**Министерство образования и науки Российской Федерации**

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Инженерная школа информационных технологий и робототехники

Отделение информационных технологий

Направление – Информационные системы и технологии

**Лабораторная работа №5**

по дисциплине: Программирование мобильных устройств и встраиваемых систем

|  |  |  |  |
| --- | --- | --- | --- |
| Выполнил: студент гр. 8И6А | \_\_\_\_\_\_\_\_\_\_\_ | И.С.Стецов | |
|  |  | |  |
| Проверил: ассистент ОИТ | \_\_\_\_\_\_\_\_\_\_\_ | | А. В. Погребной |

Томск 2020 г.

# Задание

Общее: на все фоновые запросы должен отображаться индикатор загрузки (ProgressBar), при ошибке пользователю должна предоставляться возможность переотправить запрос (кнопка «Повтор»). При повороте экрана или переход на другой экран, его состояние не должно теряться, запросы не должны переотправляться.

**Вариант 1**

Реализовать поиск по репозиториям GitHub. В тулбаре расположен SearchView. При вводе строки длиной больше чем 2 символа искать на GitHub репозитории содержащие эти символы и выводить их в списке. Каждый элемент списка содержит полное название репозитория и его описание (ограничение 5 строк). При нажатии на элемент списка открывается новая Activity. На ней выводится информация по репозиторию: название репозитория в тулбаре, полное описание репозитория, список коммитов, список issue. Реализовать обновление информации по репозиторию движением pull-to-refresh (SwipeRefreshLayout).

\

# Ход работы

Вывод информации о репозиториях представляется в виде списка. Была создана активити для вывода этой информации. Все элемементы размещены внутри SwipeRefreshLayout (ID: swipe\_container), позволяющем обновлять информацию посредством pull-to-refresh. Внутри контейнера находится NestedScrollView, внитри которого находится информация о репозитории. А внитри него находится лэйаут с полным описанием репозитория, списком коммитов и списком issues.

<?xml version="1.0" encoding="utf-8"?>

<androidx.swiperefreshlayout.widget.SwipeRefreshLayout

xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/swipe\_container"

android:layout\_width="match\_parent"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/lab5\_repo\_description">

</TextView>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Коммиты"

android:paddingBottom="12dp"

android:paddingTop="12dp">

</TextView>

<androidx.core.widget.NestedScrollView

android:layout\_width="match\_parent"

android:layout\_height="200dp"

android:id="@+id/commits">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:id="@+id/commits\_list"

android:orientation="vertical">

</LinearLayout>

</androidx.core.widget.NestedScrollView>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Issues"

android:paddingBottom="12dp"

android:paddingTop="1dp">

</TextView>

<androidx.core.widget.NestedScrollView

android:layout\_width="match\_parent"

android:layout\_height="200dp"

android:id="@+id/issues">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:id="@+id/issues\_list"

android:orientation="vertical">

</LinearLayout>

</androidx.core.widget.NestedScrollView>

</LinearLayout>

</androidx.core.widget.NestedScrollView>

</androidx.swiperefreshlayout.widget.SwipeRefreshLayout>

Для того чтобы выводить списки требуется создать файлы с xml-разметкой элементов, таких как названия репозитория, описание, коммит или issue.

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/lab5\_commit\_list\_item"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical">

<TextView

xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/commit\_date"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingLeft="12dp"

android:paddingTop="12dp"

/>

<TextView

xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/commit\_message"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingLeft="12dp"

/>

</LinearLayout>

В главную активити была добавлена статическая переменная Query, обновляющаяся при вводе поискового запроса, которая затем передается в качестве параметра задаче поиска.

@Override

@WorkerThread

protected List<Repo> executeInBackground() throws Exception {

String response = search(Lab5Activity.Query);

return parseSearch(response);

}

private String search(String query) throws Exception {

Request request = new Request.Builder()

.url("https://api.github.com/search/repositories?q=" + query)

.build();

Response response = getHttpClient().newCall(request).execute();

if (response.code() != 200) {

throw new Exception("api returned unexpected http code: " + response.code());

}

return response.body().string();

}

При парсинге ответа от API из него выделяются полное название репозитория, описание и ссылка, которая будет использована далее для получения полной информации о репозитории, если пользователь нажмет на него в списке.

private List<Repo> parseSearch(String response) throws JSONException {

JSONObject responseJson = new JSONObject(response);

