**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 №4**

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

**Вариант 1**

Организовать поиск через БД. Индексировать ФИО студента через отдельную виртуальную таблицу.

# Ход работы

Лабораторная работа 4 является продолжением предыдущей лабораторной работы, и нам требуется хранить и выделать текст цветом, но метод, который мы использовали раннее(Spannable) не подходит, так его нельзя хранить в базе данных. Поэтому вместо него мы создаём дополнительный атрибут, который представляет собой строку, указывающую на начало и конец окрашиваемого текста. Так же мы используем виртуальную таблицу.

@NonNull

@ColumnInfo(name = "string\_spans")

public String spans;

Был создан запрос, чтобы искать соответствия ФИО в базе данных и введенного пользователем слова.

@Query(

"SELECT rowid, first\_name, second\_name, last\_name, shown\_name, string\_spans FROM student WHERE " +

"shown\_name MATCH :search"

)

List<Student> searchByWord(@NonNull String search);

Были внесены изменения и в функцию search в главной активити: теперь при её вызове сначала происходит поиск в БД, а затем, в соответствии с результатами, с помощью адаптера происходит окрашивание соответствий.

В функции поиска изменяем порядок действий, теперь мы сначала производим поиск по Базе Данных, после чего окрашиваем соответствия.

List<Student> newlist = studentDao.searchByWord(query);

studentsAdapter.changeColors(newlist, query);

Создана функция, получающая на вход список отфильтрованных студентов и заданное слово, так как нельзя вносить изменения напрямую в отображаемую строку ФИО.

for (Student student : students) {

clearColor(student);

}

filteredStudents = students;

Если запрос оказывается не пустым, то происходит перебор студентов и заносим их в созданный атрибут для выделения цвета.

for (Student student : filteredList) {

clearColor(student);

List<Integer> positions = findWord(charString.toLowerCase(), student.shownName.toString().toLowerCase());

for (int i = 0; i<positions.size(); i++)

{

if (positions.get(i) != -1)

student.spans = student.spans + positions.get(i).toString() + ",";

}

student.spans = student.spans.substring

(1, student.spans.length() - 1);

}

filteredStudents = filteredList;

# Работа приложения

На главном экране приложения выводится список студентов. Как и в 3 лабораторной работе можно добавлять студентов, но теперь они не исчезнут, если перезайти в приложение, так как они хранятся в базе данных (рисунок 1)

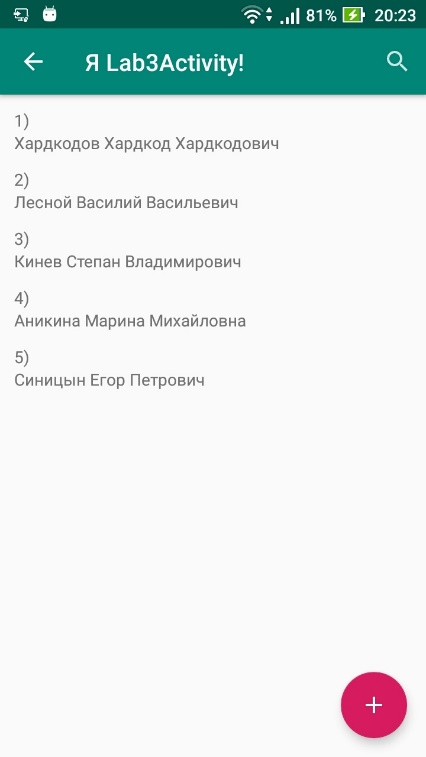
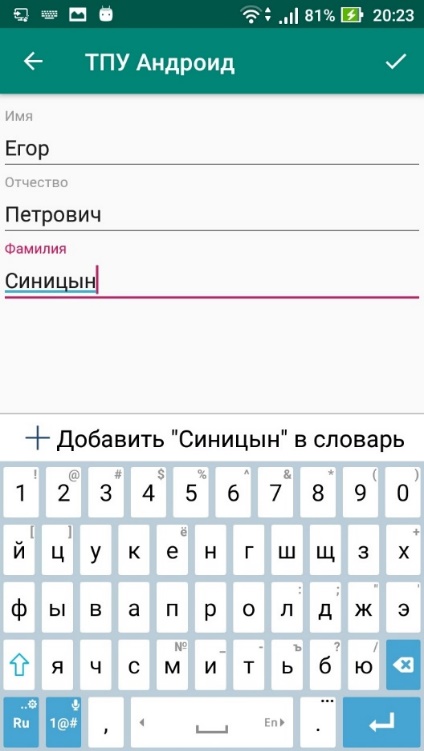
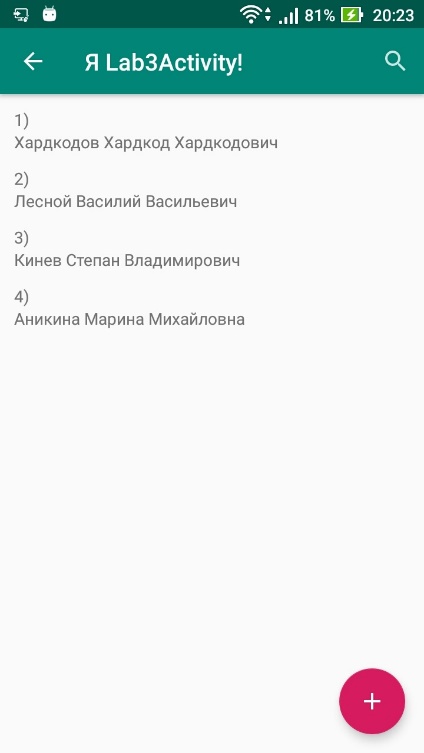


