# MD2.3

# Список дел

# Работа с графическими элементами

**Цель работы**

Освоить работу с базовыми визуальными элементами Android Studio и способы организации архитектуры приложения на Android.

**Задание**

На основе лабораторных работ «MD1.1 Основные элементы» и «MD2.2 Фильтры намерений» реализуйте приложение «Список дел», состоящее из нескольких экранов. На главном экране реализована навигация по существующим категориям. При переходе на каждую категорию реализовать возможность добавления новых дел.

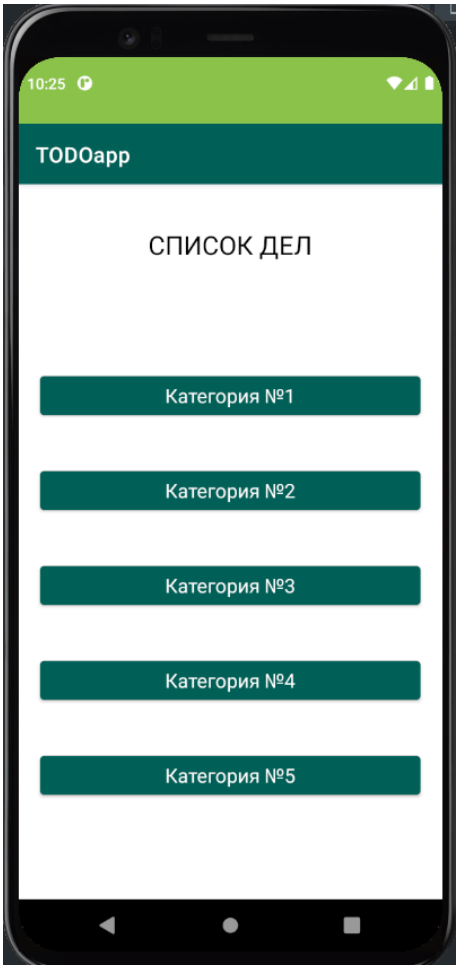
При оформлении приложения не оставляйте стандартного оформления, поменяйте цвета навигации.

Также поменяйте иконку приложения.

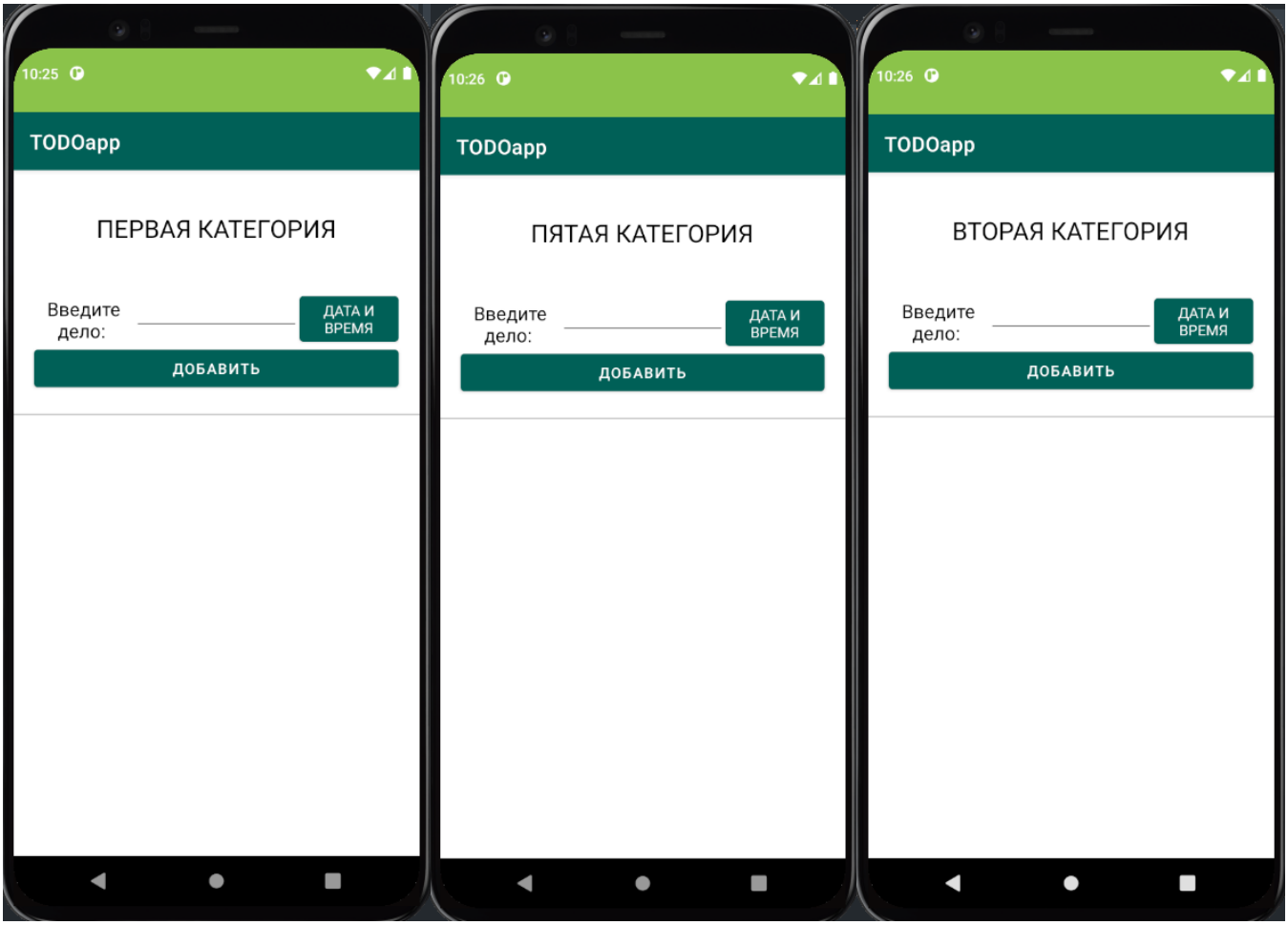
Репозиторий с проектом (Список дел):

