УПРАВЛЕНИЕ ОБРАЗОВАНИЯ ГОРОДА ПЕНЗЫ

МКУ «ЦЕНТР КОМПЛЕКСНОГО ОБСЛУЖИВАНИЯ И МЕТОДОЛОГИЧЕСКОГО ОБЕСПЕЧЕНИЯ УЧРЕЖДЕНИЙ ОБРАЗОВАНИЯ» Г. ПЕНЗЫ

МУНИЦИПАЛЬНОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

«ЛИЦЕЙ СОВРЕМЕННЫХ ТЕХНОЛОГИЙ УПРАВЛЕНИЯ № 2» Г. ПЕНЗЫ

«Старт в науку» 2020, Пенза

ПРИЛОЖЕНИЕ “ИНФА НА 5“ ДЛЯ ПОДГОТОВКИ К ЕГЭ

Выполнил:

Ларюшкин Сергей Юрьевич

ученик 11 “Г”

МБОУ ЛСТУ № 2

Научный руководитель:

Алексеева Римма Юрьевна,
учитель МБОУ ЛСТУ № 2

Пенза 2020 год

Содержание

Содержание **1**

Введение **2**

1. Аналитическая часть **3**

1.1. Аннотация проекта 3

1.2. Идея проекта 4

2. Проектная часть **5**

2.1 Этапы проекта 5

2.2 Процесс разработки 6

3. Итоги разработки **12**

Выводы **13**

Список используемых источников **14**

Приложения **15**

Приложение 1 - Код 13го решатора 15

Приложение 2 - Создание курсов 17

Приложение 3 - Класс создания страниц курсов и справочника 18

Приложение 4 - Класс представления номеров 20

Приложение 5 - Фрагмент решатора №26 23

Приложение 6 - Применение алгоритма Нарайаны 24

Приложение 7 - Применение алгоритма Дейкстры 26

Введение

В современном мире очень популярно направление информационных технологий. Появляются новые специальности, развиваются уже существующие, в вузах начинают свою работу новые факультеты. Многие ученики хотят связать свое обучение и дальнейшую карьеру именно с данным перспективным направлением. Для поступления на эти специальности требуется получить высокие баллы на ЕГЭ по информатике. С 2009 года ЕГЭ является единственной формой выпускных экзаменов в школе и основной формой вступительных экзаменов в вузы. Всем тем, кто выбрал этот сложный путь, мы хотим помочь.

Сдача ЕГЭ – задача не из легких. Многие ученики сталкиваются с проблемами на уроках в школе при подготовке к ЕГЭ. Большое количество формул, определений, типов задач не даёт покоя ни школьникам, ни их родителям, ни учителям. Ученик может не усвоить материал, даваемый на уроке, не разобраться в нем, не запомнить, не успеть записать определение или формулу, не понять алгоритм решения задач, или вовсе не узнать тему из-за пропуска занятия. Основными проблемами являются огромное количество сложного материала и ограниченное время, а также индивидуальное затруднение в определенных темах.

Для понимания принципов решения задач по информатике для ЕГЭ важным является не столько получение правильного ответа, сколько сам процесс решения, алгоритм и последовательность операций, приводящих к правильному ответу.

Поэтому мы поставили перед собой цель: помочь школьникам, сдающим ЕГЭ по информатике в усвоении материала и тренировке решения задач, созданием простого в использовании Android-приложения «Инфа на 5» с конспектами тем, знание которых понадобиться при решении ЕГЭ, базой задач, тестами, специализированным калькулятором и возможностью получить развернутое решение и ответ на задачу, с заданными вами величинами.

Для достижения поставленной цели были определены следующие задачи:

- Выбрать и описать класс задач, которые будут рассмотрены в проекте
- Разработать алгоритм решения
- Выбрать платформу для реализации приложения
- Написать программу
- Протестировать

1. Аналитическая часть

1.1. Аннотация проекта

С 2009 года ЕГЭ является единственной формой выпускных экзаменов в школе и основной формой вступительных экзаменов в вузы. Сейчас направление информационных технологий очень популярно, поэтому ЕГЭ по информатике сдают многие школьники, которым мы хотим помочь.

Многие выпускники испытывают проблемы при подготовке к ЕГЭ по информатике большое количество формул, определений, типов задач не даёт покоя ни школьникам, ни учителям, ни их родителям, даже друзьям, что теряют своего товарища за справочниками, учебниками, задачниками, за подготовительными курсами, репетиторами.

1.2. Идея проекта

Мы видим итогом своей работы мобильное приложение, которое будет помогать при подготовке к ЕГЭ на платформе Android. В приложении должны быть справочные материалы, курсы для теоретической подготовке по информатике, помощник в решении задач и задачник.

Если ученику требуется разобраться в теме, он может найти теорию и рассмотреть материал, пример решения задачи. Для того чтобы все это найти, может понадобится некоторое время. В нашем приложении можно найти и справочник, и решатор задач (с выводом решения на экран), что поможет сэкономить время и в принципе даст возможность разобраться в материале самостоятельно без особого труда.

