**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

Отчёт по лабораторной работе №3

**«RecyclerView, взаимодействие между экранами. Memory Cache»**

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

Выполнил:

Студент группы 8И6А Аманжолов А.А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил:

Преподаватель ОИТ Погребной А.В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Томск 2020 г.

**Цель работы**

Целью лабораторной работы является практическое закрепление работы с кэш памятью, работы с RecyclerView.

# Задание

На экран со студентами добавить SearchView. При поиске искать по ФИО студентов в списке на наличие введённой фразы в SearchView и выводить отфильтрованный список. Для каждого результата подсвечивать цветом кусок текста, который нашёлся (через SpannableString).

# Ход работы

Поле SpannableString это поле ФИО, к которому будут применяться выделения цветом и он же и будет выводиться в списке. В задании требуется выделять текст с помощью SpannableString, поэтому добавляем в класс студента это поле (Рис. 1).

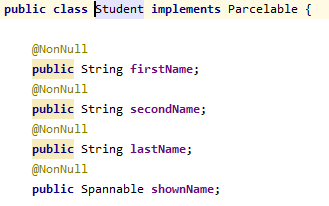


Рисунок 1 – добавление поля в класс Student

Далее нам требовалось добавить элемент поиска (Рис. 2).

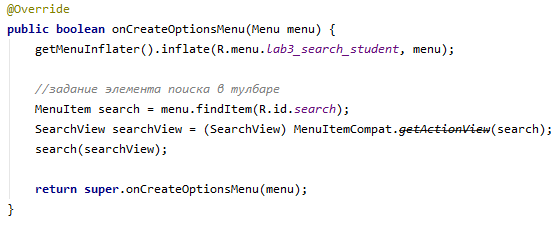


Рисунок 2 – добавление элемента поиска

Сама функция search обрабатывает изменения текста в поле поиска и запрашивает применение фильтра через адаптер. Выше сказанная функция поиска проверяет произошло ли нажатие кнопки «подтвердить» и проверяет любое изменение текста (Рис. 3).

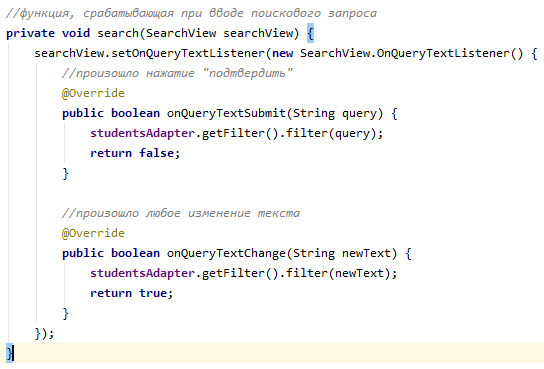


Рисунок 3 – функция поиска

Если запрос отсутствует: пользователь его стер или ещё не начал вводить, то требуется убрать все выделения и вернуть стандартный список студентов. В противном случае происходит обход всех студентов, среди которых обнаруживаются подходящие запросу с помощью функции contains(). У всех найденных таким образом студентов сначала сбрасывается предыдущее выделение, а затем выполняется поиск, на каких именно отрезках строки было найдено совпадение. По найденным отрезкам накладываются новые выделения, после чего полученный отфильтрованный список передается на вывод (Рис. 4).

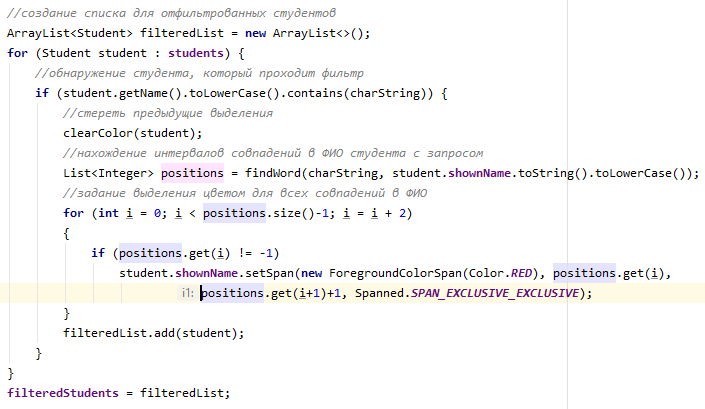


Рисунок 4 – список отфильтрованных студентов

После активируем функцию publishResults, которая обновляет список и уведомляет об его изменении (Рис. 5).

