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 №11

по дисциплине «Разработка мобильных приложений»

Выполнили:
Студенты группы ИКБО-12-22 Солобай А.П.

Проверил:
Преподаватель Степанов П.В.

Москва 2024 г.

Выполнение практической работы

Ссылка на GitHub со всеми работами: ***https://github.com/Eckorezze/Mobil***

1. Реализовать пример с использованием потоков

Листинг 1 – код в файле MainActivity

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    TextView textView = findViewById(R.id.textView);
    Button button = findViewById(R.id.button);
    button.setOnClickListener(v -> {
        Runnable runnable = () -> {
            Calendar c = Calendar.getInstance();
            int hours = c.get(Calendar.HOUR_OF_DAY);
            int minutes = c.get(Calendar.MINUTE);
            int seconds = c.get(Calendar.SECOND);
            String time = hours + ":" + minutes + ":" + seconds;
            textView.post(() -> textView.setText(time));
        };
        Thread thread = new Thread(runnable);
        thread.start();
    });
}
```

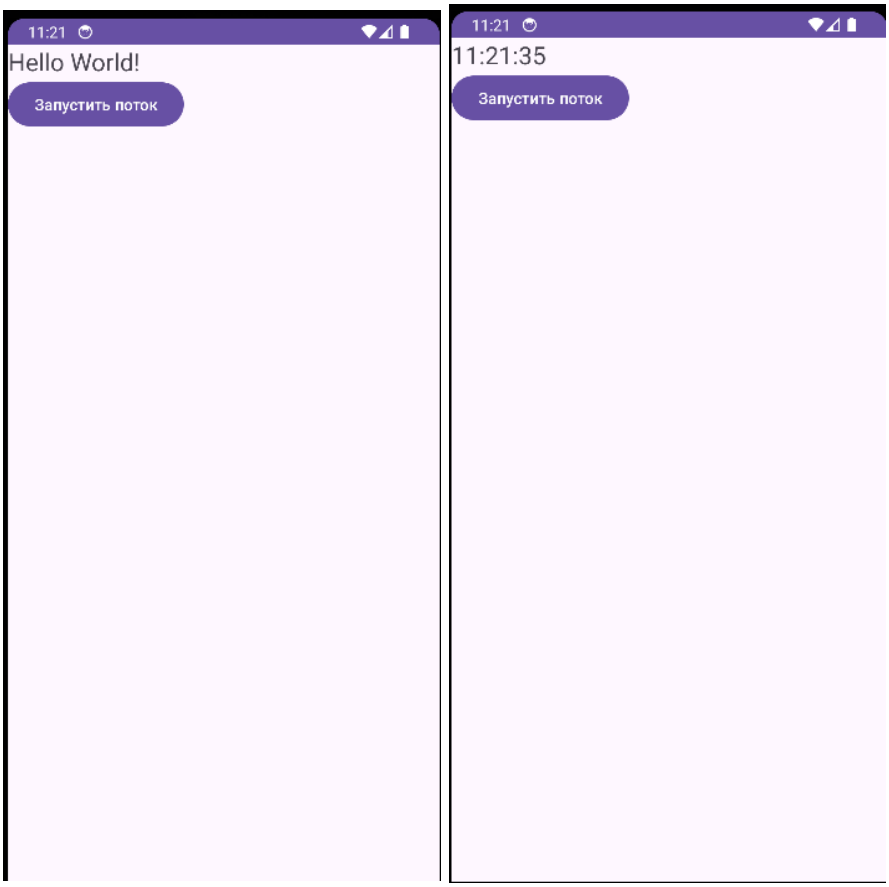

Рисунок 1,2 – тестирование кода примера

2. Реализовать потоки, фрагменты и ViewModel

Листинг 2 – код в файле MyViewModel

```
public class MyViewModel extends ViewModel {
    private final MutableLiveData<Boolean> isStarted = new
            MutableLiveData<Boolean>(false);
    private MutableLiveData<Integer> value;
    public LiveData<Integer> getValue() {
        if (value == null) {
            value = new MutableLiveData<Integer>(0);
        }
        return value;
    }
    public void execute(){
        if(Boolean.FALSE.equals(isStarted.getValue())){
            isStarted.postValue(true);
            Runnable runnable = new Runnable() {
                @Override
                public void run() {
                    for(int i = value.getValue(); i <= 100; i++){
                        try {
                            value.postValue(i);
                            Thread.sleep(400);
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                }
            };
            Thread thread = new Thread(runnable);
            thread.start();
        }
    }
}
```

Листинг 3 – код в файле MainActivity для потоков, фрагментов и ViewModel

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ProgressBar indicatorBar = findViewById(R.id.indicator);
        TextView statusView = findViewById(R.id.statusView);
        Button btnFetch = findViewById(R.id.progressBtn);
        MyViewModel model = new
                ViewModelProvider(this).get(MyViewModel.class);
        model.getValue().observe(this, value -> {
            indicatorBar.setProgress(value);
            statusView.setText("Статус: " + value);
        });
        btnFetch.setOnClickListener(v -> model.execute());
    }
}
```