https://github.com/bitcoineazy/Android_Apps/tree/main/MD23_TODOapp

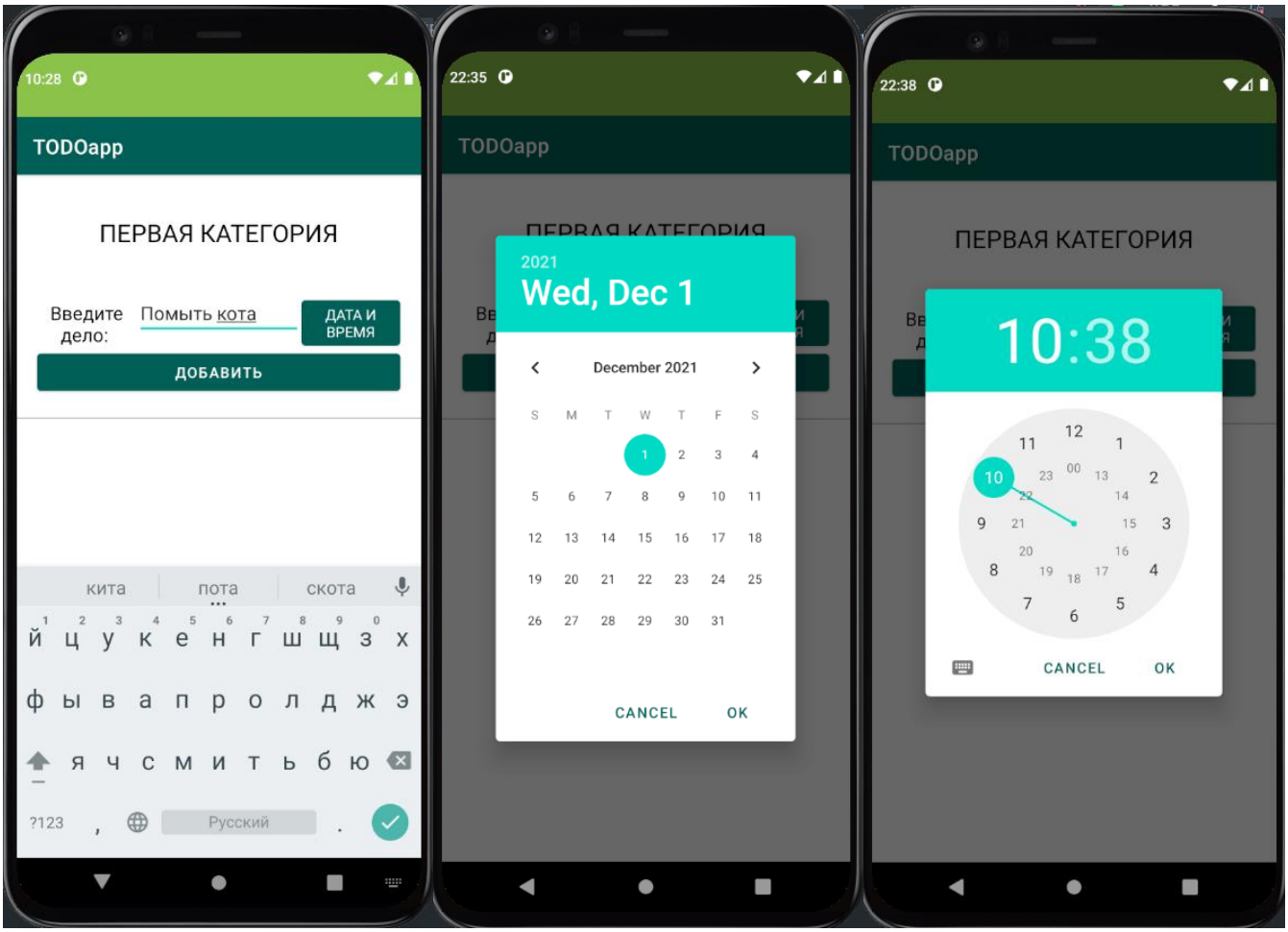
Работа приложения

