Android-App

Aplicaţia se conectează la un api ce conţine datele meteorologice în format JSON, care sunt apoi afișate sub formă de grafice pe ecran. Aplicația are la baza codul MainActivity care prelucrează datele și afișează datele.

Codul incepe astfel:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    getSupportActionBar().hide();
    setContentView(R.layout.activity_main);
    new JsonTask().execute("https://awu4j6hku3.execute-api.eu-central-1.amazonaws.com/dev/weather/hist?days=1&group=12");

}
```

În metodă “onCreate”, se ascunde bară de acțiuni utilizând ‘getSupportActionBar().hide()’ și se resetează layout-ul activității cu ‘setContentView(R.layout.activity_main)’.

```
private class JsonTask extends AsyncTask<String, String, String> {

    protected void onPreExecute() {
        super.onPreExecute();

        pd = new ProgressDialog(MainActivity.this);
        pd.setMessage("Please wait");
        pd.setCancelable(false);
        pd.show();
    }

    protected String doInBackground(String... params) {

        HttpURLConnection connection = null;
        BufferedReader reader = null;

        try {
            URL url = new URL(params[0]);
            connection = (HttpURLConnection) url.openConnection();
            connection.connect();


            InputStream stream = connection.getInputStream();

            reader = new BufferedReader(new InputStreamReader(stream));

            StringBuffer buffer = new StringBuffer();
            String line = "";

            while ((line = reader.readLine()) != null) {
                buffer.append(line+"\n");
                Log.d("Response: ", "> " + line);   //here u ll get whole response...... :-)

            }
```

Clasa ‘JsonTask’ este o clasa internă care extinde clasa ‘AsyncTask’. Acesta este responsabilă pentru efectuarea unei operațiuni de rețea în firul de execuție de fundal. Metodă ‘doInBackground’ este suprascrisa pentru a efectua operația de rețea de a obține datele JSON de la URL-ul specificat.

```
protected void onPostExecute(String result) {
    super.onPostExecute(result);
    if (pd.isShowing()){
        pd.dismiss();
    }

    JSONArray jsonarray = null;
    try {
        jsonarray = new JSONArray(result);
    } catch (JSONException e) {
        throw new RuntimeException(e);
    }

    LineGraphSeries<DataPoint> seriesT = new LineGraphSeries<>(new DataPoint[] {});
    LineGraphSeries<DataPoint> seriesH = new LineGraphSeries<>(new DataPoint[] {});
    LineGraphSeries<DataPoint> seriesW = new LineGraphSeries<>(new DataPoint[] {});
    LineGraphSeries<DataPoint> seriesP = new LineGraphSeries<>(new DataPoint[] {});

    for (int i = 0; i < jsonarray.length(); i++) {
        JSONObject jsonobject = null;
        try {
            jsonobject = jsonarray.getJSONObject(i);
        } catch (JSONException e) {
            throw new RuntimeException(e);
        }
        try {
            String time = jsonobject.getString("time");
        } catch (JSONException e) {
            throw new RuntimeException(e);
        }
        String value = null;
```

Metodă ‘onPostExecute’ este apelată după finalizarea operației de rețea și primește datele JSON preluate.

```
            try {
                value = jsonobject.getString("value");
            } catch (JSONException e) {
                throw new RuntimeException(e);
            }
            String dataType = null;
            try {
                dataType = jsonobject.getString("dataType");
            } catch (JSONException e) {
                throw new RuntimeException(e);
            }

            if (dataType.equals("temp"))
                seriesT.appendData(new DataPoint(i/5.0,Double.parseDouble(value)),true,12);
            if (dataType.equals("humidity"))
                seriesH.appendData(new DataPoint(i/5.0,Double.parseDouble(value)),true,12);
            if (dataType.equals("wind"))
                seriesW.appendData(new DataPoint(i/5.0,Double.parseDouble(value)),true,12);
            if (dataType.equals("pressure"))
                seriesP.appendData(new DataPoint(i/5.0,Double.parseDouble(value)),true,12);
        }

        GraphView graphT = (GraphView) findViewById(R.id.graphT);
        graphT.setTitle("Temperature");
        graphT.setTitleTextSize(80);
        graphT.setTitleColor(Color.RED);
        seriesT.setColor(Color.RED);
        graphT.getGridLabelRenderer().setVerticalLabelsColor(Color.WHITE);
        graphT.getGridLabelRenderer().setHorizontalLabelsColor(Color.WHITE);
        graphT.getGridLabelRenderer().setGridColor(Color.WHITE);
        graphT.addSeries(seriesT);
```

Această procesează datele și afișează graficele de linii pentru temperatura, umiditate, viteză a vântului și presiune.