MD2.2 Фильтры намерений

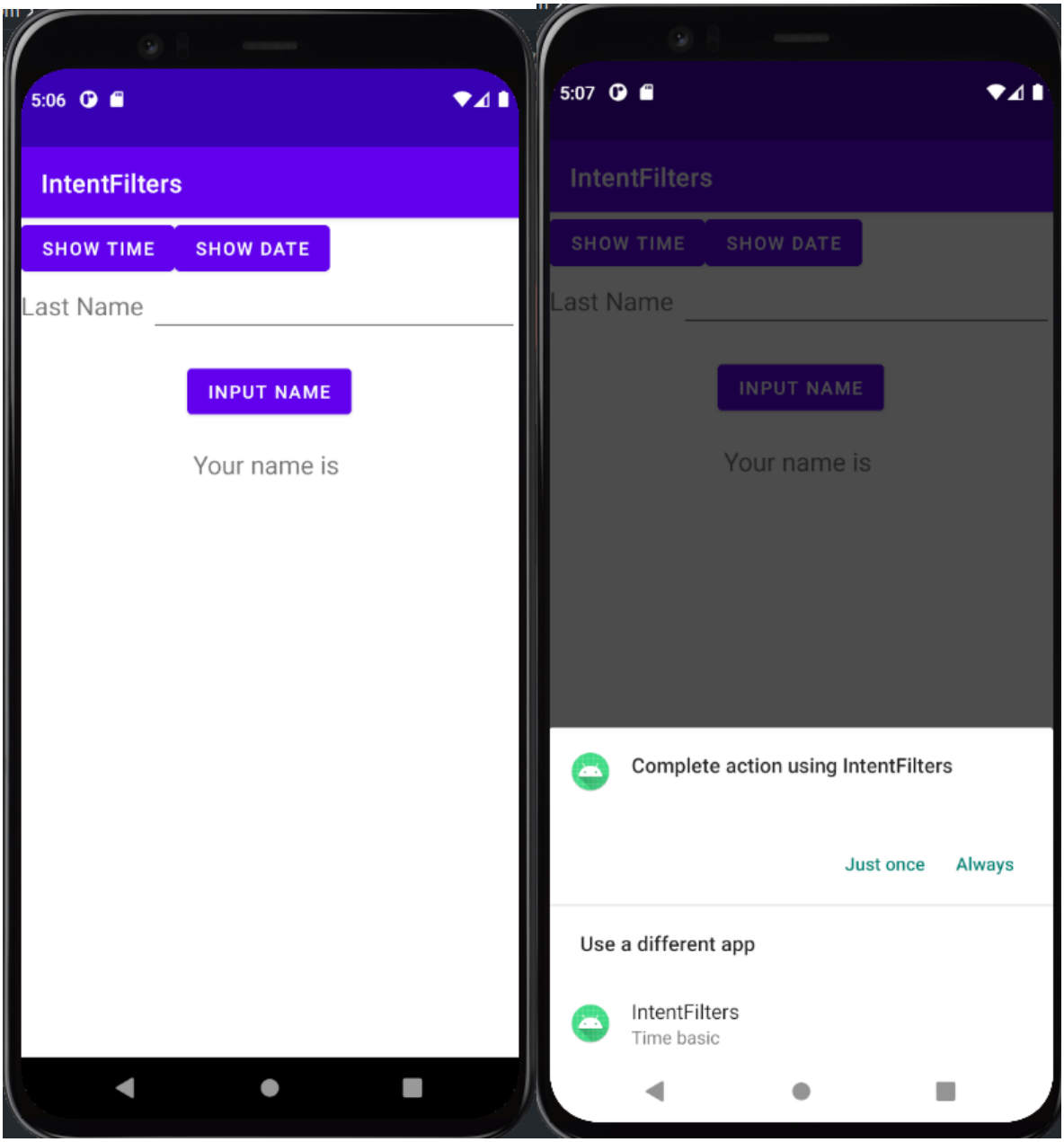
Цель работы

Научиться использовать неявный вызов активности и создавать свои фильтры намерений (intent filters) для запуска независимых компонентов приложения.

Репозиторий с проектом (Основная работа):

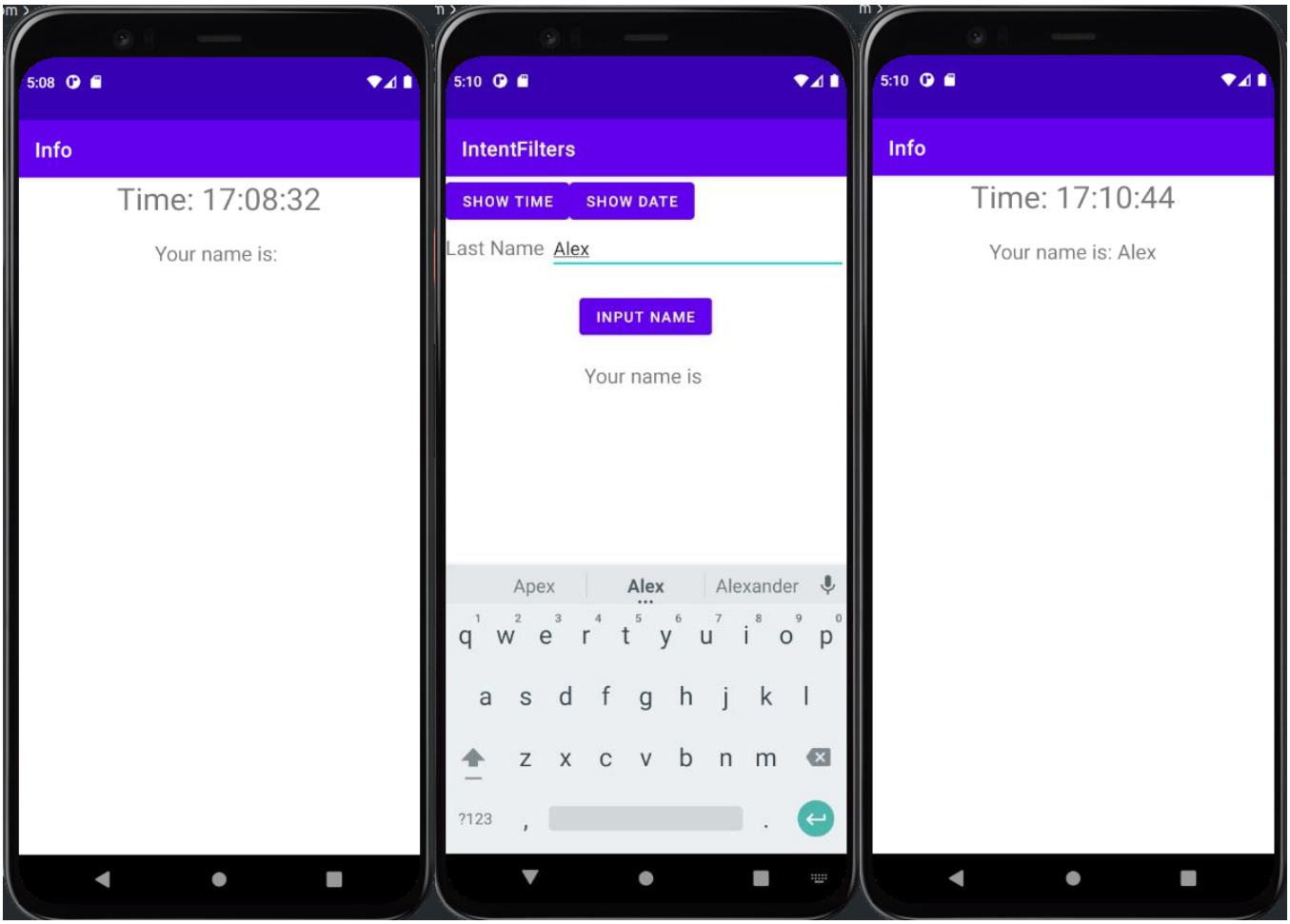
https://github.com/bitcoineazy/Android_Apps/tree/main/MD22_Intent_filters

