YALOVA UNIVERSITY

Android Programming with Huawei Mobile Services

Berk Ozyurt
Mehmet Yozgatli
Cengiz Toru

W4: ANDROID

- Android Platform Architecture
- App Components
- The Manifest
- App Resources
- Activity Lifecycle

HUAWEI

Android

Fundamentals

- App Components
  - Activities,
  - Services
  - Broadcast Receivers
  - Content Providers
- Manifest File
  - Permisions
- App Resources

Activity Lifecycle

- onCreate()
- onStart()
- onResume()
- onPause()
- onStop()
- onDestroy()

First App

- Lets make our first app for android.

Check official documents of Android;

developer.android.com/docs

App Components

App Components

App components are **the essential building blocks** of an Android app. Each component is an entry point through which the system or a user can enter your app. Some components depend on others.

There are four different types of app components:

1. Activities
2. Services
3. Broadcast receivers
4. Content providers

1. Activities

An activity is the entry point for interacting with the user. It represents a single screen with a user interface. For example, an email app might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails.

2. Services

A service is a general-purpose entry point for **keeping an app running in the background** for all kinds of reasons. It is a component that runs in the background to perform long-running operations or to perform work for remote processes.

A service **does not provide** a user interface.

3. Broadcast Receiver

A broadcast receiver is a component that enables the system to **deliver events** to the app outside of a regular user flow, allowing the app to respond to system-wide broadcast announcements. Because broadcast receivers are another well-defined entry into the app.

The system can deliver broadcasts **even to apps that aren't currently running.**

Many broadcasts originate from the system—for example, a broadcast announcing that the **screen has turned off, the battery is low, or a picture was captured**.

4. Content Provider

A content provider **manages a shared set of app data** that you can store in the file system, in a SQLite database, on the web, or on any other persistent storage location that your app can access.

Through the content provider, other apps can **query or modify the data** if the content provider allows it.

For example, the Android system provides a content provider that manages the **user's contact** information. As such, any app with the proper permissions can query the content provider, such as ContactsContract.Data, to read and write information about a particular person.

The Manifest

Android Manifest File

Before the Android system can start an app component, the **system must know** that the component exists **by reading the app's manifest file**, AndroidManifest.xml.

Your app must declare all its components in this file, which must be at the root of the app project directory.The manifest does a number of things in addition;

- Identifies any **user permissions** the app requires, such as Internet access.
- Declares the **minimum API** Level required by the app,
- Declares **hardware and software features** used or required by the app,
- Declares **API libraries** the app needs to be linked

1. Declaring Component

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >
        <activity android:name="com.example.project.ExampleActivity"
                  android:label="@string/example_label" ... >
        </activity>
        ...
    </application>
</manifest>
```

- **<activity>** elements for activities.
- **<service>** elements for services.
- **<receiver>** elements for broadcast receivers.
- **<provider>** elements for content providers.

3. Declaring app requirements

