МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 8

по дисциплине
«Разработка мобильных приложений»

Тема: «Хранение данных в Android-приложении.»

Выполнил студент группы ИКБО-33-22 Шило Ю.С.

Принял преподаватель Рысин М.Л.

Лабораторная работа выполнена «__»_____202__ г. *(подпись студента)*

«Зачтено» «__»_____202__ г. *(подпись руководителя)*

Москва 2024

A. СОХРАНЕНИЕ СОСТОЯНИЯ ПРИЛОЖЕНИЯ

Задание 1

Сформируем layout-файл главного Activity (рис. 1.1) со следующим содержимым: в корневом контейнере ConstraintLayout разместим элемент EditText (идентификатор nameBox) для ввода произвольного текста (например, имени) и кнопку для этого ввода (saveBtn, надпись «Сохранить»); а также элемент TextView (nameView) и вторую кнопку (getBtn, надпись «Восстановить») для получения сохранённого текста.

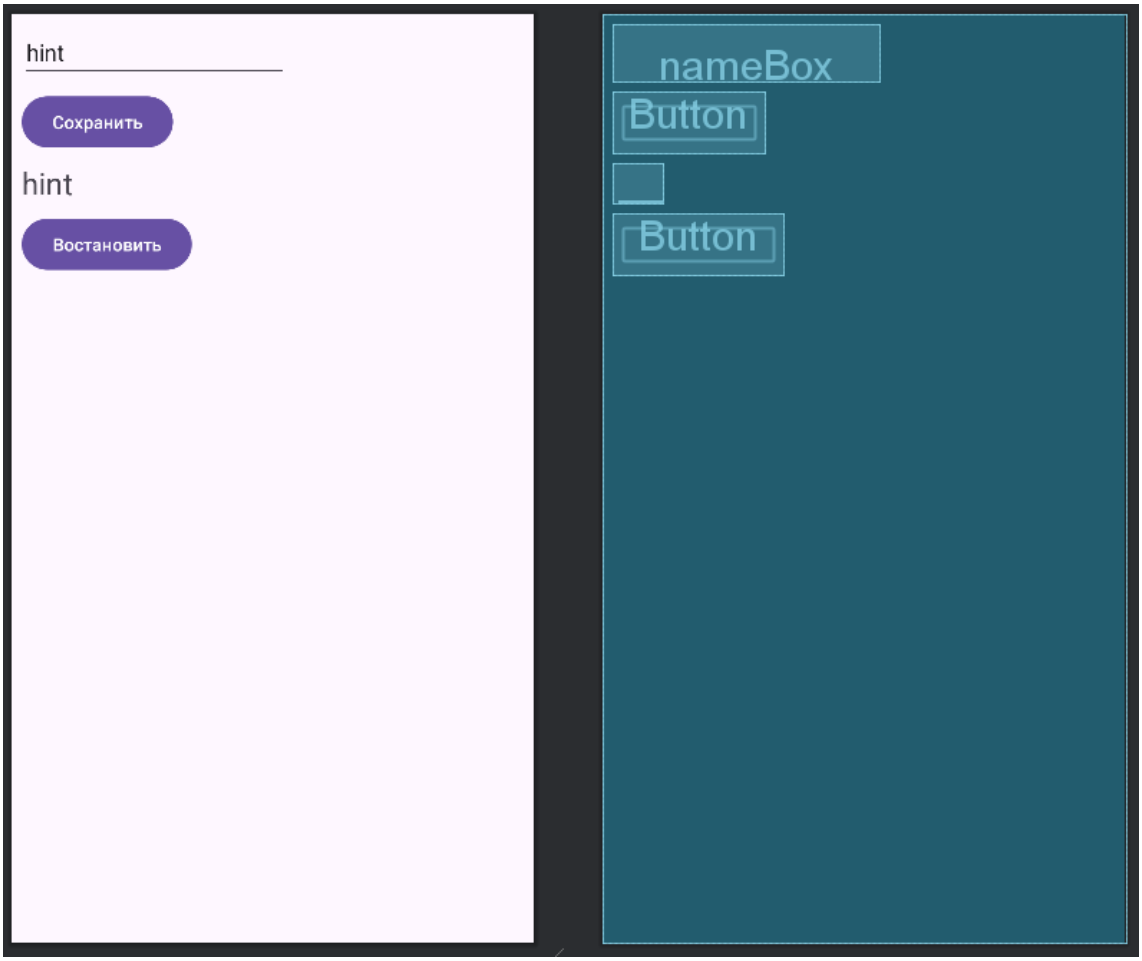


Рисунок 1.1 – Внешний вид разметки приложения

Реализуем в классе главного Activity методы сохранения и восстановления текста рисунок 1.2.

```
2 usages
private String _name = "???";
4 usages
private ActivityMainBinding _binding;

@Override
protected void onCreate(Bundle savedInstanceState) {...}

no usages
public void saveName (View v) { _name = _binding.nameBox.getText().toString(); }
no usages
public void getName (View v) { _binding.nameBox.setText(_name); }
```

Рисунок 1.2 – Содержимое файла MainActivity.java

Запустим приложение, введем какой-либо текст, сохраним его и восстановим в TextView. Перейдя к альбомному режиму – TextView окажется пустым, несмотря на то что в нём было значение. По кнопке «Восстановить» попробуем снова получить ранее сохранённое значение – оно окажется утраченным.

Чтобы избежать подобных ситуаций следует сохранять и восстанавливать состояние Activity с помощью методов onSaveInstanceState и onRestoreInstanceState. Изменим код MainActivity рисунок 1.3.

```
@Override
protected void onSaveInstanceState(@NonNull Bundle outState) {
    outState.putString(KEY, _name);
    super.onSaveInstanceState(outState);
}

@Override
protected void onRestoreInstanceState(@NonNull Bundle savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);
    _name = savedInstanceState.getString(KEY);
    _binding.nameBox.setText(_name);
}
```

Рисунок 1.3 – Добавленные методы в MainActivity.java

Запустим приложение, проверим правильность работы при поворотах экрана. Приложение предоставлено на рисунке 1.4.

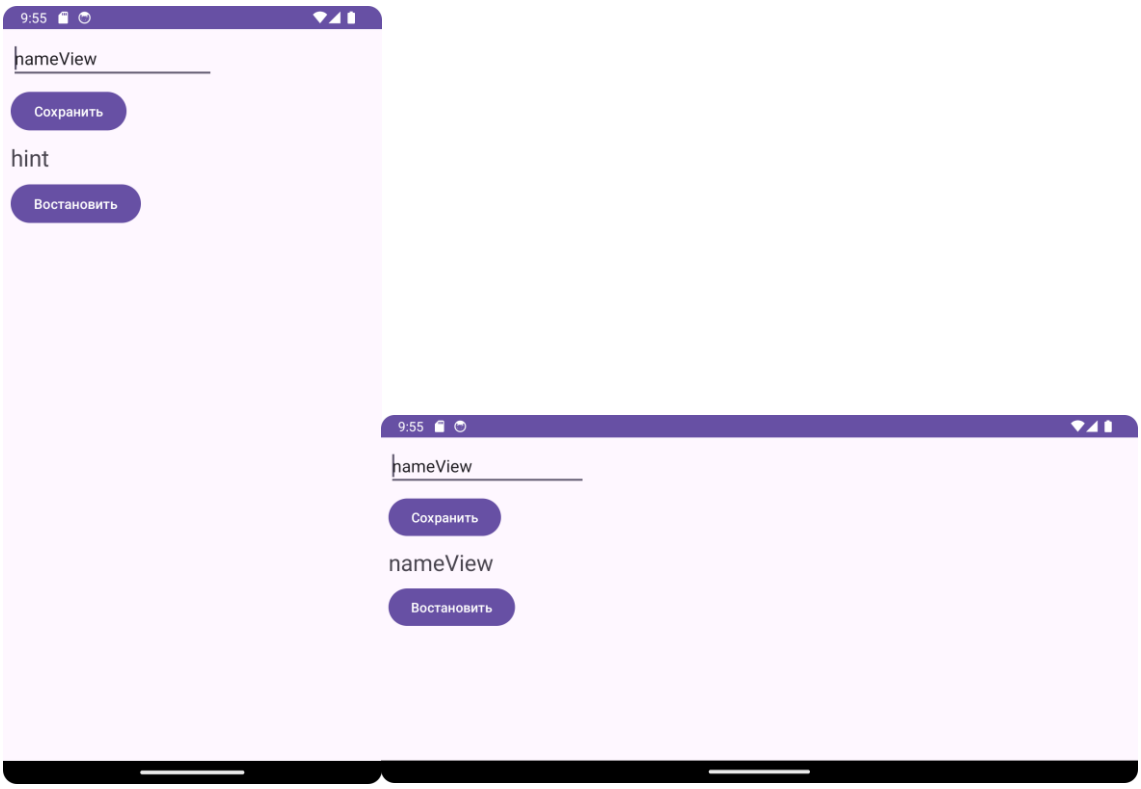


Рисунок 1.4 – Приложение запущенное на эмуляторе

Задание 2

Добавим в проект класс User и сформируем его код в рисунок 2.1.

```
no usages
public class User implements Serializable {
    3 usages
    private String _name;
    3 usages
    private int _age;

    no usages
    public User(String name, int age) {
        _name = name; _age = age;
    }


    no usages
    public int get_age() { return _age; }
    no usages
    public void set_age(int age) { _age = age; }

    no usages
    public String get_name() { return _name; }
    no usages
    public void set_name(String name) { _name = name; }
}
```

