**Ministerul Educatiei Republicii Moldova**

**Universitatea Tehnica a Moldovei**

**Departamentul ISA**

**Raport**

La lucrearea de laborator № 3

Disciplina “Programarea aplicatiilor mobile”

**Tema:** Simple HTTP Client (Web Service Emulation)

A efectuat: st. gr. TI-153 Suruceanu V.

A controlat: lect. univ. C. Rusu

Chisinau 2017

**Scopul Lucrarii:** De realizat emularea programatica a unui serviciu web.

De realizat o aplicație ce va încărca de pe o resursă web (preferată, ex: https://news.yam.md/ro/rss ) fluxul RSS al acesteia.

**Sarcina:**

Serviciul web va fi emulat programatic utilizând metodele protocolului HTTP (GET; )  
a) posibilitate de adaugare 2 sau mai multe fluxuri RSS  
b) posibilitate de a naviga catre postarea din fluxul incarcat  
c) salvarea locala a fluxului cu păstrarea sa pînă utilizatorul nu o va distruge

**Mersul lucrarii:**

1. Am creat 1 activitatea care ruleaza ca MainActivity pentru actiunea de afisare a RSS feed-ului
2. Creat o metoda (FetchFeedTask) care foloseste tipul XmlPullParser care citeste si primeste feed-ul
3. Am creat doua baze de date care pastreaza lincurile si alta care pastreaza posturile pentru citire Offline.
4. Am creat 2 activitati care gestioneaza cu lincurile si feed-ul RSS