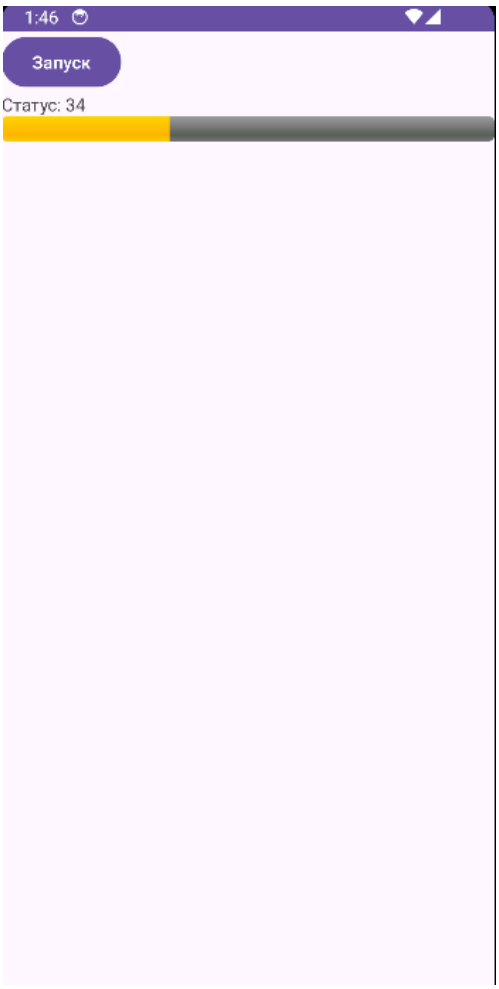

Рисунок 3 – тестирование кода примера

3. Изучить порядок и реализовать применение класса AsyncTask

Листинг 4 – код в файле MainActivity

```
int[] integers=null;
int clicks = 0;
ProgressBar indicatorBar;
TextView statusView;
TextView clicksView;
Button progressBtn;
Button clicksBtn;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    integers = new int[100];
    for(int i=0;i<100;i++) {
        integers[i] = i + 1;
    }
    indicatorBar = (ProgressBar) findViewById(R.id.indicator);
    statusView = findViewById(R.id.statusView);
    progressBtn = findViewById(R.id.progressBtn);
    progressBtn.setOnClickListener(v -> new
ProgressTask().execute());
    clicksView = findViewById(R.id.clicksView);
```

```
    clicksBtn = findViewById(R.id.clicksBtn);
    clicksBtn.setOnClickListener(v -> {
        clicks++;
        clicksView.setText("Clicks: " + clicks);
    });

}

class ProgressTask extends AsyncTask<Void, Integer, Void> {
    @Override
    protected Void doInBackground(Void... unused) {
        for (int i = 0; i<integers.length;i++) {
            publishProgress(i);
            SystemClock.sleep(400);
        }
        return(null);
    }
    @Override
    protected void onProgressUpdate(Integer... items) {
        indicatorBar.setProgress(items[0]+1);
        statusView.setText("Статус: " + String.valueOf(items[0]+1));
    }
    @Override
    protected void onPostExecute(Void unused) {
        Toast.makeText(getApplicationContext(), "Задача завершена",
                       Toast.LENGTH_SHORT)
                .show();
    }
}
```