Рисунок 2.1 – Содержимое файла User.java

Сформирем layout-файл главного Activity (рис. 2.2) со следующим содержимым: в корневом контейнере ConstraintLayout разместим два элемента

EditText (идентификаторы nameBox и yearBox) для ввода имени и возраста человека (с соответствующими подсказками в полях ввода) и кнопку для этого ввода (saveBtn, надпись «Сохранить»); а также элемент TextView (dataView) и вторую кнопку (getBtn, надпись «Получить» или «Восстановить») для получения сохранённого текста.

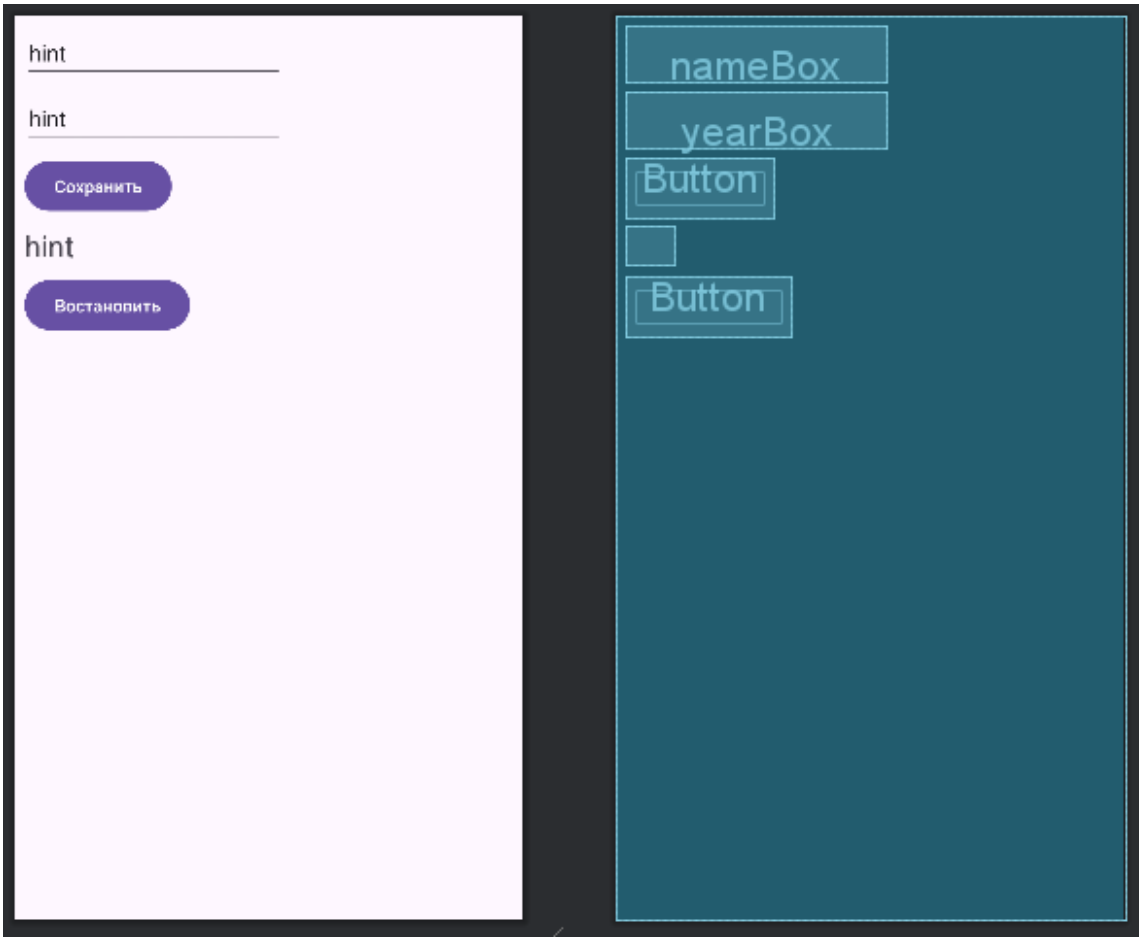


Рисунок 2.2 – Внешний вид разметки приложения

Реализуем в классе главного Activity методы сохранения и восстановления состояния Activity рисунок 2.3.

```
2 usages
final static String KEY = "STRING_KEY";
6 usages
private ActivityMainBinding _binding;
7 usages
private User _user = new User( name: "undefined",  age: 0);

@Override
protected void onCreate(Bundle savedInstanceState) {...}

@Override
protected void onSaveInstanceState(@NonNull Bundle outState) {
    outState.putSerializable(KEY, _user);
    super.onSaveInstanceState(outState);
}

@SuppressLint("SetTextI18n")
@Override
protected void onRestoreInstanceState(@NonNull Bundle savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);
    _user = (User) savedInstanceState.getSerializable(KEY);
    _binding.dataView.setText("Name: " + _user.get_name() + " Age: " + _user.get_age());
}

1 usage
public void saveName (View v) {
    String name = _binding.nameBox.getText().toString();
    int age = 0;
    try {
        age = Integer.parseInt(_binding.yearBox.getText().toString());
    }
    catch (NumberFormatException ignored) {}
    _user = new User(name, age);
}

1 usage
@SuppressLint("SetTextI18n")
public void getName (View v) {
    _binding.dataView.setText("Name: " + _user.get_name() + " Age: " + _user.get_age());
}
```

Рисунок 2.3 – Содержимое файла MainActivity.java

Запустим приложение, проверим правильность работы при поворотах экрана. Приложение предоставлено на рисунке 2.4.