Работа приложения

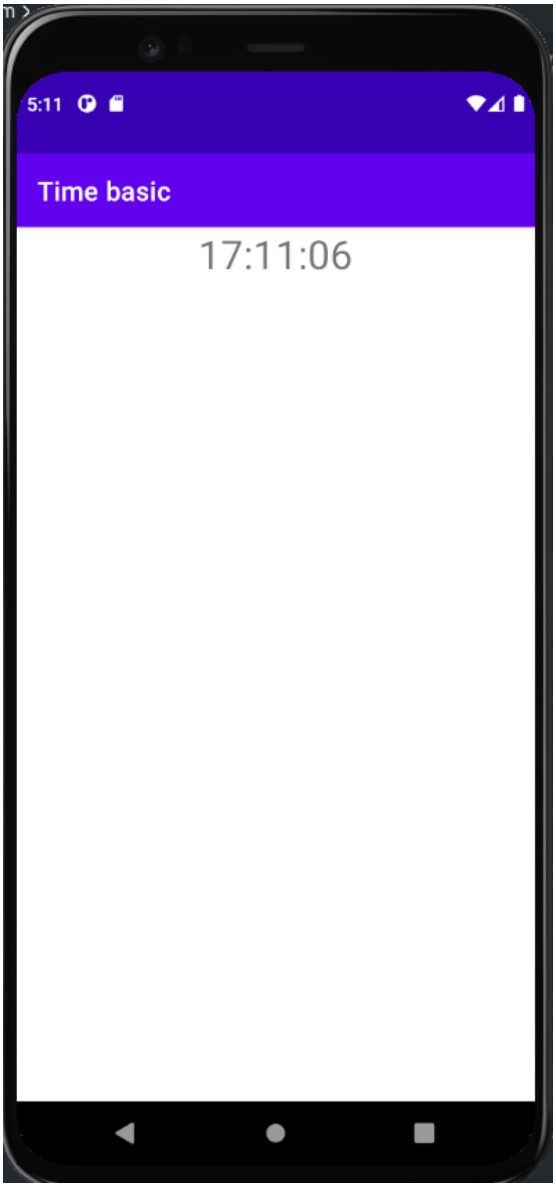


При нажатии на одну из кнопок (SHOW TIME, SHOW DATE) устройство предложит выбрать активность для запуска: SHOW TIME: 1 – Info, 2 – Time, SHOW DATE: 1 – Info, 2 - Date

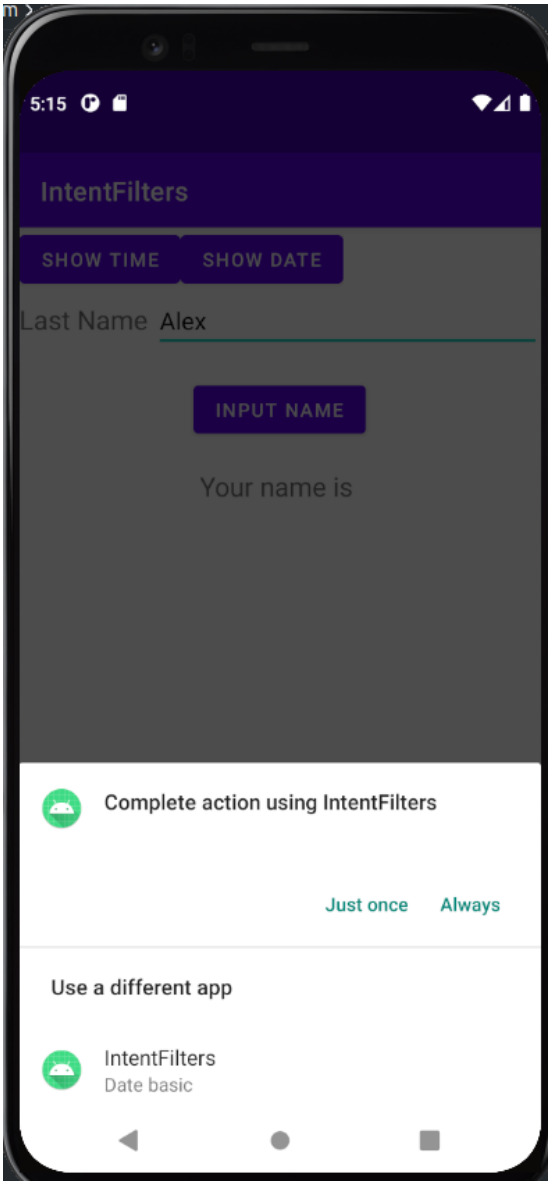
При нажатии на 1 – Info, пользователю высвечивается текущее время, если перед этим записать информацию в EditText Last Name, то программа подаст эту информацию в InfoActivity методом putExtra для дальнейшего извлечения и обработки.

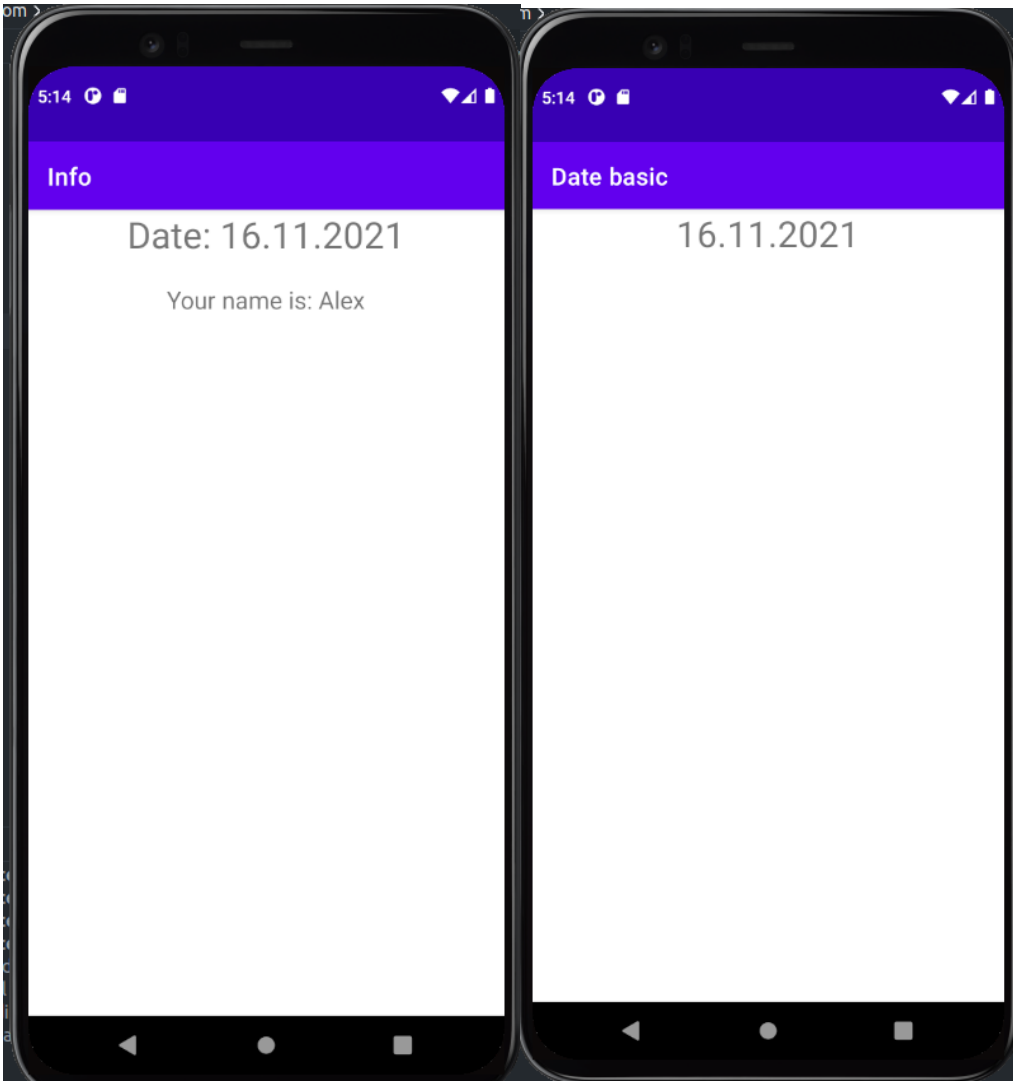


При выборе Time Basic (TimeActivity) активности выведется текущее время:

