**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное автономное образовательное учреждение высшего образования

**«Национальный исследовательский Томский политехнический университет»**

Инженерная школа информационных технологий и робототехники

Направление «Информационные системы и технологии»

Отчёт по лабораторной работе №5

**«Запросы к сети, многопоточность»**

по дисциплине **«Программирование мобильных устройств и встраиваемых систем»**

Выполнил:

Студент группы 8И6А Козловский В. Е. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил:

Ассистент ОИТ Погребной А.В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Томск 2020

# Задание

Вариант 4. 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. Запрос отправлять с задержкой в 500мс (можно использовать метод Handler::postDelayed). Если во время отправки запроса слово для поиска поменялось, прерывать запрос. Каждый элемент списка содержит полное название репозитория, его описание (ограничение 5 строк). Грузить по 20 репозиториев за запрос. При скролле списка вниз, догружать следующие 20 репозиториев. Реализовать обновление списка движением pull-to-refresh (SwipeRefreshLayout).

# Ход работы

Для отображения данных о репозиториях был создан класс “ReposAdapter”:

**package** ru.tpu.courses.lab5.adapter;  
  
**import** android.view.ViewGroup;  
  
**import** androidx.annotation.NonNull;  
**import** androidx.recyclerview.widget.RecyclerView;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**import** ru.tpu.courses.lab5.Repo;  
  
**public class** ReposAdapter **extends** RecyclerView.Adapter<RecyclerView.ViewHolder> {  
  
 **public static final int *TYPE\_REPO*** = 0;  
 **public static final int *TYPE\_DESCRIPTION*** = 1;  
  
 **private** List<Repo> **repos** = **new** ArrayList<>();  
  
 @Override  
 @NonNull  
 **public** RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, **int** viewType) {  
 **switch** (viewType) {  
 **case *TYPE\_REPO***:  
 **return new** RepoHolder(parent);  
 **case *TYPE\_DESCRIPTION***:  
 **return new** DescHolder(parent);  
 }  
 **throw new** IllegalArgumentException(**"unknown viewType = "** + viewType);  
 }  
  
 @Override  
 **public void** onBindViewHolder(@NonNull RecyclerView.ViewHolder holder, **int** position) {  
 Repo repo = **repos**.get(position/2);  
 **switch** (getItemViewType(position)) {  
 **case *TYPE\_REPO***:  
 RepoHolder repoHolder = (RepoHolder) holder;  
 repoHolder.**repo**.setText(  
 repo.**fullName** );  
 **break**;  
 **case *TYPE\_DESCRIPTION***:  
 DescHolder descHolder = (DescHolder) holder;  
 descHolder.**desc**.setText(  
 repo.**description** );  
 **break**;  
 }  
 }  
  
 @Override  
 **public int** getItemCount() {  
 **return repos**.size() \* 2;  
 }  
  
 @Override  
 **public int** getItemViewType(**int** position) {  
 **return** position % 2 == 0 ? ***TYPE\_REPO*** : ***TYPE\_DESCRIPTION***;  
 }  
  
 **public void** setRepos(List<Repo> reposList) {  
 **this**.**repos** = reposList;  
 }  
}

В основной Activity модуля были внесены изменения объекта Observer для вывода полученных объектов в список:

**public void** onSuccess(@NonNull Task<List<Repo>> task, @Nullable List<Repo> data) {  
 Log.*d*(***TAG***, **"onSuccess"**);  
 **for** (Repo repo : data) {  
 **reposCache**.addRepo(repo);  
 }  
 **reposAdapter**.setRepos(**reposCache**.getRepos());  
 **reposAdapter**.notifyDataSetChanged();  
}

При обработке ввода названий репозиториев вызывается имплементированный метод “newSearch” для отмены предыдущих запросов методом “removeCallbacks”. Новый запрос отправляется, если введено более двух символов.

**public boolean** onQueryTextChange(String s) {  
 **task**.newSearch(s);  
 **if**(s.length() > 2){  
 *threadExecutor*.execute(**task**);  
 }  
 **return false**;  
}

В класс “Repo” было добавлено поле “description” для хранения текстового описания репозитория на GitHub.