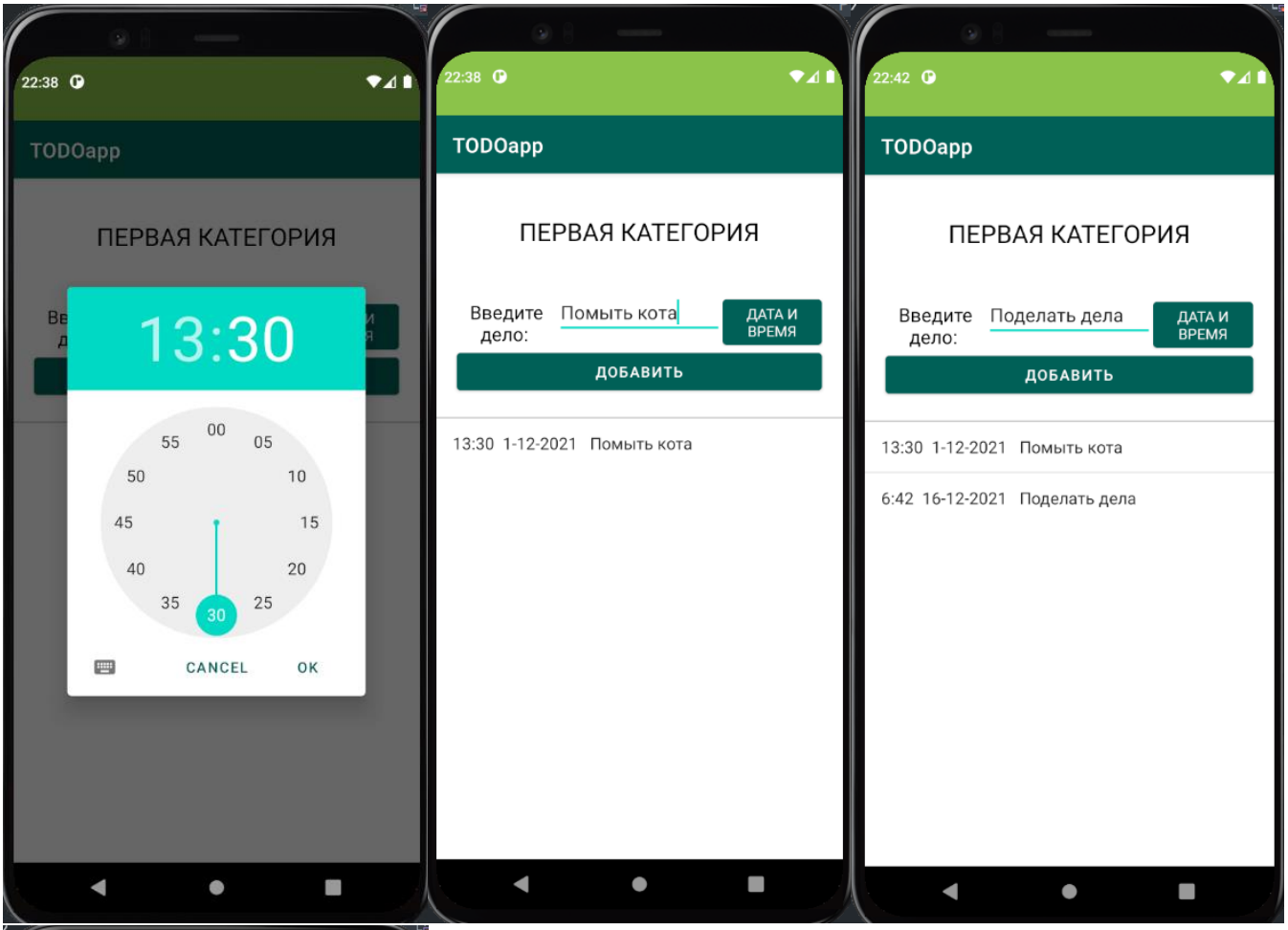


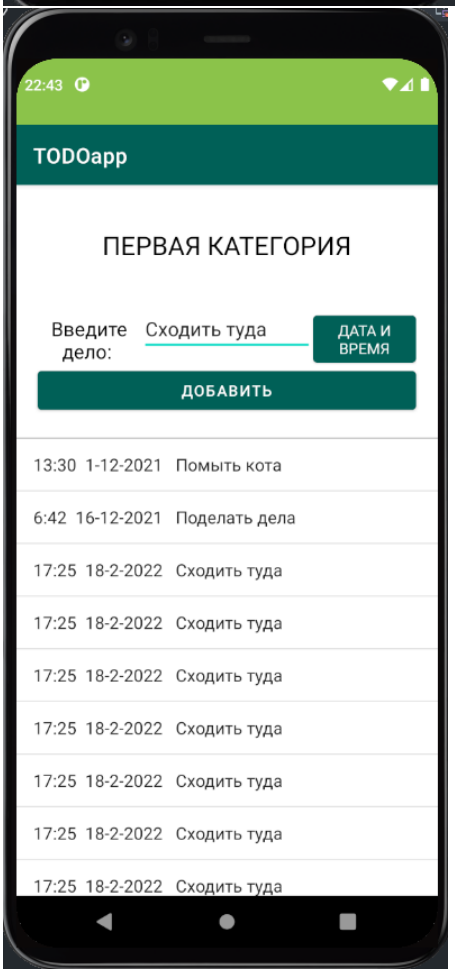

На главном экране приложения реализована навигация по категориям: 1-5



