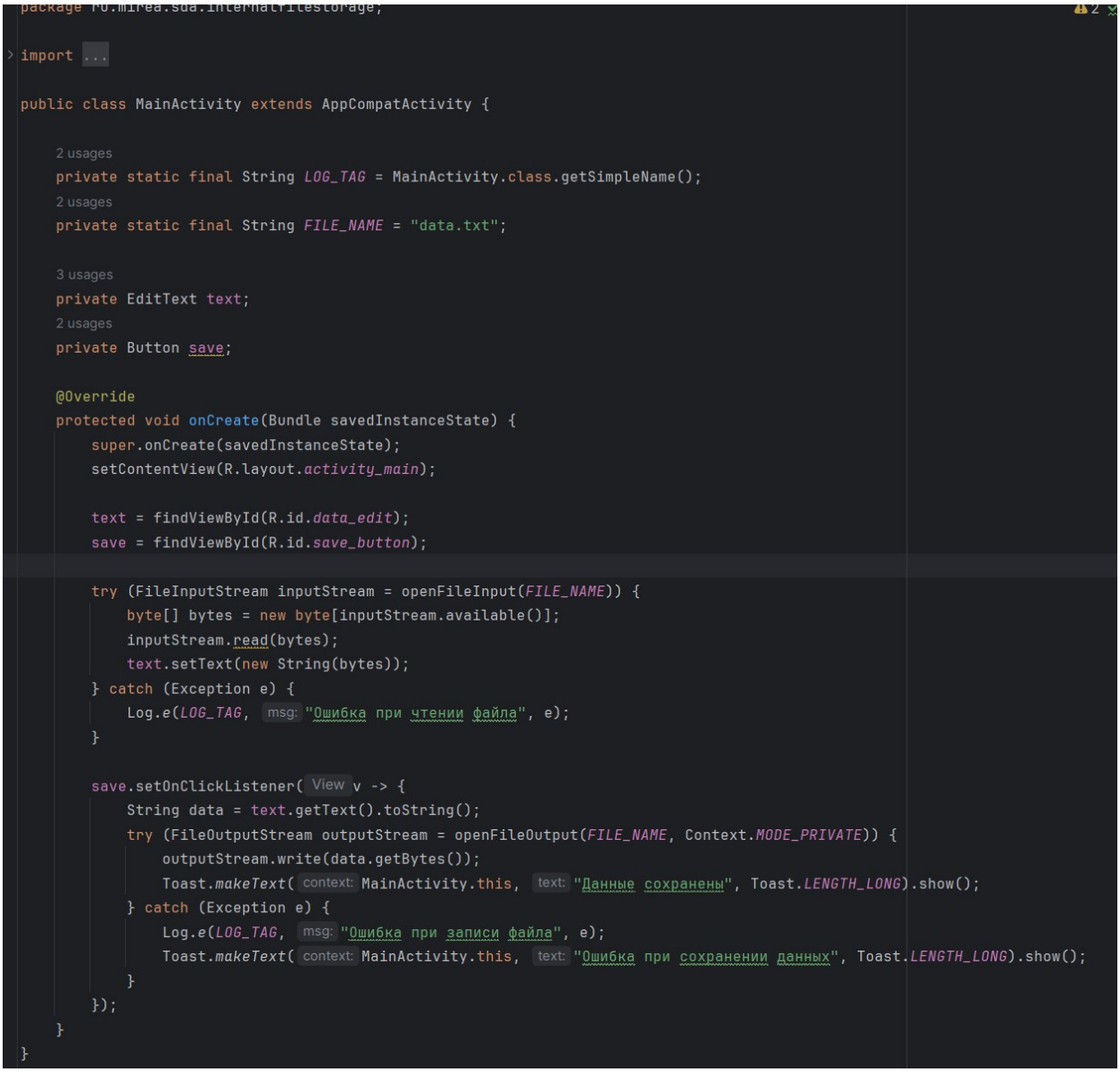
Практическая работа №6

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

Задание 1. Внутреннее хранилище

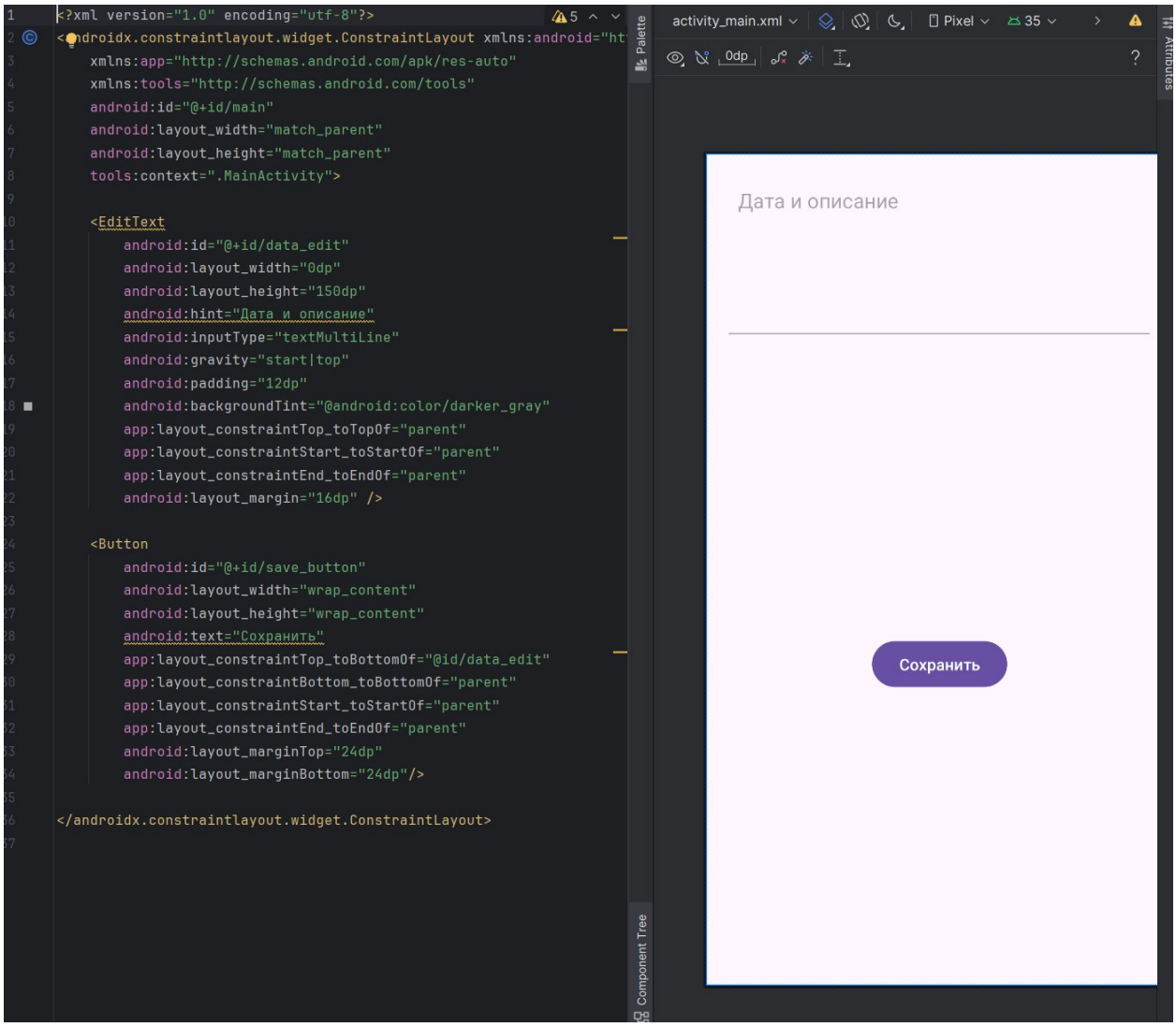
Создан модуль `internalfilestorage`. На экране размещены поле ввода (`EditText`) и кнопка (`Button`). Реализовано сохранение и загрузка пользовательского текста во внутреннем хранилище устройства с использованием методов `openFileOutput` и `openFileInput`.