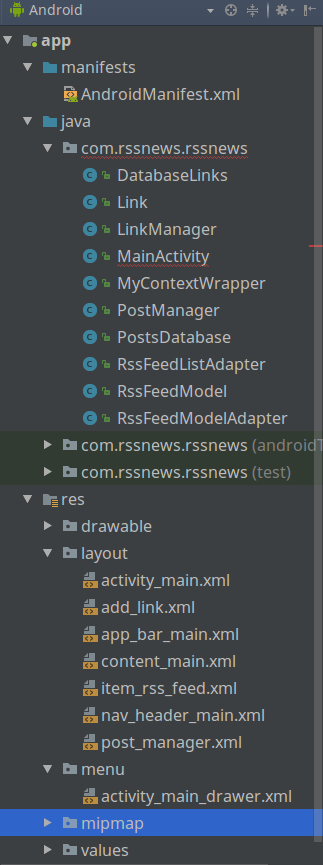
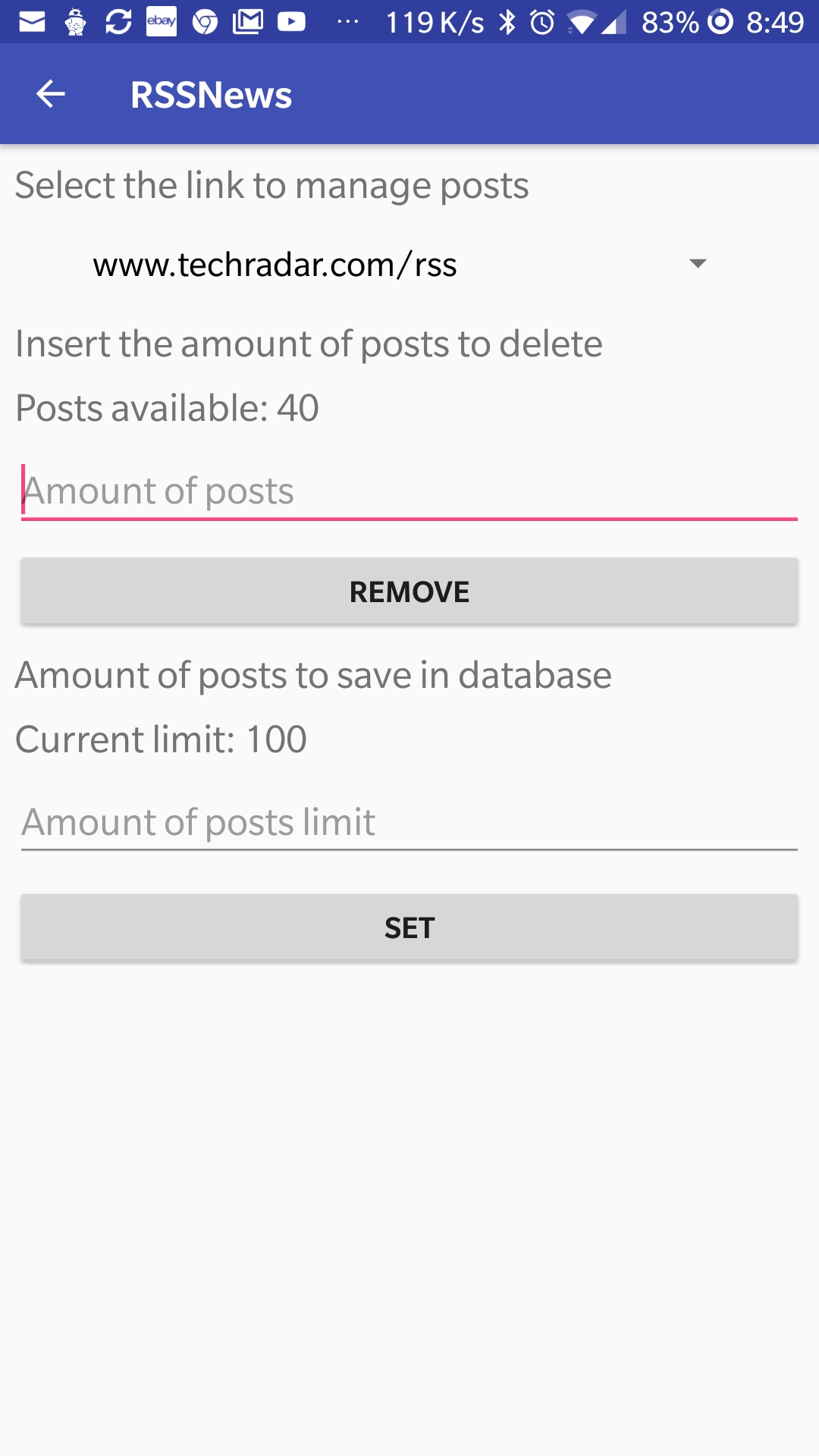
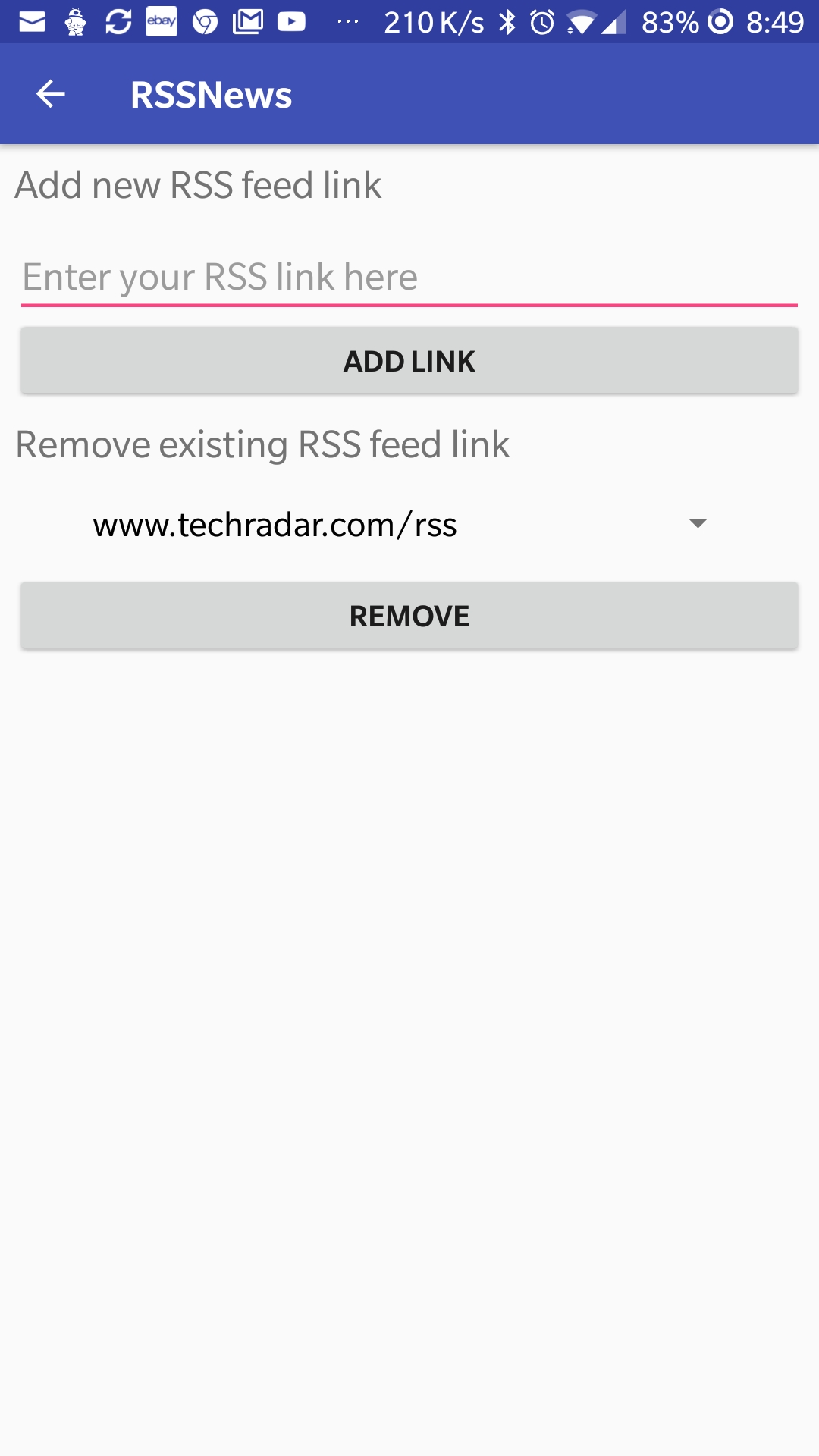