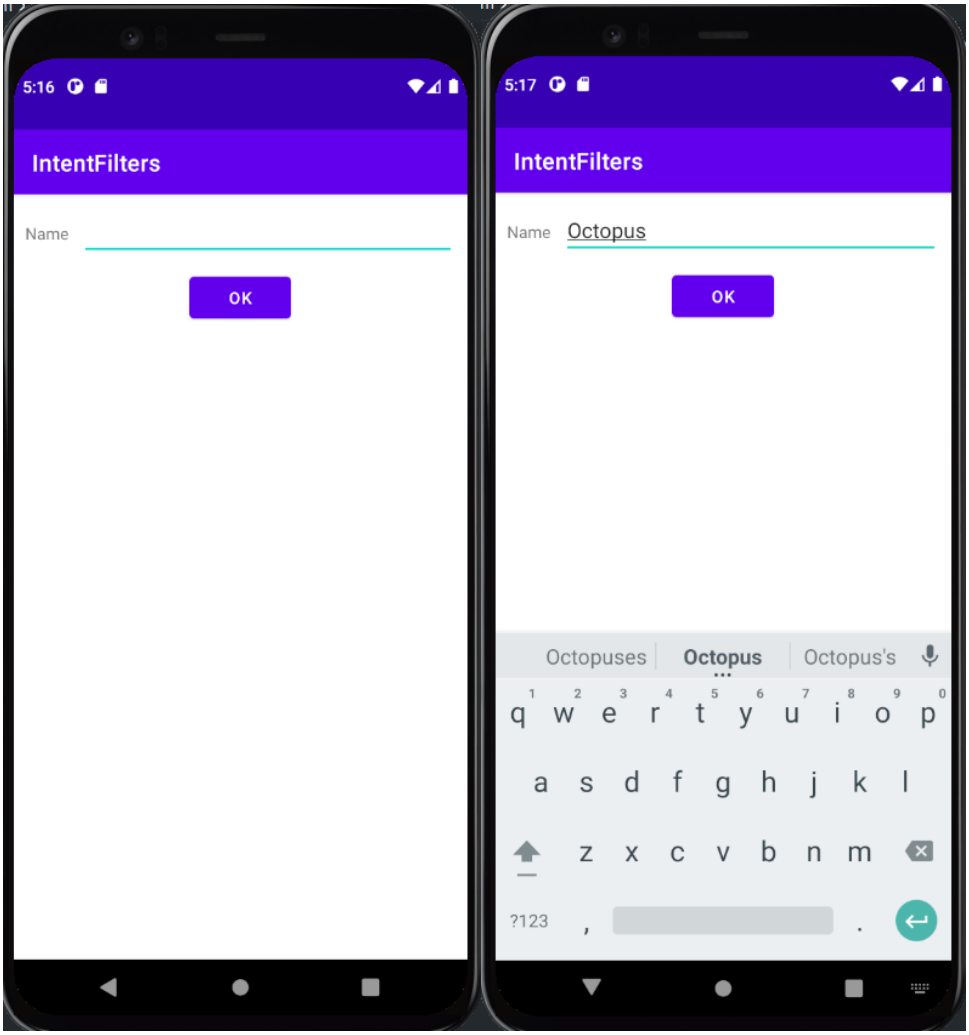


При работе с кнопкой SHOW DATE:

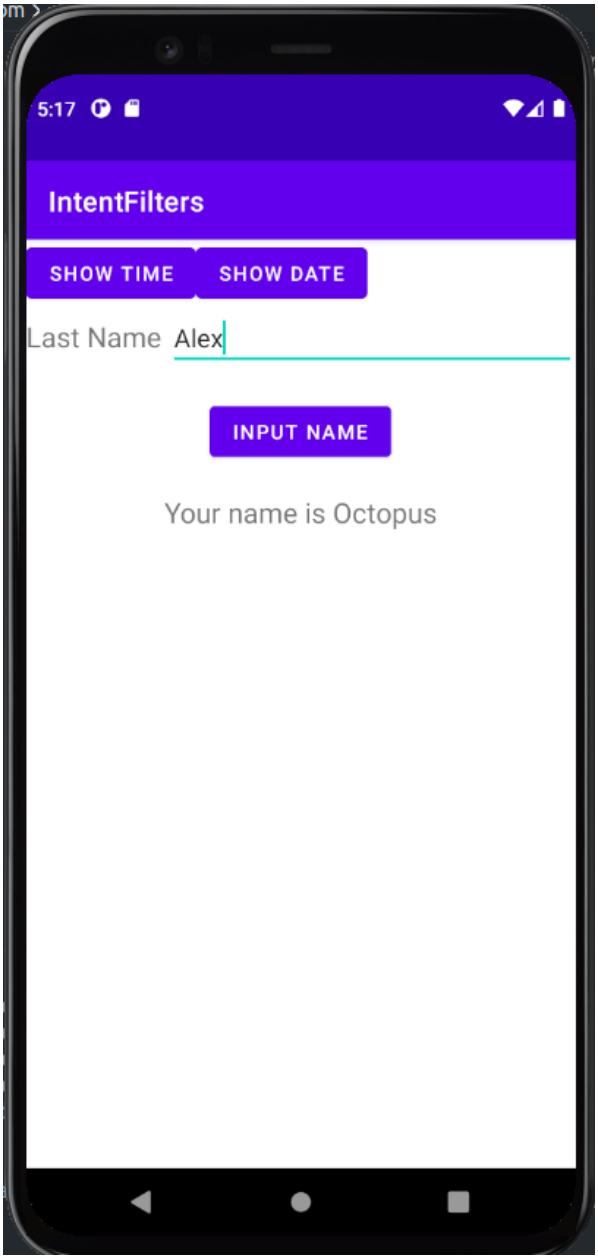

5:15
IntentFilters
SHOW TIME
SHOW DATE
Last Name
Alex
INPUT NAME
Your name is
Complete action using IntentFilters
Just once
Always
Use a different app
IntentFilters
Date basic



При нажатии на кнопку INPUT NAME в главной активности запускается новая активность NameActivity и просит пользователя ввести имя в EditText:



При нажатии кнопки OK в этой активности происходит её завершении и подача имени из поля ввода в главную активность (обратный интент). Главная активность же, в методе onActivityResult подставляет данные из getStringExtra в TextView с Your name is:



Контрольные вопросы

1. Какие существуют соглашения в порядке наименования действий?
2. Как передать информацию в активность используя неявный вызов?
3. Какие еще параметры можно задавать при создании неявного интента?
4. Зачем нужна категория в интент-фильтрах? Какие существуют категории?
5. Зачем нужен элемент <requestFocus>?
6. Зачем нужны аргументы requestCode и resultCode в обратном интенте?

1. Согласно с официальной документаций (https://developer.android.com/guide/components/intents-filters#Building)

Action

A string that specifies the generic action to perform (such as *view* or *pick*).

In the case of a broadcast intent, this is the action that took place and is being reported. The action largely determines how the rest of the intent is structured—particularly the information that is contained in the data and extras.

You can specify your own actions for use by intents within your app (or for use by other apps to invoke components in your app), but you usually specify action constants defined by the `Intent` class or other framework classes. Here are some common actions for starting an activity:

`ACTION_VIEW`

Use this action in an intent with `startActivity()` when you have some information that an activity can show to the user, such as a photo to view in a gallery app, or an address to view in a map app.

`ACTION_SEND`

Also known as the *share* intent, you should use this in an intent with `startActivity()` when you have some data that the user can share through another app, such as an email app or social sharing app.

See the `Intent` class reference for more constants that define generic actions. Other actions are defined elsewhere in the Android framework, such as in `Settings` for actions that open specific screens in the system's Settings app.

You can specify the action for an intent with `setAction()` or with an `Intent` constructor.

If you define your own actions, be sure to include your app's package name as a prefix, as shown in the following example:

Kotlin Java

```
static final String ACTION_TIMETRAVEL = "com.example.action.TIMETRAVEL";
```

принято выносить действия (Actions) в константы. Рекомендуется выбирать названия, основываясь на соглашении об именовании пакетов в Java. Действие в значительной степени определяет, как структурирована остальная часть намерения, особенно информация, содержащаяся в данных и дополнительных материалах. Такая информация позволяет явно определить совершаемое действия. В противном случае вы не можете быть уверены, какая служба ответит на намерение, и пользователь не сможет увидеть, какая служба запускается. При использовании явного вызова подаем текущий контекст и ссылаемся на

используемый класс приложения, например:

Example explicit intent

An explicit intent is one that you use to launch a specific app component, such as a particular activity or service in your app. To create an explicit intent, define the component name for the `Intent` object—all other intent properties are optional.