```
package ru.mirea.sua.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();
            }
        });
    }
}
```

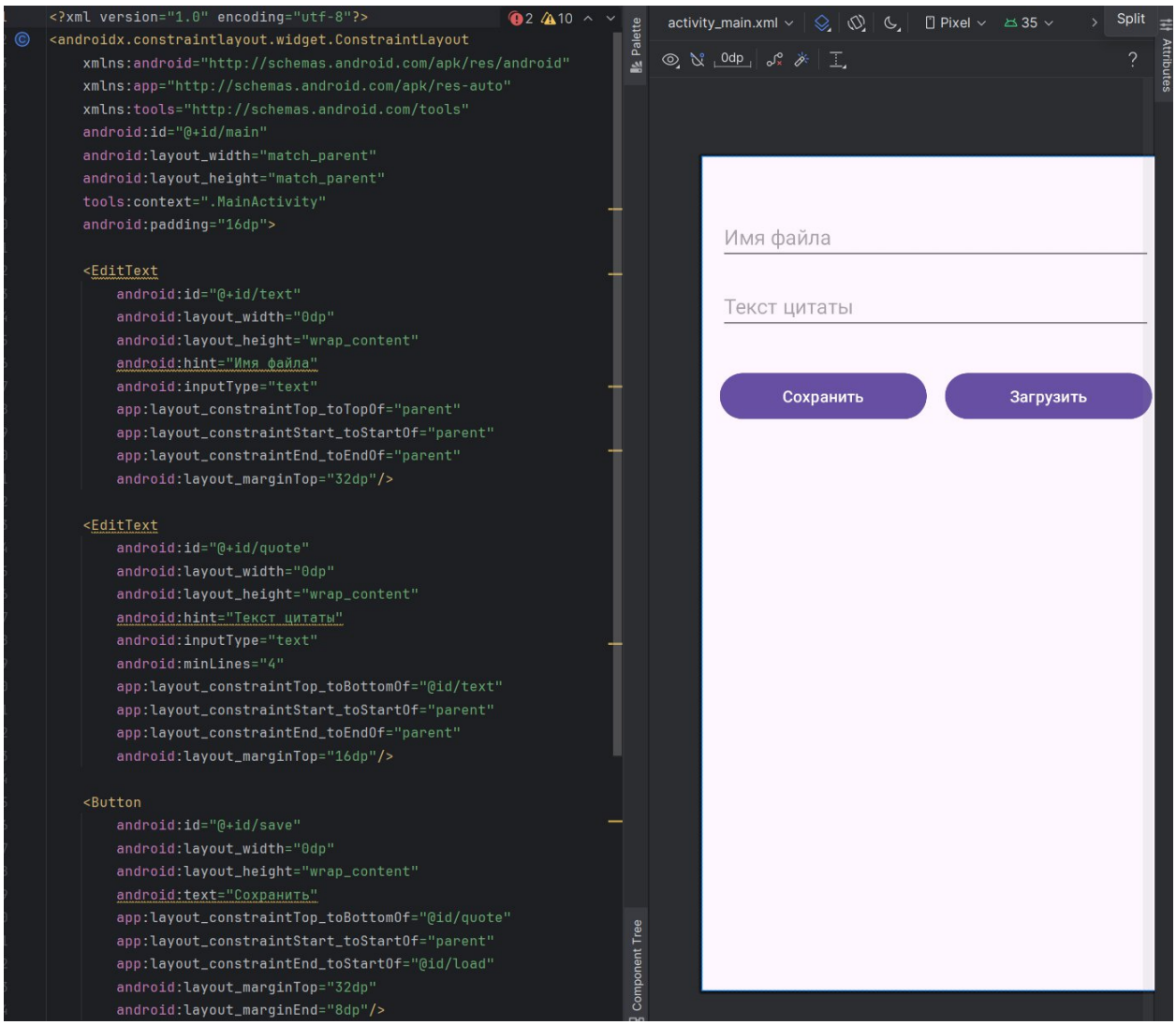
Файл data.txt создаётся в директории приложения /data/data/package_name/files. При запуске приложения содержимое файла автоматически загружается в поле ввода, а при нажатии на кнопку —