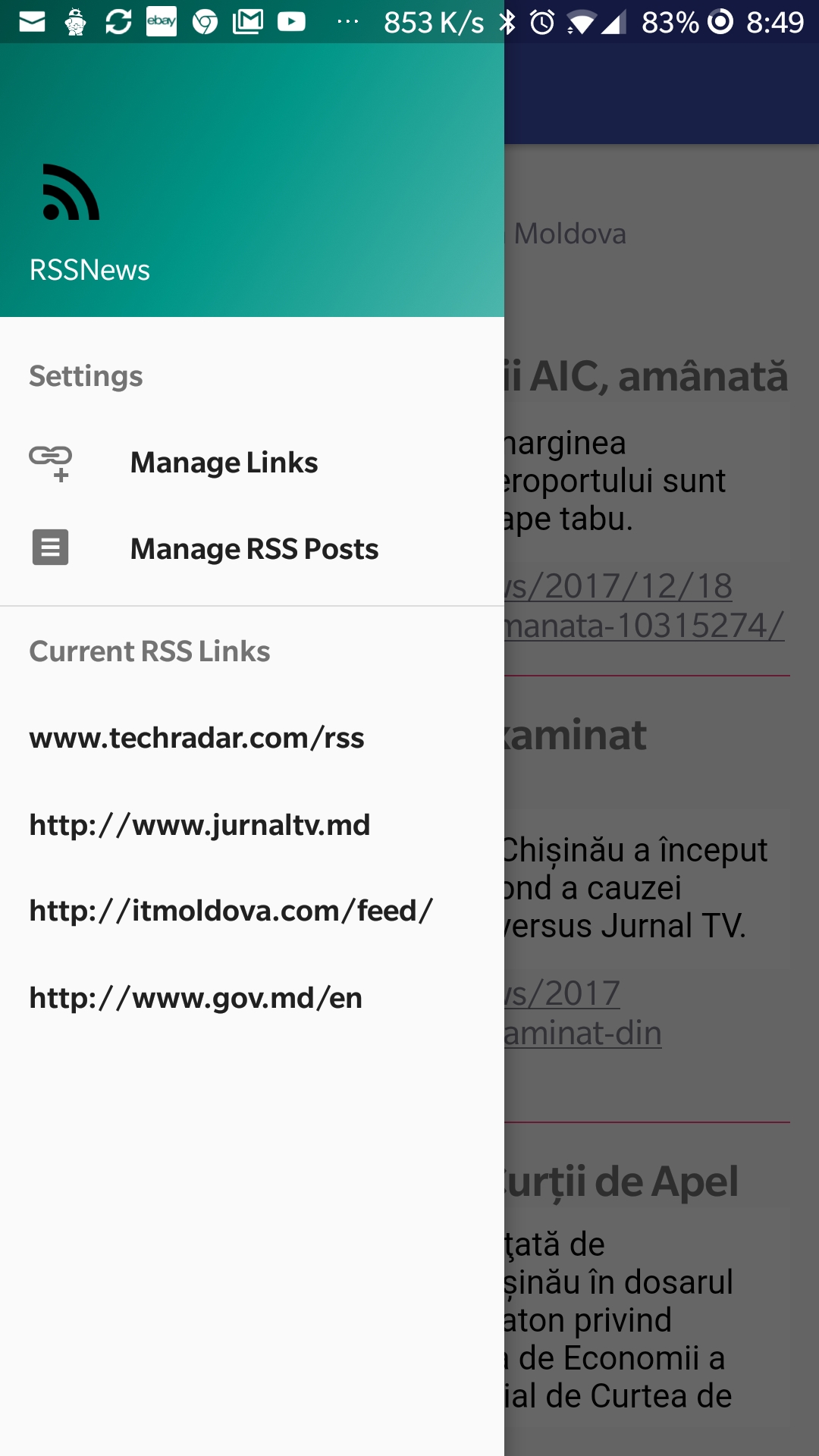