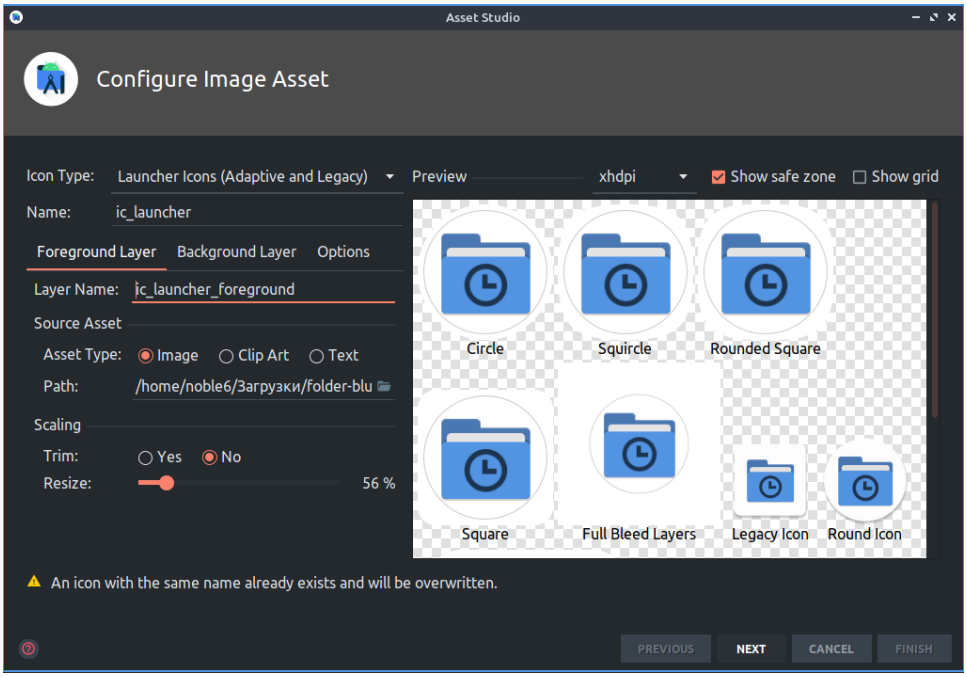
При вводе дела в EditText и выбора даты и времени можно нажать на кнопку ДОБАВИТЬ чтобы запись попала в ListView