В нашем продукте можно выделить следующий ряд положительных отличительных черт: наглядная форма вывода решения и ответа, широкий выбор тем и типов заданий для тренировки. Кроме того, несомненным плюсом является то, что решатор с введенными вами значениями в дано, калькулятор с логическими функциями, объяснение тем, тестирующая система, как по теории, так и по практике и справочник - все это собрано в одном приложении, к тому же, оно адаптировано под мобильное устройство, (так как это зачастую удобнее, чем сайт), имеет понятный интерфейс, регистрации и выхода в интернет не требуется. Все эти плюсы далеко не всегда встречаются у существующих аналогов.

2. Проектная часть

2.1 Этапы проекта

В процессе проектирования приложения работа была разбита на следующие этапы:

1. Выбор, установка и настройка среды разработки. На этом этапе была установлена среда Android Studio

2. Разработка структуры проекта - выбрана общая структура приложения, состоящая из 4-х пунктов: Справочники, курсы, решаторы, задачник. (Рис. 2.1.1)

3. Разработка программного кода - на этом этапе приступили к программированию заложенных функциональных частей приложения.

4. Тестирование и корректировка приложения - на этом этапе использовалось ручное тестирование всех функций приложения, прорешивание задач, сравнение полученных результатов с эталонными.

Рис. 2.1.1 - Структура приложения

2.2 Процесс разработки

В самом начале, ещё только начиная изучать Java и Android, алгоритмы решаторов писались на Python, затем переносились на Java. Сейчас код сразу же пишется на Kotlin и пускается в ход.

При разработке существующих решаторов для написания алгоритмов использовался язык Python, который своей простотой помогал нам быстро переносить их из головы в код.

Решатор 13го номера

Первым был сделан решатор 13-го номера (см. рис. 4.1), т.к. он являлся лучшим вариантом для того, чтоб делать под него первый решатор: он является обычной задачей на “дано-найти-решение” и при этом не слишком вариативен. Алгоритм создавался исходя из того, как люди решают задачи: мы вспоминаем, какие формулы подходят для того, чтоб найти ответ, если нам не хватает каких-нибудь значений, то мы либо ищем формулу, где нам всё известно, либо начинаем искать формулы для нахождения недостающих значений. Выходит, что мы начинаем решать в задаче о нахождении какого-нибудь *a* уже *b,* далее, если надо, то ищем по тому же алгоритму *c* и так далее.

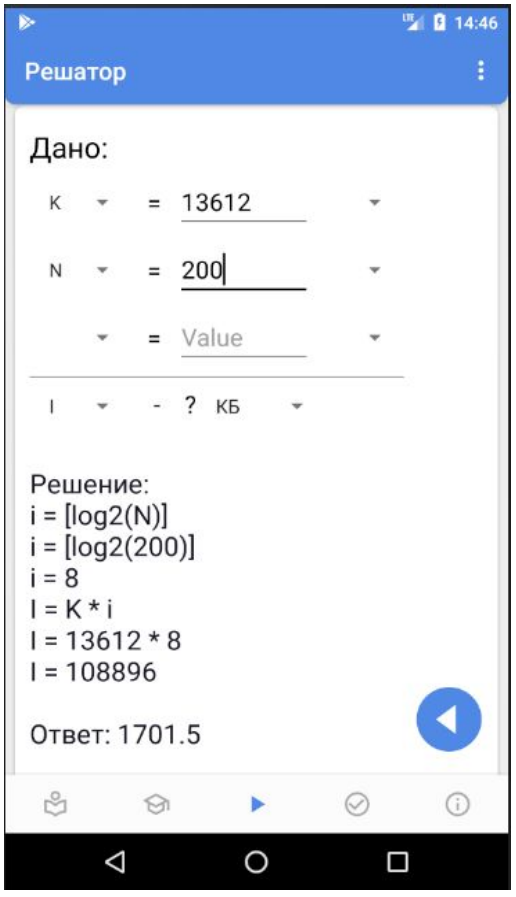


рис. 2.2.1 - Решатор 13го

Код решатора 13го, написанный на Python (см. в Приложение №1)

Решатор 26го номера

Из решаторов следующим был выбран решатор 26 задачи, который научил по-другому разрабатывать решаторы на будущее. В процессе создания стало ясно, что ввод условия должен стать более гибким, а может, даже в принципе ориентированным на схожесть с поисковой строкой, хотя последнее сейчас лишь планируется.

Хотя алгоритм на данный момент работает только с типом задач 26 про камушки, в целом, его потребуется не сильно изменить, для того чтобы он работал и на других видах этой задачи.

Программа получает входные данные в виде условия задачи, а точнее начальной ситуации, шаги, которые доступны игрокам, условия окончания игры и строит дерево всех возможных ходов игроков.

Далее определяется победитель на каждой ветке. Далее мы рекурсивно просчитываем, кто победит (см. Приложение № 5).