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

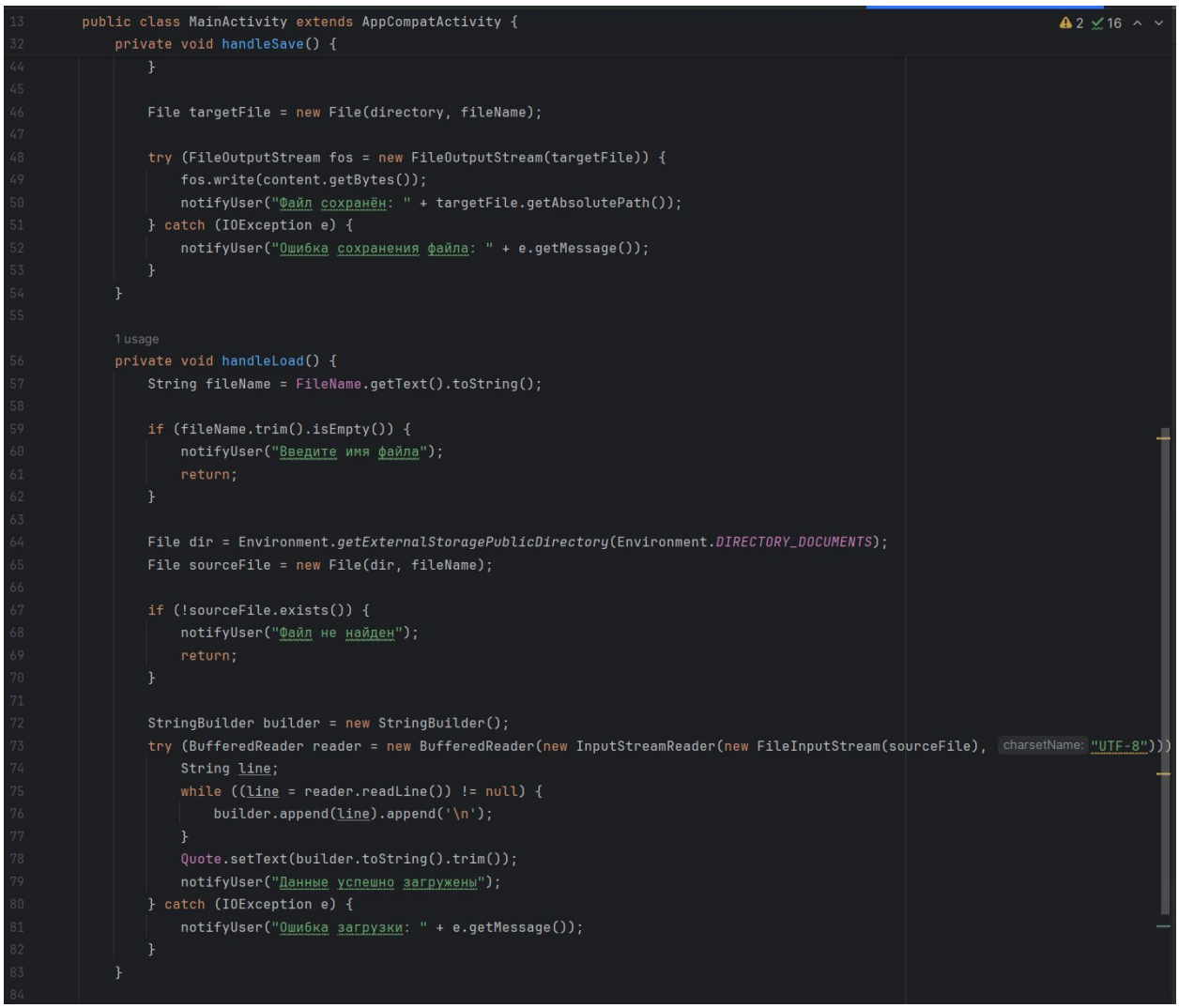


Задание 2. Внешнее хранилище

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.

