Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Петрозаводский государственный университет»
Институт математики и информационных технологий
Кафедра информатики и математического обеспечения

Отчет по итоговому проекту дисциплины «Разработка приложений для мобильных ОС»

Выполнил: студент группы 22207
Павлов Максим Павлович

Преподаватель: Димитров В. М.,
старший преподаватель

Петрозаводск
2022

Содержание

1. Введение

Тема «Системы счисления» входит в школьную программу по информатике и начинает изучаться уже с 8 класса. Она включает в себя трудоёмкие вычислительные операции, на проверку которых вручную уходит много времени. До сих пор практически не рассматривалось использование систем счисления больше 36. Хотя системы счисления активно используются в криптографии и математике [1]. Мобильное приложение, позволяющее производить вычисления и переводы над числовыми последовательностями в системах счисления, может быть полезна при решении особого круга прикладных задач.

Для создания мобильного приложения под операционную систему Android была выбрана среда разработки Android Studio и язык программирования Java. Android Studio — официальная среда разработки Android приложений, основанная на IntelliJ IDEA. Она позволяет разрабатывать приложения для смартфонов, планшетов, часов, телевизоров и других устройств на этой ОС. Android Studio предоставляет много функций для создания приложений, например: гибкую систему сборки на основе Gradle; шаблоны основных макетов и компонентов Android; обширные инструменты и фреймворки для тестирования; статический анализатор кода, позволяющий находить проблемы производительности, несовместимости версий и многое другое.

Цель итогового проекта – продемонстрировать знания и практические умения по созданию приложений под ОС Android на примере приложения «Калькулятор систем счисления» для работы в различных системах счисления.

Задачи:

1. Разработать архитектуру приложения, абстрактные функции, способные работать с двумя числами (переводить их между системами счисления) и выводить результат в зависимости от выбранной операции (сложение, умножение и т.д.)

2. Спланировать многопоточность приложения и асинхронность вычислений. Так как данное приложение может быть в дальнейшем расширено алгоритмами строковой арифметики, которые потенциально могут обрабатываться часами, то вычислительные операции над двумя числами должны быть многопоточными и выполняться параллельно.

3. Оформить приятный дизайн приложения, удобство расположения основных элементов взаимодействия с пользователем.

4. Перенести разработанные идеи в мобильное приложение.

2. Основная концепция приложения

Для работы калькулятора пользователю нужно ввести число в поле «Первое число». Затем в списке «Система счисления первого числа» необходимо указать систему счисления числа. Если это не будет сделано, то по умолчанию будет выбрана 10-ая система счисления.