Figura 1 – Structura proiectului

Figura 2 – Interfata Aplicatiei

**Concluzie**

In urma realizarii laboratorului numarul 3, am aflat metodele de creare a serviciilor pe Android, apelarea serviciilor interne (Alarm, etc..) integrarea valorilor in database resources, XmlPullParser

Anexa:

package com.rssnews.rssnews;

import android.content.Context;

import android.content.Intent;

import android.database.sqlite.SQLiteDatabase;

import android.graphics.Color;

import android.net.ConnectivityManager;

import android.net.NetworkInfo;

import android.nfc.Tag;

import android.os.AsyncTask;

import android.os.Bundle;

import android.os.Environment;

import android.support.design.widget.FloatingActionButton;

import android.support.design.widget.Snackbar;

import android.support.v4.widget.SwipeRefreshLayout;

import android.support.v7.widget.DefaultItemAnimator;

import android.support.v7.widget.LinearLayoutManager;

import android.support.v7.widget.RecyclerView;

import android.text.TextUtils;

import android.util.Log;

import android.util.Xml;

import android.view.SubMenu;

import android.view.View;

import android.support.design.widget.NavigationView;

import android.support.v4.view.GravityCompat;

import android.support.v4.widget.DrawerLayout;

import android.support.v7.app.ActionBarDrawerToggle;

import android.support.v7.app.AppCompatActivity;

import android.support.v7.widget.Toolbar;

import android.view.Menu;

import android.view.MenuItem;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Switch;

import android.widget.TextView;

import android.widget.Toast;

import org.xmlpull.v1.XmlPullParser;

import org.xmlpull.v1.XmlPullParserException;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStreamWriter;

import java.net.URL;

import java.util.ArrayList;

import java.util.List;

public class MainActivity extends AppCompatActivity

implements NavigationView.OnNavigationItemSelectedListener {

private RecyclerView mRecyclerView;

private SwipeRefreshLayout mSwipeLayout;

private TextView mFeedTitleTextView;

private TextView mFeedLinkTextView;

private TextView mFeedDescriptionTextView;

private List<RssFeedModel> mFeedModelList;

private String mFeedTitle;

private String mFeedLink;

private String mFeedDescription;

private String currentUrl;

private static int *MAX\_POST\_LIMIT* = 100;

private RssFeedModelAdapter adapter;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

Toolbar toolbar = (Toolbar) findViewById(R.id.*toolbar*);

setSupportActionBar(toolbar);

DrawerLayout drawer = (DrawerLayout) findViewById(R.id.*drawer\_layout*);

ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(

this, drawer, toolbar, R.string.*navigation\_drawer\_open*, R.string.*navigation\_drawer\_close*);

drawer.addDrawerListener(toggle);

toggle.syncState();

NavigationView navigationView = (NavigationView) findViewById(R.id.*nav\_view*);

Menu m = navigationView.getMenu();

SubMenu menuGroup = m.addSubMenu("Current RSS Links");

final List<Link> linkList = new DatabaseLinks(this).read();

String[] names = new String[linkList.size()];

if(linkList.size() > 0) {

int i = 0;

for (Link obj : linkList) {

menuGroup.add(Menu.*NONE*,obj.id,Menu.*NONE*,obj.link);

names[i] = obj.link;

i++;

}

}

PostsDatabase.*tablesName* = names;

navigationView.setNavigationItemSelectedListener(this);

mRecyclerView = (RecyclerView) findViewById(R.id.*recyclerView*);

mSwipeLayout = (SwipeRefreshLayout) findViewById(R.id.*swipeRefreshLayout*);

mFeedTitleTextView = (TextView) findViewById(R.id.*feedTitle*);

mFeedDescriptionTextView = (TextView) findViewById(R.id.*feedDescription*);

mFeedLinkTextView = (TextView) findViewById(R.id.*feedLink*);

mRecyclerView.setLayoutManager(new LinearLayoutManager(this));

if(linkList.size() > 0) currentUrl = names[0];

new FetchFeedTask().execute((Void) null);

mSwipeLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {

@Override

public void onRefresh() {

new FetchFeedTask().execute((Void) null);

}

});

}

@Override

public void onBackPressed() {

DrawerLayout drawer = (DrawerLayout) findViewById(R.id.*drawer\_layout*);

if (drawer.isDrawerOpen(GravityCompat.*START*)) {

drawer.closeDrawer(GravityCompat.*START*);

} else {

super.onBackPressed();

}

}

@SuppressWarnings("StatementWithEmptyBody")

@Override

public boolean onNavigationItemSelected(MenuItem item) {

// Handle navigation view item clicks here.

int id = item.getItemId();

final List<Link> linkList = new DatabaseLinks(this).read();

for (Link obj : linkList) {

if(obj.id == id) {

currentUrl = item.getTitle().toString();

new FetchFeedTask().execute((Void) null);

}

}

if (id == R.id.*link\_manager*) {

Intent intent = new Intent(MainActivity.this, LinkManager.class);

startActivity(intent);

} else if (id == R.id.*post\_manager*) {

Intent intent = new Intent(MainActivity.this, PostManager.class);

startActivity(intent);

