МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное образовательное учреждение высшего образования

Национальный исследовательский Томский политехнический университет

Инженерная школа информационных систем и робототехники

«Информационные системы и технологии»

Отделение Информационных систем и технологий

Отчет по лабораторной работе №5

**«Запросы к сети, многопоточность»**

по дисциплине «Программирование мобильных устройств и встраиваемых систем»

Выполнил:

студент гр. 8И6A \_\_\_\_\_\_\_\_\_\_\_\_\_ Якоби Е.В.

\_\_\_.\_\_\_.\_\_\_\_\_\_

Проверил:

ассистент ОИТ \_\_\_\_\_\_\_\_\_\_\_\_\_Погребной А.В.

\_\_\_.\_\_\_.\_\_\_\_\_\_

Томск 2020 г.

**Задание**

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).

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.

**Ход работы**

Первым делом, добавим разрешение на использование интернета в манифест активити (Рисунок 1), а так же добавим все необходимые зависимости: для использования RecyclerView и SwipeRefreshLayout и библиотеки okhttp3 для запросов к сети и поддержки логирования запросов.

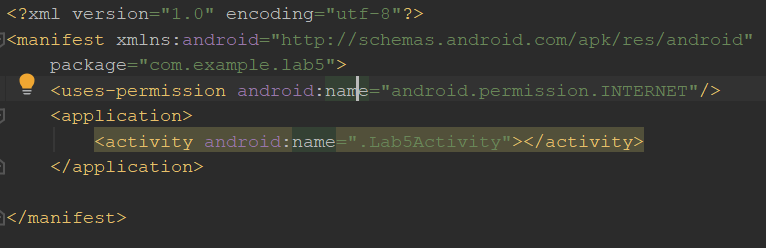


Рисунок 1 – Манифест Lab5Activity

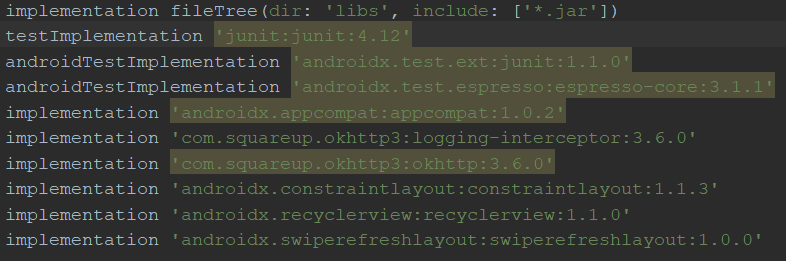


Рисунок 2 – Зависимости build.gradle

Далее, создан класс Repo (Рисунок 3) и RepoCache, реализующий паттерн Singleton для хранения полученных репозиториев и возможности получения их списка из любого места в приложении. Кроме списка репозиториев, в RepoCache сохраняется страница запроса и строка запроса (Рисунок 4)

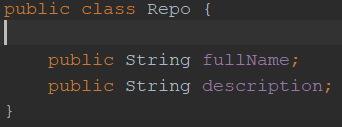


Рисунок 3 – Класс Repo

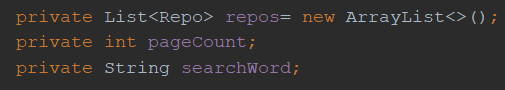


Рисунок 4 – Атрибуты класса RepoCache

Создан интерфейс Observer, с помощью которого для главного потока делаем доступными 3 состояния фонового потока (Рисунок 5). Таким образом, стало возможным получать данные из фонового потока и выводить их на экран, а так же обрабатывать ошибки, которые произошли во время выполнения запроса в фоновом потоке.