```
<manifest ... >
    <uses-feature android:name="android.hardware.camera.any"
                  android:required="true" />
    <uses-sdk android:minSdkVersion="7" android:targetSdkVersion="19" />
    ...
</manifest>
```

For example, if your app requires a camera and uses APIs introduced in Android 2.1 (API Level 7), you must declare these as requirements in your manifest file as shown in the above example.

App Resources

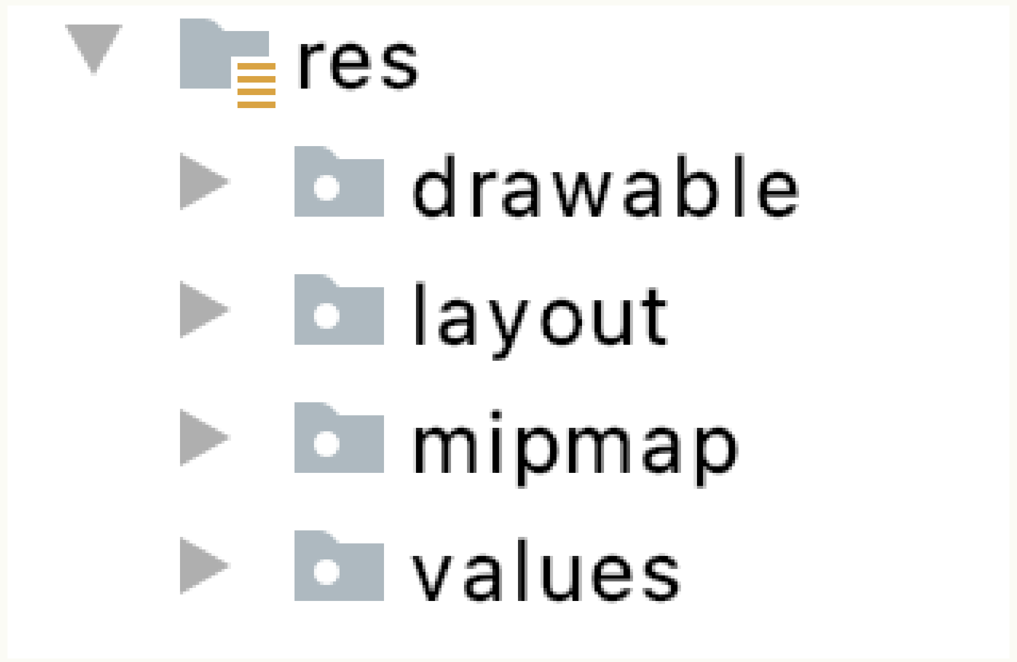
Resources

An Android app is composed of more than just code—it requires resources that are separate from the source code, such as **images, audio files, and anything relating to the visual presentation** of the app. For example, you can define animations, menus, styles, colors, and the layout of activity user interfaces with XML files.

Using app resources makes it easy to update various characteristics of your app without modifying code.

Providing sets of alternative resources enables you to **optimize your app for a variety of device configurations**, such as different languages and screen sizes.

Resources



For every resource that you include in your Android project, the SDK build **tools define a unique integer ID**, which you can use to reference the resource from your app code or from other resources defined in XML.

For example, if your app contains an image file named logo.png (saved in the res/drawable/ directory), the SDK tools generate a resource ID named **R.drawable.logo**.

Activity Lifecycle

Activity Lifecycle

As a user navigates through, out of, and back to your app, the Activity instances in your app transition through different states in their lifecycle. The Activity class provides a number of callbacks that allow the activity to know that a state has changed: that the system is creating, stopping, or resuming an activity, or destroying the process in which the activity resides.

Within the lifecycle callback methods, you can declare how your activity behaves when the user leaves and re-enters the activity.

Activity Lifecycle