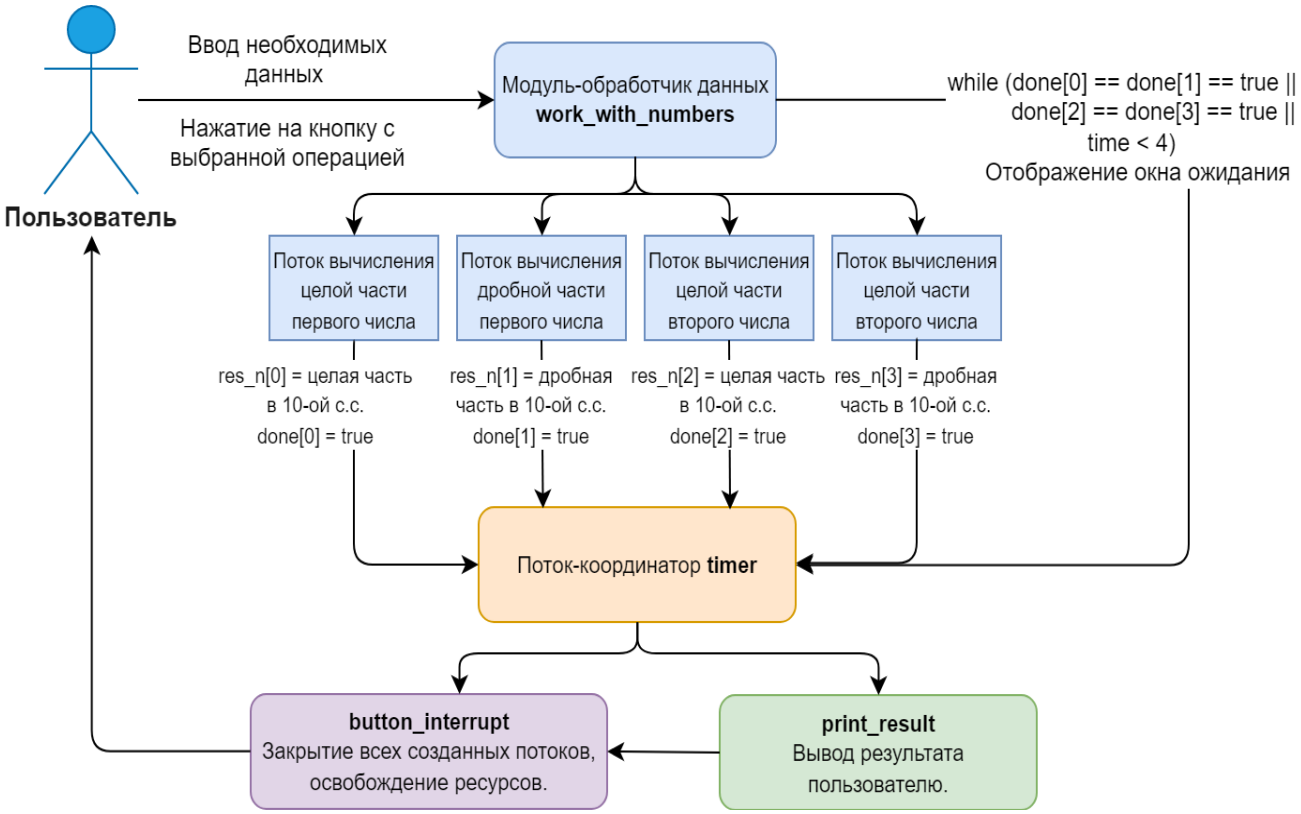

Рисунок 1. Схема работы вычислительных операций в приложении

В поле «Второе число» нужно ввести число, если необходимо произвести умножение, сложение, вычитание, деление или возведение в степень. Если необходимо просто перевести первое число, то второе число не будет учитываться, даже если оно введено. Если второе число введено, то в поле «Система счисления второго числа» необходимо указать систему счисления числа. Если это не будет сделано, то по умолчанию будет выбрана 10-ая система счисления.

В поле «Система счисления результата» необходимо указать систему счисления, в которой нужно получить результат. Если это не будет сделано, то по умолчанию будет выбрана 10-ая система счисления. Для работы калькулятора, когда введены все необходимые данные, необходимо нажать на кнопку с соответствующей операцией.

3. Интерфейс приложения

Основной экран приложения (Рисунок 2) содержит: поля ввода первого и второго числа; поля ввода систем счисления первого, второго числа и результата (число от 2 до 94); копки математических операций с введёнными числами (сложение, вычитание, умножение, деление и возведение в степень), перевода первого числа, кнопку настроек и кнопку меню. Закругления всех элементов реализованы с помощью дополнительного Relative Layout.

Рисунок 2. Основной экран приложения.
Слева – обычный, справа – по нажатию на кнопку меню

Поля ввода/вывода могут считать последовательность любой длины. Также поддерживается перемещение вправо/влево по введённым данным. По нажатии на кнопку «Меню» появляется список, с помощью которого можно переместиться на соответствующую страничку приложения (Рисунок 3) и ознакомиться с: инструкцией по работе приложения, историей систем счисления, информацией о поддерживаемых системах счисления. Все вспомогательные экраны оснащены кнопкой «Home», по нажатию на которую пользователь может вернуться на основной экран. Вся разметка страничек сделана в Relative Layout.