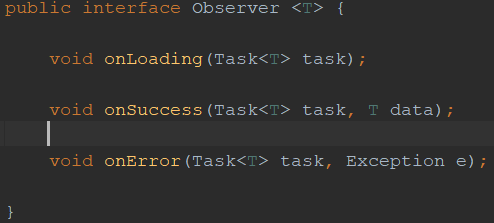


Рисунок 5 – Интерфейс Observer

Далее, создан класс SearchTask, который содержит методы для отправки запроса, получения ответа (Рисунок 6) и его обработки (Рисунок 7), где происходит запись полученных репозиториев в список класса RepoCache.

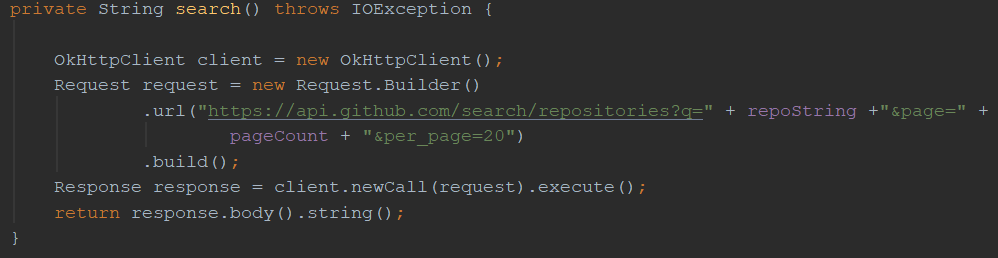


Рисунок 6 – Метод, реализующий запрос к сети и ответ

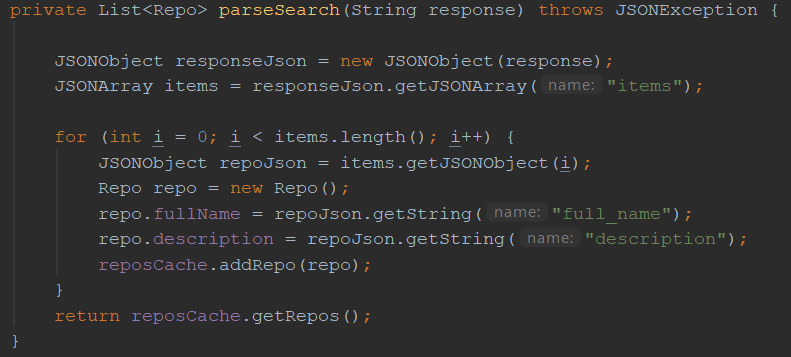


Рисунок 7 – Метод, преобразовывающий Json-объект в объект класса Repo

Создан абстрактный класс Task, в котором были связаны главный поток и созданный ранее интерфейс, методом post (Рисунок 8). Созданный ранее класс SearchTask был унаследован от данного класса и метод executeInBackground был переназначен. (Рисунок 9).

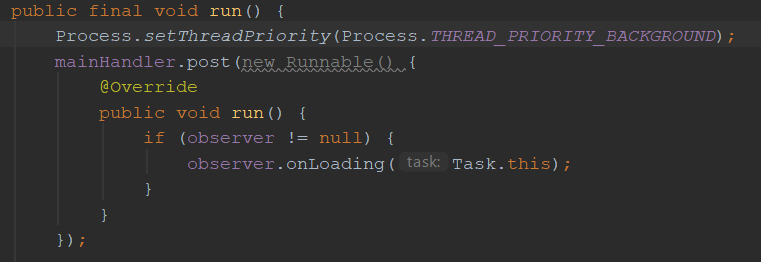


Рисунок 8 – Привязка метода onLoading интерфейса Observer к началу выполнения фонового потока

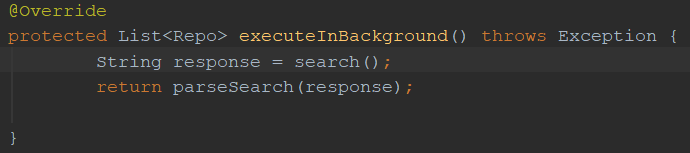


Рисунок 9 – Переназначение метода

В папку drawable ресурсов, добавлен значок Search. В классе Lab5Activity был переопределен метод onCreateOptionsMenu, в котором обработан метод searchView onQueryTextChange, срабатывающий при изменении в нем текста (Рисунок 10). Метод loadRepos показан на рисунке 11. Здесь создается новый поток для обращения к сети с задержкой в 500 мс.

