**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 | \_\_\_\_\_\_\_\_\_\_\_ | Н. П. Шкулов | |
|  |  | |  |
| Проверил: ассистент ОИТ | \_\_\_\_\_\_\_\_\_\_\_ | | А. В. Погребной |

Томск 2020 г.

# Задание

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

# Ход работы

Согласно заданию, для выполнения этой лабораторной работы нужно использовать виртуальную таблицу, для чего в описании сущности студентов было добавлено ключевое слово fts4. С другой стороны, эта работа является продолжением предыдущей, поэтому нужно было обеспечить и хранение выделения текста цветом. Так как Spannable непосредственно хранить в базе данных нельзя, вместо этого был добавлен дополнительный атрибут spans, представляющий собой строку, в которой, разделенные запятыми, указываются начала и концы окрашиваемых отрезков.

@Fts4

@Entity

public class Student implements Parcelable {

@PrimaryKey(autoGenerate = true)

@ColumnInfo(name = "rowid")

public int id;

@NonNull

@ColumnInfo(name = "first\_name")

public String firstName;

@NonNull

@ColumnInfo(name = "second\_name")

public String secondName;

@NonNull

@ColumnInfo(name = "last\_name")

public String lastName;

//отображаемое имя, представляющее собой соединение ФИО

@NonNull

@ColumnInfo(name = "shown\_name")

public String shownName;

//хранит информацию об окрашиваемых отрезках

@NonNull

@ColumnInfo(name = "string\_spans")

public String spans;

... }

Виртуальные таблицы поддерживают полнотекстовый поиск, для чего в запросе используется специальное ключевое слово MATCH; соответственно, был создан запрос, который будет искать соответствия среди ФИО в базе данных с введенным словом.

@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.

private void search(SearchView searchView) {

searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {

@Override

public boolean onQueryTextSubmit(String query) {

//выполнение поиска по введенному слову в БД

List<Student> newlist =

studentDao.searchByWord(query);

//окрашивание цветом совпадений с запросом

studentsAdapter.changeColors(newlist, query);

return false;

}

@Override

public boolean onQueryTextChange(String newText) {

List<Student> newlist =

studentDao.searchByWord(newText);

studentsAdapter.changeColors(newlist, newText);

return true;

}

});

}

В связи с тем, что теперь нельзя вносить изменения непосредственно в отображаемую строку ФИО, в адаптере была создана функция changeColors(), получающая на вход список отфильтрованных студентов и искомое слово. Так же, как и в функции search() из предыдущей лабораторной, которая отвечала за фильтрацию и окрашивание, сначала происходит обработка пустой строки: в этом случае нужно сбросить все окрашивания.

if (charString.isEmpty())

{

//убрать выделение цветом у всех студентов

for (Student student : students) {

clearColor(student);

}

//вернуть для отображения весь список

filteredStudents = students;

}

В противном случае, если запрос не пустой, происходит перебор отфильтрованных студентов с занесением в их атрибут spans информации для выделения цветом.

//перебор всех найденных студентов

for (Student student : filteredList) {

//стереть предыдущие выделения

clearColor(student);

//найти позиции для выделения цветом

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;

Таким образом, информация для выделения цвета стала сохраняться в БД. Обрабатывается она в функции onBindViewHolder, связывающей адаптер с холдером, в результате чего на вывод в холдере в список поступает уже окрашенная строка.

if ((!student.spans.equals(" ")))

{

Spannable coloredString = new SpannableString(student.shownName);

//разбиение spans по запятым, чтобы найти позиции

List<String> spns = Arrays.asList(student.spans.split(","));

for (int i = 0; i < spns.size() - 1; i=i+2)

{

int position1 = Integer.parseInt(spns.get(i));

int position2 = Integer.parseInt(spns.get(i+1)) + 1;

coloredString.setSpan(new ForegroundColorSpan(Color.RED),

position1, position2, Spanned.SPAN\_EXCLUSIVE\_EXCLUSIVE);

}

studentHolder.student.setText(

coloredString

);

}

else

studentHolder.student.setText(student.shownName);

# Работа приложения

При запуске приложения открывается экран со списком ранее добавленных в базу данных студентов, как можно видеть на рисунке 1.