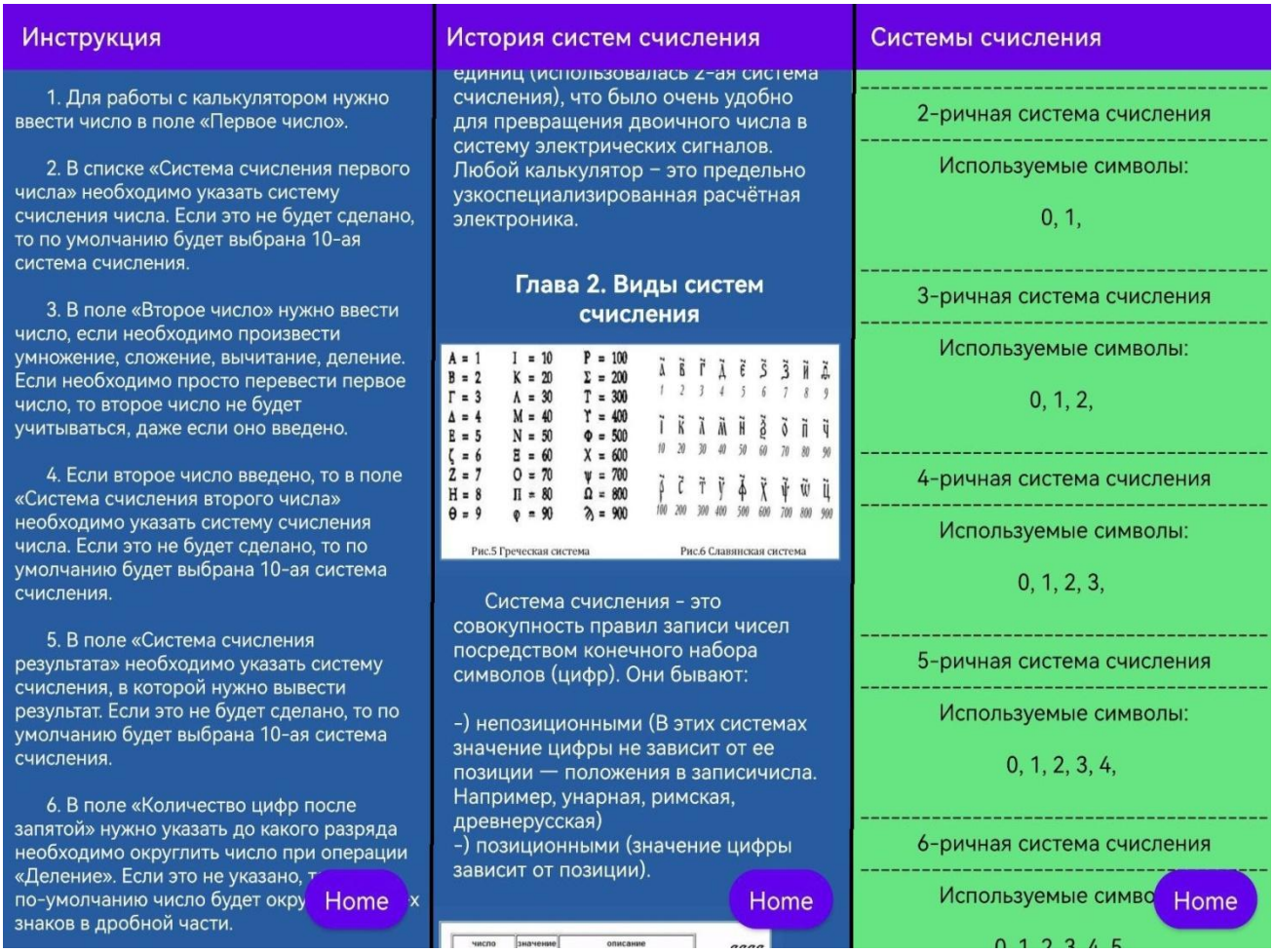

Рисунок 3. Вспомогательные экраны приложения.
Слева – инструкция по работе приложения, по центру – история систем счисления, справа – поддерживаемые системы счисления

Благодаря грамотно расставленным якорям дизайн приложения будет масштабирован в соответствии с разрешением экрана устройства, и будет выглядеть удовлетворительно абсолютно на всех устройствах. Также программа оснащена приятными GIF-анимациями, за их воспроизведение отвечает библиотека Glide [2].

4. Программная реализация

Для организации работы приложения, упрощения дальнейших улучшений и поддержания программного кода была создана модульная архитектура ПО (Рисунок 1). Каждый модуль базируется на трёх основных алгоритмах: перевод целой части числа, дробной части в десятичную систему счисления и обратно.