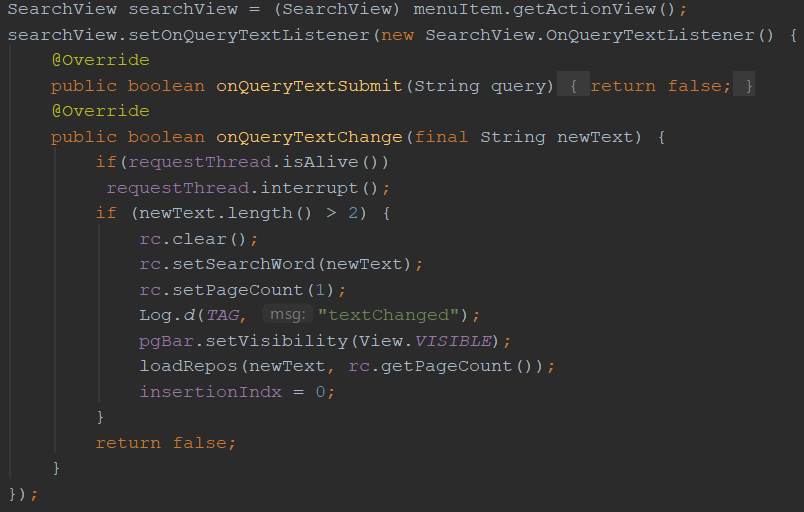


Рисунок 10 – Обработка изменения текста в searchView

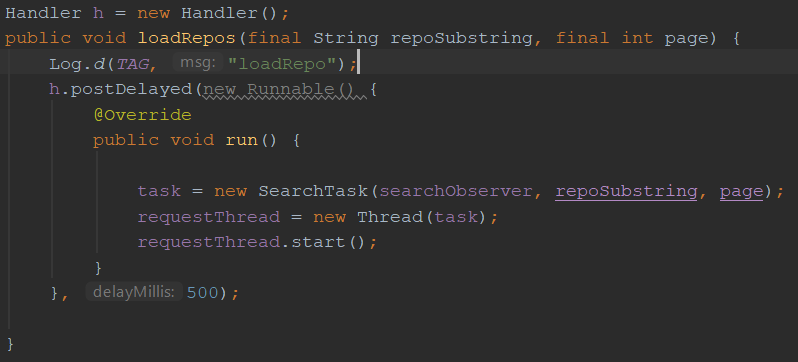


Рисунок 10 – Метод loadRepos

На экран был добавлен recyclerView. Созданы классы для его обработки: RepoHolder, который содержит 2 textView для названия и описания репозитория и RepoAdapter для обработки заполнения нового элемента RepoHolder.

Также, в главном классе был создан экземпляр интерфейса Observer, где происходит получение данных в методе onSuccess и их передача в Adapter recyclerView.

Для обновления данных recycler view был обернут в swiperefreshlayout, который обработан в методе onCreate (Рисунок 11). Также, в данном методе добавлен onScrolListener (Рисунок 12), который отслеживает достижение конца списка и вызывает метод loadRepos, который создает поток для запроса следующей страницы репозиториев (одна страница содержит 20 репозиториев).