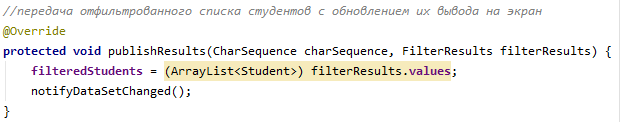


Рисунок 5 – функция обновления сипска

**Результат**

На главном экране приложения выводится список студентов. В правом нижнем углу есть кнопка, для добавления студентов (Рис. 6):

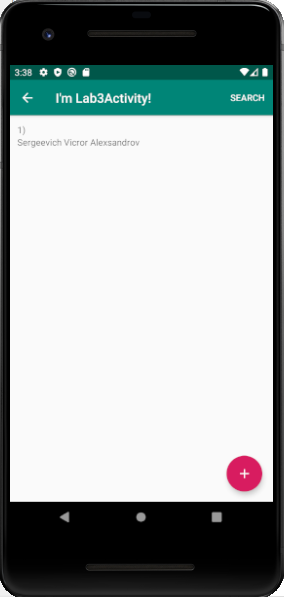


Рисунок 6 – Список студентов и кнопка

При нажатии на кнопку открывается экран с добавлением студента (Рис. 7).

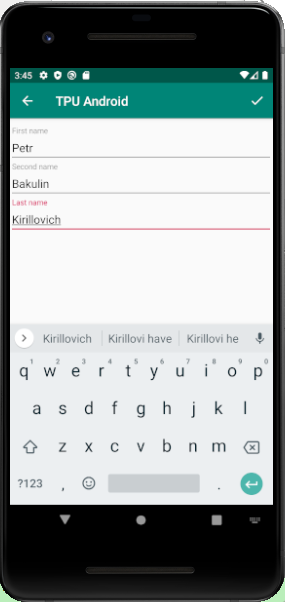


Рисунок 7 – Добавление нового студента

Студент добавился в основной список (Рис. 8).

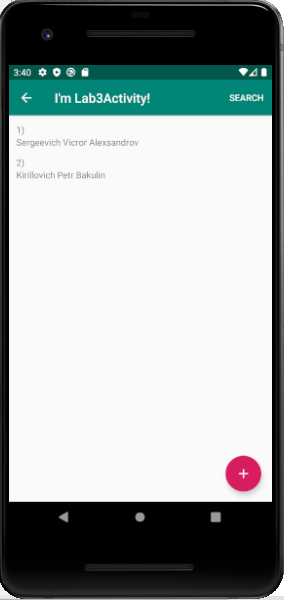


Рисунок 8 – Список с новым студентом

На рисунке 9 показано, что при вводе запроса список автоматически отфильтровывается: на экране остаются только студенты, имеющие совпадения в ФИО с запросом, и эти совпадения окрашиваются в красный. При вводе в строку поиска букв список автоматически фильтруется и выходят студенты с совпадениями, окрашиваясь цветом.

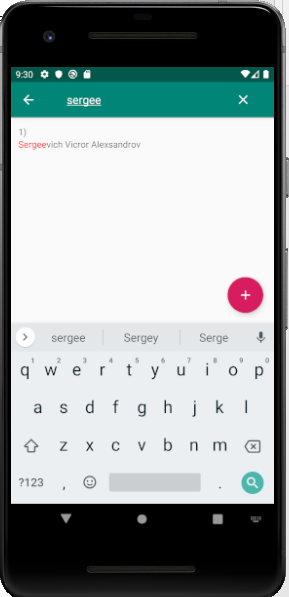


Рисунок 9 – Поиск студента

**Вывод**

В ходе выполнения лабораторной работы было создано приложения, которое позволяет хранить список студентов, добавлять новых студентов. Был реализован поиск студентов по соответствиям, подсвечивая их.