Пример работы 26го решатора представлен ниже

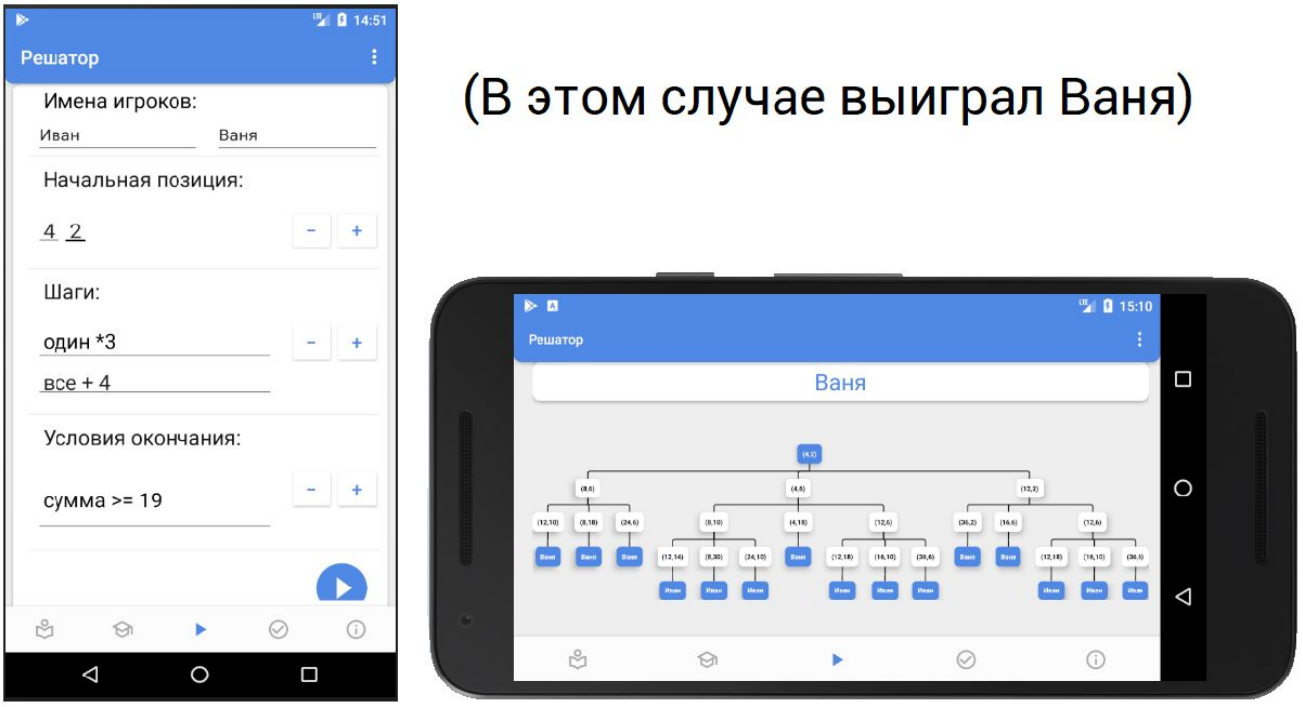


рис. 2.2.2 - Решатор 26го в действии

Решатор 3го номера

Одним из новейших решаторов является решатор 3го номера. 3й номер в егэ представляет собой решение задачи на нахождение кратчайшего пути или же просто дороги в графе. Граф может быть представлен рисунком или матрицей, может быть дана и матрица, и граф, где в матрице одни названия пунктов, а в графе другие и тогда уже начинает работать алгоритм по сопоставлению пунктов матрицы с названиями в графе. Человек это делает легко, а компьютер же перебирает перестановки таблицы чтоб найти ту таблицу, где строки и столбцы стоят так чтоб названия строк и столбцов были такие: А, Б, В…. Все нужные перестановки создаются алгоритмом Нарайаны, внутрь которого добавлена проверка на эквивалентность путей в таблицах. (Приложение № 6). Кратчайший же путь вычисляется по алгоритму Дейкстры (Приложении № 7).