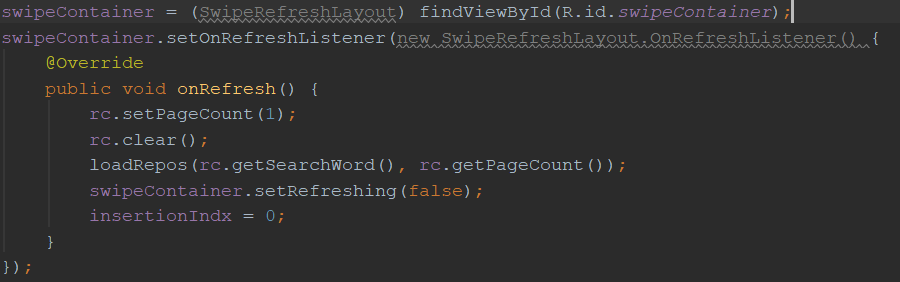


Рисунок 11 – Обработка SwipeRefreshLayout

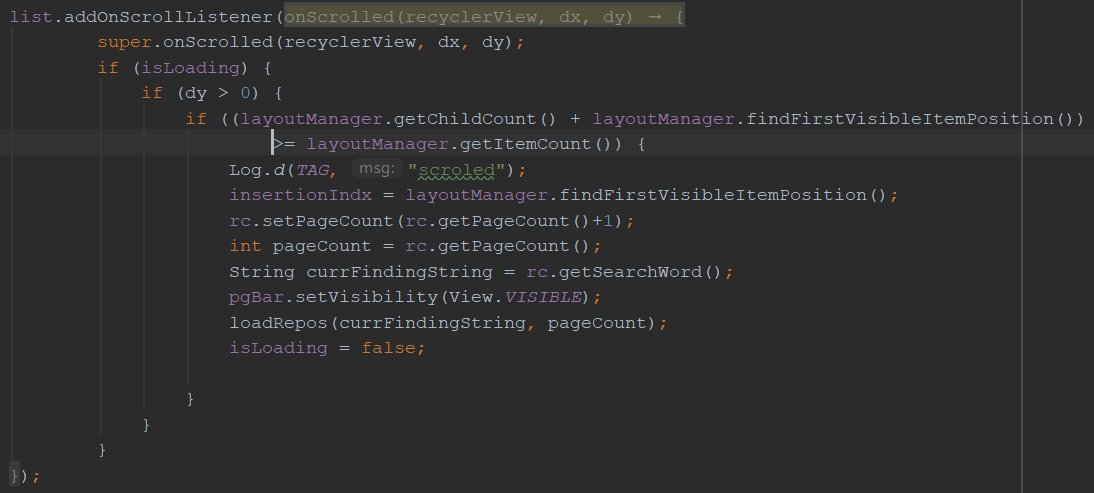


Рисунок 12 – Обработка onScrollListener

Также, в некоторых методах Lab4Activity происходит управление видимостью Layout, который отображает сообщение об ошибки и кнопку для повторной отправки данных, и progressBar. Обработка нажатия кнопки повторной отправки данных происходит в методе onError экземпляра интерфейса Observer (Рисунок 12).