Перевод целой части числа в десятичную систему счисления:

```
public String ceil_to_10(String a, int original_system) {
    // a - строка, содержащее число для перевода
    // original_system - система счисления полученного чисел
    // result_system - система счисления результата
    if (a.isEmpty() || a == " ")
        return "Неправильный ввод числа или его системы счисления!";
    //Проверка на минус
    short minus = 1;
    StringBuilder r = new StringBuilder();
    for (int i = 0; i < a.length(); i++)
        if (a.charAt(i) == '-') minus *= -1; else r.append(a.charAt(i));
    a = r.toString();

    long k =0;
    long d = a.length() - 1;
    long result;
        boolean Current = false;

    // Переводим число в 10-ую систему счисления
    for (int i = 0; i < a.length(); i++) {
        result = 0;
        // Диапазоны ASCII
        if (((int) a.charAt(i) > 47) && (int) a.charAt(i) < 58)
            result = (int) a.charAt(i) - 48;
        if (((int) a.charAt(i) > 64) && (int) a.charAt(i) < 91)
            result = (int) a.charAt(i) - 55;
        if (((int) a.charAt(i) > 32) && (int) a.charAt(i) < 48)
            result = (int) a.charAt(i) + 3;
        if (((int) a.charAt(i) > 90) && (int) a.charAt(i) < 127)
            result = (int) a.charAt(i) - 40;
        if (((int) a.charAt(i) > 57) && (int) a.charAt(i) < 65)
            result = (int) a.charAt(i) - 29;

        if (result >= original_system) Current = true;
        if (a.charAt(i) != '0' && !Current)
            k += result * Math.pow(original_system, d);
        d -= 1;
    }

    if (k > Long.MAX_VALUE-1) return "Слишком большое число. Такие пока что не
поддерживаются";
    if (!Current) return String.valueOf(k*minus);
    return "Неправильный ввод числа или его системы счисления!";
}
```

Помимо перевода чисел между системами счислений данные функции также осуществляют проверки на корректность полученных данных. И в случае

ошибки они возвращают соответствующее сообщение. Функция, вызвавшая их, обрабатывает данное сообщение и записывает в массив результата res_n уже не число, а данное сообщение, вместе с тем, устанавливая флаг ошибки. При выводе, так как данный флаг установлен, то будет выведено данное сообщение (сообщение последнее ошибки при переводе). Если ошибок будет несколько, то пользователю выведется только последнее из них. Также проверяется и корректность введённых символов. Например, в двоичной системе счисления можно использовать только 0 и 1, другие цифры считаются некорректным вводом. При работе с дробной частью число округляется до 6 знаков после запятой. Округление происходит согласно правилам BigDecimal.setScale(6, RoundingMode.HALF_UP), т. е. округление вверх с избытком.

Перевод целой части числа из десятичной системы счисления в нужную:

```
public String ceil_to_res(long k, int result_system){
    // result_system - система счисления результата
=    String minus = "";
    if (k == 0) return "0";
    if (k > Long.MAX_VALUE-1) return "Слишком большое число. Такие пока что не
поддерживаются";
    if (k < 0) { minus = "-"; k = -k;}

    String a;
    StringBuilder aBuilder = new StringBuilder();
    if (result_system <= 10) {
        // Если система счисления результата <= 10, то метод перевода следующий
        while (k > 0) {
            aBuilder.append(k % result_system);
            k = k / result_system;
        }
    } else {
        // Иначе система счисления результата > 10, и метод перевода другой
        while (k > 0) {
            if (k % result_system < 10) {
                aBuilder.append(k % result_system);
            }
            else {
                long i = (k % result_system);
                // Диапозоны ASCII
                if (i < 10)
                    aBuilder.append((char) (i + 48));
                if (i > 9 && i < 36)
                    aBuilder.append((char) (i + 55));
                if (i > 35 && i < 51)
                    aBuilder.append((char) (i - 3));
                if (i > 50 && i < 87)
                    aBuilder.append((char) (i + 40));
                if (i > 86 && i < 94)
                    aBuilder.append((char) (i + 29));
            }
            k = k / result_system;
        }
    }
    a = aBuilder.toString();
    return minus + new StringBuilder(a).reverse();
}
```