For example, if you built a service in your app, named `DownloadService`, designed to download a file from the web, you can start it with the following code:

Kotlin Java

```
// Executed in an Activity, so 'this' is the Context
// The fileUrl is a string URL, such as "http://www.example.com/image.png"
Intent downloadIntent = new Intent(this, DownloadService.class);
downloadIntent.setData(Uri.parse(fileUrl));
startService(downloadIntent);
```

The `Intent(Context, Class)` constructor supplies the app `Context` and the component a `Class` object. As such, this intent explicitly starts the `DownloadService` class in the app.

2. Использование неявного вызова полезно, когда ваше приложение не может выполнить действие, но другие приложения, вероятно, могут, и вы хотите, чтобы пользователь выбрал, какое приложение использовать. Например, надо запустить камеру вернуть изображение, для этого обращаемся к вызову ACTION_IMAGE_CAPTURE у модуля MediaStore. Intent captureIntent = new Intent(MediaStore.*ACTION_IMAGE_CAPTURE)* Неявные намерения — это механизм, позволяющий запрашивать анонимные компоненты приложений с помощью действий. Создавая новое неявное намерение для передачи в метод **startActivity()**, необходимо назначить действие, которое должно выполниться, а также при желании указать вспомогательный путь **URI** к тем данным, что нужно обработать. Вы также можете передать дополнительные данные в другую активность, используя параметр намерения *extras*. Пример, открыть браузер и перейти по ссылке:

```
Uri address = Uri.parse("https://" + etLink.getText().toString());
Intent openBrowser = new Intent(Intent.ACTION_VIEW, address);
startActivity(openBrowser);
```

Example implicit intent

An implicit intent specifies an action that can invoke any app on the device able to perform the action. Using an implicit intent is useful when your app cannot perform the action, but other apps probably can and you'd like the user to pick which app to use.

For example, if you have content that you want the user to share with other people, create an intent with the `ACTION_SEND` action and add extras that specify the content to share. When you call `startActivity()` with that intent, the user can pick an app through which to share the content.

Kotlin Java

```
// Create the text message with a string.
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");

// Try to invoke the intent.
try {
    startActivity(sendIntent);
} catch (ActivityNotFoundException e) {
    // Define what your app should do if no activity can handle the intent.
}
```

3. Можно задать:

a. Действие - определяет действие, которое будет выполнено. Класс Intent содержит множество констант действия. Название метода определяет ряд параметров и возвращаемое значение. Вы можете также определить собственные действия для активизации активности. В этом случае вы должны включать имя пакета приложения в качестве префикса, например com.samples.yourproject.CUSTOM_ACTiON. Действие в объекте Intent устанавливается в методе setAction() и читается методом getAction();

b. Данные - это URI данных и тип MIME для этих данных. Разные активности соединены с разными видами спецификаций данных.

c. Категория - строка, содержащая дополнительную информацию о виде компонента, который должен обработать намерение. В объект Intent можно поместить любое количество описаний категорий. Класс Intent определяет несколько констант CATEGORY, например, CATEGORY_BROWSABLE

d. Дополнения - пары ключ-значения для дополнительной информации, которую нужно поставить компоненту, обращающемуся с намерением. Например, действие ACTION_TIMEZONE_CHANGED имеет дополнение time-zone, которое идентифицирует новый часовой пояс, ACTION_HEADSET_PLUG имеет дополнение state, указывающее, включены ли наушники или отключены, а также дополнение name для типа наушников. Объект Intent имеет ряд методов put...() для вставки различных типов дополнительных данных и подобного набора

методов get...() для чтения данных. Дополнения устанавливаются и читаются как объекты Bundle с использованием методов putExtras() и getExtras();

e. Флаги - указывают системе, как запускать активность (например, какому заданию должна принадлежать активность) и как обработать это после того, как активность запустили (например, принадлежит ли она списку недавних активностей). Все флаги определены в классе Intent.

4. Категория - нужна для описания, при каких обстоятельствах должно обслуживаться действие. Использует атрибут android:name, Каждый тег intent-filter способен содержать несколько тегов category. Вы можете задать собственные категории или же брать стандартные значения, предоставляемые системой. Список категорий:
    a. ALTERNATIVE - Наличие данной категории говорит о том, что действие должно быть доступно в качестве альтернативного тому, которое выполняется по умолчанию для элемента этого типа данных. Например, если действие по умолчанию для контакта — просмотр, то в качестве альтернативы его также можно редактировать
    b. SELECTED_ALTERNATIVE - То же самое, что и ALTERNATIVE, но вместо одиночного действия с использованием утверждения намерения, которое описано выше, применяется в тех случаях, когда нужен список различных возможностей. Одной из функций фильтра намерений может стать динамическое заполнение контекстного меню с помощью действий.
    c. BROWSABLE - Говорит о том, что действие доступно из браузера. Когда намерение срабатывает в браузере, оно всегда содержит данную категорию. Если вы хотите, чтобы приложение реагировало на действия, инициированные браузером (такие как перехват ссылок на конкретный сайт), то должны добавить в его манифест категорию BROWSABLE.
    d. DEFAULT - Установите эту категорию, чтобы сделать компонент обработчиком по умолчанию для действия, выполняемого с указанным типом данных внутри Фильтра намерений. Это необходимо и для Активностей, которые запускаются с помощью явных Намерений
    e. GADGET - Наличие этой категории указывает на то, что данная активность может запускаться внутри другой активности.
    f. HOME - Устанавливая эту категорию и не указывая при этом действия, вы создаете альтернативу для стандартного домашнего экрана.
    g. LAUNCHER - Используя эту категорию, вы помещаете Активность в окно для запуска приложений.
5. Тег <requestFocus> нужен для установки фокуса на нужном компоненте, например на одном TextView из множества или на нужно EditText.
6. Аргумент requestCode при вызове обратного интента позволяет определить id запускаемого интента чтобы отличать пришедшие результаты. Аргумент resultCode позволяет понять успешно ли прошел вызов интента или нет.