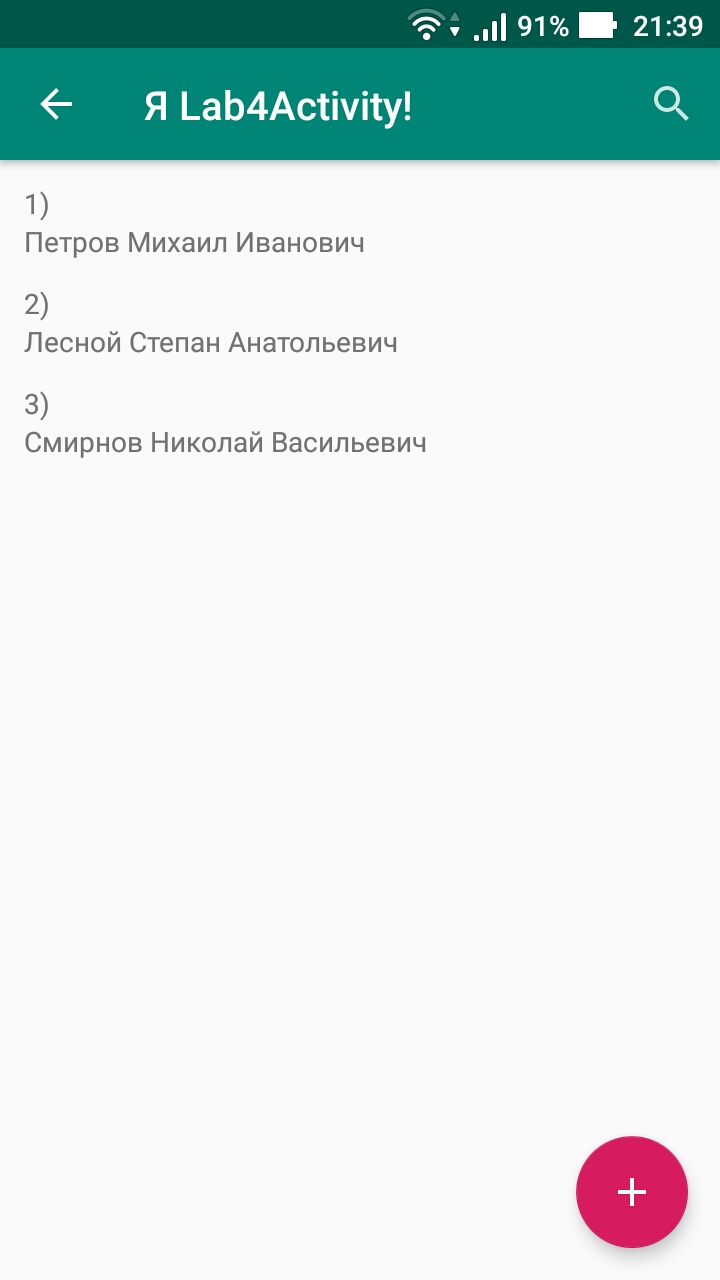


Рисунок 1 – Список добавленных ранее студентов

Также, как и в предыдущей лабораторной работе, можно добавлять новых студентов, как показано на рисунках 2 и 3, но теперь они будут сохраняться в БД, а не в кэш, и они не исчезнут при следующем запуске приложения.