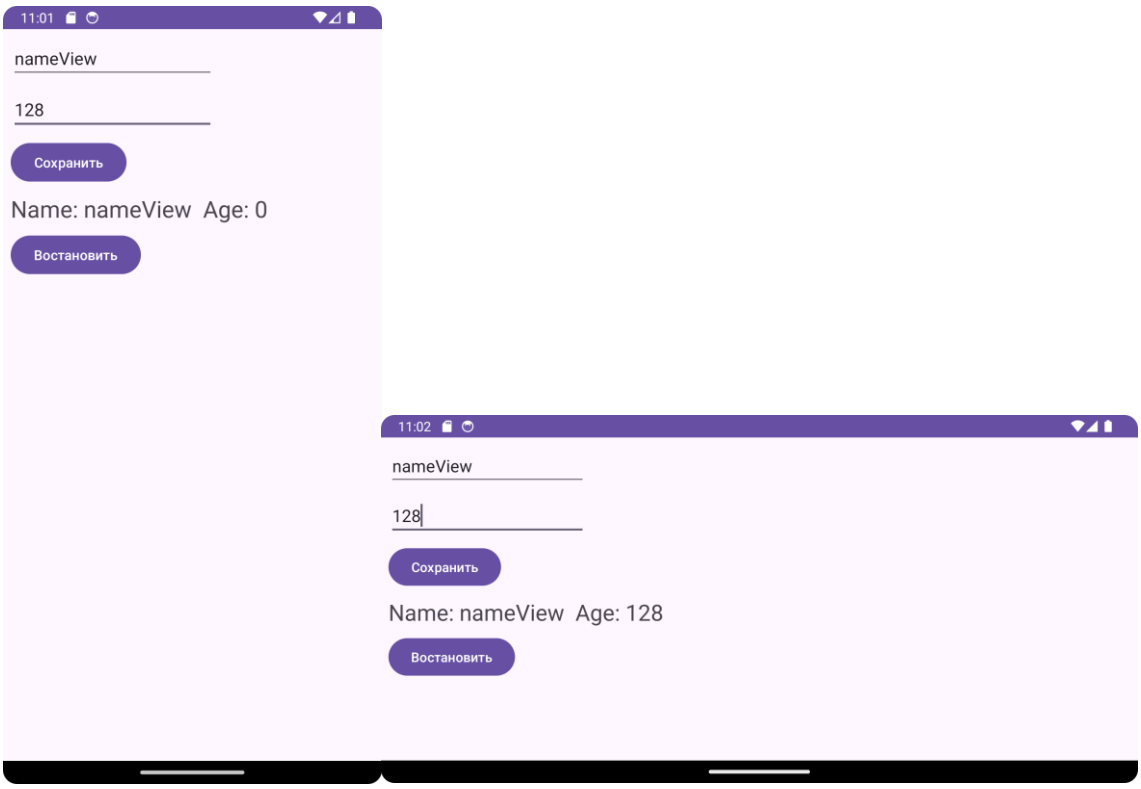


Рисунок 2.4 – Приложение запущенное на эмуляторе

B. SHAREDPREFERENCES ДЛЯ СОЗДАНИЯ И ПОЛУЧЕНИЯ НАСТРОЕК

Задание 3

Сформируем layout-файл главного Activity (рис. 3.1) со следующим содержимым: в корневом контейнере ConstraintLayout разместим элемент EditText (идентификатор nameBox) для ввода имени человека (с соответствующей подсказкой в поле ввода) и кнопку для этого ввода (saveButton, надпись «Сохранить»); а также элемент TextView (nameView) и вторую кнопку (getButton, надпись «Восстановить») для получения сохранённого имени.

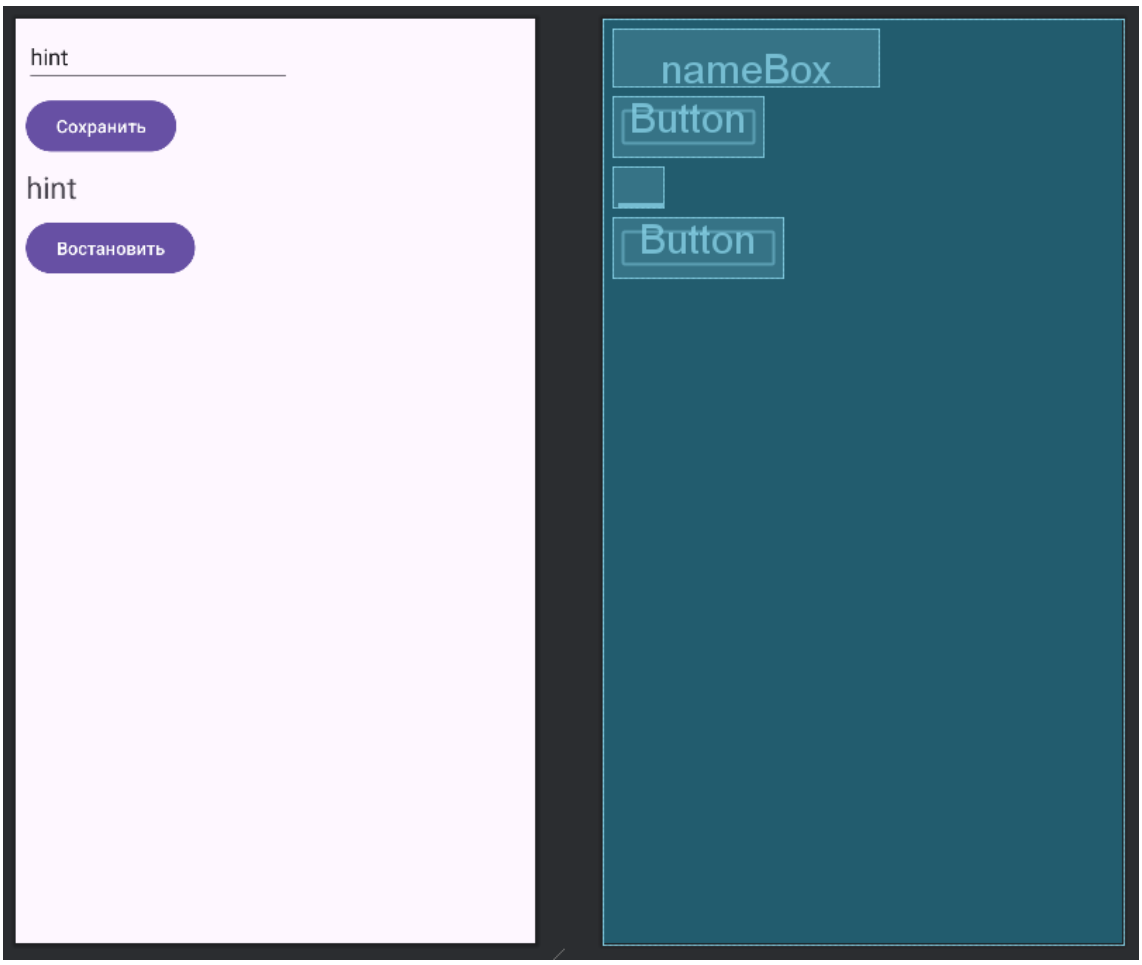


Рисунок 3.1 – Внешний вид разметки приложения

Определим методы обработки нажатия кнопок в классе MainActivity. А также переопределим методы onCreate и onPause, чтобы при выходе из приложения значение текстового поля автоматически сохранялось, а при запуске – восстанавливалось. Код предоставлен на рисунке 3.2.

```
1 usage
private static final String PREFS_FILE = "Account", PREF_NAME = "Name";
5 usages
private SharedPreferences _settings;
6 usages
private ActivityMainBinding _binding;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    _binding = ActivityMainBinding.inflate(getLayoutInflater());
    setContentView(_binding.getRoot());

    _settings = getSharedPreferences(PREFS_FILE, MODE_PRIVATE);

    String name = _settings.getString(PREF_NAME, defValue: "не определенно");
    _binding.dataView.setText(name);
}
@Override
protected void onPause() {
    super.onPause();

    String name = _binding.nameBox.getText().toString();

    SharedPreferences.Editor prefEditor = _settings.edit();
    prefEditor.putString(PREF_NAME, name);
    prefEditor.apply();
}
    1 usage
    public void saveName (View v) {
    String name = _binding.nameBox.getText().toString();

    SharedPreferences.Editor prefEditor = _settings.edit();
    prefEditor.putString(PREF_NAME, name);
    prefEditor.apply();
}
1 usage
public void getName (View v) {
    String name = _settings.getString(PREF_NAME, defValue: "не определенно");
    _binding.dataView.setText(name);
}
```

Рисунок 3.2 – Содержимое файла MainActivity.java

Запустим приложение. Нажмите кнопку «Восстановить»: при отсутствии настроек при попытке их получить, приложение выведет значение по умолчанию. Запущенное приложение на эмуляторе предоставлено на рисунке 3.3.