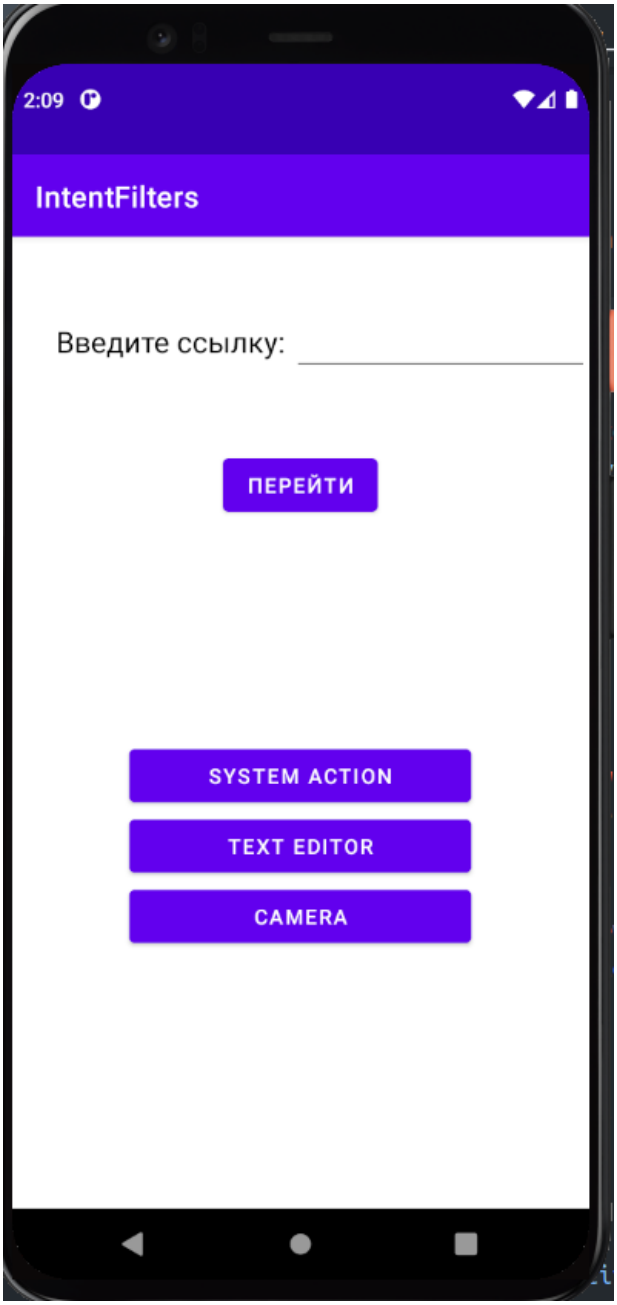
Дополнительные задания

1. Создайте приложение, на главном окне которого будет расположено поле ввода текста и при нажатии на кнопку “перейти” будет запускаться браузер по введенному пользователем адресу.
2. Создайте приложение, отвечающее на какое-либо стандартное системное действие. Проверьте его работоспособность.
3. Создайте приложение, которое выводит текстовую надпись и предлагает выбрать цвет и выравнивание надписи. Выбор должен производиться в двух разных активностях. При возврате в основную активность форматирование надписи должно меняться.
4. (*) Создайте приложение, запускающее приложение камеры. Когда пользователь делает снимок, он должен вернуться в наше приложение, и оно должно отобразить его в виде миниатюры на экране.

Репозиторий с проектом (Дополнительные задания):

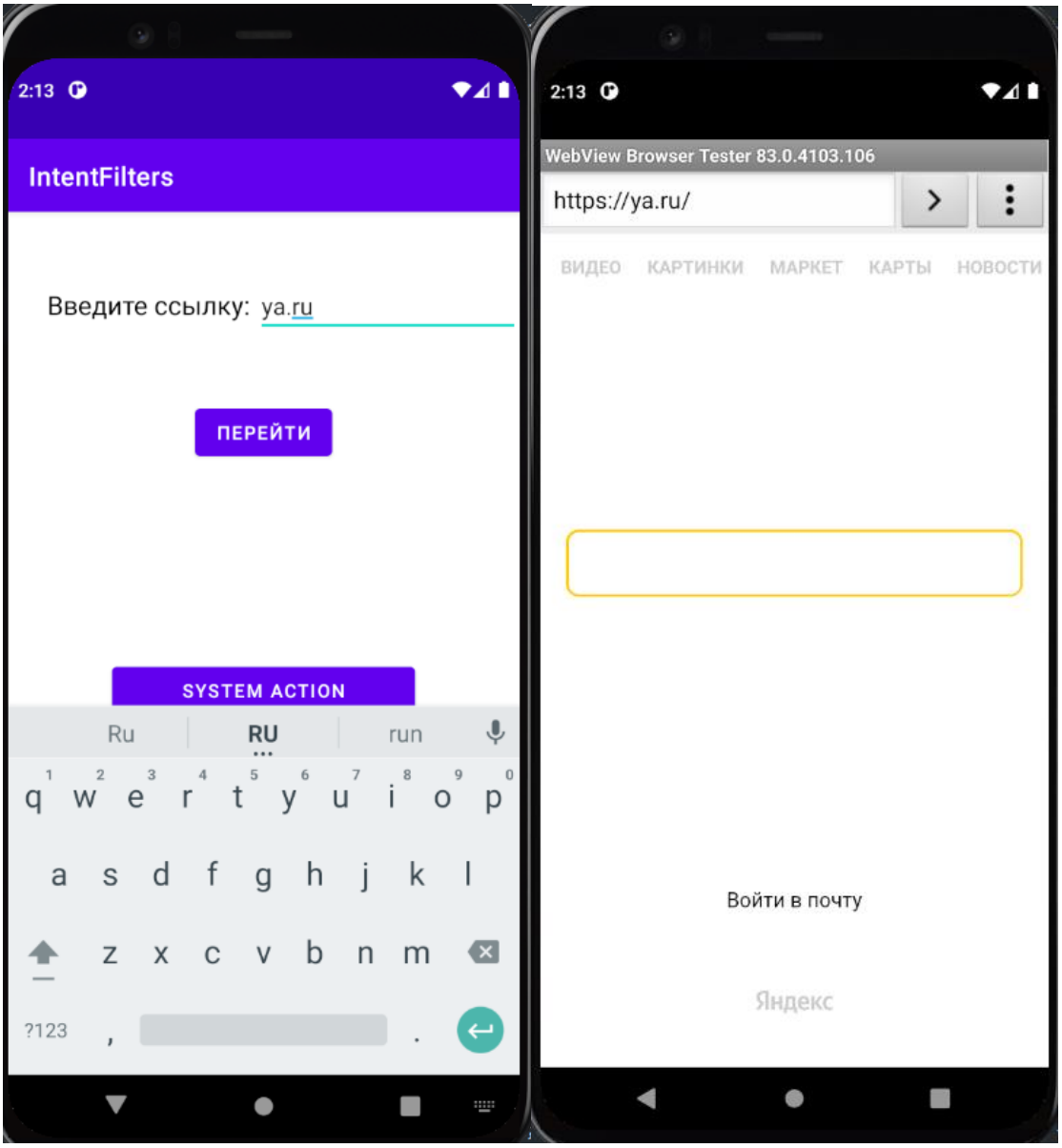
https://github.com/bitcoineazy/Android_Apps/tree/main/MD22_Additional_tasks