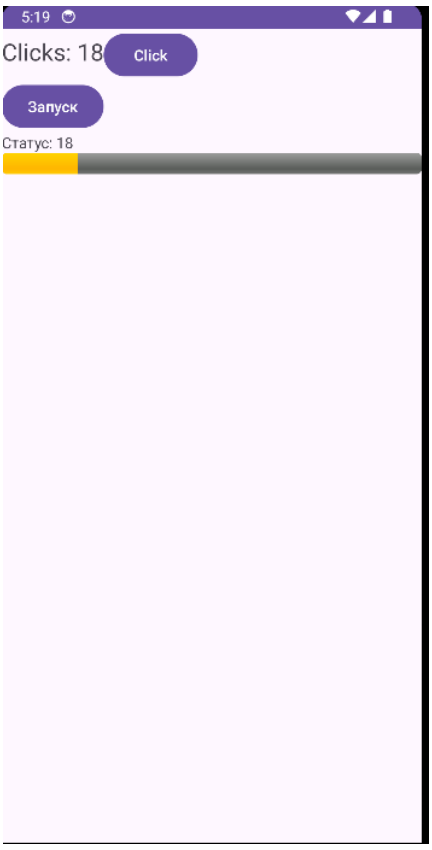

Рисунок 4 – тестирование кода примера

4. Реализовать применение AsyncTask с фрагментом

Листинг 5 – код в файле ProgressFragment

```
public class ProgressFragment extends Fragment {
    int[] integers=null;
    ProgressBar indicatorBar;
    TextView statusView;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setRetainInstance(true);
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_progress,
container, false);
        integers = new int[100];
        for(int i=0;i<100;i++) {
            integers[i] = i + 1;
        }
        indicatorBar = (ProgressBar)
view.findViewById(R.id.indicator);
        statusView = (TextView) view.findViewById(R.id.statusView);
        Button btnFetch =
(Button)view.findViewById(R.id.progressBtn);
        btnFetch.setOnClickListener(v -> new
ProgressTask().execute());
        return view;
    }
    class ProgressTask extends AsyncTask<Void, Integer, Void> {
        @Override
        protected Void doInBackground(Void... unused) {
            for (int i = 0; i<integers.length;i++) {
                publishProgress(i);
                SystemClock.sleep(400);
            }
            return null;
        }
        @Override
        protected void onProgressUpdate(Integer... items) {
            indicatorBar.setProgress(items[0]+1);
            statusView.setText("Статус: " +
String.valueOf(items[0]+1));
        }
        @Override
        protected void onPostExecute(Void unused) {
            Toast.makeText(getActivity(), "Задача завершена",
                           Toast.LENGTH_SHORT)
                    .show();
        }
    }
}
```

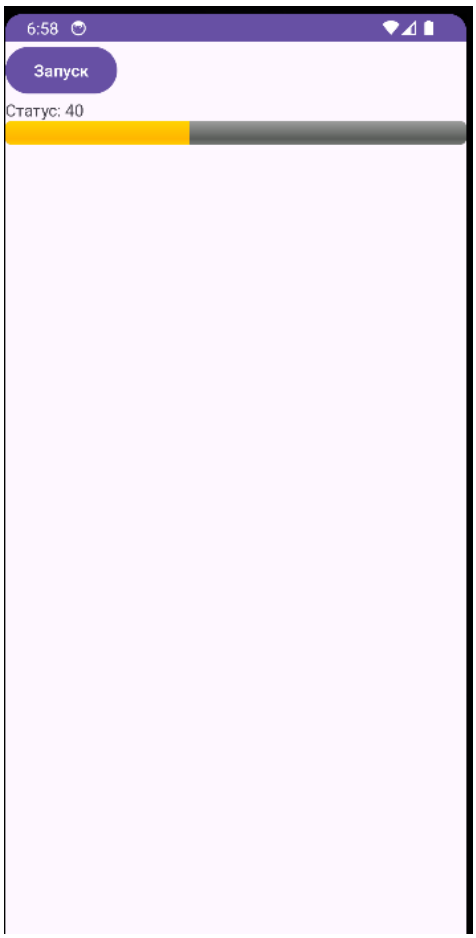

Рисунок 5 – тестирование кода примера

5. Изучить работу с сетью и классом WebView. Реализовать пример.

Листинг 6 – код в файле MainActivity

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

    WebView browser=findViewById(R.id.webBrowser);
    WebSettings webSettings = browser.getSettings();
    webSettings.setJavaScriptEnabled(true);
    browser.loadUrl("https://vk.com/lettons");