List<Repo> repos = new ArrayList<>();

JSONArray items = responseJson.getJSONArray("items");

for (int i = 0; i < items.length(); i++) {

JSONObject repoJson = items.getJSONObject(i);

Repo repo = new Repo();

repo.fullName = repoJson.getString("full\_name");

repo.url = repoJson.getString("html\_url");

repo.description = repoJson.getString("description");

repos.add(repo);

}

return repos;

}

В обсервере было задано появление ProgressBar на время загрузки и его исчезновение при её окончании или возникновении ошибки (в этом случае появляется также диалог с предложением попробовать повторить отправку запроса).

private Observer<List<Repo>> searchObserver = new

Observer<List<Repo>>() {

@Override

public void onLoading(@NonNull Task<List<Repo>> task) {

ProgressBar progressBar = findViewById(R.id.progress\_bar);

progressBar.setVisibility(ProgressBar.VISIBLE);

}

@Override

public void onSuccess(@NonNull Task<List<Repo>> task, @Nullable List<Repo> data) {

//загрузка в список найденных репозиториев

reposAdapter.setRepos(data);

ProgressBar progressBar = findViewById(R.id.progress\_bar);

progressBar.setVisibility(ProgressBar.INVISIBLE);

reposAdapter.notifyDataSetChanged();

}

@Override

public void onError(@NonNull Task<List<Repo>> task, @NonNull Exception e) {

ProgressBar progressBar = findViewById(R.id.progress\_bar);

progressBar.setVisibility(ProgressBar.INVISIBLE);

showDialog(DIALOG);

}

};

Сам диалог содержит заголовок «Произошла ошибка», сообщение «Повторить запрос?» и две кнопки, на которые установлен Listener.

protected Dialog onCreateDialog(int id) {

if (id == DIALOG)

{

AlertDialog.Builder adb = new AlertDialog.Builder(this);

//заголовок

adb.setTitle("Произошла ошибка");

adb.setMessage("Повторить запрос?");

adb.setIcon(android.R.drawable.ic\_dialog\_alert);

adb.setPositiveButton("Да", repeatSearchListener);

adb.setNegativeButton("Нет", repeatSearchListener);

return adb.create();

}

return super.onCreateDialog(id);

}

В самом Listener’е обрабатывается нажатая кнопка. Если была нажата кнопка «Да» - создается и выполняется такая же задача по поиску репозиториев ещё раз.

DialogInterface.OnClickListener repeatSearchListener = new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int answer) {

switch (answer)

{

case Dialog.BUTTON\_POSITIVE:

task = new SearchTask(searchObserver);

threadExecutor.execute(task);

break;

case Dialog.BUTTON\_NEGATIVE:

break;

}

}

};

Для этого был использован Intent и функция putExtra. На активити репозитория выводятся полное имя и описание репозитория, поэтому они туда передаются. Также происходит передача и ссылки, которая будет преобразовываться в ссылку для получения коммитов и issues.

case TYPE\_DESCRIPTION:

DescriptionHolder descriptionHolder = (DescriptionHolder) holder;

Repo repo1 = repos.get( position / 2);

descriptionHolder.description.setText(

repo1.description

);

descriptionHolder.description.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent intent = new Intent(context, RepoActivity.class);

intent.putExtra("name", repos.get(position/2).fullName);

intent.putExtra("url", repos.get(position / 2).url);

intent.putExtra("description", repos.get(position/2).description);

context.startActivity(intent);

}

});

break;

Соответственно, при создании активити репозитория получает эти данные.

Bundle arguments = getIntent().getExtras();

nameOfRepo = (String) arguments.getSerializable("name");

repoUrl = (String) arguments.getSerializable("url");

repoDescription = (String) arguments.getSerializable("description");

setTitle(nameOfRepo);

Как при создании активити, так и при выполнении действия pull-to-refresh создаются и выполняются одновременно две задачи: для коммитов и для issues.

@Override

public void onRefresh() {

new Handler().postDelayed(new Runnable() {

@Override

public void run() {

// Отменяем анимацию обновления

swipeRefreshLayout.setRefreshing(false);

taskCommits = new SearchCommitsTask(searchObserver);

taskIssues = new SearchIssuesTask(searchIssuesObserver);

//выполнение через пул потоков

threadExecutor.execute(taskCommits);

threadExecutor.execute(taskIssues);

}

}, 4000);

}

При обработке JSON’а нужно выделить поля, соответствующие сообщению, автору коммита, а также дате.