Работа приложения

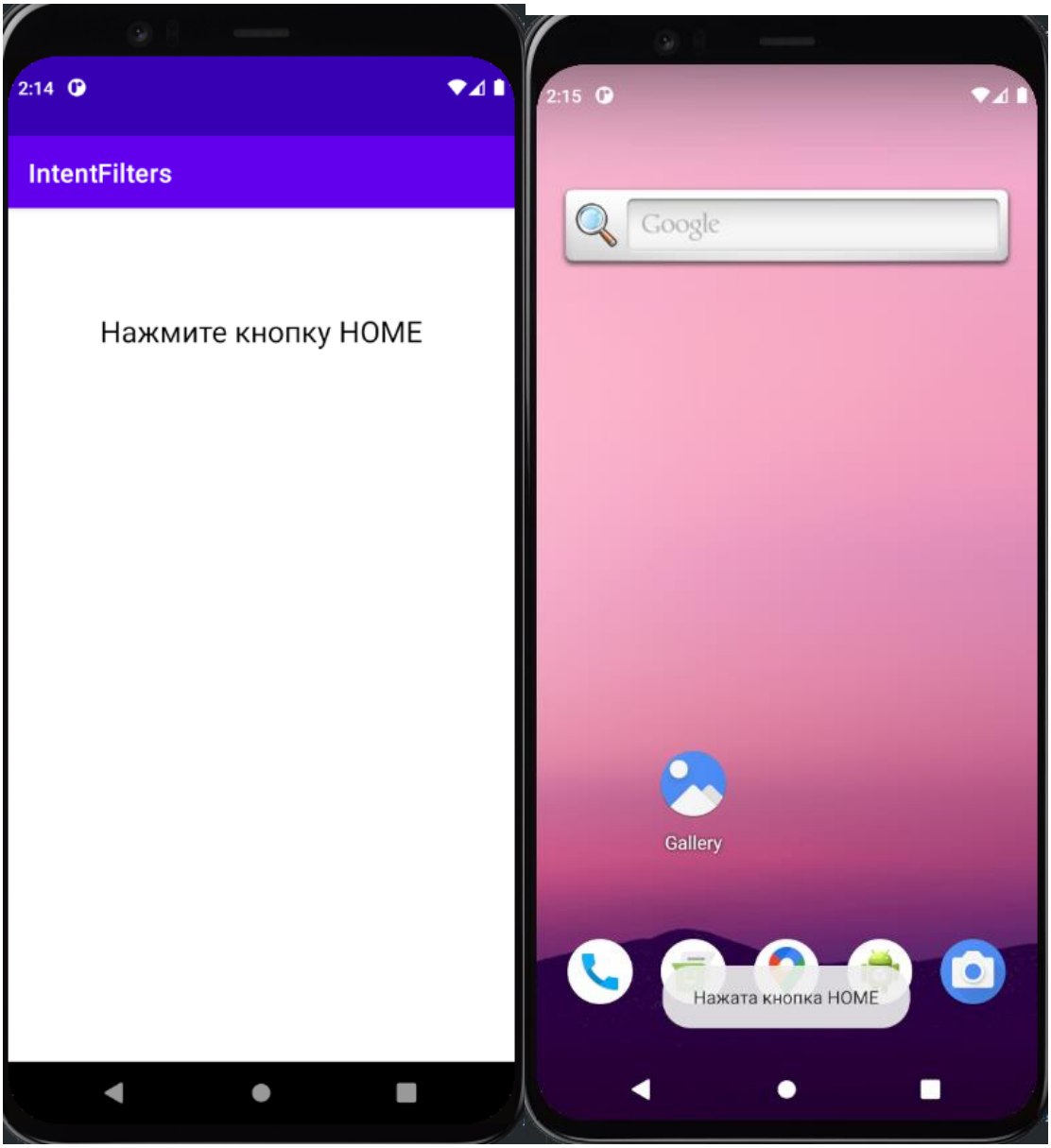


Дополнительные задания выполнены в этом приложении. На главном экране можно ввести ссылку и перейти по ней в браузер (1 доп задание), кнопки SYSTEM ACTION, TEXT EDITOR, CAMERA (2, 3, 4 доп задания).

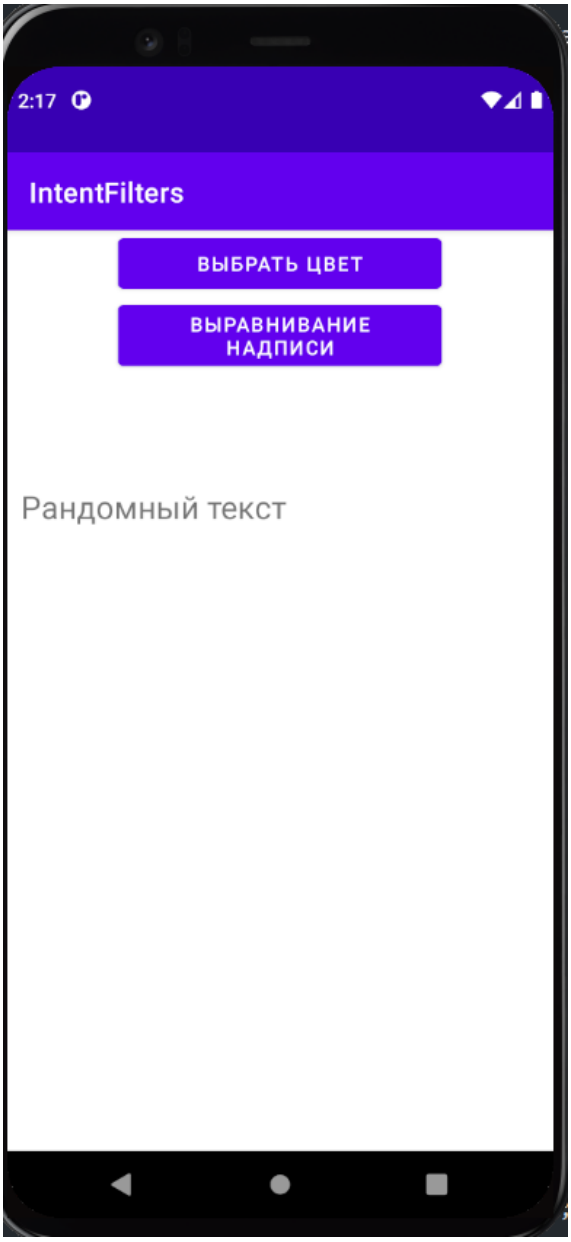
К ссылке добавляется приставка https:// и она открывается в браузере.



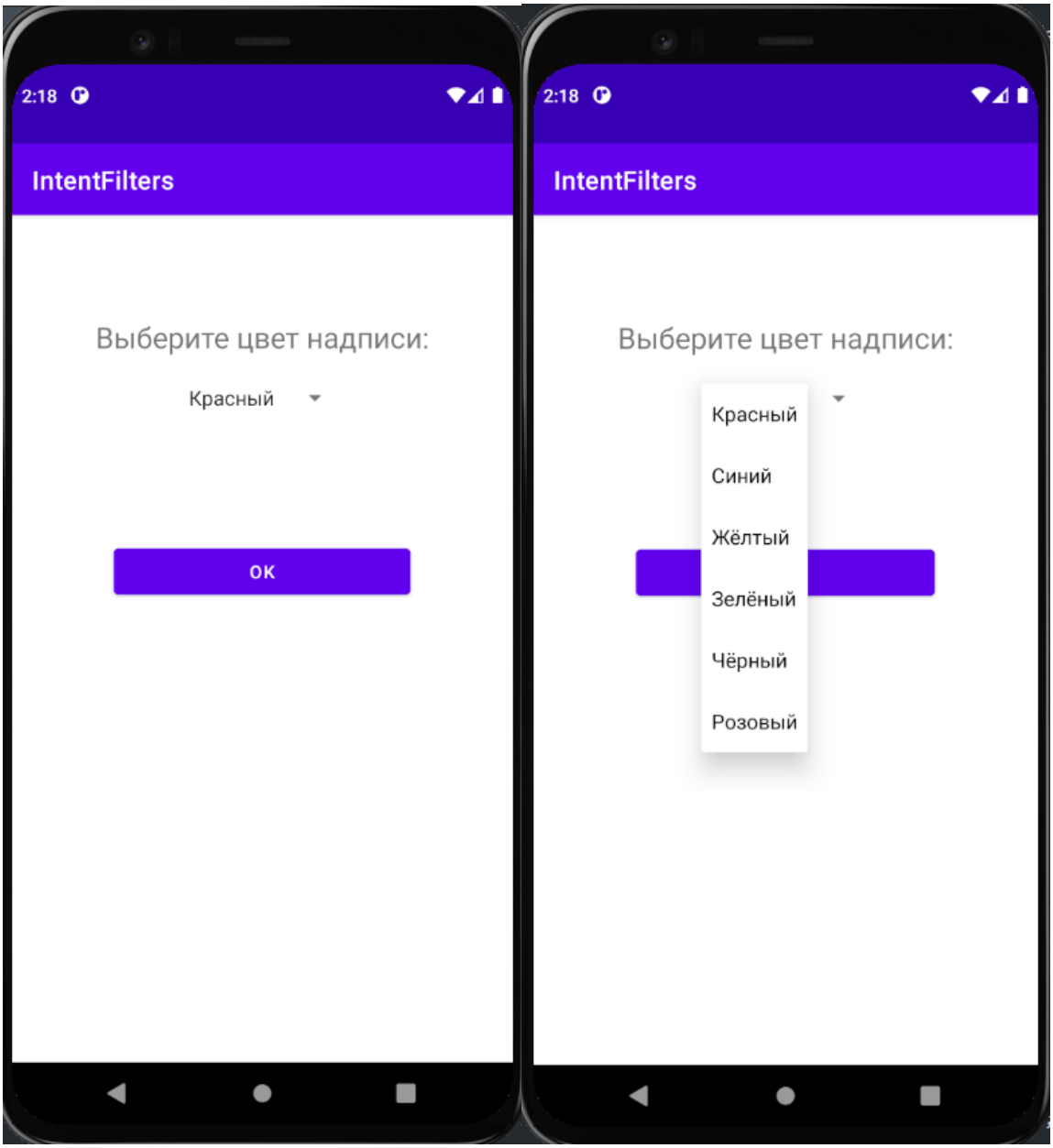
Нажав на кнопку SYSTEM ACTION, откроется активность SystemAction и появится сообщение Нажмите кнопку HOME, при нажатии на которую приложение обработает системное действие, и пользователь получит уведомления от Toast, что он совершил переход домой.



