МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И. ВЕРНАДСКОГО»
ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
Кафедра компьютерной инженерии и моделирования

«Вывод списков и RecyclerView»

Лабораторная работа
по дисциплине «Программирование в системах мобильной связи»
студента 4 курса группы ПИ-182(2)
Змитрович Никита Сергеевич

направления подготовки 09.03.04 «ПРОГРАММНАЯ ИНЖЕНЕРИЯ»

Проверил Матюнина Я.Ю.
Ассистент кафедры компьютерной инженерии и моделирования _____
(оценка)

_____
(подпись, дата)

Симферополь, 2022

Цель:

Необходимо доработать приложение, создаваемое на лекциях.

Ход работы

1. Отклик на нажатия

Для того, чтобы повесить слушатель на весь ViewHolder нужно передать корневую View.

```
public CrimeHolder(@NonNull View itemView) {
    super(itemView);
    titleTextView = itemView.findViewById(R.id.crime_title);
    dateTextView = itemView.findViewById(R.id.crime_date);
    root = itemView;
}
```

Рисунок 1 Корневое View во ViewHolder

А далее повесить слушатель на корневую View.

```
public void bindCrime(final Crime crime) {
    this.crime = crime;
    titleTextView.setText(crime.getTitle());
    dateTextView.setText(crime.getDate().toString());
    root.setOnClickListener(view -> {
        Toast.makeText(view.getContext(), crime.getTitle(), Toast.LENGTH_SHORT).show();
    });
}
```

Рисунок 2 Установка слушателя на корневую View

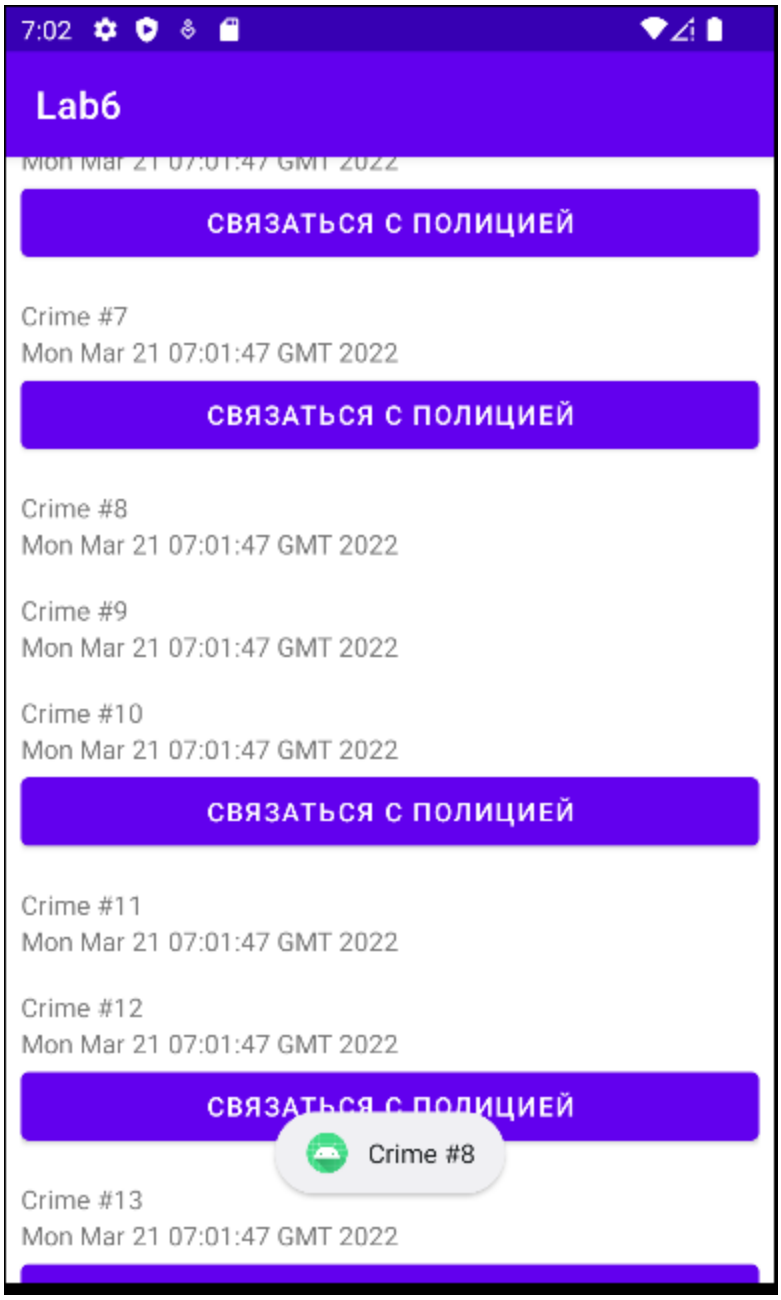

Рисунок 3 Результат

2. Типы View в RecyclerView

Для начала создадим layout и отдельный ViewHolder для серьезных преступлений.

```java
public class SeriousCrimeHolder extends RecyclerView.ViewHolder {

    public TextView titleTextView;
    public TextView dateTextView;
    public View root;

    private Crime crime;

    public SeriousCrimeHolder(@NonNull View itemView) {
        super(itemView);
        titleTextView = itemView.findViewById(R.id.crime_title);
        dateTextView = itemView.findViewById(R.id.crime_date);
        root = itemView;
    }

    public void bindCrime(final Crime crime) {
        this.crime = crime;
        titleTextView.setText(crime.getTitle());
        dateTextView.setText(crime.getDate().toString());
        root.setOnClickListener(view -> {
            Toast.makeText(view.getContext(), crime.getTitle(), Toast.LENGTH_SHORT).show();
        });
    }

}
```