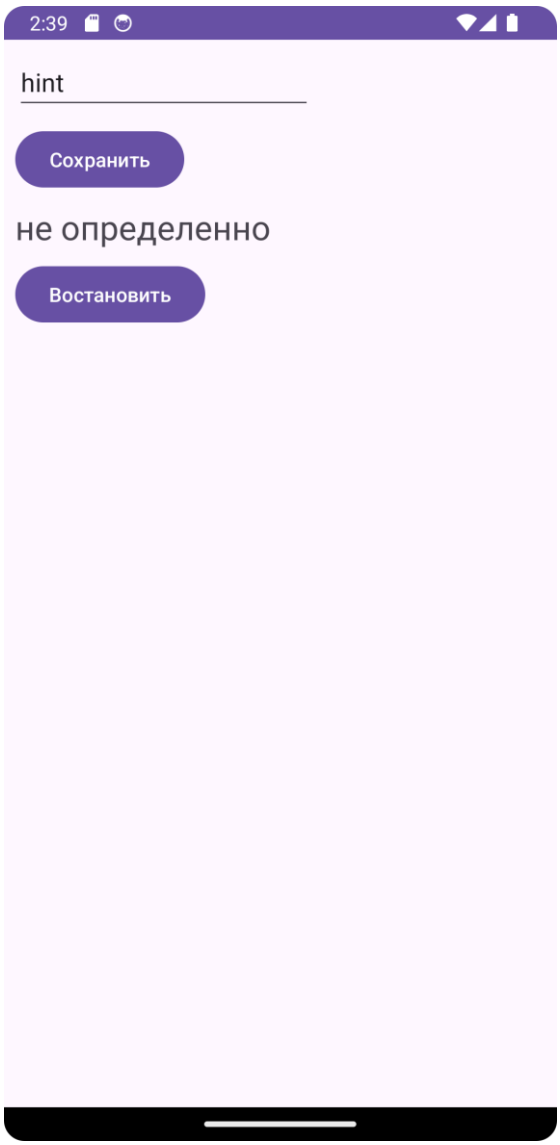


Рисунок 3.3 – Приложение запущенное на эмуляторе

C. РАБОТА С ФАЙЛОВОЙ СИСТЕМОЙ

Чтение и запись файлов во внутреннем хранилище

Задание 4

Сформируем layout-файл главного Activity (рис. 4.1) со следующим содержимым: в корневом контейнере ConstraintLayout разместим элемент EditText (идентификатор editor) для ввода текста (с соответствующей подсказкой в поле ввода) и кнопку для сохранения ввода в файл (save_text, надпись «Записать»); а также элемент TextView (text) и вторую кнопку (open_text, надпись «Прочитать») для получения из файла сохранённого текста.

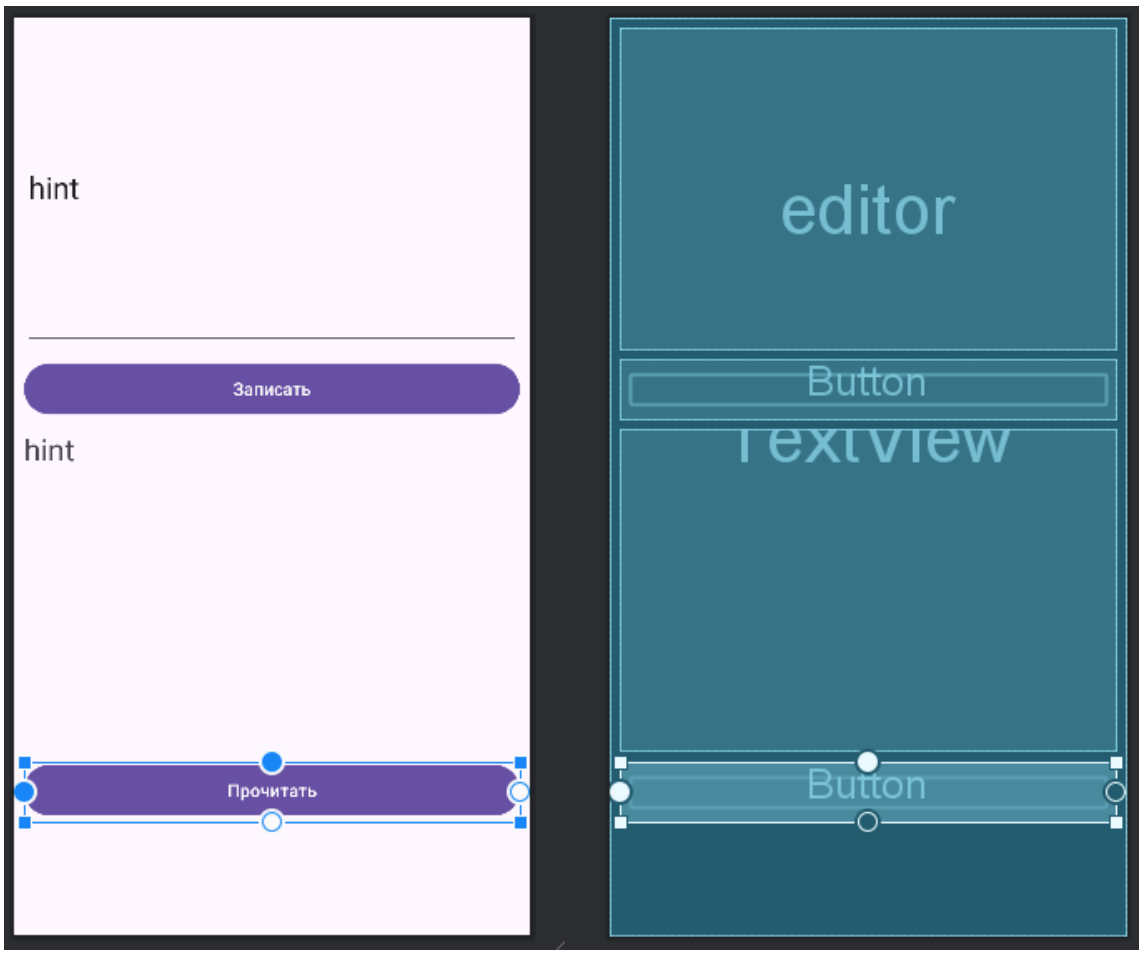


Рисунок 3.1 – Внешний вид разметки приложения

Определим в главном Activity обработчики кнопок в соответствии рисунок 3.2.

```
1 usage
public void saveText (View view) {
    FileOutputStream fos = null;
    try {
        String text = _binding.editor.getText().toString();
        fos = openFileOutput(FILE_NAME, MODE_PRIVATE);
        fos.write(text.getBytes());
        Toast.makeText( context: this,  text: "Файл сохранен", Toast.LENGTH_SHORT).show();
    } catch (IOException ex) {
        Toast.makeText( context: this, ex.getMessage(), Toast.LENGTH_SHORT).show();
        throw new RuntimeException(ex);
    } finally {
        try {
            if (fos != null) fos.close();
        } catch (IOException ex) {
            Toast.makeText( context: this, ex.getMessage(), Toast.LENGTH_SHORT).show();
        }
    }
}

1 usage
public void openText(View view) {
    FileInputStream fin = null;
    try {
        fin = openFileInput(FILE_NAME);
        byte[] bytes = new byte[fin.available()];
        fin.read(bytes);
        String text = new String(bytes);
        _binding.text.setText(text);
    } catch (IOException ex) {
        Toast.makeText( context: this, ex.getMessage(), Toast.LENGTH_SHORT).show();
    } finally {
        try {
            if (fin != null) fin.close();
        } catch (IOException ex) {
            Toast.makeText( context: this, ex.getMessage(), Toast.LENGTH_SHORT).show();
        }
    }
}
```

Рисунок 4.2 – Методы, отвечающие за отработку нажатий по кнопкам

Запущенное приложение на эмуляторе предоставлено на рисунке 4.3.