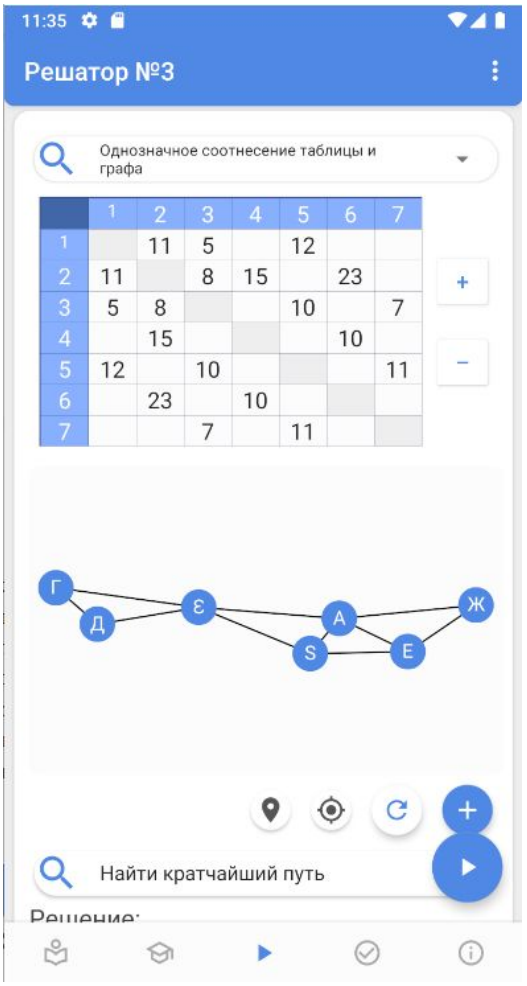


рис. 2.2.3 - Ввод условия

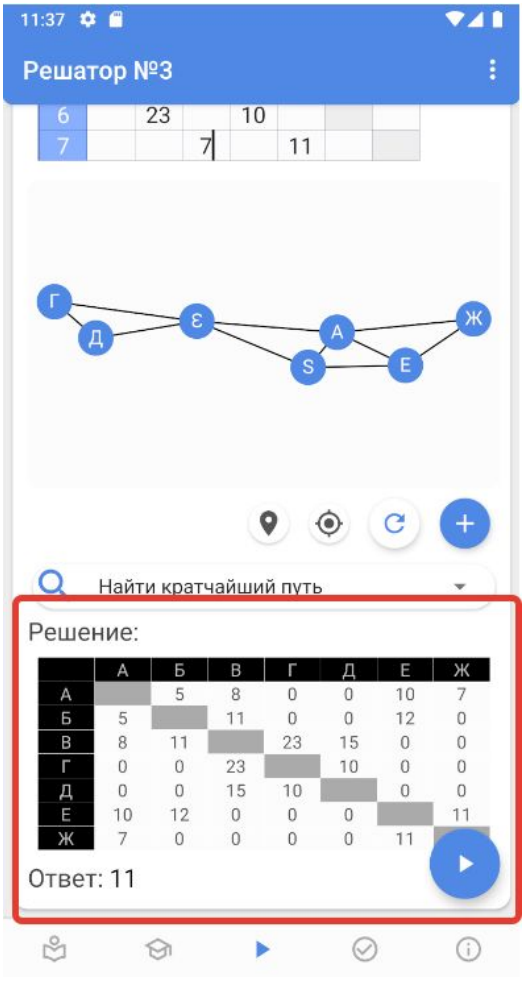
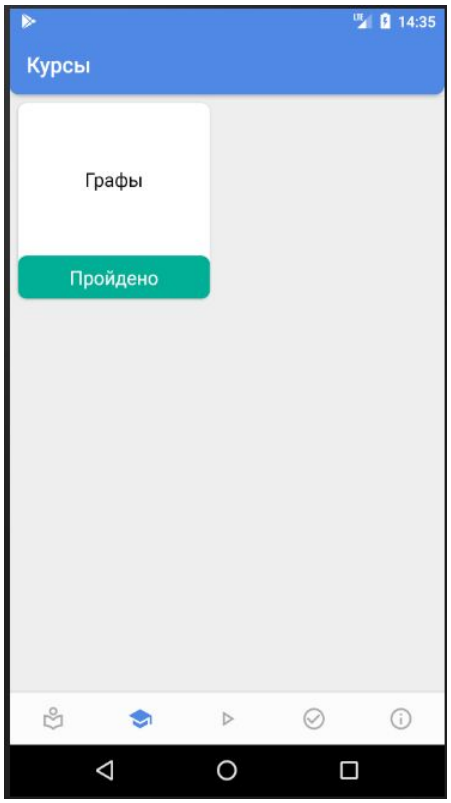


рис 2.2.4 - Решение и ответ

Курсы

При разработке курсов был создан отдельный Fragment, который должен был облегчить работу с разметкой. Делать каждую страничку курса было бы долго и муторно, т.е. неэффективно. Этот Fragment был назван FactoryEducationFragment (см. Приложение № 3), он представляет из себя класс, который может удобно заполнять страничку из курса заданиями, текстом, заголовками и картинками (в зависимости от будущих потребностей в класс могут быть добавлены дополнительные методы)(см. Приложение № 2).



Страница выбора курса, рис. 2.2.5

Курс “Графы” 1 стр, рис. 2.2.6

Курс “Графы” 2 стр, рис. 2.2.7

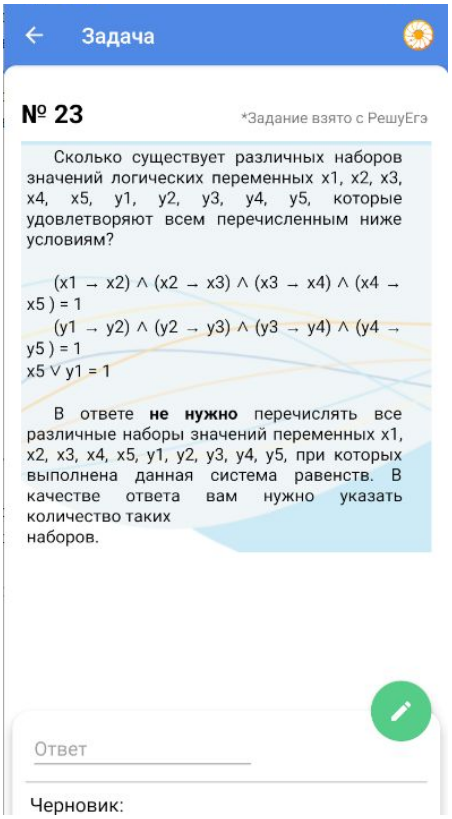


рис. 2.2.8 - Лист задачника

Устроен задачник достаточно просто: на экран выводится номер задачи, условие и поле ввода для ответа пользователя. Таким образом, ученик может ввести свой ответ и проверить его, нажав на специальную кнопку. (рис. 4.6)

Снизу используется выдвигающийся экран см. Источник № 7 для того чтоб юзер мог в начале прочесть задание, затем, нажав кнопку получить поле ввода ответа, если нужен черновик, то можно выдвинуть и его.