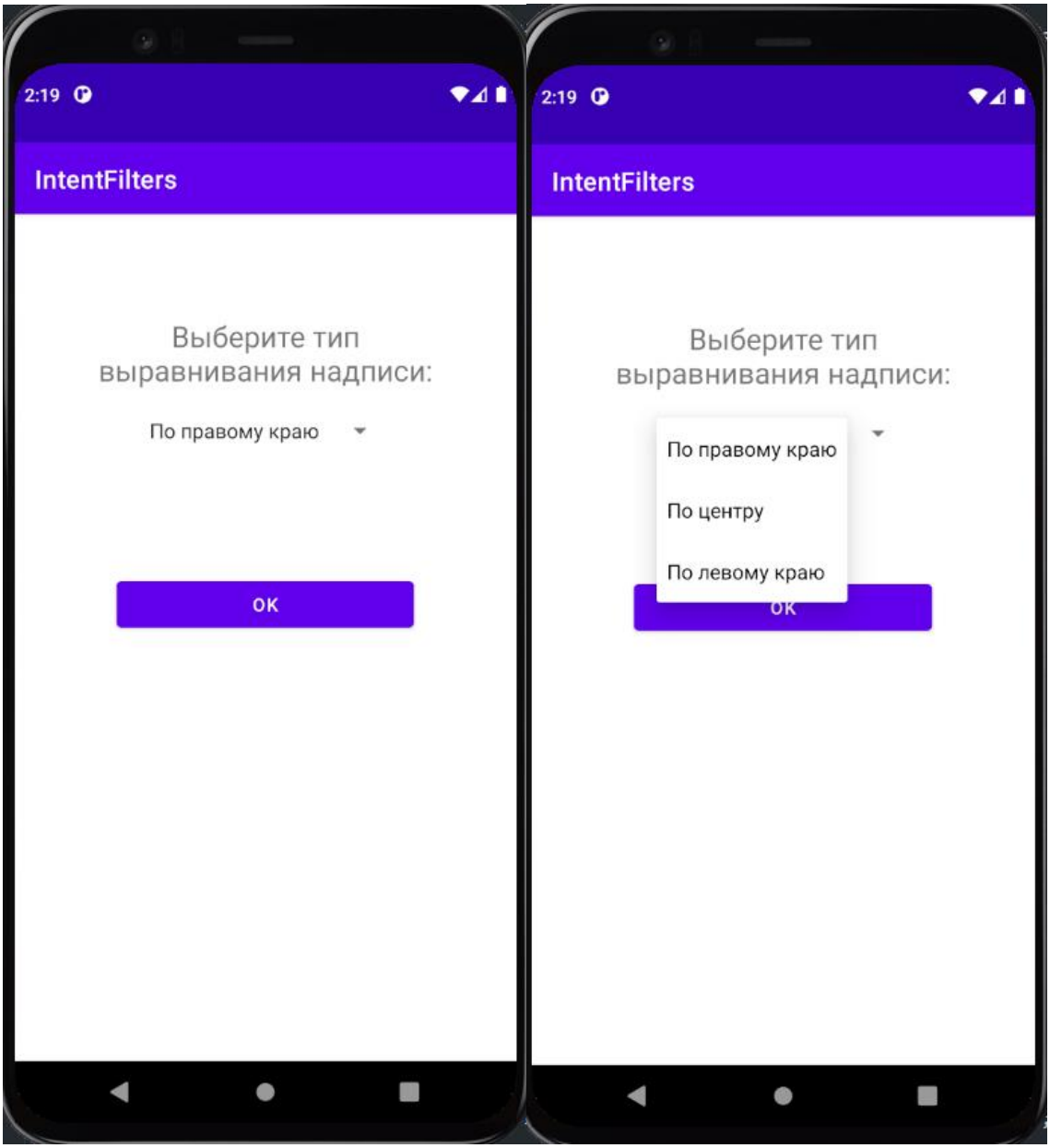
В активности TEXT EDITOR (EditorActivity) пользователь может выбрать цвет надписи и выравнивание