Рисунок 1 – Добавление студента

Когда вводим в строку поиска частиб фамилии, имени или отчества начинает работать полнотекстовый поиск по виртуальной таблице, в результате чего выводятся студенты, ФИО которых имеет соответствие с запросом, и это соответствие окрашивается в красный, как показано на рисунках 4 и 5.

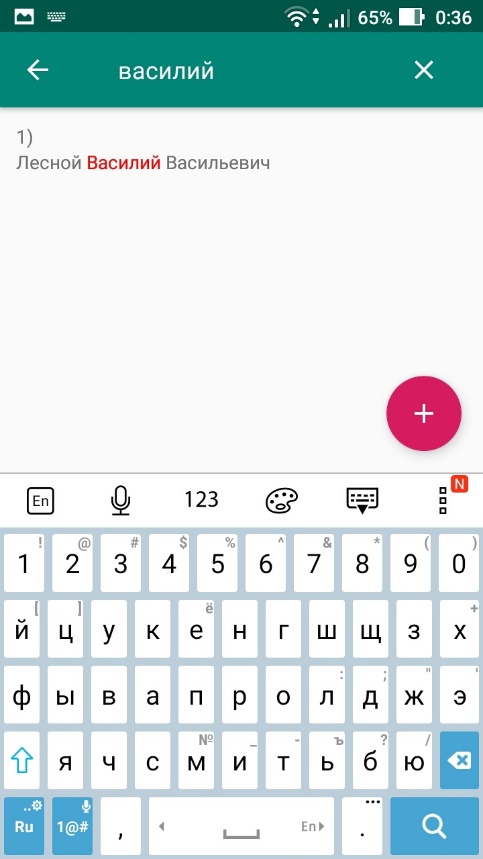


Рисунок 4 – Единичное совпадение

Добавлен ещё один студент для проверки множественного совпадения.

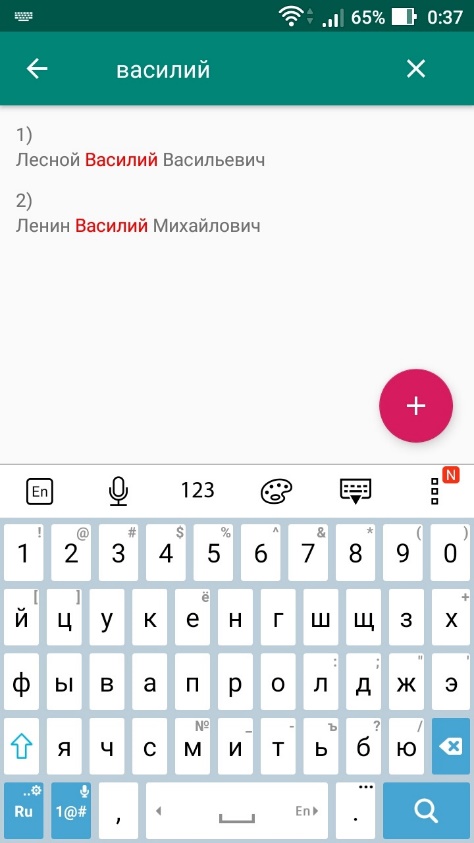


Рисунок 5 – Множественное совпадение

# Итоги работы

В ходе выполнения лабораторной работы была улучшена программа созданная на предыдущей лабораторной работе. Способ хранения данных изменён на хранение через базу данных. Теперь после перезапуска программы добавленные данные останутся на месте, а не пропадут, как это было ранее.