Для обеспечения задержки между запросами к Api был введен флаг “waitDelay”, в зависимости от которого функция отправки запроса будет выполнена сразу либо после ожидания 500мс:

@Override  
@WorkerThread  
**protected** List<Repo> executeInBackground() **throws** Exception {  
 **if**(*waitDelay*)  
 **synchronized**(**this**) {  
 wait(500);  
 }  
 String response = search();  
 **return** parseSearch(response);  
}

# Результат

При переходе к модулю 5 лабораторной работы на экран устройства выводится Toolbar поиска (рис. 1).

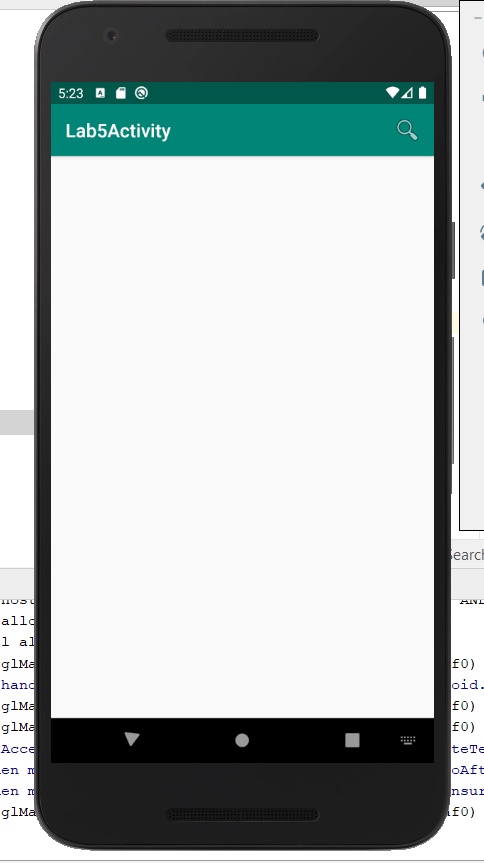


Рисунок 1 – Начальный вид модуля

При нажатии на кнопку поиска, появляется поле SearchView (рис. 2), где пользователь вводит название репозитория.

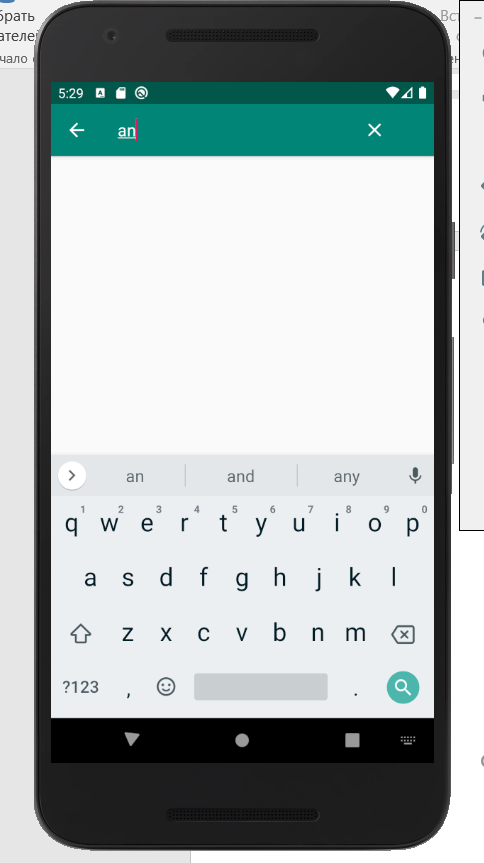
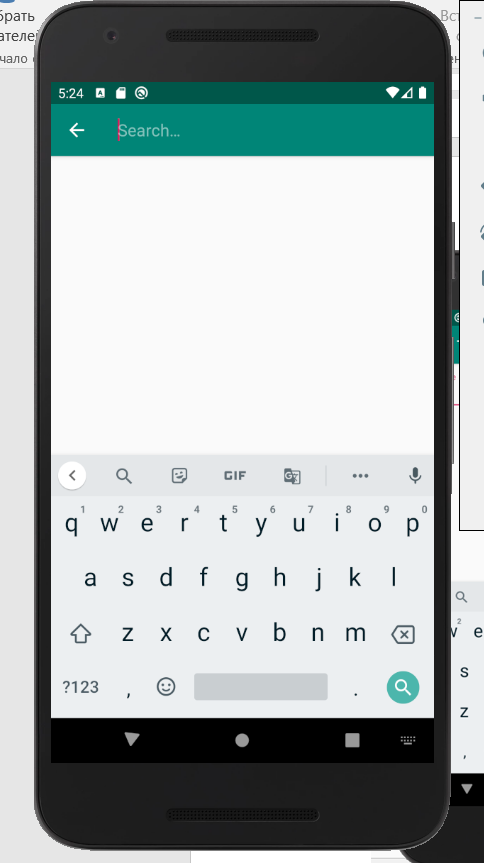