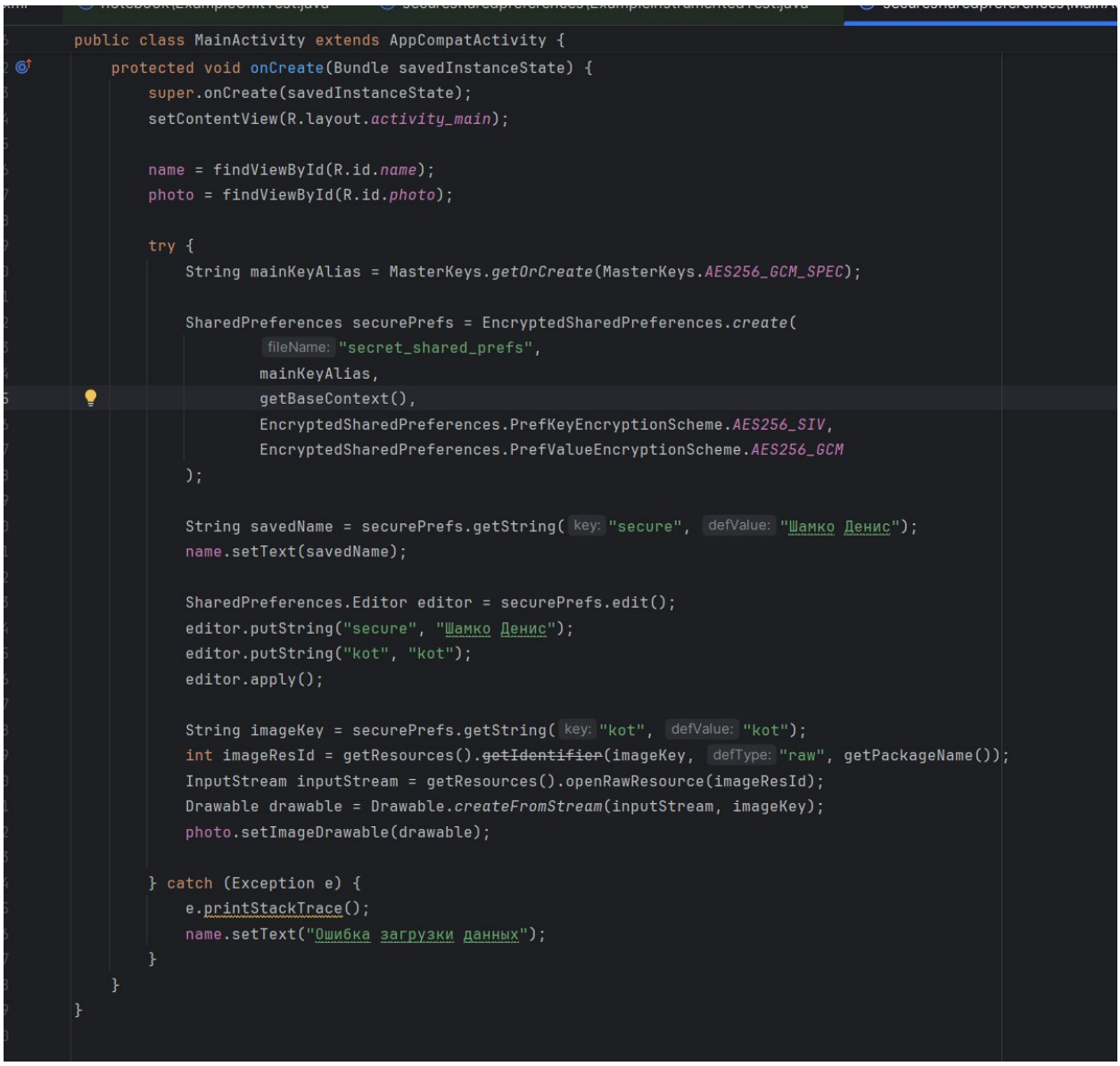


```
public class MainActivity extends AppCompatActivity {
    private void handleSave() {
        }

        File targetFile = new File(directory, fileName);

        try (FileOutputStream fos = new FileOutputStream(targetFile)) {
            fos.write(content.getBytes());
            notifyUser("Файл сохранён: " + targetFile.getAbsolutePath());
        } catch (IOException e) {
            notifyUser("Ошибка сохранения файла: " + e.getMessage());
        }
    }

    1 usage
    private void handleLoad() {
        String fileName = FileName.getText().toString();

        if (fileName.trim().isEmpty()) {
            notifyUser("Введите имя файла");
            return;
        }

        File dir = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOCUMENTS);
        File sourceFile = new File(dir, fileName);

        if (!sourceFile.exists()) {
            notifyUser("Файл не найден");
            return;
        }

        StringBuilder builder = new StringBuilder();
        try (BufferedReader reader = new BufferedReader(new InputStreamReader(new FileInputStream(sourceFile), charsetName: "UTF-8"))) {
            String line;
            while ((line = reader.readLine()) != null) {
                builder.append(line).append('\n');
            }
            Quote.setText(builder.toString().trim());
            notifyUser("Данные успешно загружены");
        } catch (IOException e) {
            notifyUser("Ошибка загрузки: " + e.getMessage());
        }
    }
```

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.