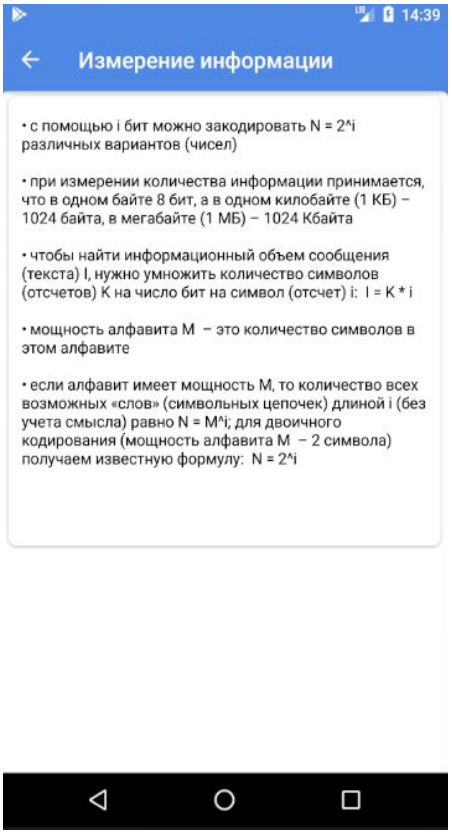


рис. 2.2.9 - Лист справочника

В приложении присутствует ещё и справочник, который предоставляет краткую информацию о какой-либо теме из школьного курса информатики (рис. 4.7). Класс, который отображает номера см. Приложение № 4

Рис 2.2.10 - Калькулятор

В процессе создания калькулятора мы осознали, что не надо изобретать велосипед и взяли уже готовую библиотеку для обработки математических выражений из строки “MXparser”, позже эта же библиотека была использована в решаторе 26 номера и планируется её начать использовать в задании № 13. (рис. 4.8)

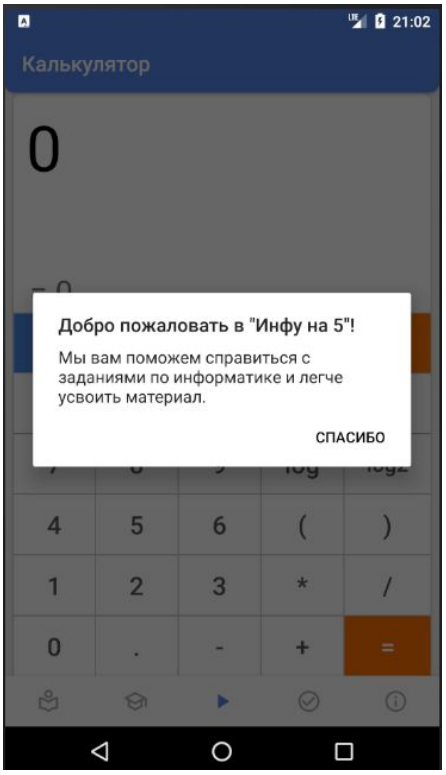


Рис. 2.2.11 - Приветствие

При первом запуске нашего приложения пользователь попадает на приветственный экран (Рис. 4.9)

3. Итоги разработки

В нашем продукте можно выделить следующий ряд положительных отличительных черт: наглядная форма вывода решения и ответа, широкий выбор тем и типов заданий для тренировки. Кроме того, несомненным плюсом является то, что решатор с введенными вами значениями в дано, калькулятор с логическими функциями, объяснение тем, тестирующая система, как по теории, так и по практике и справочник - все это собрано в одном приложении, к тому же, оно адаптировано **под мобильное устройство**, имеет удобный и понятный интерфейс, регистрации и выхода в интернет не требуется. Все эти плюсы далеко не всегда встречаются у существующих аналогов.

Выводы

По результатам проделанной работы можно отметить, что мы рассмотрели программирование алгоритмов на языке Python, разработку мобильных приложений под Android в среде Android Studio; изучили алгоритмы решения задач из ЕГЭ по информатике. Нам удалось собрать полезные материалы, объяснение тем, тренировочные тесты, решаторы, выводящие не только ответ к вашей задаче, но и пошаговое решение в простом и доступном любому пользователю приложении на Android. Следует уточнить, что в приложении реализованы решаторы только для нескольких заданий. Решаторы для остальных заданий мы планируем разработать в дальнейшем.

Данное приложение уже прошло тестирование на наших одноклассниках и учениках параллельных классов и получило одобрение от первых пользователей. К очному этапу планируется продолжить разработку приложения. В процессе работы над проектом появились новые перспективы и возможности. Нам предстоит разработать еще несколько решаторов, расширить базу курсов и тематических тестов, дополнить справочные материалы, улучшить взаимодействие пользователя с приложением. Его смогут активно использовать все те, кому нужно сдавать ЕГЭ по информатике или просто подготовиться по этому предмету. Можно с уверенностью утверждать, что поставленные задачи выполнены, цель достигнута.