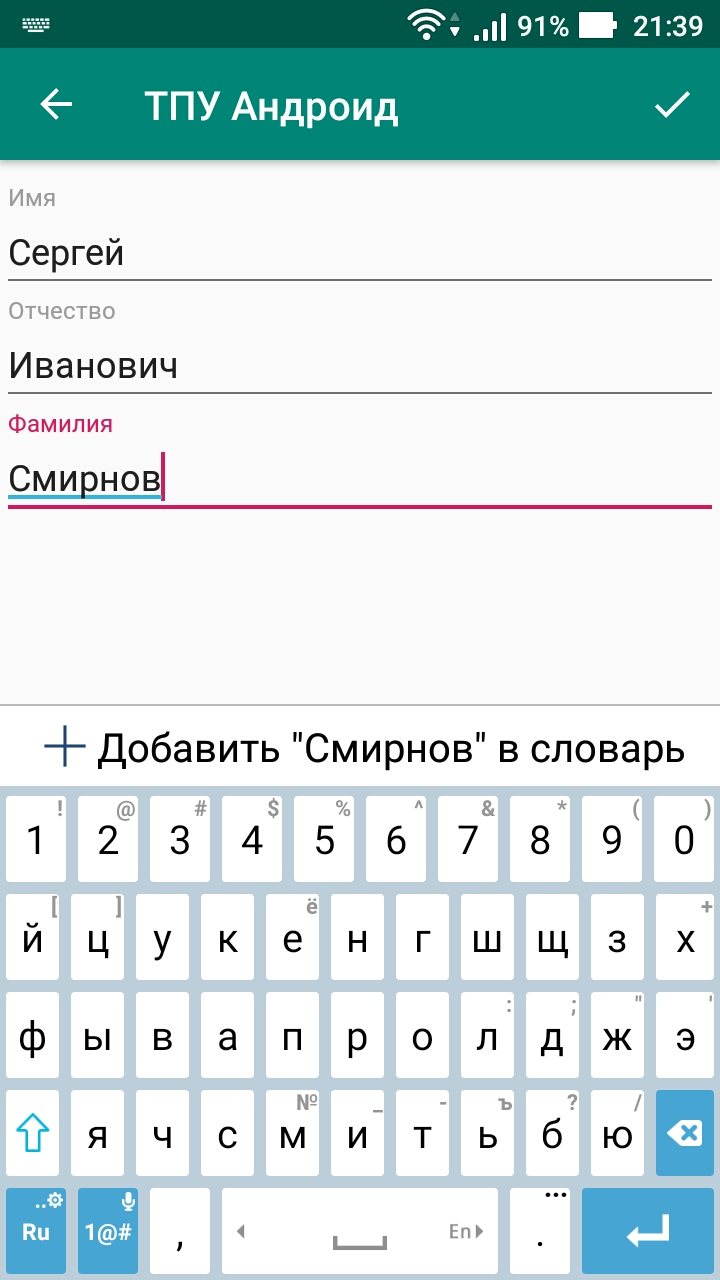


Рисунок 2 – Добавление нового студента

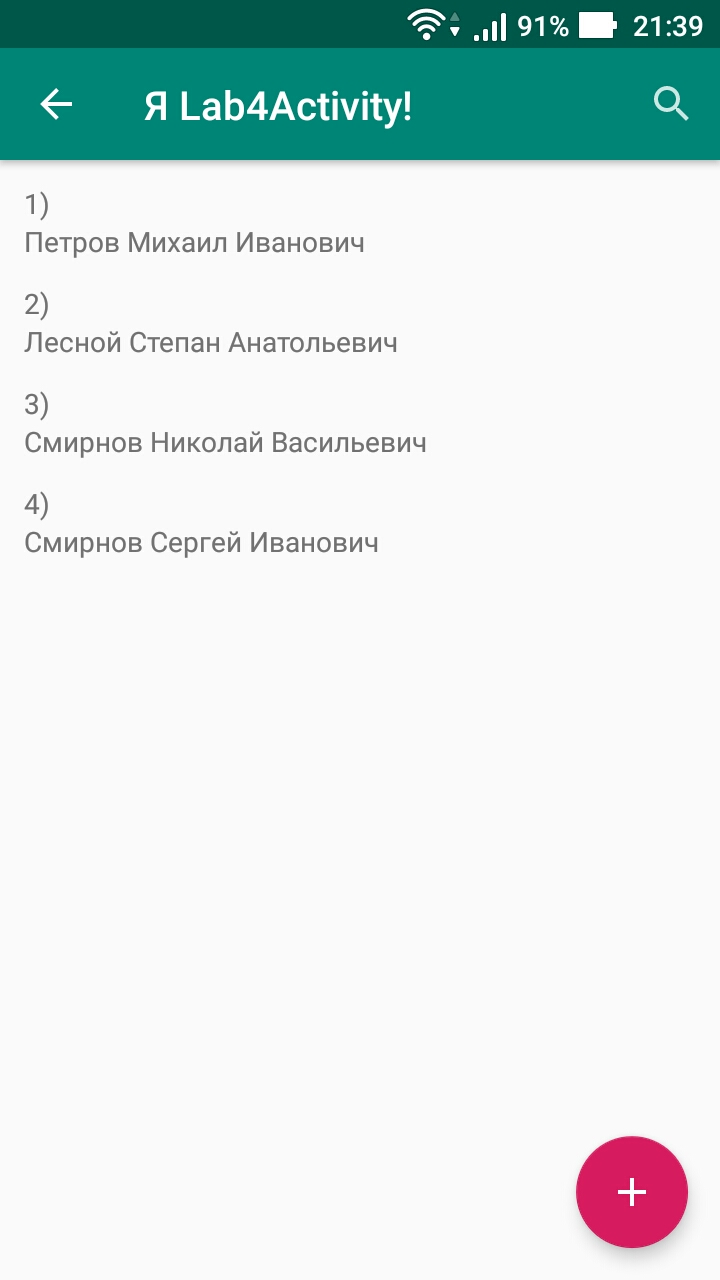


Рисунок 3 – Появление добавленного студента в списке

При вводе в строке поиска части 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.