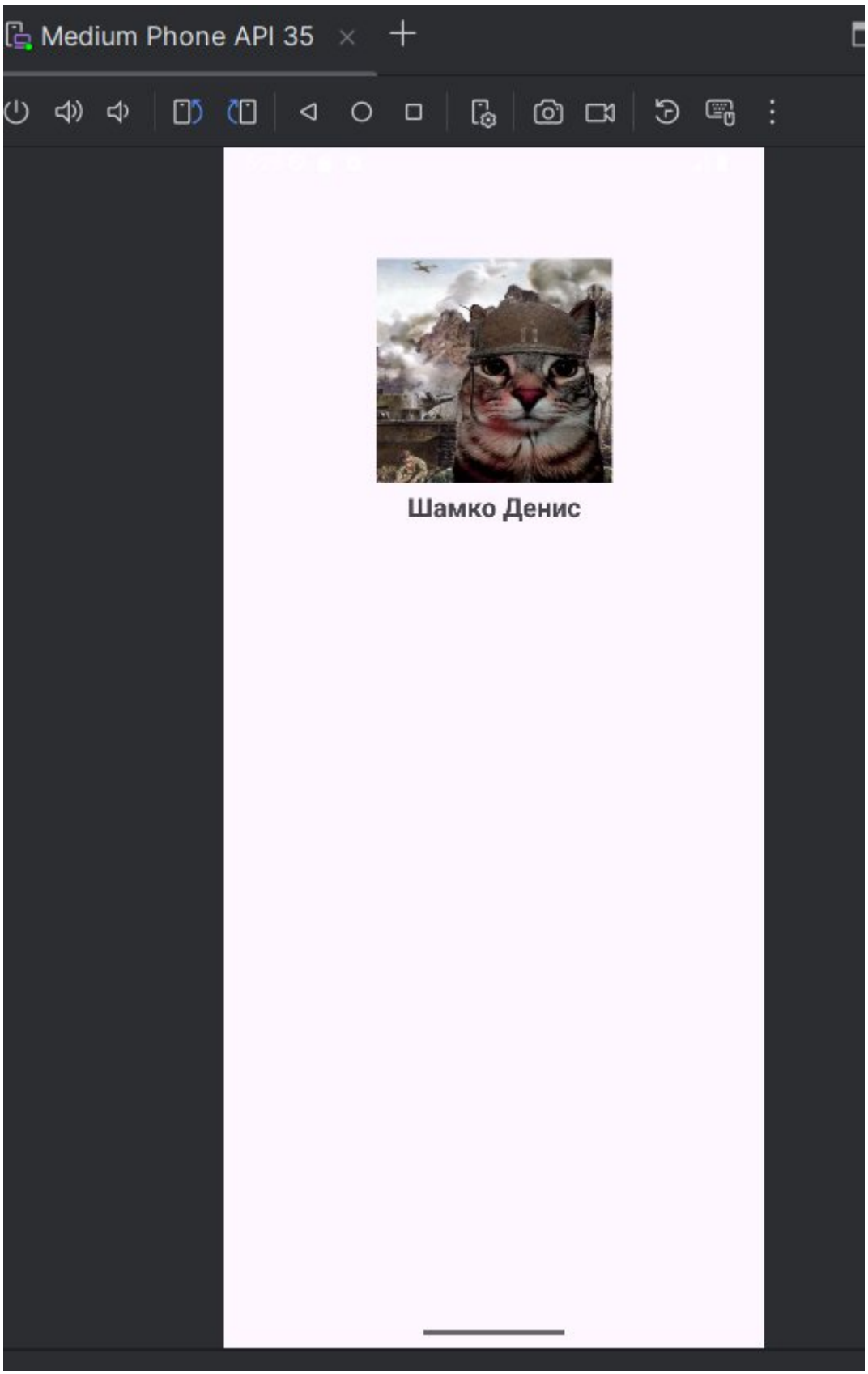
Задание 3. SharedPreferences и EncryptedSharedPreferences (securesharedpreferences)

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.



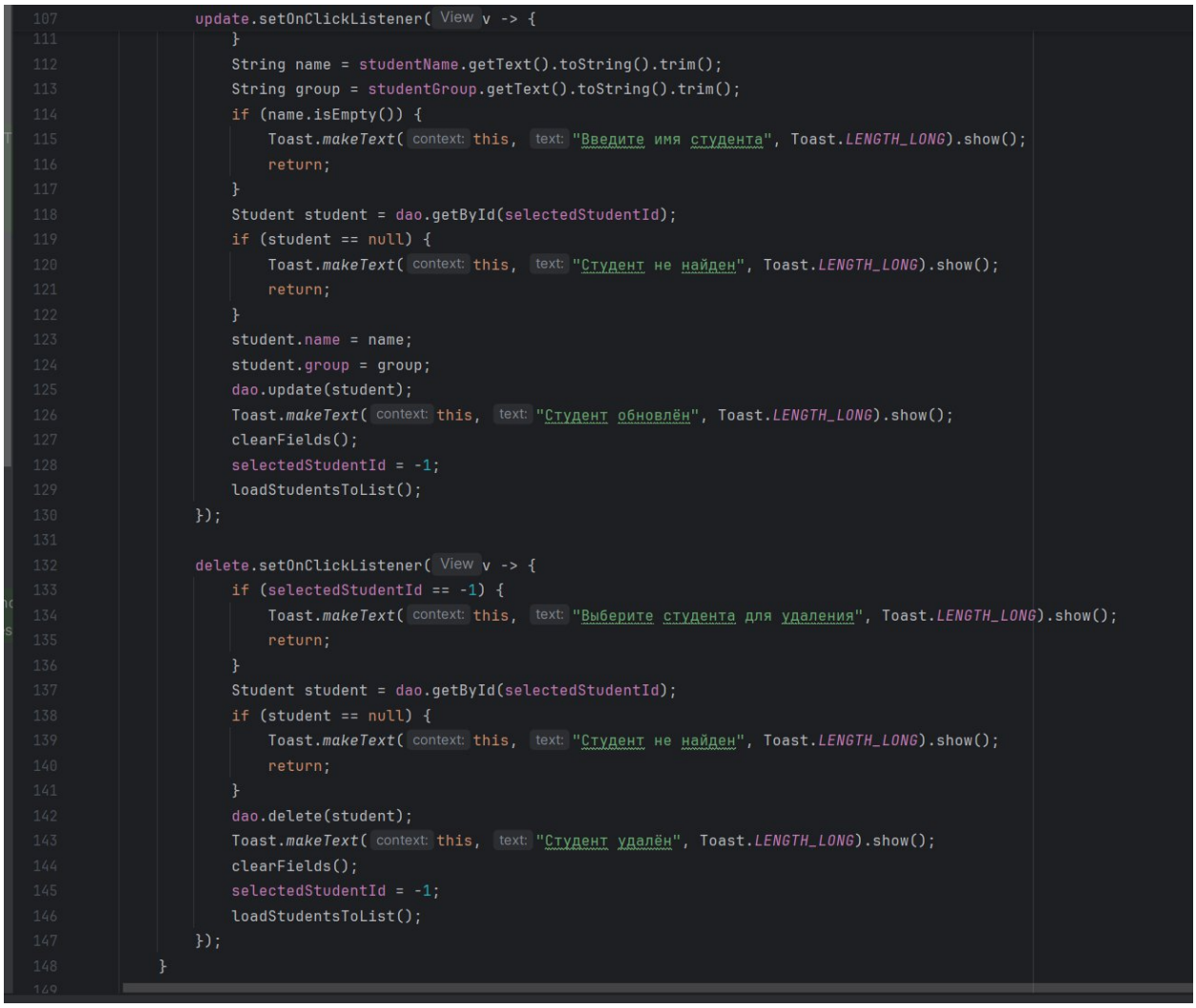
Визуальный интерфейс реализован вручную через findViewById, используется TextView и ImageView. Данные шифруются с помощью схем AES256_SIV (ключи) и AES256_GCM (значения).