Рисунок 4 ViewHolder для серьезных преступлений

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="8dp">

    <TextView
        android:id="@+id/crime_title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Crime Title" />

    <TextView
        android:id="@+id/crime_date"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Crime Date" />

    <Button
        android:id="@+id/contact_police"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/contact_police"/>


</LinearLayout>
```

Рисунок 5 layout для серьезных преступлений

А далее создадим различные view type, чтобы отличать.

```
private static final int DEFAULT_CRIME = 0;
private static final int SERIOUS_CRIME = 1;
```

Рисунок 6 View типы

Зададим присвоения типа в зависимости от поля isRequiredPolice.

```java
@Override
public int getItemViewType(int position) {
    if (crimeList.get(position).isRequirePolice()) {
        return SERIOUS_CRIME;
    }

    return DEFAULT_CRIME;
}
```

Рисунок 7 Логика присвоения типа

```java
@Override
public void onBindViewHolder(@NonNull RecyclerView.ViewHolder holder, int position) {
    switch (holder.getItemViewType()) {
        case SERIOUS_CRIME:
            ((SeriousCrimeHolder)holder).bindCrime(crimeList.get(position));
            break;
        default:
            ((CrimeHolder)holder).bindCrime(crimeList.get(position));
            break;
    }
}
```

Рисунок 8 Логика создания ViewHolder

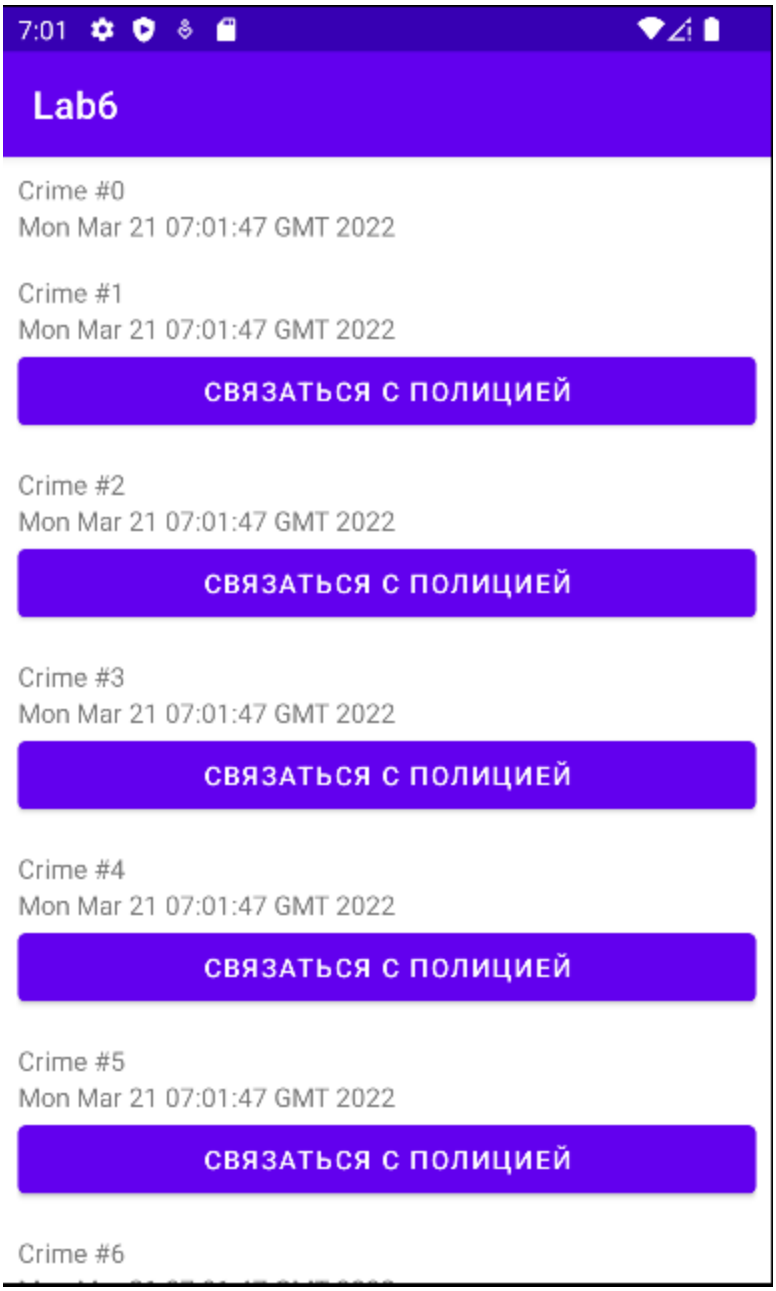

Рисунок 9 Результат

Вывод:

Доработали приложение создаваемое на лекциях.