JSONArray responseJson = new JSONArray(response);

for (int i = 0; i < responseJson.length(); i++)

{

JSONObject commit = responseJson.getJSONObject(i);

Commit newCommit = new Commit();

JSONObject commitInformation = commit.getJSONObject("commit");

newCommit.message = commitInformation.getString("message");

JSONObject committer = commitInformation.getJSONObject("committer");

newCommit.date = committer.getString("date");

commits.add(newCommit);

}

# Работа приложения

Вводим в поисковую строку запрос и нажимаем кнопку подтвердить. Выполняется поиск на Гитхабе и выводится их названия в виде списка(рисунок 1)



Рисунок 1 – Поиск по репозиториям

\Демонстрация загрузки данных после отправки запроса(рисунок 2)



Рисунок 2 – Загрузка данных

При загрузке данных, если происходит ошибка, то предлагается повторить запрос(рисунок 3).

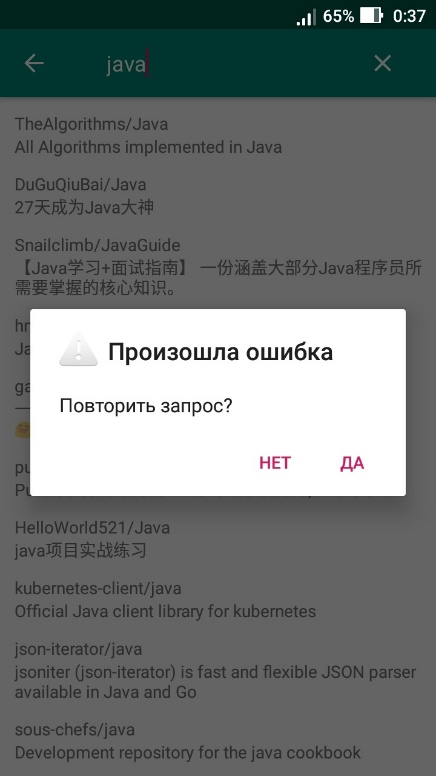


Рисунок 3 – Сообщение об ошибке

При выборе какого либо репозитория, открывается экран с названием этого репозитория и его описанием(рисунок 4).

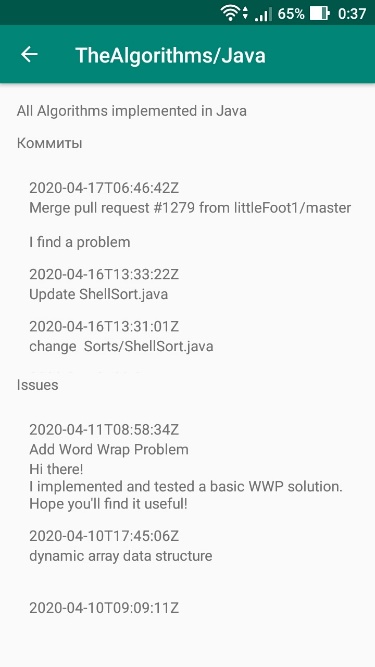


Рисунок 4 – Просмотр репозитория

При свайпе информация загружается заново отображая символ загрузки(рисунок 6).

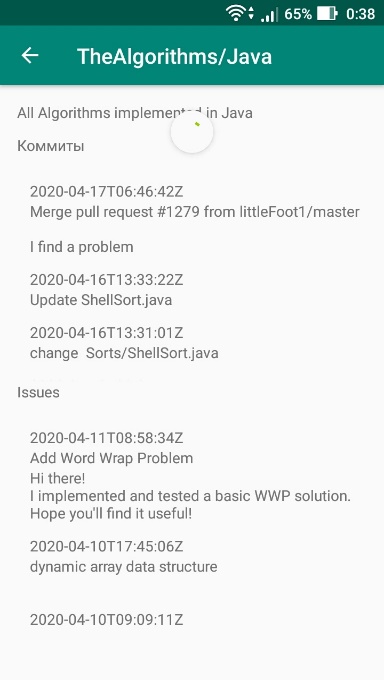


Рисунок 5 – Повторная загрузка данных

# Вывод

В ходе выполнения лабораторной работы было создано приложение по поиску данных о репозиториях GitHub с дополнительными задачами. Для решения задачи потребовалось использовать знания, приобретённые в предыдущих лабораторных работах.