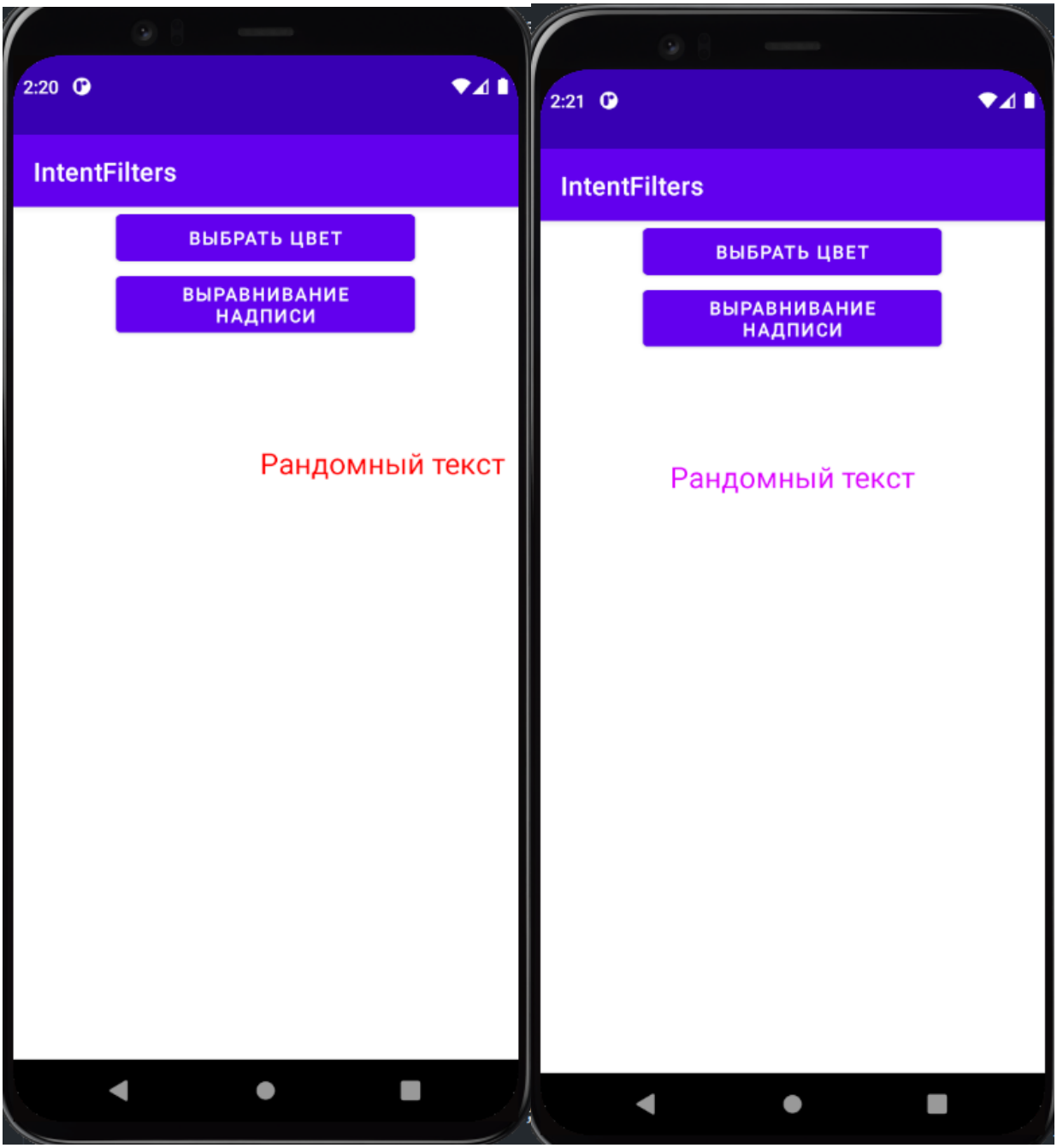
Выбор цвета, активность ColorEditor



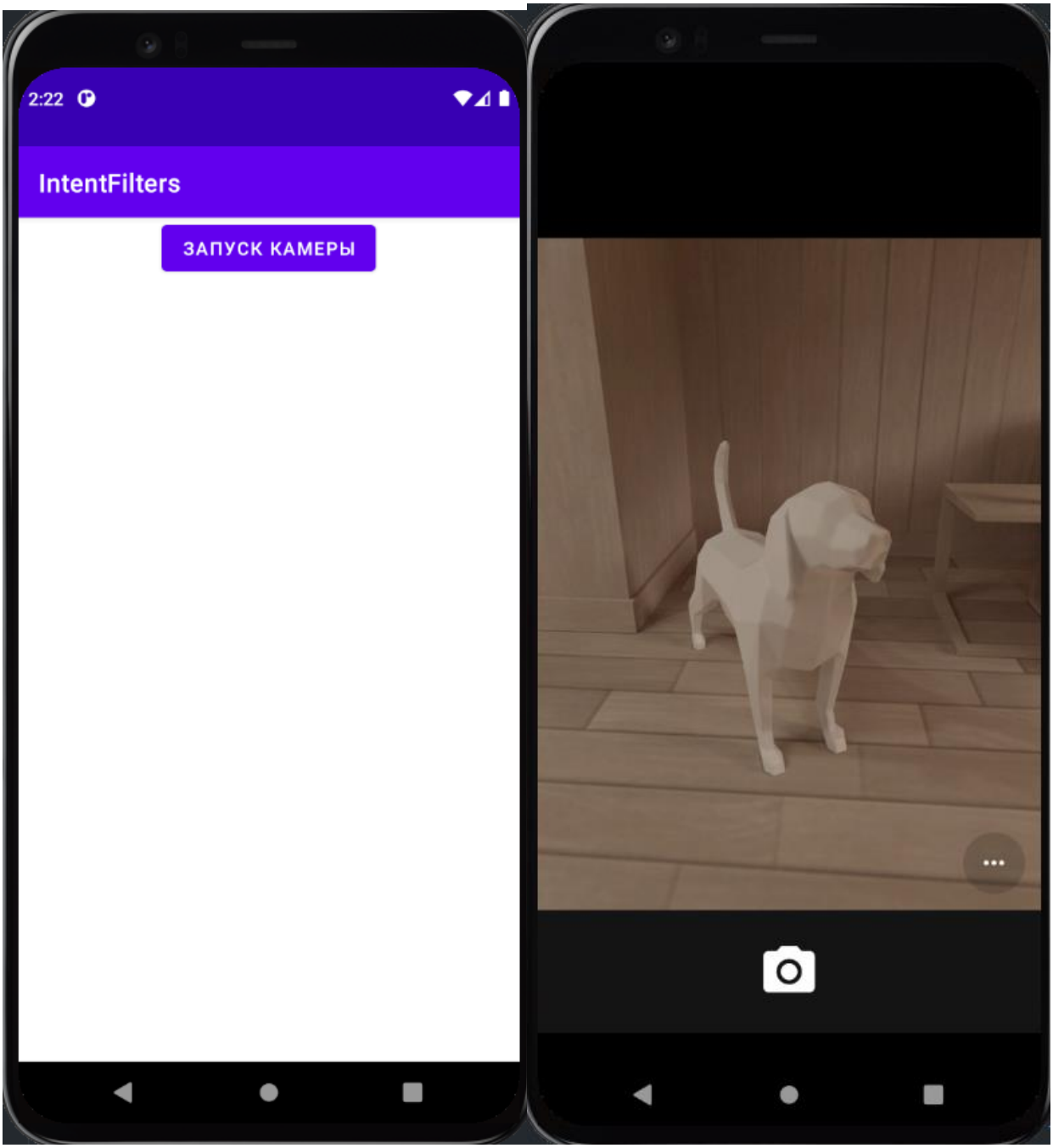
Выбор выравнивания, активность AlignmentEditor



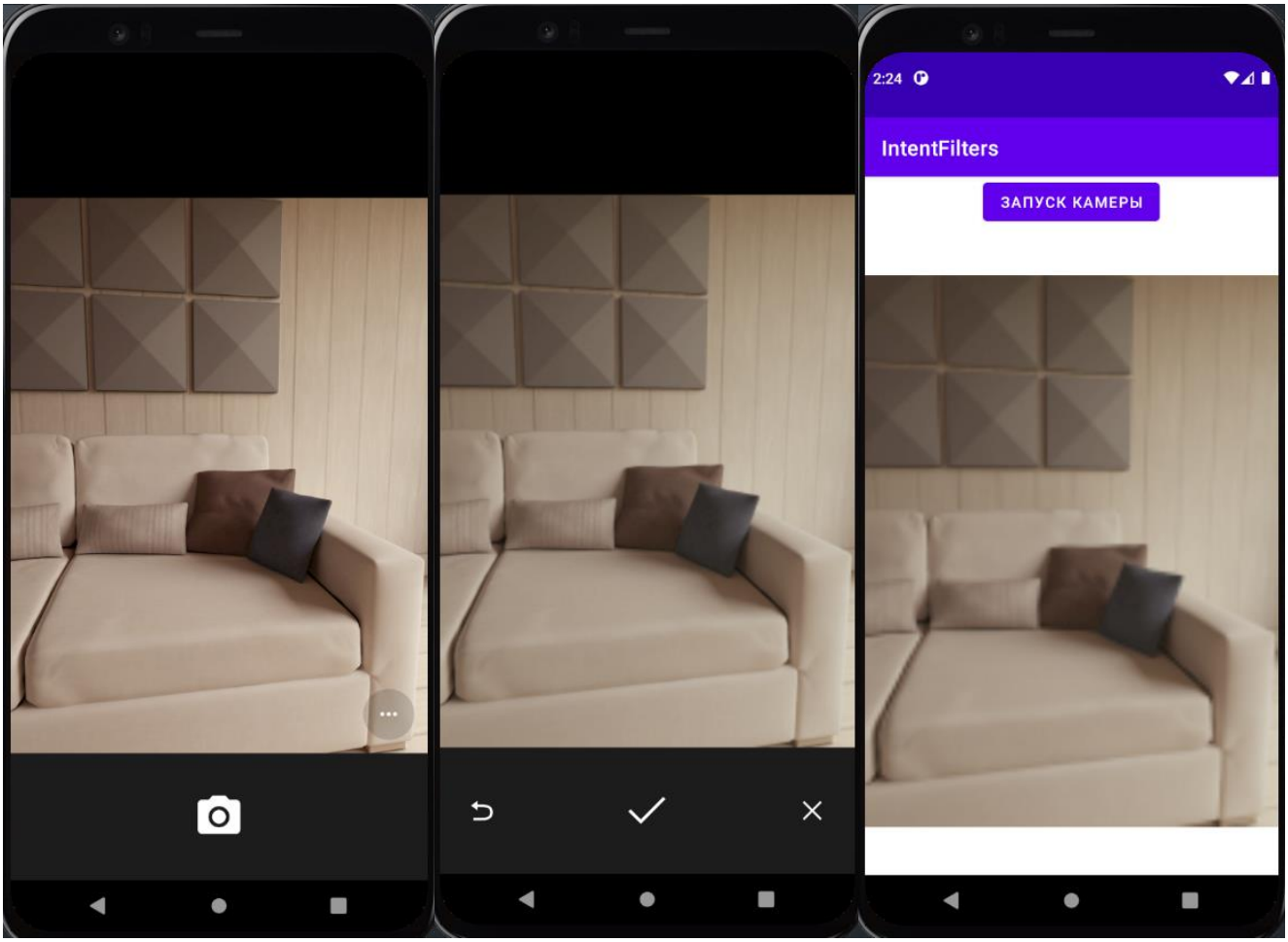
Выберем красный цвет текста и выравнивание по левому краю, а также розовый цвет и выравнивание по центру



В активности CAMERA (CameraActivity) кнопка ЗАПУСК КАМЕРЫ открывает приложение для съемки



После снимка пользователь может нажать выбрать использовать ли эту фотография и при нажатии на галочку она появится в ImageView активности



Также проверил работоспособность на своём устройстве:

CameraActivity при нажатии на кнопку вызывает обратный интент для захвата фото (с обработкой ошибки отсутствия камеры) и с помощью метода setImageBitmap редактирует используемый ресурс изображения ImageView

```
    public void onClick(View view) {
        try {
            Intent captureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
            startActivityForResult(captureIntent, CAPTURE_PHOTO);
        } catch (ActivityNotFoundException e) {
            String errorMessage = "Устройство не поддерживает фото-съёмку";
            Toast toast = Toast.makeText( context: this, errorMessage, Toast.LENGTH_SHORT);
            toast.show();
        }
    }

    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode == RESULT_OK) {
            if (requestCode == CAPTURE_PHOTO) {
                Bitmap bitmap = (Bitmap) data.getExtras().get("data");
                ivCaptured.setImageBitmap(bitmap);
            }
        }
    }
}
```