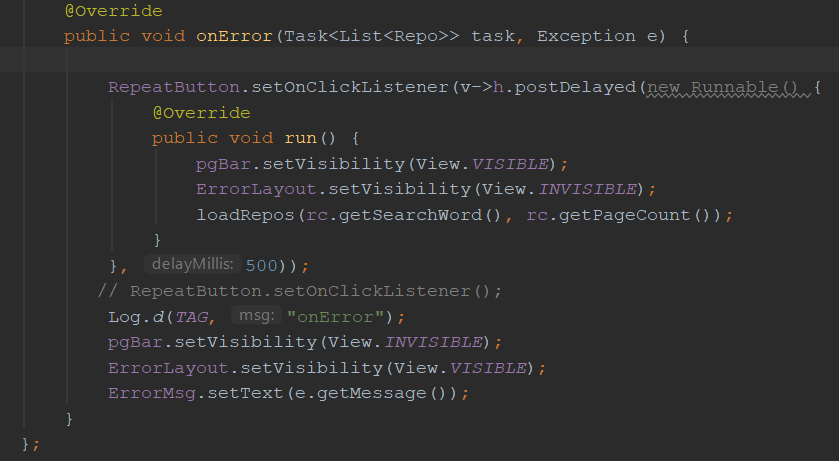


Рисунок 13 – Метод onError

**Результат работы**

В результате выполнения лабораторной работы был создан модуль android-приложения, в котором происходит обработка и отображение названия и описания полученных репозиториев по введенной поисковой строке (Рисунок 14). При повороте экрана все данные и позиция recyclerView остаются прежними (Рисунок 15). При достижении конца списка происходит загрузка следующих 20 репозиториев. После загрузки, recyclerView сохраняет позицию. Загрузка происходит с сопровождением отображения progressBar. При ошибке появляется сообщение с кнопкой, позволяющей повторить запрос (Рисунок 16). При движении pull-to-refresh отображается первые 20 новых полученных репозиториев.

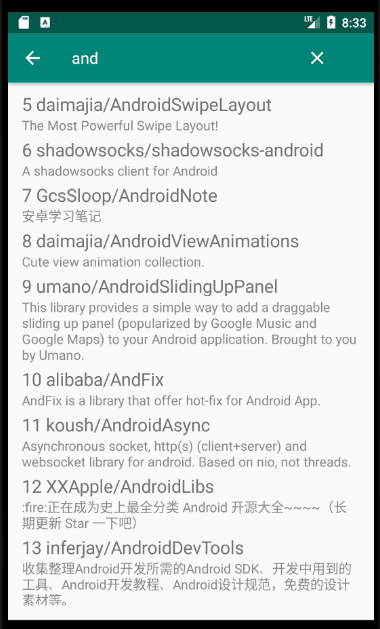


Рисунок 14 – Полученные данные по запросу «and»

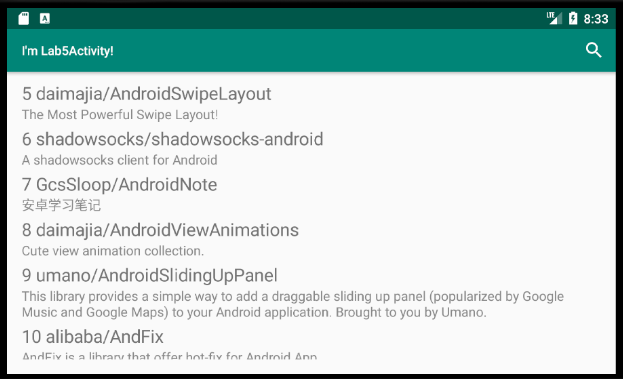


Рисунок 15 – Отображение recyclerView при повороте экрана

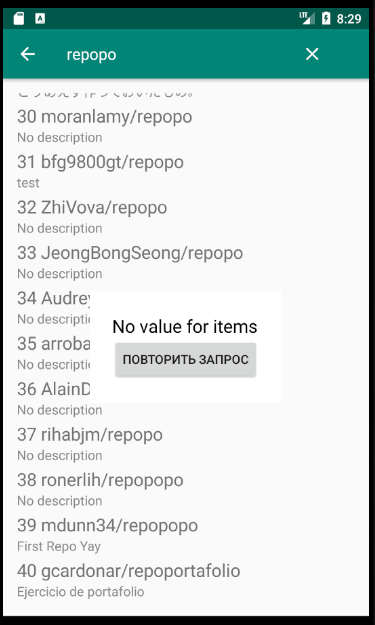


Рисунок 16 – Сообщение об ошибке

**Вывод**

В процессе выполнения данной лабораторной работы были получены следующие навыки android-разработки:

1. Создание потоков, прерывание потоков,
2. Знакомство с паттерном Наблюдатель,
3. Выполнение запросов к сети,
4. Знакомство с SwipeRefreshLayout,
5. Выполнение кода с задержкой с помощью класса Hundler.