Список используемых источников

1. Руководство по дизайну и коду: рекомендации по разработки приложений, URL: https://material.io/
2. Освой программирование играючи: изучаем Android, Александр Климов, 2019, URL: http://developer.alexanderklimov.ru/android/
3. Start Android - учебник по Android для начинающих и продвинутых, URL: https://startandroid.ru/
4. mXparser - Math Expression Evaluator / Parser: библиотека вычисления математических выражений, URL: http://mathparser.org/
5. ФОКСФОРД: онлайн подготовка к ЕГЭ, «ЦОО Нетология-групп» 2009–2020, URL: https://foxford.ru/
6. СДАМ ГИА: РЕШУ ЕГЭ, Образовательный портал для подготовки к экзаменам, URL: https://inf-ege.sdamgia.ru/
7. Android: выдвигающийся экран снизу: перевод статьи Emrullah Luleci, URL: https://habr.com/ru/post/309200/
8. Документация для Android-разработчиков, Google, URL: https://developer.android.com/docs

Приложения

Приложение 1 - Код 13го решатора

```
from math import *

formulas = {
        "i":  {"N":  ["ceil(log2(N))",  lambda  given_val:
ceil(log2(given_val['N']))],
           "KI": ["K / I", lambda given_val: given_val['K'] /
given_val['I']]},
      "I": {"Ki": ["K * i", lambda given_val: given_val['K'] *
given_val['i']]},
      "K": {"Ii": ["I / i", lambda given_val: given_val['I'] /
given_val['i']]},
         "N":  {"i":  ["2  **  i",  lambda  given_val:  2  **
given_val['i']]}}

OPERATORS = {'+': (1, lambda x, y: x + y), '-': (1, lambda
x, y: x - y),
             '*': (2, lambda x, y: x * y), '/': (2, lambda
x, y: x / y)}

def calc(sort):
    tmp = []
    for i in sort:
        if i in OPERATORS:
            y = tmp.pop()
            x = tmp.pop()
            tmp.append(OPERATORS[i][1](x, y))
        else:
            tmp.append(i)
    return tmp[0]

def main_func(given_val, find):
    given = {i for i in given_val.keys()}
    __find = formulas[find]
    for i in __find.keys():
        if (set(list(i)) & given) == set(list(i)):
            hv = __find[i]
            print(find, '=', hv[0])
```

```
                for v in given:
                    if v in i:
                                            hv[0]  =  hv[0].replace(v,
str(given_val[v]))
                print(find, '=', hv[0])
                print(find, '=', hv[1](given_val))
                return given_val, hv[1](given_val)
            for j in i:
                if j not in given:
                    helper_value = main_func(given_val, j)[-1]
                    given_val[j] = helper_value
                    given = {i for i in given_val.keys()}
            if (set(list(i)) & given) == set(list(i)):
                hv = __find[i]
                print(find, '=', hv[0])

                for v in given:
                    if v in i:
                                            hv[0]  =  hv[0].replace(v,
str(given_val[v]))
                print(find, '=', hv[0])
                print(find, '=', hv[1](given_val))

                return given_val, hv[1](given_val)

    def main():
        # given_val = {"N": 256}
        # find = "i"
        # print("Дано: \nN = 256\ni - ?")
        given_val = {"K": "10000", "N": '256'}
        find = "I"
        print("Дано: ")
            print('\n'.join([i+'  =  '+given_val[i]  for  i  in
given_val.keys()]))
        print("--------------\n" +
              find+" - (?)\n\n"
              "Решение:")

        for i in given_val.keys():
            given_val[i] = eval(given_val[i])

        given_val, a = main_func(given_val, find)
        print("Ответ:", a)

    if __name__ == "__main__":
        main()
```

Приложение 2 - Создание курсов

```
private void upload_courses() {

   mode_fragments.put("course_graphs", new
ArrayList<>());

   actionBarNames.put("course_graphs", new
ArrayList<>());


   mode_fragments.get("course_graphs").add(new
FactoryEducationFragment()

            .newTitle(R.string.graphs_1)

            .newText(R.string.graphs_2)

            .newImage(R.mipmap.graph_visualisation));

   actionBarNames.get("course_graphs").add("Графы");

   mode_fragments.get("course_graphs").add(new
FactoryEducationFragment()

            .newText(R.string.graphs_task1)

            .newChoice(R.array.choice_graphs_task1));

   actionBarNames.get("course_graphs").add("Графы");

}
```

Приложение 3 - Класс создания страниц курсов и справочника

```
public class FactoryEducationFragment extends Fragment {
   List<List> task_list = new ArrayList<>();
   final String[] TASKS = {"image", "text", "title", "choice",
"enter", "text_list"};
   String answer = null;
   String user_answer = "";
   boolean enable_continue = true;

   EditText enter;

   View root_view;

   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                            Bundle savedInstanceState) {
       root_view =
inflater.inflate(R.layout.fragment_factory_education, container,
false);
       evaluateTasks();
       if (enable_continue)

root_view.findViewById(R.id.button5).setVisibility(View.VISIBLE);
       else

root_view.findViewById(R.id.button5).setVisibility(View.INVISIBLE)
;
       return root_view;
   }

   public FactoryEducationFragment newImage(int image_id) {
return newTask("image", image_id);}
   public FactoryEducationFragment newText(int text_id) { return
newTask("text", text_id);}
   public FactoryEducationFragment newTitle(int text_id) { return
newTask("title", text_id);}
   public FactoryEducationFragment newChoice(int choice_array_id)
{ return newTask("choice", choice_array_id);}
   public FactoryEducationFragment newEnter(int answer_id) {
return newTask("enter", answer_id);}
   public FactoryEducationFragment newTextList(int text_array_id)
{  return newTask("text_list", text_array_id);}
```

```
    public FactoryEducationFragment enableContinue(boolean a)
{...}

    public FactoryEducationFragment newTask(String name, int
obj_id) {...}
    private void evaluateTasks() {...}

    private void addImage(List task) {...}
    private void addText(List task) {
        TextView textView = new TextView(getContext());
        textView.setText((int)task.get(1));

        LinearLayout.LayoutParams imageViewLayoutParams = new
LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
LinearLayout.LayoutParams.WRAP_CONTENT);
        textView.setLayoutParams(imageViewLayoutParams);

        TypedArray ta =
getContext().obtainStyledAttributes(R.style.EducationText,
R.styleable.StandartText);

textView.setTextColor(ta.getColor(R.styleable.StandartText_android
_textColor, -1));
        ta.recycle();

        ((LinearLayout)
root_view.findViewById(R.id.factory_base)).addView(textView);
    }
    private void addTitle(List task) {...}
    private void addChoice(List task) {...}
    private void addEnter(List task) {...}
    private void addTextList(List task) {...}

    public Boolean onClickNext() {
        return (answer == null || answer.equals(user_answer));
    }
    public void save() {
        try {
            Log.i("----------------------", "save: " +
enter.getText());
            user_answer = "" + enter.getText();
        } catch (NullPointerException e) {
            if (!user_answer.equals(""))
                Log.i("----------------------", "save: " +
user_answer);
        }
    }
}
```