Перевод дробной части числа в десятичную систему счисления:

```
public String fractional_to_10(String a, int original_system) {
    if (a.isEmpty() || a == " ") return "Неправильный ввод числа или его системы
счисления!";
    a = "0." + a;
    double s = 0.0;
    int stop = 150;
    boolean Current = false;
    long k = a.length();

    for (int i = 2; i < k; i++) {
        // Диапозоны ASCII
        if (((int) a.charAt(i) > 47) && (int) a.charAt(i) < 58) {
          s += ((int) a.charAt(i) - 48) * (1 / Math.pow(original_system, i - 1));
          if ((int) a.charAt(i) - 48 >= original_system) Current = true;
        }
        else if (((int) a.charAt(i) > 64) && (int) a.charAt(i) < 91) {
          s += ((int) a.charAt(i) - 55) * (1 / Math.pow(original_system, i - 1));
          if ((int) a.charAt(i) - 55 >= original_system) Current = true;
        }
        else if (((int) a.charAt(i) > 32) && (int) a.charAt(i) < 48) {
          s += ((int) a.charAt(i) + 3) * (1 / Math.pow(original_system, i - 1));
          if ((int) a.charAt(i) + 3 >= original_system) Current = true;
        }
        else if (((int) a.charAt(i) > 90) && (int) a.charAt(i) < 127) {
          s += ((int) a.charAt(i) - 40) * (1 / Math.pow(original_system, i - 1));
          if ((int) a.charAt(i) - 40 >= original_system) Current = true;
        }
        else if (((int) a.charAt(i) > 57) && (int) a.charAt(i) < 65) {
          s += ((int) a.charAt(i) - 29) * (1 / Math.pow(original system, i - 1));
          if ((int) a.charAt(i) - 29 >= original_system) Current = true;
        }
        else {
            int number = parseInt(String.valueOf(a.charAt(i)));
            s += number * (1 / Math.pow(original_system, i - 1));
            if (number >= original_system) Current = true;
        }
        stop -= 1;
        if (stop == 0) break;
    }
    a = String.valueOf((s));
    if (!Current) return a.substring(2);

    return "Неправильный ввод числа или его системы счисления!";
}
```

Математические операции осуществляться по одному типовому алгоритму. Введённое число преобразовывается в строковую переменную. По наличию точки определяется наличие дробной части. Строка разбивается на две части: целую и дробную (при наличии). Обе части переводятся в десятичную систему счисления. Определение представления текстовых символов (отличных от цифр) осуществляется с помощью ASCII (разность номера текущего символа в таблице и символа «0») для более приятного восприятия символьного результата. Это позволяет расширить системы счисления дальше тридцатишестиричной (в данной

реализации предельная система счисления 93). После символов английского алфавита будут использованы соответствующие символы ASCII (используются лишь однобайтовые символы основной латиницы, начиная с символа «0» и заканчивая «~»). Первый диапазон символов – цифры ASCII (номера 48-57), второй – буквы латинского алфавита в верхнем регистре (65-90), третий – символы и знаки препинания (33-47), четвертый – специальные символы (58-64), пятый – буквы латиницы в нижнем регистре и оставшиеся однобайтовые символы (91-126). Разбиение ASCII на такие диапазоны и связь их порядковых номеров с основанием системы счисления обусловлена уже устоявшимися обозначениями.

Перевод дробной части числа из десятичной системы счисления в нужную:

```
public String fractional_to_res(double a, long result_system) {
    if (a == 0.0) return "0";
    StringBuilder s = new StringBuilder();
    int stop = 150;
    while (a > 1e-6) {
        Double number = a * result_system;
        number = new BigDecimal(String.valueOf(number)).setScale(6,
RoundingMode.HALF_UP).doubleValue();
        String h = String.valueOf(number);

        StringBuilder r = new StringBuilder();
        for (int i = 0; i < h.length(); i++)
            if (h.charAt(i) == '.') break; else r.append(h.charAt(i));


        int c = Integer.parseInt(r.toString());
        // Диапозоны ASCII
        if (c < 10) s.append((char) (c + 48));
        if (c > 9 && c < 36) s.append((char) (c + 55));
        if (c > 35 && c < 51) s.append((char) (c - 3));
        if (c > 50 && c < 87) s.append((char) (c + 40));
        if (c > 86 && c < 94) s.append((char) (c + 29));

        stop -= 1; if (stop == 0) break;
        StringBuilder drod = new StringBuilder("0.");
        int dlin = h.length(), i = 0;

        while (i < dlin) {
            if (h.charAt(i) == '.') break;
            i += 1;
        }
        i += 1;

        while (i < dlin) {
            drod.append(h.charAt(i));
            i += 1;
        }

        a = Double.parseDouble(drod.toString());
    }
    return (s.toString());
}
```