Иконка приложения выглядит так, стандартная тема оформления также изменена




## Контрольные вопросы

1. Что такое событийно-ориентированное программирование?
2. Какие события вы использовали в своем приложении?
3. С помощью какого класса предоставляется доступ к ресурсам из кода Java?
4. Какие основные квалификаторы ресурсов вы знаете?
5. Какие основные XML-атрибуты используются для задания расположения виджета на экране?
6. Какие основные XML-атрибуты используются для задания отображения виджета на экране?
7. Какие существуют соглашения в порядке наименования действий?
8. Как передать информацию в активность используя неявный вызов?
9. Какие еще параметры можно задавать при создании неявного интента?
10. Зачем нужна категория в интент-фильтрах? Какие существуют категории?
11. Зачем нужен элемент <requestFocus>?
12. Зачем нужны аргументы requestCode и resultCode в обратном интенте?
13. Зачем делить приложение на несколько окно? Почему нельзя использовать разные расположения?

14. Что такое интент и зачем он нужен?
15. Как вызвать определенное окно своего приложение? А другого?
16. Что такое таск? Почему при перемещении между окнами работает кнопка “Назад”?

1. Событийно-ориентированное программирование — это способ построение проекта из двух частей: выборка событий и обработка событий
2. MainActivity: OnCreate, OnClick, Category: OnCreate, OnClick, onDateSet, onTimeSet
3. С помощью класса R
4. Orientation, Language and region, Layout Direction, DPI, API level, Screen size
5. layout_margin, layout_gravity, layout_x, layout_y
6. layout_weight — вес элемента относительно других, text, layout_width -ширина элемента, layout_height — высота
7. 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.

8. 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.
}
```

9. 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.

10. 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.

11. Тег <requestFocus> нужен для установки фокуса на нужном компоненте, например на одном TextView из множества или на нужном EditText.
12. 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.
13. Для распределения функционала, многозадачности, удобства. Если использовать разные расположения, то большой проект и весь функционал придется уместить в одну активность и layout, чтобы найти информацию или изменения в активности придется очень долго скроллить экран, проект также теряет возможность обработки множества событий.
14. Интент - Намерение - это абстрактное описание операции, которую необходимо выполнить. Это механизм для описания одного действия. Примеры: открыть камеру, перейти по ссылке в браузер, вызвать другую активность. Намерение предоставляет средство для выполнения поздней привязки во время выполнения между кодом в разных приложениях. Его наиболее важное применение заключается в запуске мероприятий, где его можно рассматривать как связующее звено между мероприятиями. По сути, это пассивная структура данных, содержащая абстрактное описание выполняемого действия. https://developer.android.com/reference/android/content/Intent
15. Вызвать определенное окно своего приложения можно с помощью метода startActivity, аргументом которого является интент (намерение) с нужной нам активностью. Вызвать окно другого приложения, можно с помощью интеграции в текущий проект и вышеописанных действиях, при помощи использования сторонних библиотек или при помощи подключения к стороннему приложению и использования его ресурсов API.
16. Task - это группа активностей, с помощью которых пользователь выполняет определенную операцию. Это стэк истории активностей, запущенных пользователем. Ориентируясь на Task программа понимает в какую активность вернуть пользователя при нажатии кнопки “Назад” или любого другого действия смены состояния.

**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

Отчёт:
https://github.com/bitcoineazy/Android_Apps/blob/main/MD22_Intent_filters/Туголуков_Матвей_MD2.2.pdf