Приложение 4 - Класс представления номеров

```
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.graphics.Color;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.webkit.WebView;
import android.widget.EditText;
import android.widget.TextView;

import java.io.IOException;
import java.util.Random;


public class TaskViewFragment extends Fragment {
    View rootView;
    int curr;
    int num;
    String[] tasks = new String[]{"We have't tasks for this
number yet, *answer is \"\"*", ""};

    private DatabaseHelper mDBHelper;
    private SQLiteDatabase mDb;


    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable
ViewGroup container,
                             @Nullable Bundle
savedInstanceState) {
        rootView =
```

```
                inflater.inflate(R.layout.task_view, container,
false);
        startTaskSet();
        return rootView;
    }

    public void setTask(int i) {
        curr = i;
    }
    public void startTaskSet() {
        mDBHelper = new DatabaseHelper(rootView.getContext());

        try {
            mDBHelper.updateDataBase();
        } catch (IOException mIOException) {
            throw new Error("UnableToUpdateDatabase");
        }

        try {
            mDb = mDBHelper.getWritableDatabase();
        } catch (SQLException mSQLException) {
            throw mSQLException;
        }

        Cursor cursor;
        Random random = new Random();

        cursor = mDb.rawQuery("SELECT task, answer FROM tasks
WHERE number="+curr, null);
        cursor.moveToFirst();

        num = random.nextInt(cursor.getCount() / 2);
        cursor.moveToPosition(num*2);
        tasks = new String[]{cursor.getString(0),
cursor.getString(1)};
        cursor.close();

        ((TextView)
rootView.findViewById(R.id.task_theme)).setText("№ " + curr);
        WebView task_text =
rootView.findViewById(R.id.task_text);
```

```
        task_text.loadDataWithBaseURL(null,
tasks[0].replaceFirst("\"width:650px\"", "\"width:90%\""),
                "text/html", "en_US", null);
        task_text.setBackgroundColor(Color.TRANSPARENT);
        task_text.getSettings().setBuiltInZoomControls(true);

        (rootView.findViewById(R.id.task_answer))
                .setMinimumWidth(  (int)((EditText)
rootView.findViewById(R.id.task_answer))
                        .getTextSize() * tasks[1].length());
    }

    public void onClickCheckAnswer(View view) {
        if (tasks[1].equals(((EditText)
rootView.findViewById(R.id.task_answer)).getText().toString())
) {

rootView.findViewById(R.id.task_check).getBackground().setTint
(getResources().getColor(R.color.colorAppTrue));
            ((EditText)
rootView.findViewById(R.id.task_answer)).setText("");
            //setTask(curr);
            startTaskSet();
        }
        else

rootView.findViewById(R.id.task_check).getBackground().setTint
(getResources().getColor(R.color.colorAppFalse));
    }
    }
    public void setRootView(View rootView) {
        this.rootView = rootView;
    }
}
```

Приложение 5 - Фрагмент решатора №26

просчитывается, кто из игроков победит

```
private Boolean tree_logic_and(Map<String, Object> tree) {
    List<Boolean> temp = new ArrayList<>();
    for (String i: tree.keySet()) {
        try {
            temp.add(tree_logic_or((Map<String, Object>) tree.get(i)));
        } catch (ClassCastException cce) {
            temp.add(tree_logic_or((Integer) tree.get(i)));
        }
    }
    return all(temp);
}
private Boolean tree_logic_and(Integer arg) { return arg == 1; }

private Boolean tree_logic_or(Map<String, Object> tree) {
    List<Boolean> temp = new ArrayList<>();
    for (String i: tree.keySet()) {
        try {
            temp.add(tree_logic_and((Map<String, Object>) tree.get(i)));
        } catch (ClassCastException cce) {
            temp.add(tree_logic_and((Integer) tree.get(i)));
        }

    }
    return any(temp);
}
private Boolean tree_logic_or(Integer arg) { return arg == 1; }
```