Аналогичные операции производятся и со вторым числом. Затем над целым и дробными частями осуществляются математические операции, обе части результата переводятся в нужную систему счисления и выводятся пользователю. Такой способ реализации основан на общепринятых правилах перевода в системах счисления (совпадает с «ручным» переводом), что позволяет проконтролировать основные этапы вычисления.

Функция вывода результата пользователю

```
void print_result(String First_n, int First_s, String Second_n, int Second_s, int
Res_s, int mode){
        runOnUiThread(() -> {
            TextView time_tv = rLayout.findViewById(R.id.time_tv);
            TextView user message = rLayout.findViewById(R.id.user message);
            TextView message_to_user = rLayout.findViewById(R.id.message_to_user);
            Button button_stop = rLayout.findViewById(R.id.button_stop);
            time_tv.setText("");
            user_message.setText("Результат");

            Log.d("BETA_RESULT", res_n[0]+" | "+res_n[1]+" | "+res_n[2]+" |
"+res_n[3]);
            Double  result, result_1 = 0.0, result_2 = 0.0;
            //Проверка на минус
            short minus_1 = 1, minus_2 = 1;
            if (res_n[0] != null && error == 0) { result_1 +=
Double.parseDouble(res_n[0]); if (res_n[0].contains("-")) minus_1 *= -1; }
            if (res_n[2] != null && error == 0) { result_2 +=
Double.parseDouble(res_n[2]); if (res_n[2].contains("-")) minus_2 *= -1; }
            if (res_n[1] != null && error == 0) result_1 += minus_1 *
Double.parseDouble("0."+res_n[1]);
            if (res_n[3] != null && error == 0) result_2 += minus_2 *
Double.parseDouble("0."+res_n[3]);

            if (mode == 0 && error == 0) {
                 result = result_1 + result_2;
                Log.d("BETA_RESULT", result + " "+ result_1 + " " + result_2);
                message_to_user.setText("\t\t" + First_n + "(" + First_s + ") + " +
Second_n + "(" + Second_s + ") = " + new Perevod().ceil_to_res(result.intValue(),
Res_s) +"." + new Perevod().fractional_to_res(Math.abs(result)-
Math.abs(result.intValue()), Res_s) + "(" + Res_s + ")");
            }
            else if (mode == 1 && error == 0) {
                result = result_1 - result_2;
                message_to_user.setText("\t\t" + First_n + "(" + First_s + ") - " +
Second_n + "(" + Second_s + ") = " + new Perevod().ceil_to_res(result.intValue(),
Res_s) + "." + new Perevod().fractional_to_res(Math.abs(result)-
Math.abs(result.intValue()), Res_s) + "(" + Res_s + ")");
            }
            else if (mode == 2 && error == 0) {
                if (result_2 == 0.0) result_2 = 1.0;
                result = result_1 / result_2;
                message_to_user.setText("\t\t" + First_n + "(" + First_s + ") / " +
Second_n + "(" + Second_s + ") = " + new Perevod().ceil_to_res(result.intValue(),
Res_s) + "." + new Perevod().fractional_to_res(Math.abs(result)-
Math.abs(result.intValue()), Res_s) + "(" + Res_s + ")");
            }
            else if (mode == 3 && error == 0) {
```

```
                result = result_1 * result_2;
                message_to_user.setText("\t\t" + First_n + "(" + First_s + ") * " +
Second_n + "(" + Second_s + ") = " + new Perevod().ceil_to_res(result.intValue(),
Res_s) + "." + new Perevod().fractional_to_res(Math.abs(result)-
Math.abs(result.intValue()), Res_s) + "(" + Res_s + ")");
            }
            else if (mode == 4 && error == 0) {
                result = Math.pow(result_1, result_2);
                message_to_user.setText("\t\t" + First_n + "(" + First_s + ") ^ " +
Second_n + "(" + Second_s + ") = " + new Perevod().ceil_to_res(result.intValue(),
Res_s) + "." + new Perevod().fractional_to_res(Math.abs(result)-
Math.abs(result.intValue()), Res_s) + "(" + Res_s + ")");
            }
            else if(error == 0)
                message_to_user.setText("\t\t"+First_n+"("+First_s+") = "+new
Perevod().ceil_to_res(result_1.intValue(), Res_s)+"."+ new
Perevod().fractional_to_res(Math.abs(result_1) - Math.abs(result_1.intValue()),
Res_s)+"("+Res_s+")");
            else
                message_to_user.setText(res_n[error-1]);
            button_stop.setText("Продолжить");
        });
}
```