}
```

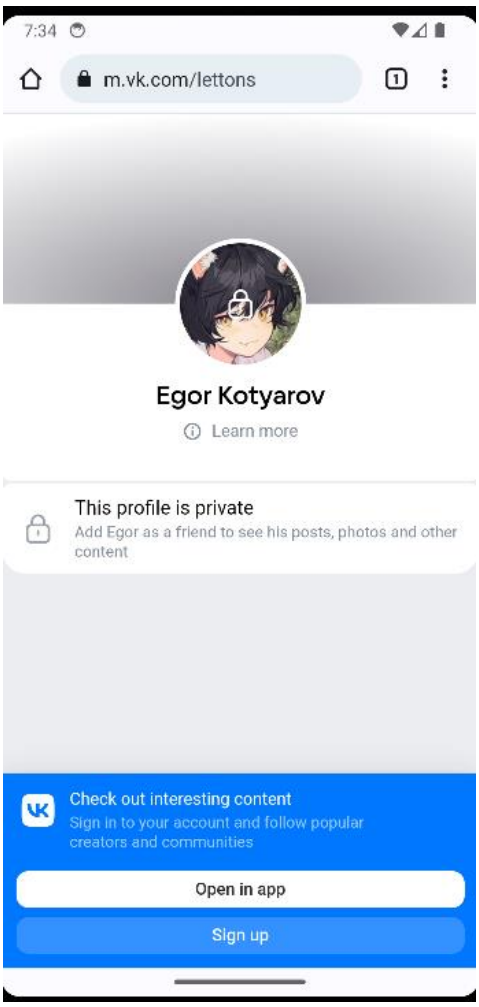

Рисунок 6 – тестирование кода примера

6. Реализовать пример загрузки данных с применением класса HttpsURLConnection

Листинг 7 – код в файле MainActivity

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

    TextView contentView = findViewById(R.id.content);
    WebView webView = findViewById(R.id.webView);
    webView.getSettings().setJavaScriptEnabled(true);
    Button btnFetch = findViewById(R.id.downloadBtn);
    btnFetch.setOnClickListener(v -> {
        contentView.setText("Загрузка...");
        new Thread(() -> {
            try{
                String content =
getContent("https://vk.com/lettons");
                webView.post(() -> {


webView.loadDataWithBaseURL("https://vk.com/lettons",content,
```

```
                                "text/html", "UTF-8",
"https://vk.com/lettons");
                        Toast.makeText(getApplicationContext(), "Данные
загружены", Toast.LENGTH_SHORT).show();
                    });
                    contentView.post(() -> contentView.setText(content));
                }
                catch (IOException ex){
                    contentView.post(() -> {
                        contentView.setText("Ошибка: " +
ex.getMessage());
                        Toast.makeText(getApplicationContext(), "Ошибка",
                                Toast.LENGTH_SHORT).show();
                    });
                }
            }).start();
        });
    }
    private String getContent(String path) throws IOException {
        BufferedReader reader=null;
        InputStream stream = null;
        HttpsURLConnection connection = null;
        try {
            URL url=new URL(path);
            connection =(HttpsURLConnection)url.openConnection();
            connection.setRequestMethod("GET");
            connection.setReadTimeout(10000);
            connection.connect();
            stream = connection.getInputStream();
            reader= new BufferedReader(new InputStreamReader(stream));
            StringBuilder buf=new StringBuilder();
            String line;
            while ((line=reader.readLine()) != null) {
                buf.append(line).append("\n");
            }
            return(buf.toString());
        }
        finally {
            if (reader != null) {
                reader.close();
            }
            if (stream != null) {
                stream.close();
            }
            if (connection != null) {
                connection.disconnect();
            }
        }
    }
```

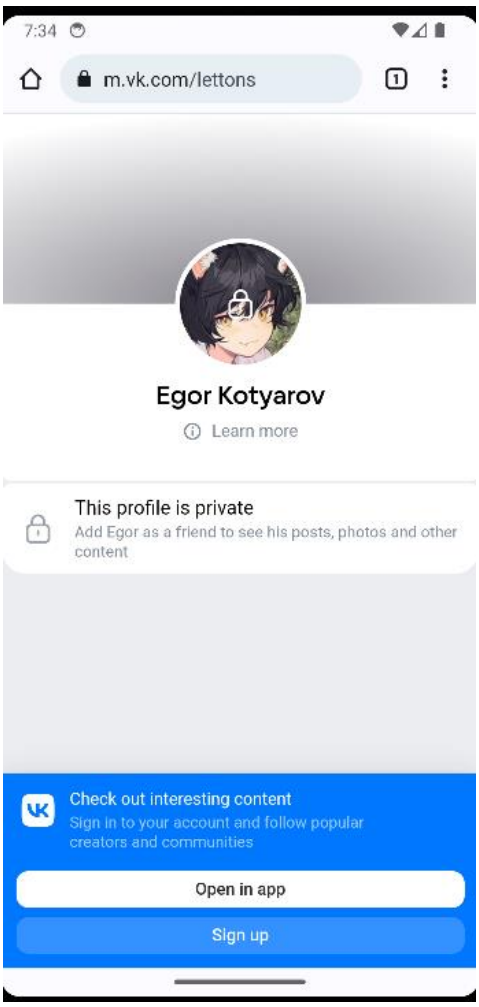

Рисунок 7 – тестирование кода примера

Вывод

В ходе работы мне удалось научиться реализовывать многопоточность и асинхронность. Были освоены навыки работы с потоками, фрагментами, ViewModel и сетью.