Задание 4. Room Database (employeedb)

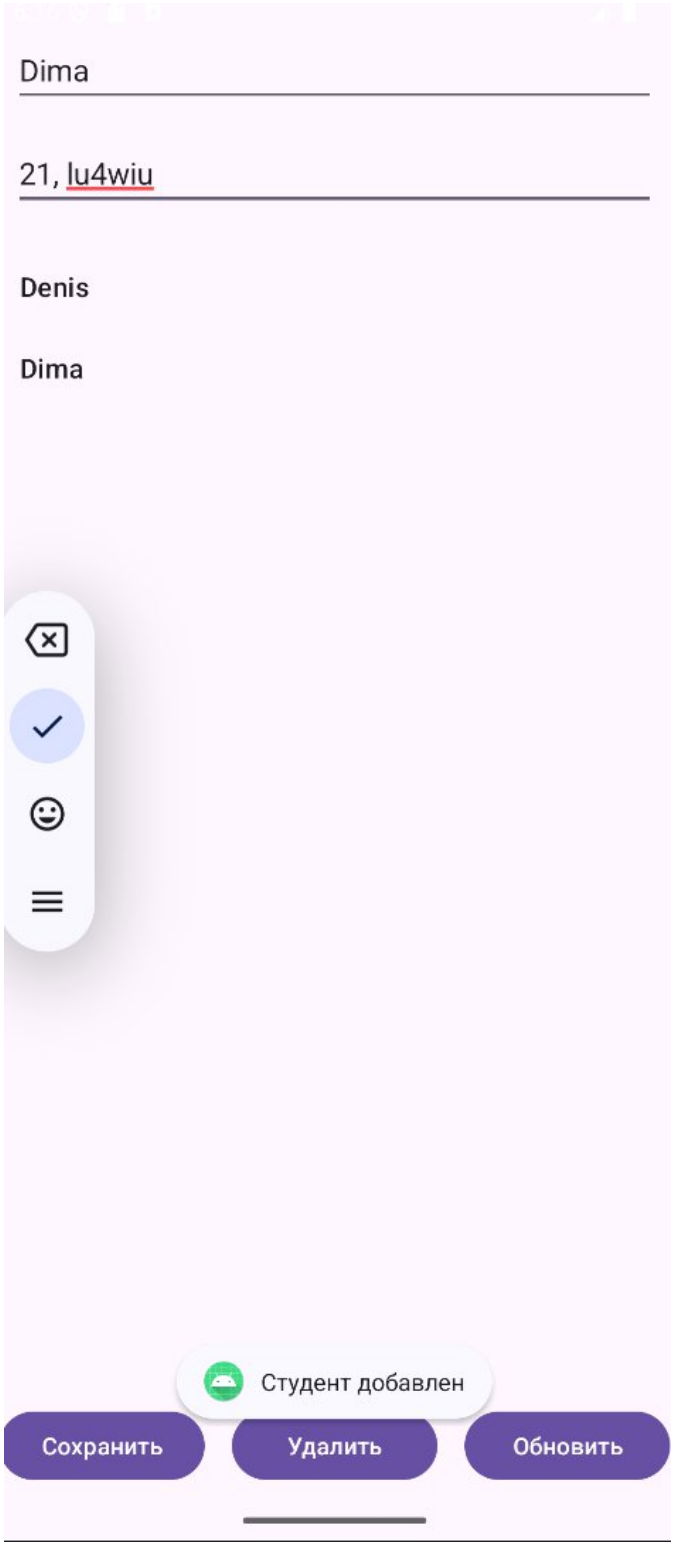
Модуль employeedb реализует хранение информации о студентах с использованием библиотеки Room:

- @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.