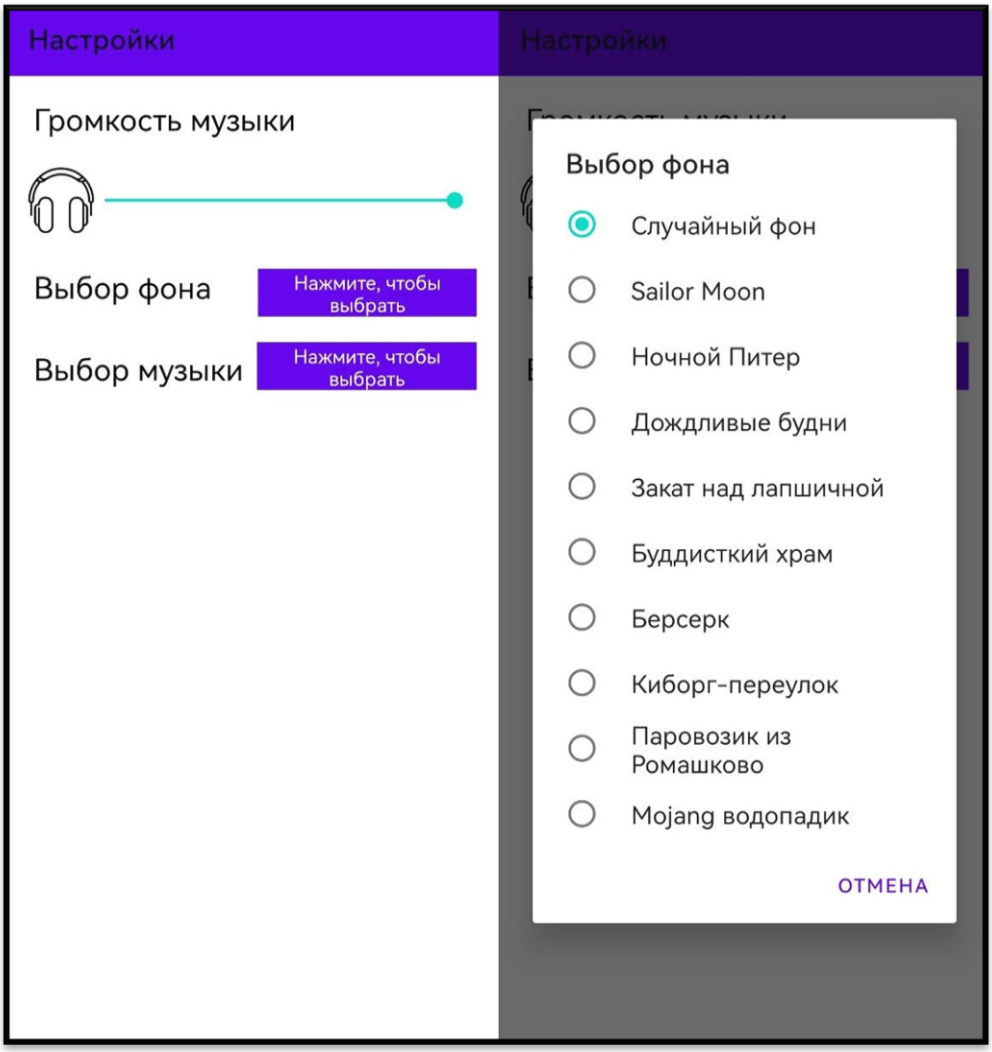

Рисунок 4. Настройки приложения

Функция вывода результата действует согласно выбранному пользователем mode: -1 – перевод первого числа, 0 – сумма двух чисел, 1 – разность, 2 – частное, 3 – произведение, 4 – возведение в степень. Если при переводе были замечены ошибки, то выводится соответствующее сообщение, предварительно записанное в res_n[error-1].