Good implementation of the lifecycle callbacks can help ensure that your app avoids:

- Crashing if the user receives a phone call or switches to another app while using your app.
- Consuming valuable system resources when the user is not actively using it.
- Losing the user's progress if they leave your app and return to it at a later time.
- Crashing or losing the user's progress when the screen rotates between landscape and portrait orientation.
-

To navigate transitions between stages of the activity lifecycle, the Activity class provides a core set of six callbacks: **onCreate(), onStart(), onResume(), onPause(), onStop(), and onDestroy().** The system invokes each of these callbacks as an activity enters a new state.

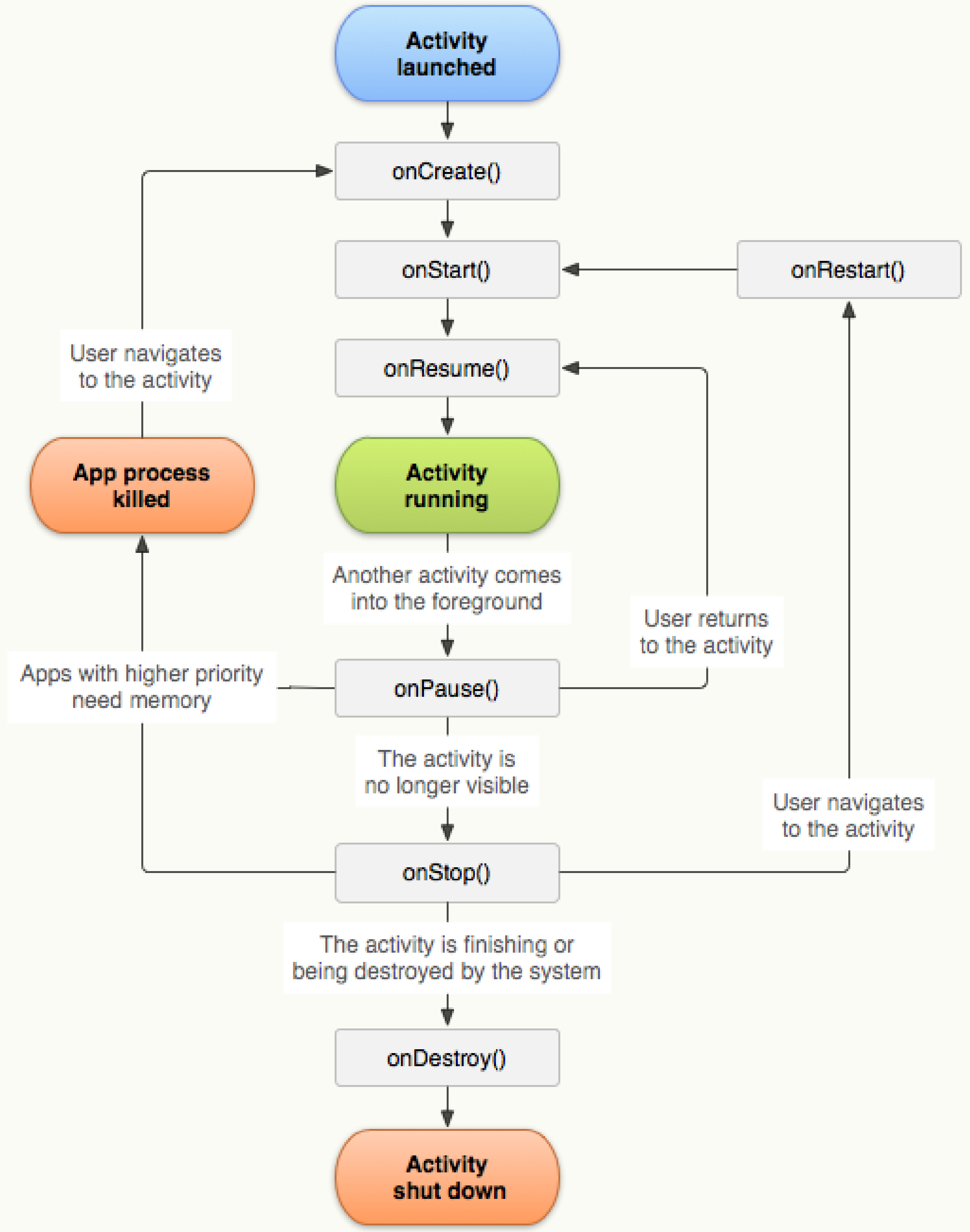
Activity Lifecycle

Good implementation of the lifecycle callbacks can help ensure that your app avoids:

- Crashing if the user receives a phone call or switches to another app while using your app.
- Consuming valuable system resources when the user is not actively using it.
- Losing the user's progress if they leave your app and return to it at a later time.
- Crashing or losing the user's progress when the screen rotates between landscape and portrait orientation.
-

To navigate transitions between stages of the activity lifecycle, the Activity class provides a core set of six callbacks: **onCreate(), onStart(), onResume(), onPause(), onStop(), and onDestroy().** The system invokes each of these callbacks as an activity enters a new state.

Activity Lifecycle


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Activity Lifecycle

onCreate()

You must implement this callback, which fires when the system first creates the activity. On activity creation, the activity enters the Created state. In the onCreate() method, you perform basic application startup logic that should happen only once for the entire life of the activity.

The following example of the onCreate() method shows fundamental setup for the activity, such as declaring the user interface (defined in an XML layout file), defining member variables, and configuring some of the UI. In this example, the XML layout file is specified by passing file's resource ID R.layout.main_activity to setContentView().