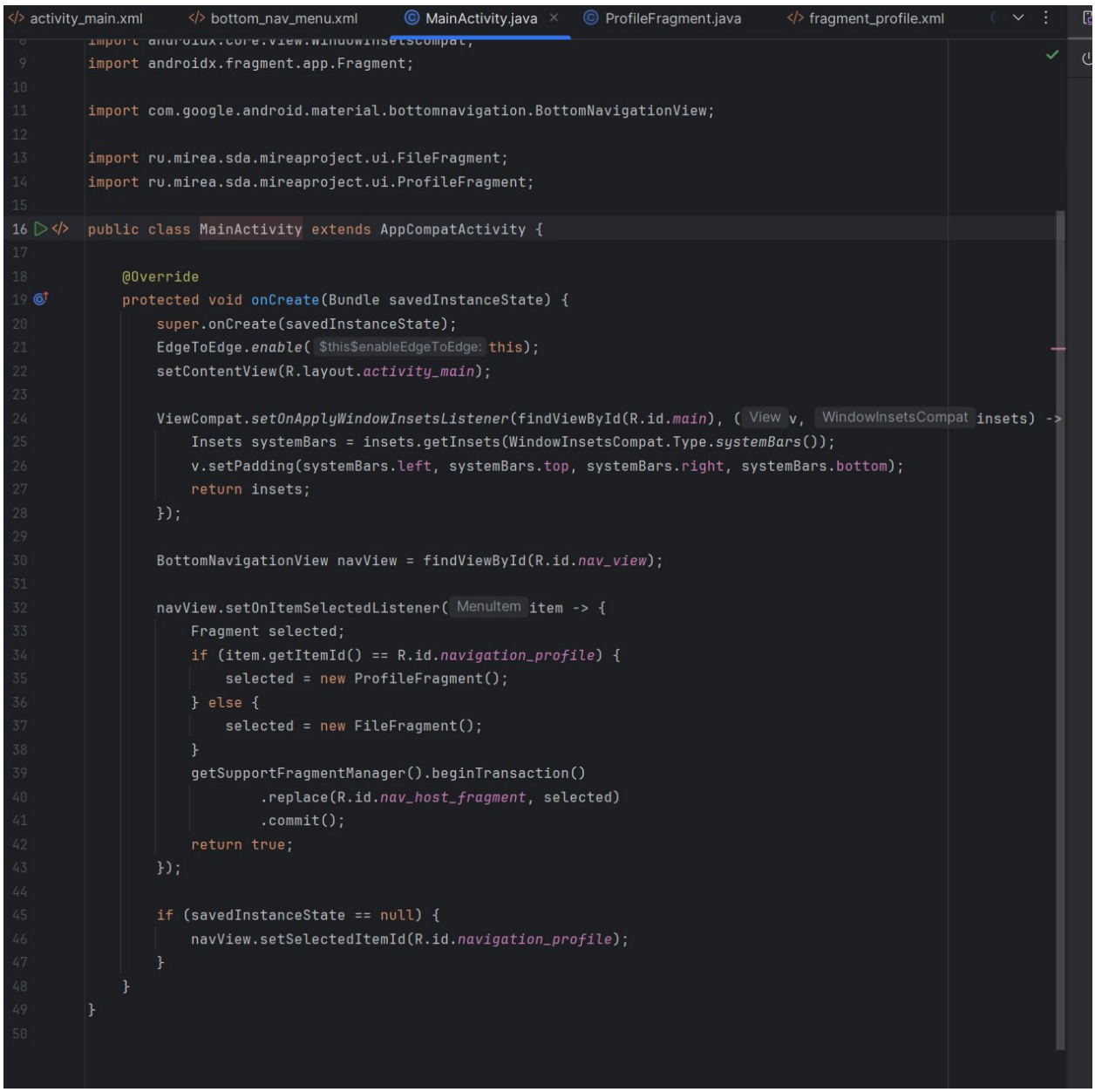


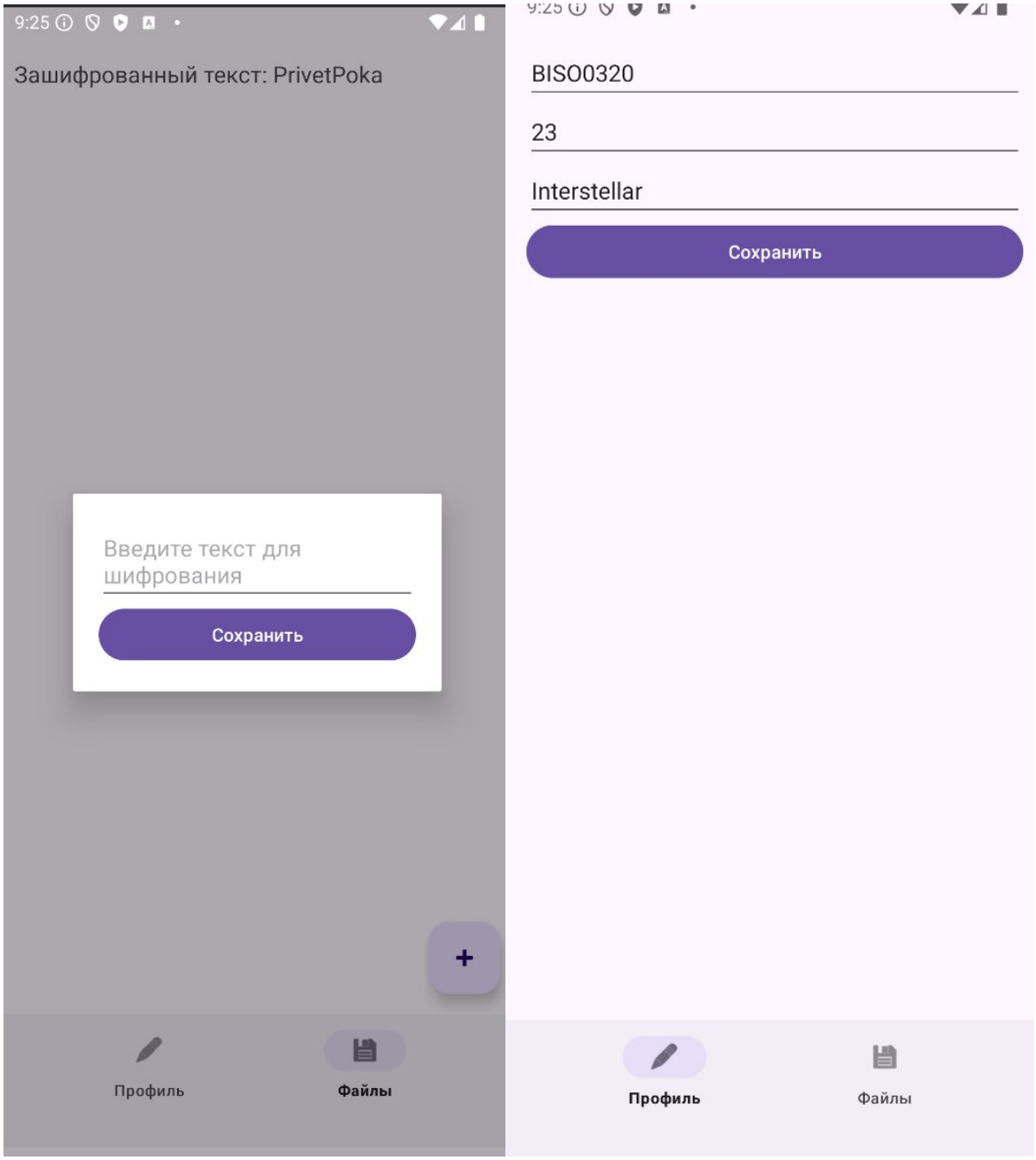
```
        update.setOnClickListener( View v -> {
            }
            String name = studentName.getText().toString().trim();
            String group = studentGroup.getText().toString().trim();
            if (name.isEmpty()) {
                Toast.makeText( context: this,  text: "Введите имя студента", Toast.LENGTH_LONG).show();
                return;
            }
            Student student = dao.getById(selectedStudentId);
            if (student == null) {
                Toast.makeText( context: this,  text: "Студент не найден", Toast.LENGTH_LONG).show();
                return;
            }
            student.name = name;
            student.group = group;
            dao.update(student);
            Toast.makeText( context: this,  text: "Студент обновлён", Toast.LENGTH_LONG).show();
            clearFields();
            selectedStudentId = -1;
            loadStudentsToList();
        });

        delete.setOnClickListener( View v -> {
            if (selectedStudentId == -1) {
                Toast.makeText( context: this,  text: "Выберите студента для удаления", Toast.LENGTH_LONG).show();
                return;
            }
            Student student = dao.getById(selectedStudentId);
            if (student == null) {
                Toast.makeText( context: this,  text: "Студент не найден", Toast.LENGTH_LONG).show();
                return;
            }
            dao.delete(student);
            Toast.makeText( context: this,  text: "Студент удалён", Toast.LENGTH_LONG).show();
            clearFields();
            selectedStudentId = -1;
            loadStudentsToList();