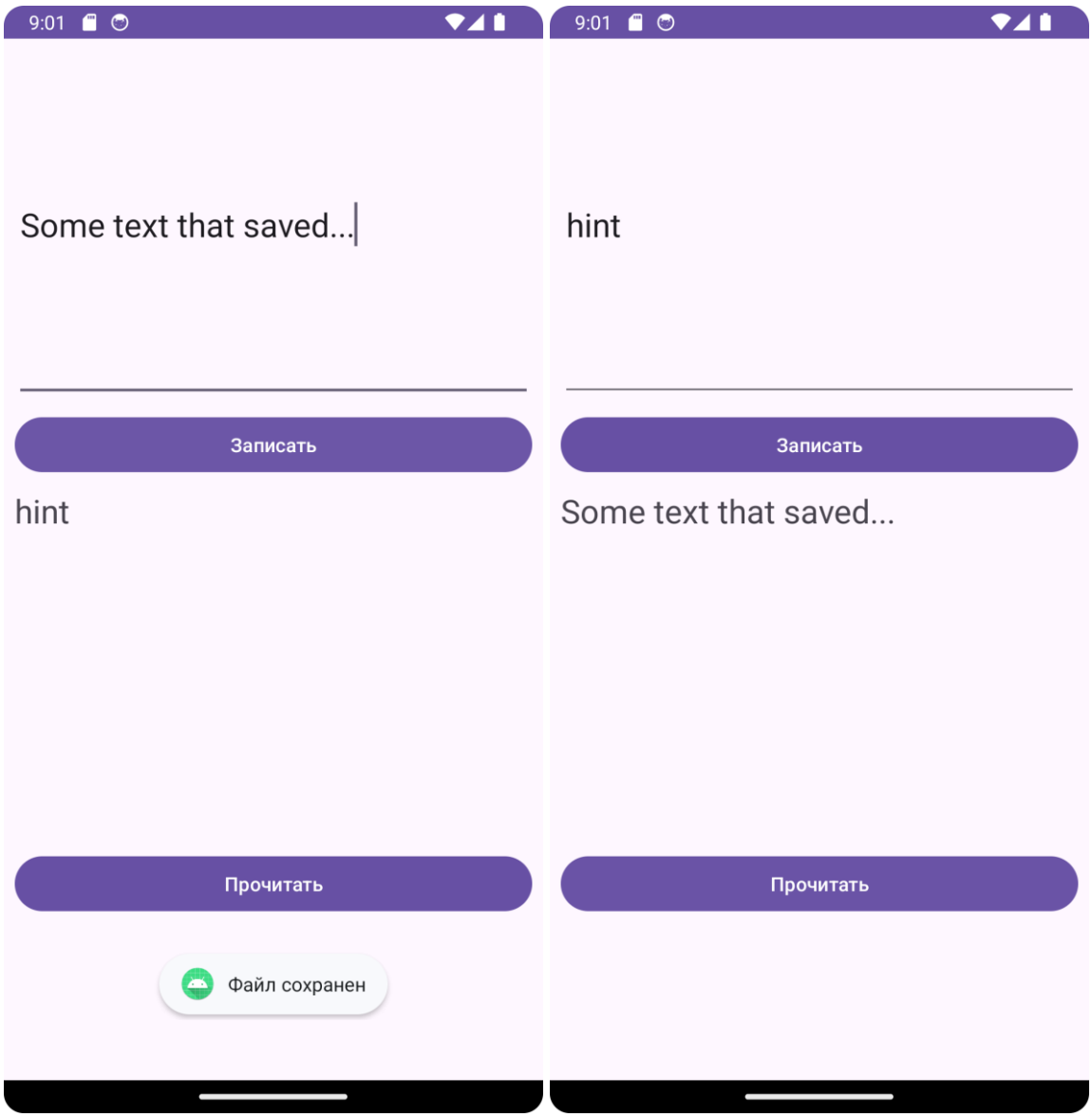


Рисунок 4.3 – Приложение запущенное на эмуляторе

Работа с файлами на внешнем накопителе

Выше был рассмотрен вариант работы с файлами из каталога приложения. По умолчанию такие файлы доступны только самому приложению. Возможно также помещать и работать с файлами из внешнего хранилища устройства. Это предоставит доступ к данным файлам и другим программам.

В предыдущем проекте отредактируйте код главного Activity для работы с файлами на внешнем накопителе рисунок 4.4.

```
3 usages
private File getExternalPath() { return new File(getExternalFilesDir( type: null), FILE_NAME); }
1 usage
public void saveText (View view) {
    FileOutputStream fos = null;
    try {
        String text = _binding.editor.getText().toString();
        fos = new FileOutputStream(getExternalPath()); //
        fos.write(text.getBytes());
        Toast.makeText( context: this,  text: "Файл сохранен", Toast.LENGTH_SHORT).show();
    } catch (IOException ex) {
        Toast.makeText( context: this, ex.getMessage(), Toast.LENGTH_SHORT).show();
        throw new RuntimeException(ex);
    } finally {
        try {
            if (fos != null) fos.close();
        } catch (IOException ex) {
            Toast.makeText( context: this, ex.getMessage(), Toast.LENGTH_SHORT).show();
        }
    }
}
1 usage
public void openText(View view) {
    File file = getExternalPath(); //
    if (!file.exists()) return; //
    FileInputStream fin = null;
    try {
        fin = new FileInputStream(getExternalPath()); //
        byte[] bytes = new byte[fin.available()];
        fin.read(bytes);
        String text = new String(bytes);
        _binding.text.setText(text);
    } catch (IOException ex) {
        Toast.makeText( context: this, ex.getMessage(), Toast.LENGTH_SHORT).show();
    } finally {
        try {
            if (fin != null) fin.close();
        } catch (IOException ex) {
            Toast.makeText( context: this, ex.getMessage(), Toast.LENGTH_SHORT).show();
        }
    }
}
```

Рисунок 4.4 – Изменённые методы

После запуска нашего приложения и нажатия на кнопку “Сохранить” сохранённый файл мы сможем найти в папке приложения в .../data рисунок 4.5.

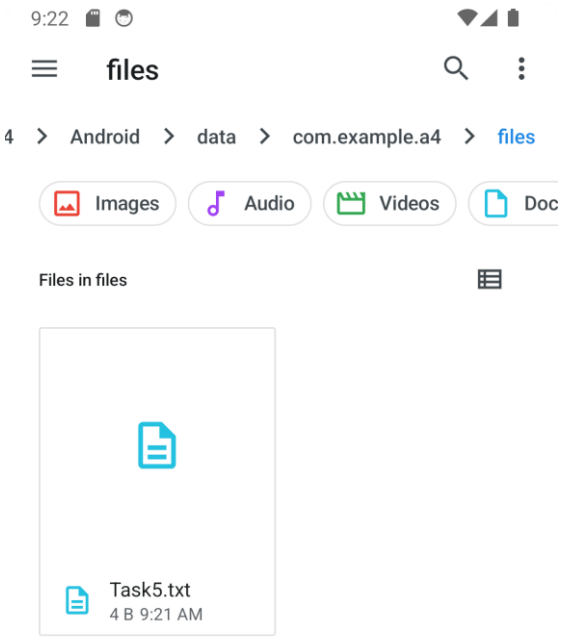


Рисунок 4.5 – Путь к нашему файлу в проводнике

D. ФОРМАТ JSON

Задание 5

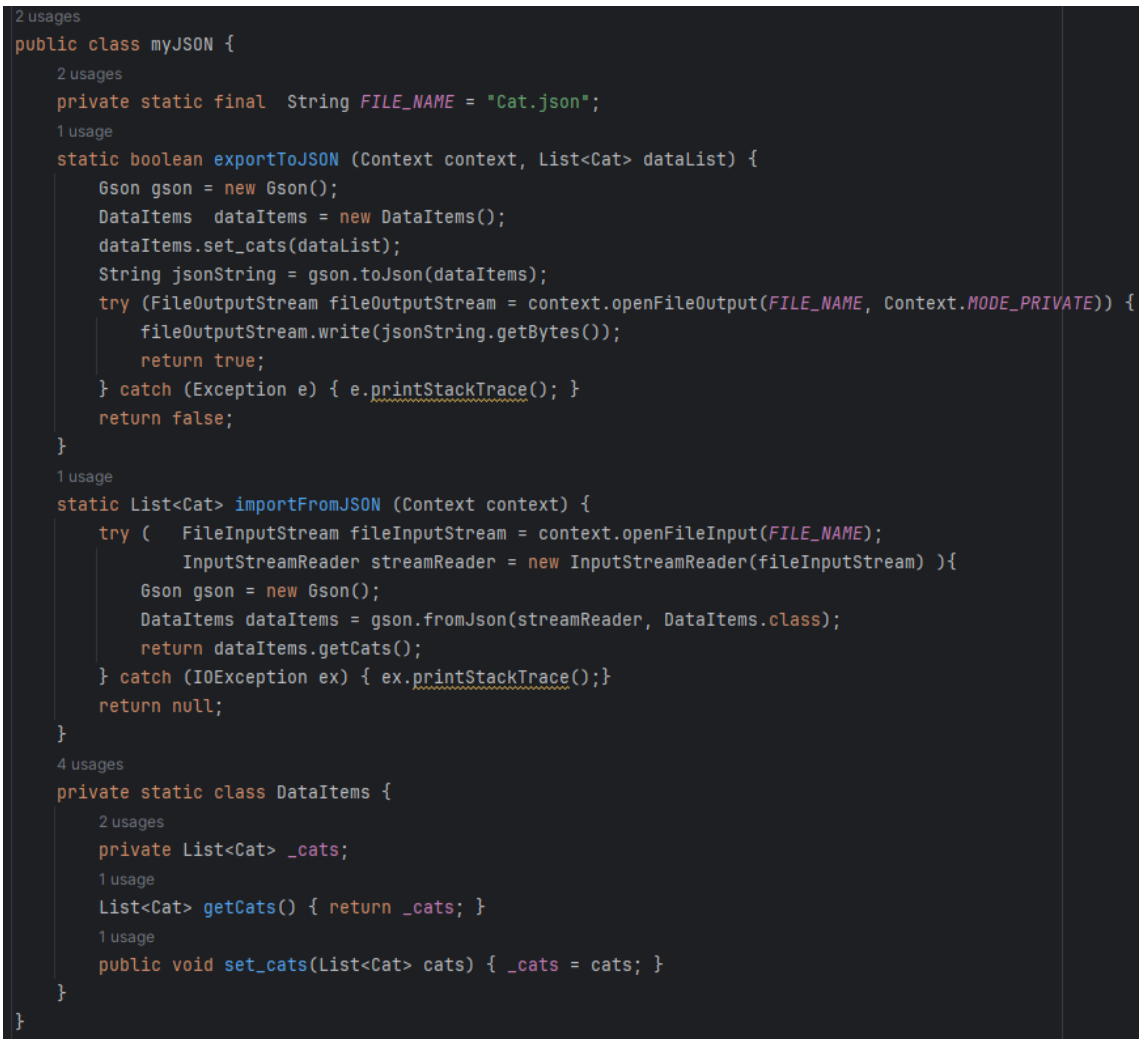
Создадим новый класс Cat рисунок 5.1.

```
no usages
public class Cat {
    4 usages
    private String _name;
    4 usages
    private int _age;

    no usages
    Cat (String name, int age) { _name = name; _age = age; }

    no usages
    public String get_name() { return _name; }
    no usages
    public void set_name(String name) { _name = name; }

    no usages
    public int get_age() { return _age; }
    no usages
    public void set_age(int age) { _age = age; }

    @Override
    public String toString() {
        return "Кличка кота: " + _name  + ", Возраст: " + _age;
    }
}
```