Рисунок 2 – Ввод в поле поиска репозиториев

После ввода трех и более символов названия происходит поиск по репозиториям GitHub и 20 результатов отображается в RecyclerView-списке (рис. 3).

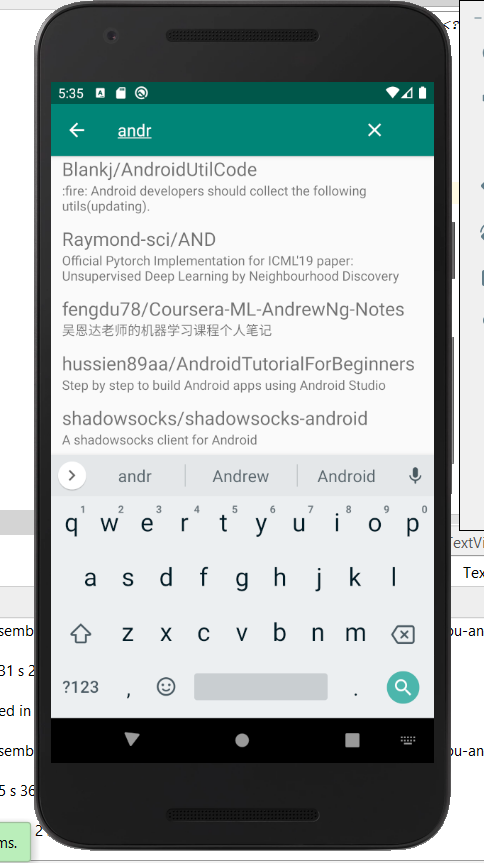


Рисунок 3 – Результат поиска репозиториев

При этом состояние списка сохранено в ReposCache и не изменяется при изменении ориентации, выходе/входе из модуля (рис. 4).

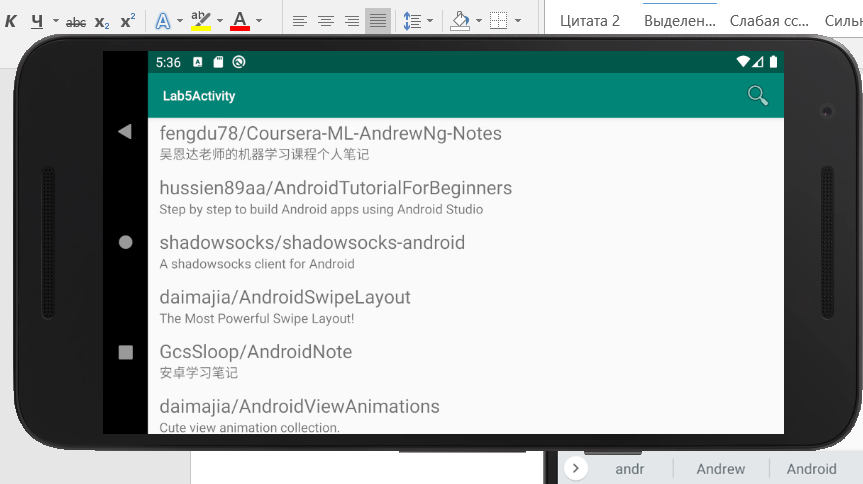


Рисунок 4 – Состояние списка сохранено

# Вывод

В ходе выполнения лабораторной работы были закреплены на практике знания принципов работы компонентов Cache, Adapter, ViewHolder. Реализовано приложение с выполнением кода в параллельных потоках и обращением к ресурсам API по http.