        });
    }
```



Контрольное задание

В рамках контрольного задания в проект MireaProject добавлены фрагменты для ввода и сохранения пользовательских данных через SharedPreferences, а также для работы с файлами — реализована базовая обработка текста (например, шифрование). По нажатию на FloatingActionButton открывается диалоговое окно для создания записи. Навигация между фрагментами осуществляется через Navigation Drawer.


activity_main.xml
bottom_nav_menu.xml
MainActivity.java
ProfileFragment.java
fragment_profile.xml
import androidx.fragment.app.Fragment;
import com.google.android.material.bottomnavigation.BottomNavigationView;
import ru.mirea.sda.mireaproject.ui.FileFragment;
import ru.mirea.sda.mireaproject.ui.ProfileFragment;
public class MainActivity extends AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
EdgeToEdge.enable($this$enableEdgeToEdge: this);
setContentView(R.layout.activity_main);
ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (View v, WindowInsetsCompat insets) ->
Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
return insets;
});
BottomNavigationView navView = findViewById(R.id.nav_view);
navView.setOnItemSelectedListener(MenuItem item -> {
Fragment selected;
if (item.getItemId() == R.id.navigation_profile) {
selected = new ProfileFragment();
} else {
selected = new FileFragment();
}
getSupportFragmentManager().beginTransaction()
.replace(R.id.nav_host_fragment, selected)
.commit();
return true;
});
if (savedInstanceState == null) {
navView.setSelectedItemId(R.id.navigation_profile);
}
}
}


9:25
Зашифрованный текст: PrivetPoka
Введите текст для шифрования
Сохранить
+
Профиль
Файлы
BISO0320
23
Interstellar
Сохранить
Профиль
Файлы