/package_name/files. При запуске приложения содержимое файла
автоматически загружается в поле ввода, а при нажатии на кнопку —

сохраняется текущий текст. Использован `findViewById`, взаимодействие с UI осуществляется без применения `ViewBinding`.



Задание 2. Внешнее хранилище

Модуль `notebook` реализует приложение-блокнот. Интерфейс включает два поля ввода: имя файла и текст цитаты, а также две кнопки — «Сохранить» и «Загрузить».



Сохранение выполняется в директорию Documents на внешнем хранилище с использованием `Environment.getExternalStoragePublicDirectory`. Программа проверяет наличие директории и при необходимости создаёт её. Запись осуществляется через `FileOutputStream`.

```

13 public class MainActivity extends AppCompatActivity {
14     private void handleSave() {
15     }
16
17     File targetFile = new File(directory, fileName);
18
19     try (FileOutputStream fos = new FileOutputStream(targetFile)) {
20         fos.write(content.getBytes());
21         notifyUser("Файл сохранён: " + targetFile.getAbsolutePath());
22     } catch (IOException e) {
23         notifyUser("Ошибка сохранения файла: " + e.getMessage());
24     }
25 }
26
27 1 usage
28 private void handleLoad() {
29     String fileName = fileName.getText().toString();
30
31     if (fileName.trim().isEmpty()) {
32         notifyUser("Введите имя файла");
33         return;
34     }
35
36     File dir = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOCUMENTS);
37     File sourceFile = new File(dir, fileName);
38
39     if (!sourceFile.exists()) {
40         notifyUser("Файл не найден");
41         return;
42     }
43
44     StringBuilder builder = new StringBuilder();
45     try (BufferedReader reader = new BufferedReader(new InputStreamReader(new FileInputStream(sourceFile), charsetName: "UTF-8"))) {
46         String line;
47         while ((line = reader.readLine()) != null) {
48             builder.append(line).append('\n');
49         }
50         Quote.setText(builder.toString().trim());
51         notifyUser("Данные успешно загружены");
52     } catch (IOException e) {
53         notifyUser("Ошибка загрузки: " + e.getMessage());
54     }
55 }
56
57 }

```

Для чтения данных используется `BufferedReader` с `InputStreamReader`. При нажатии кнопки «Загрузить» содержимое файла подставляется в поле ввода цитаты. Добавлены уведомления через `Toast` и обработка ошибок. Работа с UI реализована через `findViewById`, `ViewBinding` не используется.

Задание 3. `SharedPreferences` и `EncryptedSharedPreferences` (`securesharedpreferences`)

Создан модуль `securesharedpreferences`, демонстрирующий безопасное хранение и отображение пользовательских данных. На главном экране отображается имя пользователя, сохранённое с помощью `EncryptedSharedPreferences`, и его фотография, загруженная из ресурсов `raw`.

```

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

        name = findViewById(R.id.name);
        photo = findViewById(R.id.photo);

        try {
            String mainKeyAlias = MasterKeys.getOrCreate(MasterKeys.AES256_GCM_SPEC);

            SharedPreferences securePrefs = EncryptedSharedPreferences.create(
                fileName: "secret_shared_prefs",
                mainKeyAlias,
                getBaseContext(),
                EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,
                EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM
            );

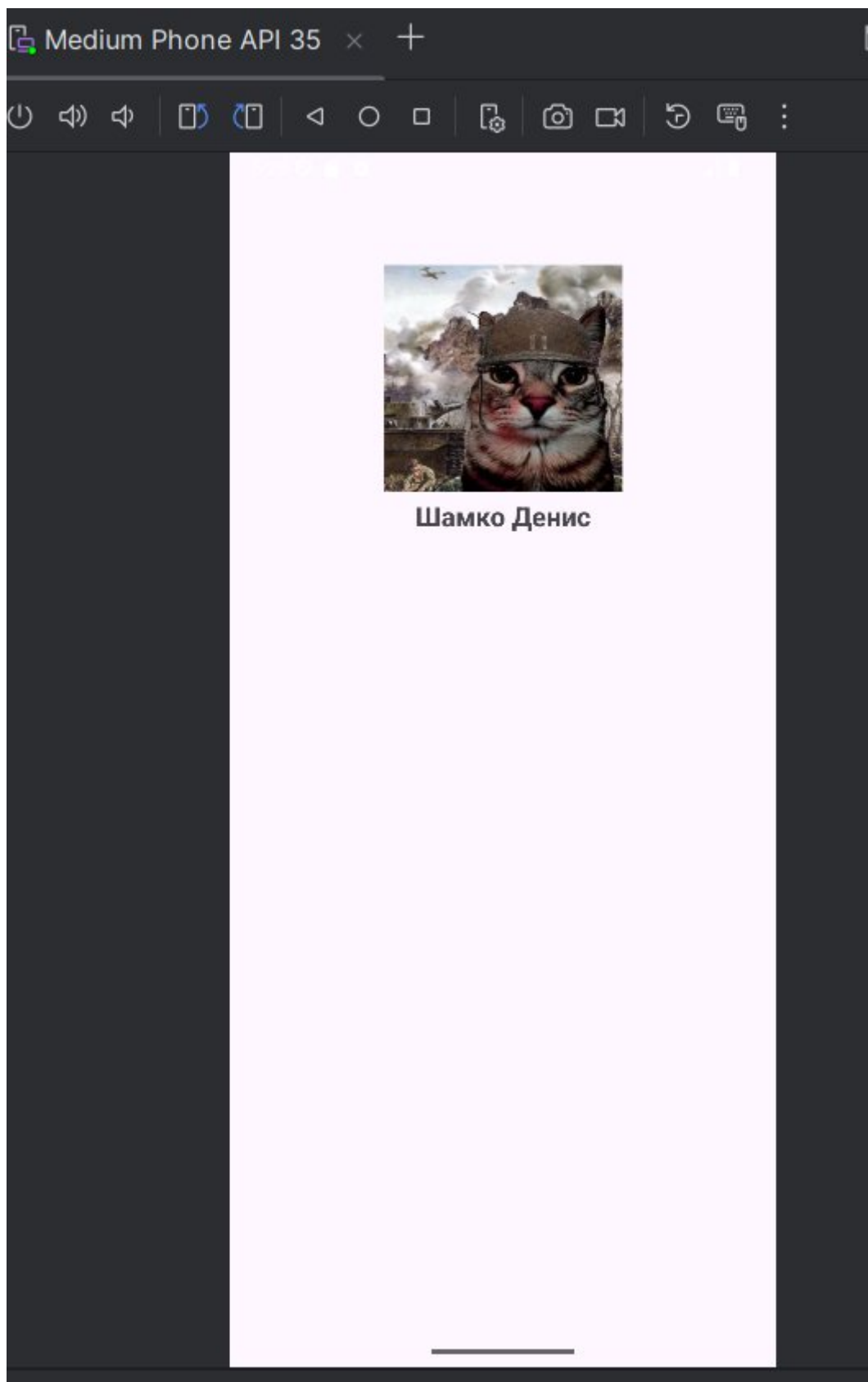
            String savedName = securePrefs.getString(key: "secure", defValue: "Шамко Денис");
            name.setText(savedName);

            SharedPreferences.Editor editor = securePrefs.edit();
            editor.putString("secure", "Шамко Денис");
            editor.putString("kot", "kot");
            editor.apply();

            String imageKey = securePrefs.getString(key: "kot", defValue: "kot");
            int imageResId = getResources().getIdentifier(imageKey, defType: "raw", getPackageName());
            InputStream inputStream = getResources().openRawResource(imageResId);
            Drawable drawable = Drawable.createFromStream(inputStream, imageKey);
            photo.setImageDrawable(drawable);
        } catch (Exception e) {
            e.printStackTrace();
            name.setText("Ошибка загрузки данных");
        }
    }
}