Activity Lifecycle

onStart()
When the activity enters the Started state, the system invokes this callback. The onStart() call makes the activity visible to the user, as the app prepares for the activity to enter the foreground and become interactive. For example, this method is where the app initializes the code that maintains the UI.

The onStart() method completes very quickly and, as with the Created state, the activity does not stay resident in the Started state. Once this callback finishes, the activity enters the Resumed state, and the system invokes the onResume() method.

Activity Lifecycle

onResume()
When the activity enters the Resumed state, it comes to the foreground, and then the system invokes the onResume() callback. This is the state in which the app interacts with the user. The app stays in this state until something happens to take focus away from the app. Such an event might be, for instance, receiving a phone call, the user's navigating to another activity, or the device screen's turning off.

When an interruptive event occurs, the activity enters the Paused state, and the system invokes the onPause() callback.

Activity Lifecycle

onPause()

The system calls this method as the first indication that the user is leaving your activity (though it does not always mean the activity is being destroyed); it indicates that the activity is no longer in the foreground (though it may still be visible if the user is in multi-window mode). Use the onPause() method to pause or adjust operations that should not continue (or should continue in moderation) while the Activity is in the Paused state, and that you expect to resume shortly. There are several reasons why an activity may enter this state. For example, some event interrupts app execution, as described in the onResume() section. This is the most common case.

Activity Lifecycle

onStop()