Приложение 6 - Применение алгоритма Нарайаны

```
   class Resh3t1 {
      private fun swap(a: ArrayList<ArrayList<Int>>, x: Int,
y: Int): ArrayList<ArrayList<Int>> {
          val b: ArrayList<ArrayList<Int>> = ArrayList()
          for (i in 0 until a.size) {
              b.add(ArrayList())
              for (j in 0 until a.size) {
                  b.last().add(a
                          [if (i == x) y else if (i == y) x
else i]
                          [if (j == x) y else if (j == y) x
else j]
                  )
              }
          }
          return b
      }

      private fun check(a: ArrayList<ArrayList<Int>>, b:
ArrayList<ArrayList<Int>>): Boolean {
          for (i in 0 until a.size) {
              for (j in 0 until a.size) {
                  if (!((a[i][j] - b[i][j] == 0) || (a[i][j] >
0 && b[i][j] > 0))) {
                      return false
                  }
              }
          }
          return true
      }

      fun run(a: ArrayList<ArrayList<Int>>, bb:
ArrayList<ArrayList<Int>>): ArrayList<Int> {

          val x = ArrayList<Int>()
          var b = bb

          for (i in 0 until a.size) {
              x.add(i)
          }

          while (true) {
              if (check(a, b)) break;
```

```
                var j = 0
                for (i in 0 until x.size - 1) {
                    if (x[i] < x[i + 1]) {
                        j = i
                    }
                }
                var l = 0
                for (i in j + 1 until x.size) {
                    if (x[i] > x[j]) {
                        l = i
                    }
                }
                if (j == 0 && l == 0) break;
                x[j] = x[l].also { x[l] = x[j] }
                b = swap(b, j, l)
                for (r in 0 until (b.size - j + 1) / 2) {
                    x[r + j + 1] = x[b.size - 1 - r].also {
x[b.size - 1 - r] = x[r + j + 1] }
                    b = swap(b, r + j + 1, b.size - 1 - r);
                }
            }
            if (!check(a, b)) throw Exception("Что-то не
сходится")

            return x
        }

    }
```

Приложение 7 - Применение алгоритма Дейкстры

```
public class Deikstra {
    private List<Rib> ribs;
    private int peaksCount, ribsCount;
    private int taskFrom, taskTo;

    private static class Rib {
        int from;
        int  to;
        int weight;

        private Rib(int from, int to, int weight) {
            this.from = from;
            this.to = to;
            this.weight = weight;
        }
    }

        public Deikstra initialize(int peaksCount, int
ribsCount, ArrayList<ArrayList<Integer>> ribs, int taskFrom,
int taskTo) {
        this.peaksCount = peaksCount;
        this.ribsCount = ribsCount;

        this.ribs = new LinkedList<>();
        for (int i = 0; i < ribsCount; i++) {
            for (int j = 0; j < ribsCount; j++) {
                try {
                    if (ribs.get(i).get(j) != 0)
                                this.ribs.add(new Rib(i, j,
ribs.get(i).get(j)));
                } catch (Exception e) {}
            }
        }
        this.taskFrom = taskFrom;
        this.taskTo = taskTo;

        return this;
    }

    public int main() throws Exception {
```

```
            if(ribsCount != 0) {
                List<Integer> usedPeaks = new LinkedList<>();
                usedPeaks.add(taskFrom);
                int currentPeak = taskFrom;
                int result = 0;

                  //костыль на случай, если из заданного старта
есть прямой
                   //путь в заданный финиш, т.к. итоговый путь
может оказаться
                //больше него
                int directLineWeight = 0;
                for(Rib r : ribs) {
                    if(r.from == taskFrom && r.to == taskTo) {
                        directLineWeight = r.weight;
                    }
                }

                 while(currentPeak != taskTo && usedPeaks.size()
!= peaksCount) {
                                List<Rib> currentRibsList = new
LinkedList<>();

                               //составляем список возможных для
использования ребер
                    for(Rib r : ribs) {
                                    if (r.from == currentPeak &&
!usedPeaks.contains(r.to)) {
                                currentRibsList.add(r);
                        }
                    }

                         //если не найдено ни одного подходящего
ребра
                    if(currentRibsList.size() == 0) {
                        return directLineWeight;
                    }
                    ribs.removeAll(currentRibsList);

                     //ищем в этом списке ребро, с использованием
которого
```

```
                    //удастся достичь наименьшего НА ДАННЫЙ
МОМЕНТ веса
                int minWeight = Integer.MAX_VALUE;
                Rib ribToUse = null;
                for(Rib r : currentRibsList) {
                    if(r.weight + result < minWeight) {
                        minWeight = r.weight;
                        ribToUse = r;
                    }
                }
                currentRibsList.remove(ribToUse);
                ribs.addAll(currentRibsList);
                            //меняем текущие значения для
переменных-показателей статуса
                currentPeak = ribToUse.to;
                usedPeaks.add(ribToUse.to);
                result += ribToUse.weight;
                     //выводит список вершин, обойденных на
данный момент
            /*for(int r : usedPeaks) {
               System.out.print(r);
            }
            System.out.println();*/
            }

            //если нужного пути не существует, выводится -1
             if(currentPeak != taskTo &&  usedPeaks.size() ==
peaksCount){
                return directLineWeight;
            } else {
                    if(directLineWeight != 0 && result >
directLineWeight) {
                    return directLineWeight;
                } else {
                    return result;
                }
            }
        } else {
            throw new Exception("All wrong");
        }
    }
}
```