}

DrawerLayout drawer = (DrawerLayout) findViewById(R.id.*drawer\_layout*);

drawer.closeDrawer(GravityCompat.*START*);

return true;

}

public List<RssFeedModel> parseFeed(InputStream inputStream) throws XmlPullParserException, IOException {

String title = null;

String link = null;

String description = null;

boolean isItem = false;

List<RssFeedModel> items = new ArrayList<>();

try {

XmlPullParser xmlPullParser = Xml.*newPullParser*();

xmlPullParser.setFeature(XmlPullParser.*FEATURE\_PROCESS\_NAMESPACES*, false);

xmlPullParser.setInput(inputStream, null);

xmlPullParser.nextTag();

while (xmlPullParser.next() != XmlPullParser.*END\_DOCUMENT*) {

int eventType = xmlPullParser.getEventType();

String name = xmlPullParser.getName();

if(name == null)

continue;

if(eventType == XmlPullParser.*END\_TAG*) {

if(name.equalsIgnoreCase("item")) {

isItem = false;

}

continue;

}

if (eventType == XmlPullParser.*START\_TAG*) {

if(name.equalsIgnoreCase("item")) {

isItem = true;

continue;

}

}

Log.*d*("MainActivity", "Parsing name ==> " + name);

String result = "";

if (xmlPullParser.next() == XmlPullParser.*TEXT*) {

result = xmlPullParser.getText();

xmlPullParser.nextTag();

}

if (name.equalsIgnoreCase("title")) {

title = result;

} else if (name.equalsIgnoreCase("link")) {

link = result;

} else if (name.equalsIgnoreCase("description")) {

description = result;

}

if (title != null && link != null && description != null) {

if(isItem) {

PostsDatabase db = new PostsDatabase(MainActivity.this);

if (db.countDB(currentUrl) < *MAX\_POST\_LIMIT*) {

RssFeedModel item = new RssFeedModel(title, link, description);

db.addData(item,currentUrl);

}

}

else {

mFeedTitle = title;

mFeedLink = link;

mFeedDescription = description;

}

title = null;

link = null;

description = null;

isItem = false;

}

}

List<RssFeedModel> tmp = new PostsDatabase(this).read(currentUrl);

int i = *MAX\_POST\_LIMIT*;

for(RssFeedModel obj : tmp) {

if (i!=0) {

items.add(obj);

i--;

}

else break;

}

return items;

} finally {

inputStream.close();

}

}

private class FetchFeedTask extends AsyncTask<Void, Void, Boolean> {

private String urlLink;

@Override

protected void onPreExecute() {

mSwipeLayout.setRefreshing(true);

mFeedTitle = null;

mFeedLink = null;

mFeedDescription = null;

mFeedTitleTextView.setText("Feed Title: " + mFeedTitle);

mFeedDescriptionTextView.setText("Feed Description: " + mFeedDescription);

mFeedLinkTextView.setText("Feed Link: " + mFeedLink);

urlLink = currentUrl;

}

@Override

protected Boolean doInBackground(Void... voids) {

if (TextUtils.*isEmpty*(urlLink))

return false;

try {

if(!urlLink.startsWith("http://") && !urlLink.startsWith("https://"))

urlLink = "http://" + urlLink;

URL url = new URL(urlLink);

InputStream inputStream = url.openConnection().getInputStream();

mFeedModelList = parseFeed(inputStream);

return true;

} catch (IOException e) {

Log.*e*("ERROR", "Error", e);

} catch (XmlPullParserException e) {

Log.*e*("ERROR", "Error", e);

}

return false;