Рисунок 5.1 – Содержимое класса Cat

Объекты этого класса будем сериализовать в формат json и затем обратно десериализовать из файла в объекты.

Для работы с форматом json добавьте в проект класс myJSON рисунок 5.2.



```
2 usages
public class myJSON {
    2 usages
    private static final  String FILE_NAME = "Cat.json";
    1 usage
    static boolean exportToJSON (Context context, List<Cat> dataList) {
        Gson gson = new Gson();
        DataItems  dataItems = new DataItems();
        dataItems.set_cats(dataList);
        String jsonString = gson.toJson(dataItems);
        try (FileOutputStream fileOutputStream = context.openFileOutput(FILE_NAME, Context.MODE_PRIVATE)) {
            fileOutputStream.write(jsonString.getBytes());
            return true;
        } catch (Exception e) { e.printStackTrace(); }
        return false;
    }
    1 usage
    static List<Cat> importFromJSON (Context context) {
        try (   FileInputStream fileInputStream = context.openFileInput(FILE_NAME);
                InputStreamReader streamReader = new InputStreamReader(fileInputStream) ){
            Gson gson = new Gson();
            DataItems dataItems = gson.fromJson(streamReader, DataItems.class);
            return dataItems.getCats();
        } catch (IOException ex) { ex.printStackTrace();}
        return null;
    }
    4 usages
    private static class DataItems {
        2 usages
        private List<Cat> _cats;
        1 usage
        List<Cat> getCats() { return _cats; }
        1 usage
        public void set_cats(List<Cat> cats) { _cats = cats; }
    }
}
```

Рисунок 5.2 – Файл myJSON.java

Сформируем layout-файл главного Activity (рис. 5.3) со следующим содержимым: в корневом контейнере ConstraintLayout разместите два элемента EditText (идентификаторы nameText и ageText) для ввода текста (с соответствующей подсказкой в поле ввода) и кнопку для добавления ввода в список (addButton, обработчик add); две кнопки (saveButton и openButton, обработчики save и open) для сериализации (записи в json-файл) и десериализации из файла соответственно; а также элемент ListView (идентификатор list) для вывода списка на экран.

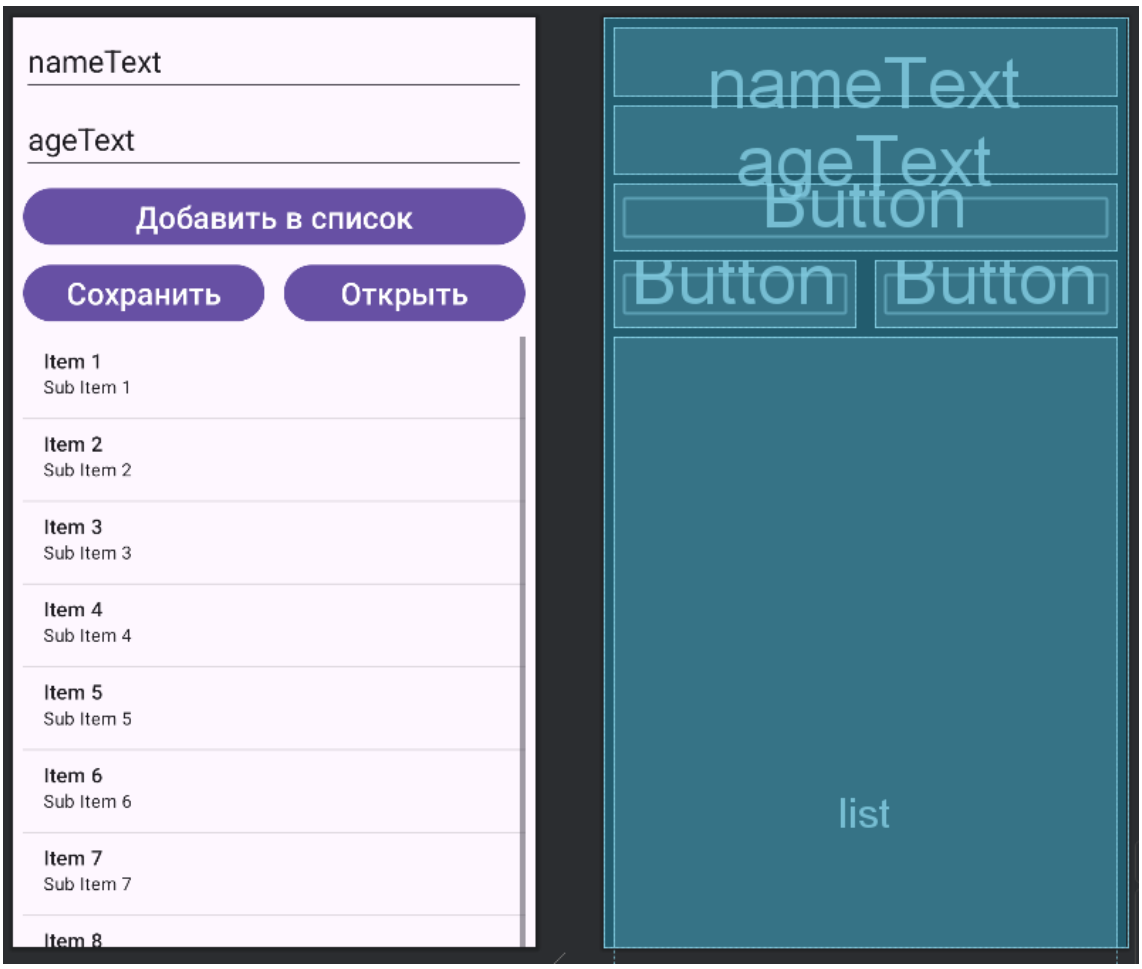


Рисунок 3.1 – Внешний вид разметки приложения

Определим код главного Activity рисунок 5.4.

```
private ActivityMainBinding _binding;
5 usages
private ArrayAdapter<Cat> _adapter;
7 usages
private List<Cat> _cats;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    _binding = ActivityMainBinding.inflate(getLayoutInflater());
    setContentView(_binding.getRoot());
    _cats = new ArrayList<>();
    _adapter = new ArrayAdapter<>( context: this, android.R.layout.simple_list_item_1, _cats);
    _binding.list.setAdapter(_adapter);
}
1 usage
public void add(View view) {
    String name = _binding.nameText.getText().toString();
    int age = Integer.parseInt(_binding.ageText.getText().toString());
    Cat cat = new Cat(name, age);
    _cats.add(cat);
    _adapter.notifyDataSetChanged();
}
1 usage
public void save(View view) {
    boolean result = myJSON.exportToJSON( context: this, _cats);
    if (result)
        Toast.makeText( context: this,  text: "Данные сохранены", Toast.LENGTH_SHORT).show();
    else
        Toast.makeText( context: this,  text: "Не удалось сохранить данные", Toast.LENGTH_SHORT).show();

}
1 usage
public void open(View view) {
    _cats = myJSON.importFromJSON( context: this);
    if (_cats != null) {
        _adapter = new ArrayAdapter<>( context: this, android.R.layout.simple_list_item_1, _cats);
        _binding.list.setAdapter(_adapter);
        Toast.makeText( context: this,  text: "Данные востановлены", Toast.LENGTH_SHORT).show();
    }
    else
        Toast.makeText( context: this,  text: "Не удалось открыть данные", Toast.LENGTH_SHORT).show();
}
```

Рисунок 5.3 – Файл Main.Activity.java

Запущенное приложение на эмуляторе предоставлено на рисунке 5.4.