Отдельные потоки для работы с целой и дробной частью чисел:

```
void ceil_part(String First_n, int First_s, int index){
    /*
     * Переводич целую часть в фоновом потоке
     */
    ceil_part = new Thread(() -> {
        res_n[index]= new Perevod().ceil_to_10(First_n, First_s);
        if (res_n[index].equals("Неправильный ввод числа или его системы
счисления!") ||
                res_n[index].equals("Слишком большое число. Такие пока что не
поддерживаются")) error = index+1;
        done[index] = true;
    });
    ceil_part.start();
}

void fractional part(String First n, int First s, int index) {
    /*
     * Переводич дробную часть в фоновом потоке
     */
    fractional_part = new Thread(() -> {
        res_n[index]= new Perevod().fractional_to_10(First_n, First_s);
        if (res_n[index].equals("Неправильный ввод числа или его системы
счисления!") ||
                res_n[index].equals("Слишком большое число. Такие пока что не
поддерживаются")) error = index+1;
        done[index] = true;
    });
    fractional_part.start();
}
```

Приложение также оснащено настройками (Рисунок 4). Они реализованы с помощью SharedPreference, т. е. выбранные настройки сохраняются в приложении. И даже после перезагрузки телефона, или переустановки приложения, они сохранятся и их можно будет использовать. Настройки позволяют пользователю непосредственно настроить громкость музыки в приложении, выбрать фон (всегда случайный или один из предложенных), выбрать музыку (случайную или одну из предложенных). Для отображения Toolbar используется дополнительный Relative Layout.

5. Заключение

В ходе итогового проекта были задействованы основные навыки создания приложений под ОС Android: базовая верстка экранов; организация многоэкранного приложения (5 экранов); работа с отдельными потоками и ассинхроннное выполнение частей кода; вынесение кода на отдельные функции, составление из них отдельных тематических классов (6 Java-классов).

Данная программа может использоваться как для проверки небольших вычислений, сделанных вручную, так и для длинной арифметики с высокой точностью в различных системах счисления (включая привычную десятичную). Возможно дальнейшее добавление дополнительных функций, реализующих другие математические операции (вычисление факториала числа и другие), поддержание работы приложения, его оптимизация (например, перевод алгоритма работы на низкоуровневый язык) и увеличение максимальной поддерживаемой системы счисления, путём использования диапазонов символов Unicode, перевод вычислений на строковую арифметику, чтобы уйти от целочисленных лимитов типов данных.

Системы счисления активно использовались на этапах зарождения ЭВМ, телеграфии (кодирование букв и чисел последовательностью нулей и единиц), создания штрих-кодов [3]. Но в наши дни одна из главных сфер применения данного приложения – кодирование (сжатие) числовой последовательности (например, цепного кода). «…Любую упорядоченную последовательность можно представить в виде ряда упорядоченных множеств и для любого упорядоченного множества можно создать свою позиционную систему счисления…» [4]. Рационально использовать данный алгоритм и для отправки данных, так как при передаче данных используются другие системы счисления (например, кодирование base64), когда есть дополнительные ограничения на то, какие значения могут иметь передаваемые байты (например, это должны быть только печатные символы ASCII).

Список литературы

1. Будкина И. М. Научному прогрессу - творчество молодых: материалы XV международной молодежной научной конференции по естественнонаучным и техническим дисциплинам (Йошкар-Ола, 17-18 апреля 2020 года): в 2 ч. / редкол.: Д. В. Иванов [и др.]. / И. М. Будкина, А. М. Кораблев // Системы счисления. Йошкар-Ола: Поволжский государственный технологический университет, 2020. Ч. 1. С. 14—16.

2. Glide Documents [Электронный ресурс]: https://github.com/bumptech/glide (дата обращения: 10.12.2022).
3. Гашков С. Б. Системы счисления и их применение. Москва : МЦНМО, 2004. 52 с.
4. Лебедева А. В. Система счисления рядов упорядоченных множеств // Математика и математическое моделирование: сборник материалов XII всероссийской молодежной научно-инновационной школы. Саратов: Интерконтакт, 2018. С. 21—22.