}

@Override

protected void onPostExecute(Boolean success) {

mSwipeLayout.setRefreshing(false);

if (success) {

mFeedTitleTextView.setText("Feed Title: " + mFeedTitle);

mFeedTitleTextView.setTextColor(R.color.*defaultTextViewColor*);

mFeedDescriptionTextView.setText("Feed Description: " + mFeedDescription);

mFeedDescriptionTextView.setTextColor(R.color.*defaultTextViewColor*);

mFeedLinkTextView.setText("Feed Link: " + mFeedLink);

mFeedLinkTextView.setTextColor(R.color.*defaultTextViewColor*);

// Fill RecyclerView

adapter = new RssFeedModelAdapter(mFeedModelList);

RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(getApplicationContext());

mRecyclerView.setLayoutManager(mLayoutManager);

mRecyclerView.setItemAnimator(new DefaultItemAnimator());

mRecyclerView.setAdapter(adapter);

} else {

if(!isNetworkAvailable()) {

Toast.*makeText*(MainActivity.this,"No internet connection!", Toast.*LENGTH\_LONG*).show();

mFeedTitleTextView.setText("NO INTERNET CONNECTION!");

mFeedTitleTextView.setTextColor(Color.*RED*);

mFeedDescriptionTextView.setText("READING FROM DB");

mFeedDescriptionTextView.setTextColor(Color.*RED*);

mFeedLinkTextView.setText("CONNECT TO INTERNET TO UPDATE RSS FEED");

mFeedLinkTextView.setTextColor(Color.*RED*);

}

Toast.*makeText*(MainActivity.this,"Enter a valid Rss feed url", Toast.*LENGTH\_LONG*).show();

List<RssFeedModel> tmp = null;

if(currentUrl != null)

tmp = new PostsDatabase(MainActivity.this).read(currentUrl);

if(tmp != null){

mFeedModelList = new ArrayList<RssFeedModel>();

int i = *MAX\_POST\_LIMIT*;

for(RssFeedModel obj : tmp) {

if (i!=0) {

mFeedModelList.add(obj);

i--;

}

else break;

}

adapter = new RssFeedModelAdapter(mFeedModelList);

RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(getApplicationContext());

mRecyclerView.setLayoutManager(mLayoutManager);

mRecyclerView.setItemAnimator(new DefaultItemAnimator());

mRecyclerView.setAdapter(adapter);

}

}

}

}

public static void setMaxPostsAmount(int value) {

*MAX\_POST\_LIMIT* = value;

}

public static int getMaxPostsAmount() {

return *MAX\_POST\_LIMIT*;

}

public boolean isExternalStorageWritable() {

String state = Environment.*getExternalStorageState*();

if (Environment.*MEDIA\_MOUNTED*.equals(state)) {

return true;

}

return false;

}

public void setMaxPostLimitFromFile(int data) throws IOException {

try {

FileOutputStream fou = openFileOutput("limit.txt", *MODE\_WORLD\_READABLE*);

OutputStreamWriter outputStreamWriter = new OutputStreamWriter(fou);

outputStreamWriter.write(Integer.*toString*(data));

outputStreamWriter.close();

}

catch (IOException e) {

Log.*e*("Exception", "File write failed: " + e.toString());

}

}

public int getMaxPostLimitFromFile() throws IOException {

FileInputStream fis;

int n;

fis = openFileInput("limit.txt");

StringBuffer fileContent = new StringBuffer("");

byte[] buffer = new byte[1024];

while ((n = fis.read(buffer)) != -1)

{

fileContent.append(new String(buffer, 0, n));

}

Log.*d*("LIMIT", String.*valueOf*(fileContent));

//string temp contains all the data of the file.

fis.close();

return Integer.*parseInt*(String.*valueOf*(fileContent));

}

private boolean isNetworkAvailable() {

ConnectivityManager connectivityManager

= (ConnectivityManager) getSystemService(Context.*CONNECTIVITY\_SERVICE*);

NetworkInfo activeNetworkInfo = connectivityManager.getActiveNetworkInfo();

return activeNetworkInfo != null && activeNetworkInfo.isConnected();

}

}