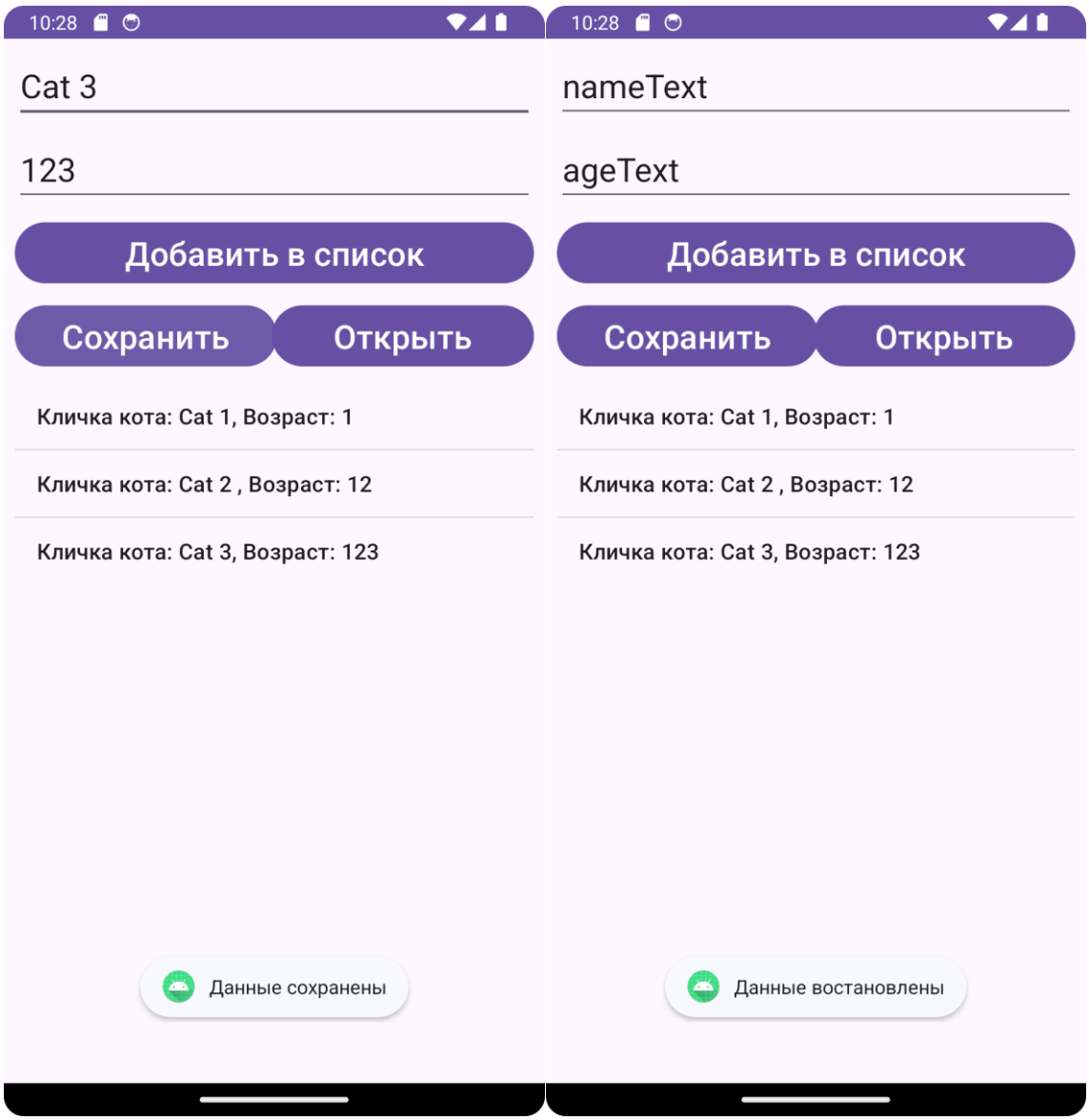


Рисунок 5.4 – Результат работы приложения

Е. ПРАКТИЧЕСКИЕ ЗАДАНИЯ

Задание 6

Реализуйте приложение для отображения содержимого большого текстового файла с возможностью прокрутки и отображения любой его части. Проверьте его работу на реальном файле.

Код отвечающий за открытие файла показан на рисунке 6.1

```
1 usage   shoki (main)
private void LoadText(View view) {
    File downloadDir = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS);
    File file = new File(downloadDir,  child: "Text.txt");
    FileInputStream fileInput = null;
    try {
        fileInput = new FileInputStream(file);
        byte[] bytes = new byte[fileInput.available()];
        fileInput.read(bytes);
        binding.text.setText(new String(bytes));
    } catch (IOException ex) {
        Toast.makeText( context: this, ex.getMessage(), Toast.LENGTH_SHORT).show();
        Log.d( tag: "my", ex.getMessage());
    } finally {
        try {
            if (fileInput != null) fileInput.close();
        } catch (IOException ex) {
            Toast.makeText( context: this, ex.getMessage(), Toast.LENGTH_SHORT).show();
            Log.d( tag: "my", ex.getMessage());
        }
    }
}
```

Рисунок 6.1 – Код для открытия локального файла

В качестве файла была использована книга Льва Николаевича Толстого «Война и мир». Приложение, запущенное на эмуляторе показано на рисунке 6.2.