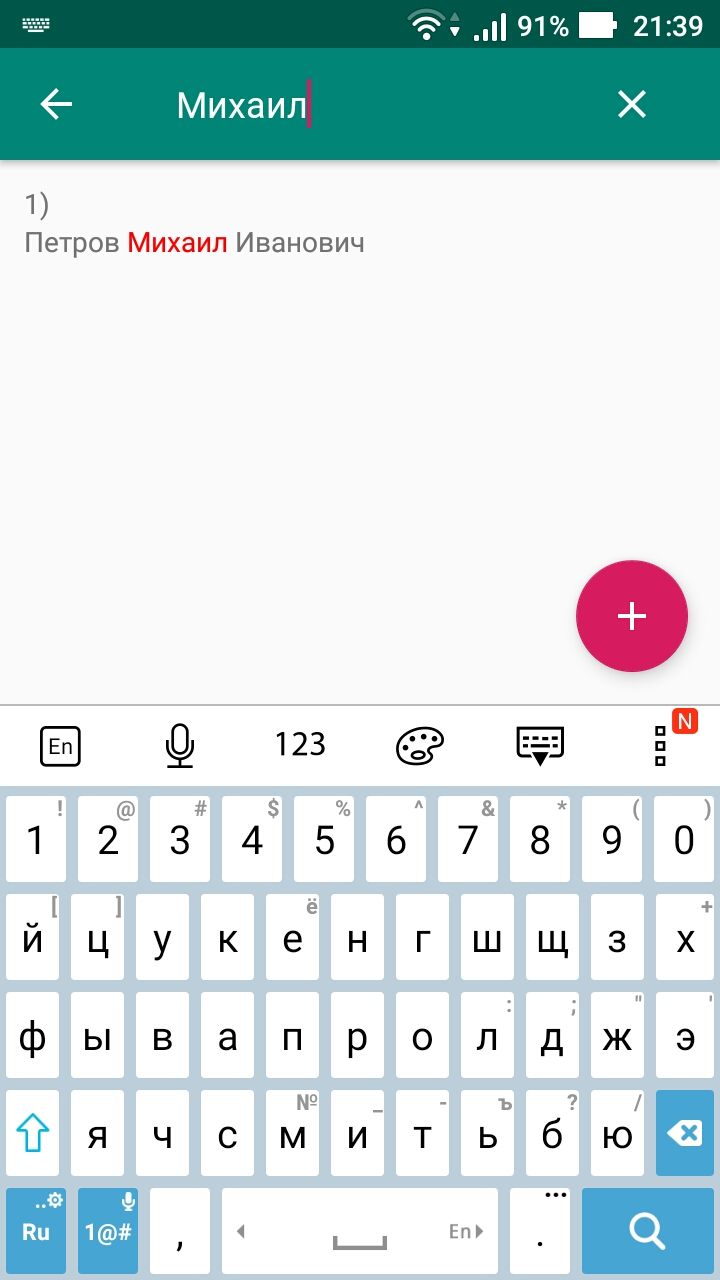


Рисунок 4 – Единичное совпадение

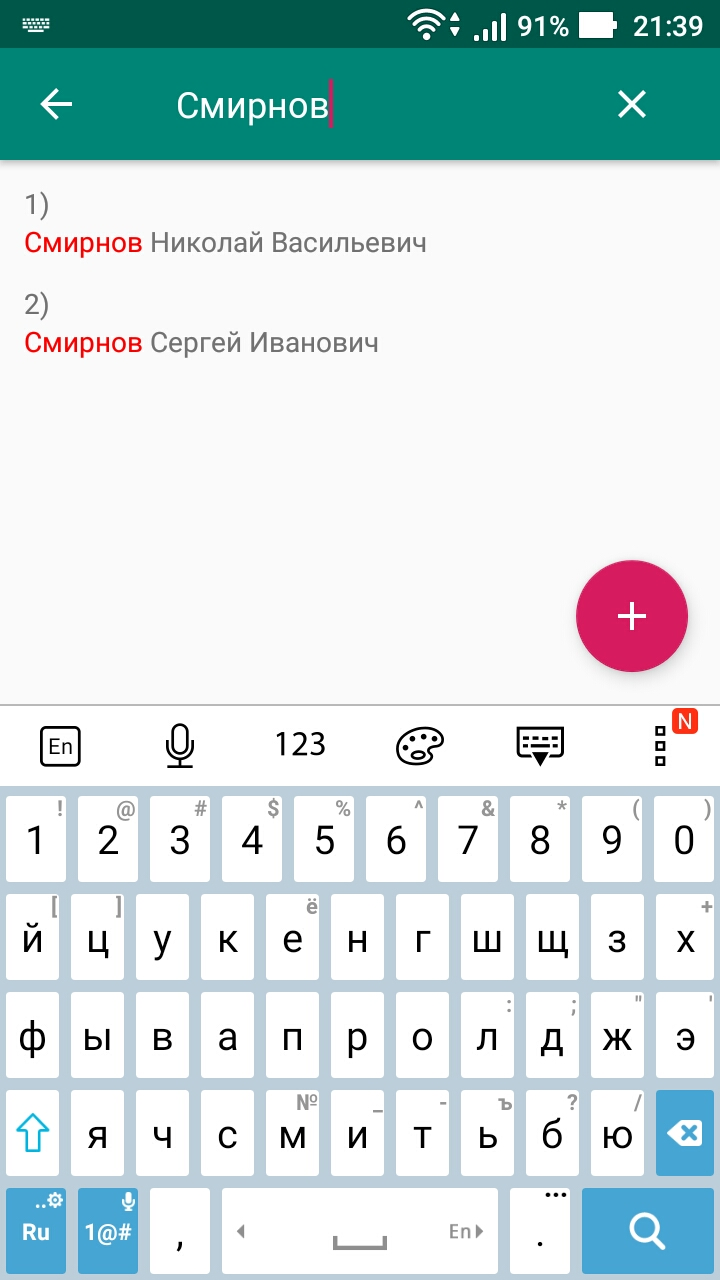


Рисунок 5 – Множественное совпадение

# Итоги работы

В результате выполнения лабораторной работы было создано мобильное приложение, которое, как и приложение из предыдущей лабораторной работы, позволяет хранить, добавлять и фильтровать по поисковому запросу список студентов, однако здесь использовался другой способ хранения данных – через базу данных. В результате, в отличие от данных, хранимых в кэше, эти данные не пропадают при следующем запуске приложения. К тому же, использованная в лабораторной работе виртуальная таблица имеет встроенную функцию полнотекстового поиска, используемую посредством ключевого слова MATCH в запросе, которая работает производительнее, чем обычный запрос с использованием LIKE, за счет индексирования хранящихся в виртуальной таблице данных.