When your activity is no longer visible to the user, it has entered the Stopped state, and the system invokes the onStop() callback. This may occur, for example, when a newly launched activity covers the entire screen. The system may also call onStop() when the activity has finished running, and is about to be terminated.

Using onStop() instead of onPause() ensures that UI-related work continues, even when the user is viewing your activity in multi-window mode.

Activity Lifecycle

onDestroy()

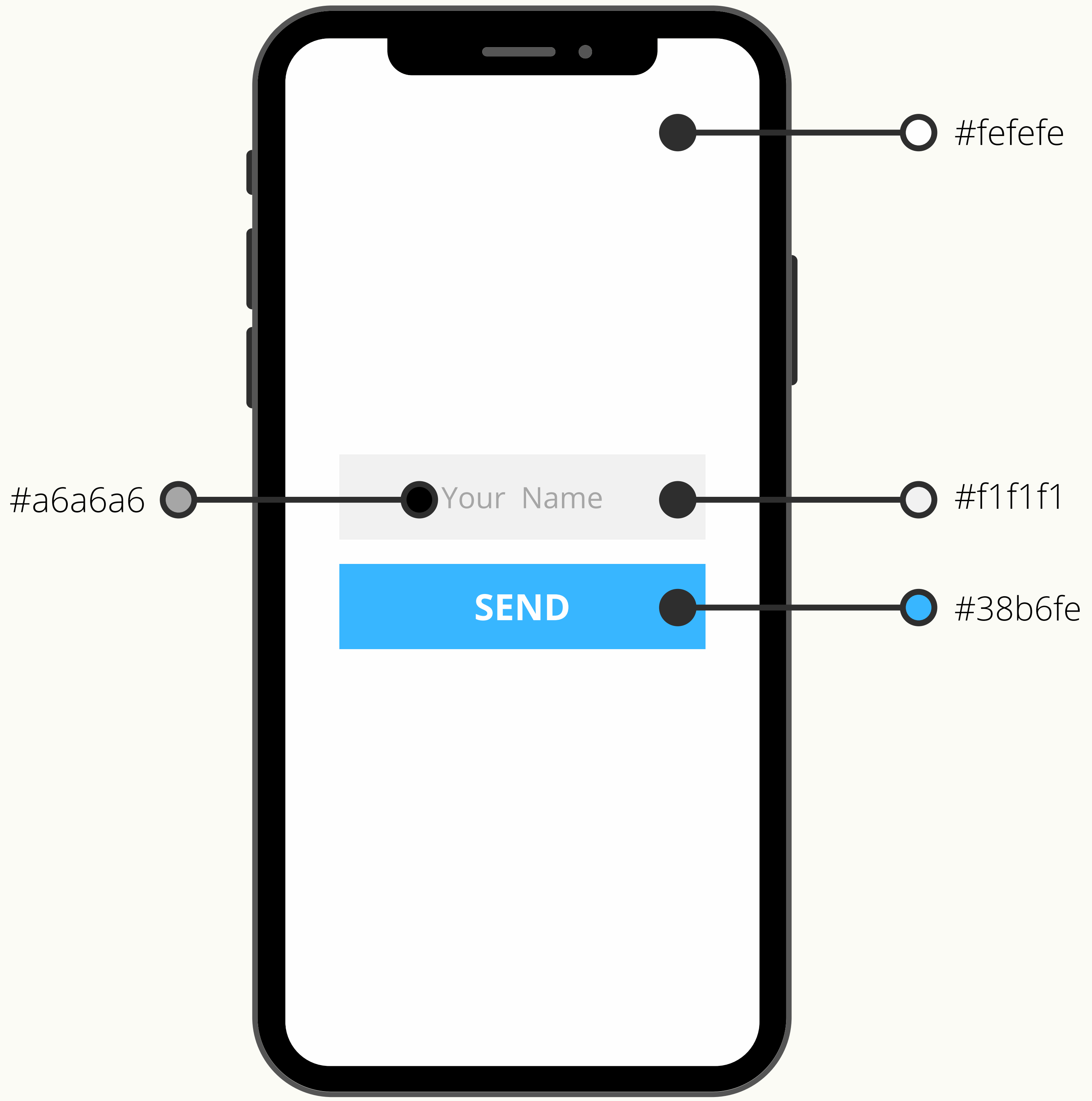
onDestroy() is called before the activity is destroyed. The system invokes this callback either because:

- the activity is finishing (due to the user completely dismissing the activity or due to finish() being called on the activity), or
- the system is temporarily destroying the activity due to a configuration change (such as device rotation or multi-window mode)

If the activity is finishing, onDestroy() is the final lifecycle callback the activity receives. If onDestroy() is called as the result of a configuration change, the system immediately creates a new activity instance and then calls onCreate() on that new instance in the new configuration.

First App

First Activity



Design

Please follow design which shows at left side.

View Components

EditText

Button

Validation Rules

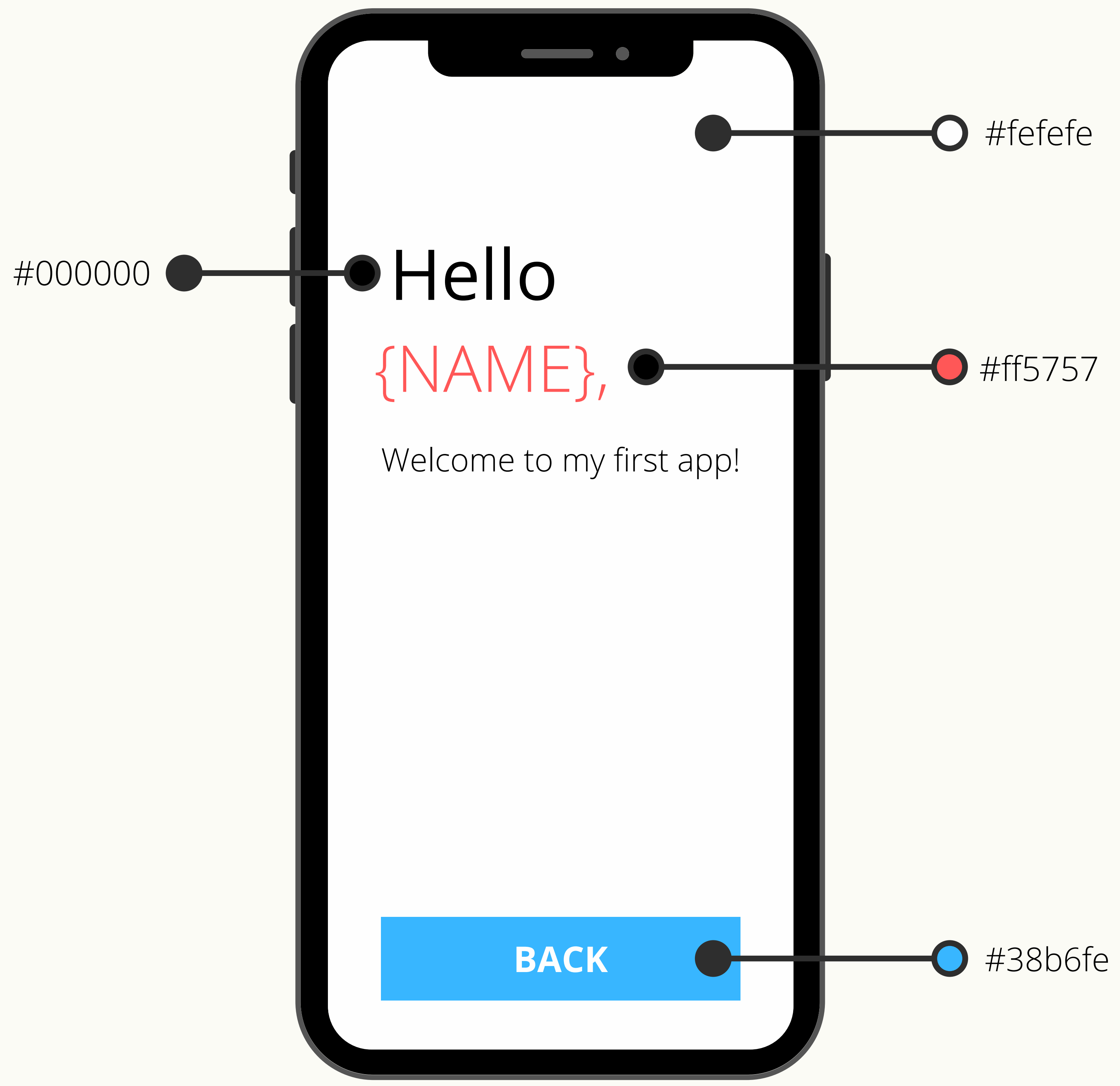
Name cannot be less than 3 characters.

Name cannot be more than 20 characters.

Actions

When user click to send button;
Check validation rules, if name is not valid, show toast message, if name is valid, open second activity and send name to second activity.

Second Activity



Design
Please follow design which shows at left side.

View Components
TextView
Button

Validation Rules
There is not any validation rules.

Actions
When user click to back button;
Second activity should be closed and first activity should be present as initial.

HUAWEI

LET'S TALK
CONTACT INFORMATION

Berk Ozyurt
berk.ozyurt1@huawei.com

Mehmet Yozgatli
mehmet.yozgatli1@huawei.com

Cengiz Toru
cengiz.toru@huawei.com

Telegram Channel
will be created

Official Website
https://developer.huawei.com