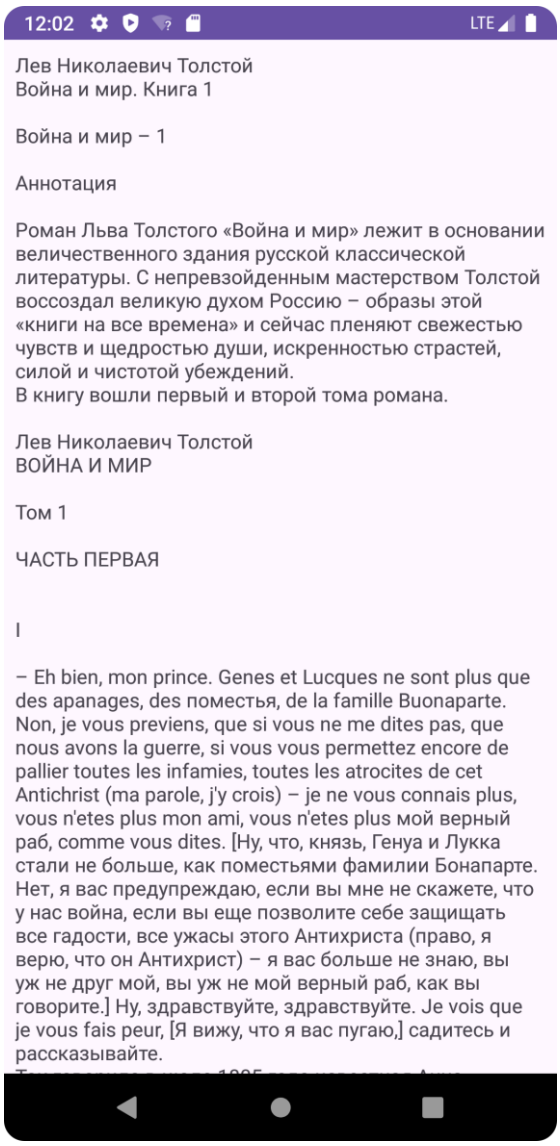
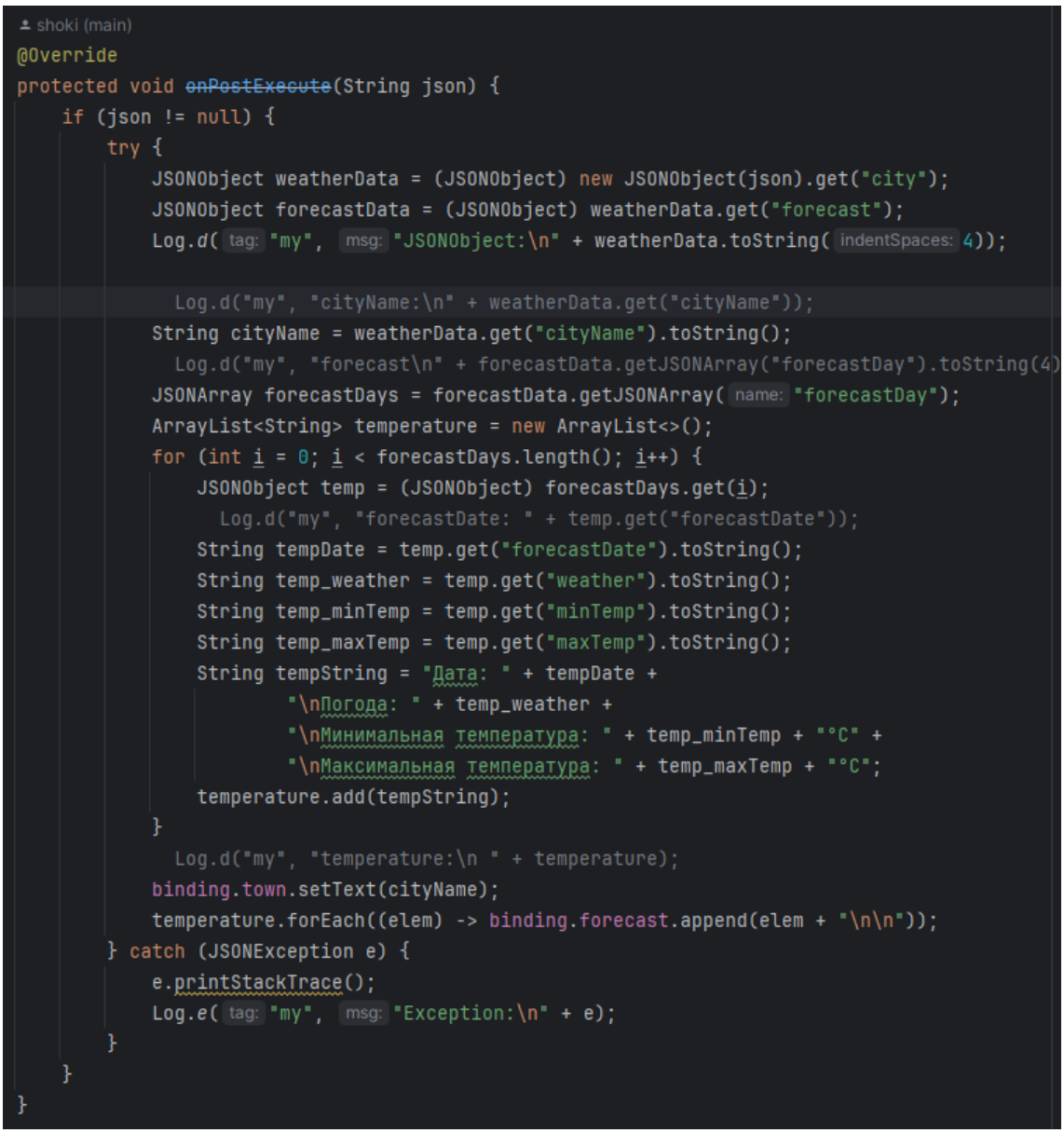


Рисунок 6.2 – Результат работы приложения

Задание 7

Реализуем приложение «Погода», в котором данные для отображения будут браться из json-файла, поставляемого Интернет-ресурсом «Всемирный информационный погодный сервис».

Парсер json файла показан на рисунке 7.1.



```
shoki (main)
@Override
protected void onPostExecute(String json) {
    if (json != null) {
        try {
            JSONObject weatherData = (JSONObject) new JSONObject(json).get("city");
            JSONObject forecastData = (JSONObject) weatherData.get("forecast");
            Log.d( tag: "my",  msg: "JSONObject:\n" + weatherData.toString( indentSpaces: 4));

              Log.d("my", "cityName:\n" + weatherData.get("cityName"));
            String cityName = weatherData.get("cityName").toString();
              Log.d("my", "forecast\n" + forecastData.getJSONArray("forecastDay").toString(4)
            JSONArray forecastDays = forecastData.getJSONArray( name: "forecastDay");
            ArrayList<String> temperature = new ArrayList<>();
            for (int i = 0; i < forecastDays.length(); i++) {
                JSONObject temp = (JSONObject) forecastDays.get(i);
                  Log.d("my", "forecastDate: " + temp.get("forecastDate"));
                String tempDate = temp.get("forecastDate").toString();
                String temp_weather = temp.get("weather").toString();
                String temp_minTemp = temp.get("minTemp").toString();
                String temp_maxTemp = temp.get("maxTemp").toString();
                String tempString = "Дата: " + tempDate +
                        "\nПогода: " + temp_weather +
                        "\nМинимальная температура: " + temp_minTemp + "°C" +
                        "\nМаксимальная температура: " + temp_maxTemp + "°C";
                temperature.add(tempString);
            }
              Log.d("my", "temperature:\n " + temperature);
            binding.town.setText(cityName);
            temperature.forEach((elem) -> binding.forecast.append(elem + "\n\n"));
        } catch (JSONException e) {
            e.printStackTrace();
            Log.e( tag: "my",  msg: "Exception:\n" + e);
        }
    }
}
```

Рисунок 7.1 – Парсер json файла

Реализованное приложение при запуске скачивает json файл с сайта и после парса файлы выводят ее данные 7.2.

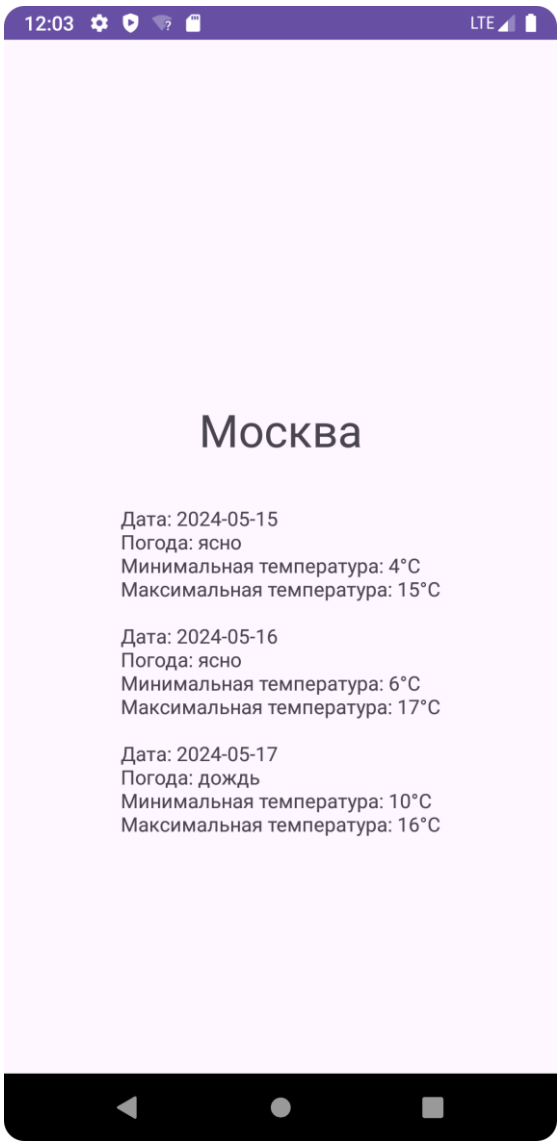


Рисунок 7.2 – Результат работы приложения

ВЫВОД

В результате выполнения практической работы были получены навыки по работе с json файлами с записью и чтению из памяти из устройства. А также как работать с запросами на сайты.