```

В MainActivity создаётся мастер-ключ через MasterKeys.getOrCreate, после чего данные сохраняются в файл secret_shared_prefs. При запуске приложения из файла извлекается имя и ключ изображения, которые используются для отображения текста и фотографии в ImageView.



Визуальный интерфейс реализован вручную через `findViewById`, используется `TextView` и `ImageView`. Данные шифруются с помощью схем `AES256_SIV` (ключи) и `AES256_GCM` (значения).

Задание 4. Room Database (employeeedb)

Модуль employeeedb реализует хранение информации о студентах с использованием библиотеки Room:

- @Entity класс Student содержит поля id, name и group;
- @Dao интерфейс StudentDao реализует методы getAll(), getById(id), insert, update, delete;
- @Database класс AppDatabase управляет базой данных.

В MainActivity реализованы функции добавления, обновления, удаления и отображения списка студентов в RecyclerView. Выбор элемента из списка позволяет редактировать его содержимое.

```
107         update.setOnClickListener( View v -> {
111             }
112             String name = studentName.getText().toString().trim();
113             String group = studentGroup.getText().toString().trim();
114             if (name.isEmpty()) {
115                 Toast.makeText( context: this, text: "Введите имя студента", Toast.LENGTH_LONG).show();
116                 return;
117             }
118             Student student = dao.getById(selectedStudentId);
119             if (student == null) {
120                 Toast.makeText( context: this, text: "Студент не найден", Toast.LENGTH_LONG).show();
121                 return;
122             }
123             student.name = name;
124             student.group = group;
125             dao.update(student);
126             Toast.makeText( context: this, text: "Студент обновлён", Toast.LENGTH_LONG).show();
127             clearFields();
128             selectedStudentId = -1;
129             loadStudentsToList();
130         });
131
132         delete.setOnClickListener( View v -> {
133             if (selectedStudentId == -1) {
134                 Toast.makeText( context: this, text: "Выберите студента для удаления", Toast.LENGTH_LONG).show();
135                 return;
136             }
137             Student student = dao.getById(selectedStudentId);
138             if (student == null) {
139                 Toast.makeText( context: this, text: "Студент не найден", Toast.LENGTH_LONG).show();
140                 return;
141             }
142             dao.delete(student);
143             Toast.makeText( context: this, text: "Студент удалён", Toast.LENGTH_LONG).show();
144             clearFields();
145             selectedStudentId = -1;
146             loadStudentsToList();
147         });
148     }
149 }
```

Dima

21, lu4wiu

Denis

Dima



Студент добавлен

Сохранить

Удалить

Обновить