Introduction to Android

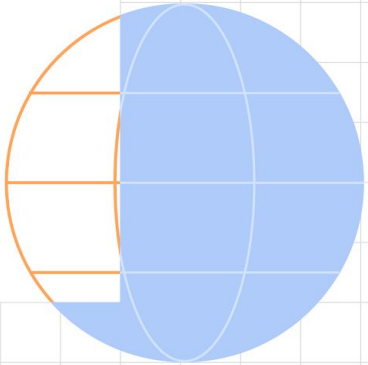
Get to know Android project basics, its structure, key components, UI and more

Contents

- AndroidStudio
- Project Structure
- Gradle build script
- Android key components
- Building UI (Compose/XML)
- Dependencies, libraries
- AndroidX, Jetpack

GDG

Android Studio



Android Studio

Essentially the only IDE you'll ever need for Android

- Built by **Jetbrains** and Google
- Impossible number of tools, features and random things you don't even know about
- Lots of quality of life things, like adding images, vectors, generating code
- **IntelliSense**

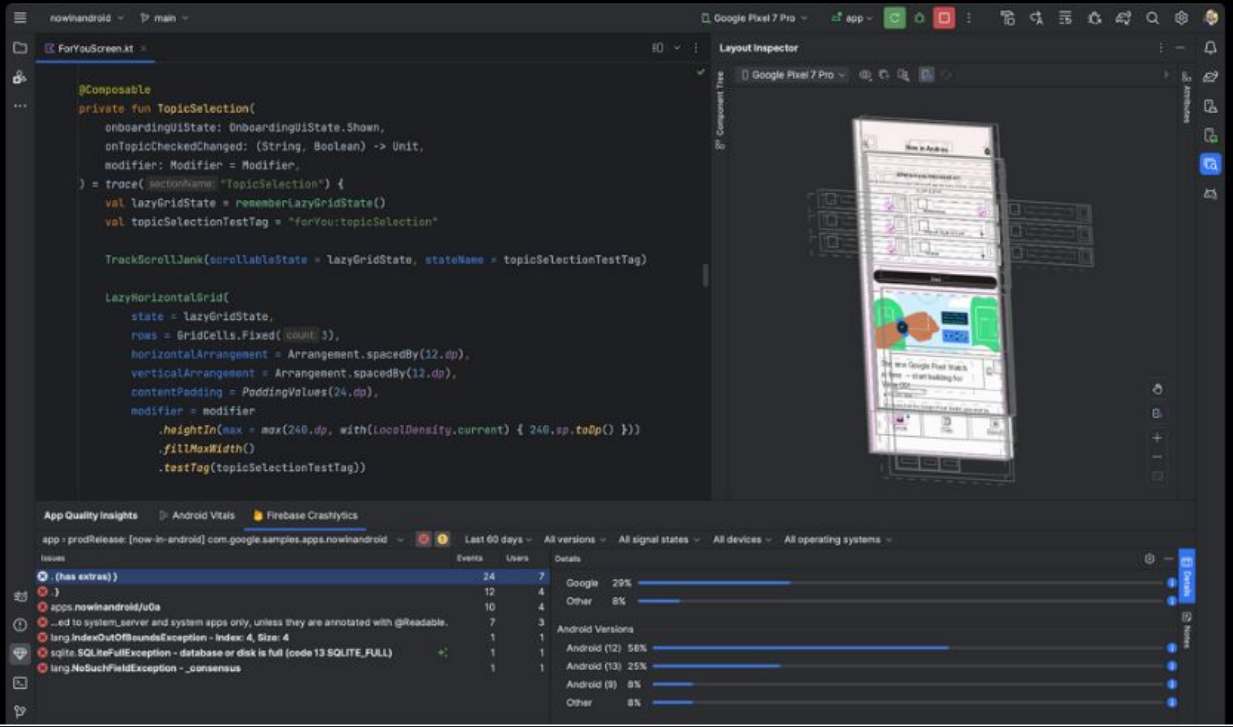
https://developer.android.com/studio

Android Studio

Get the official Integrated Development Environment (IDE) for Android app development.

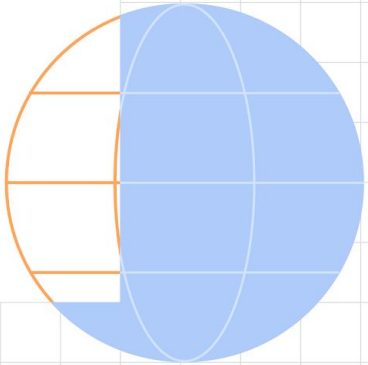
Download Android Studio Iguana

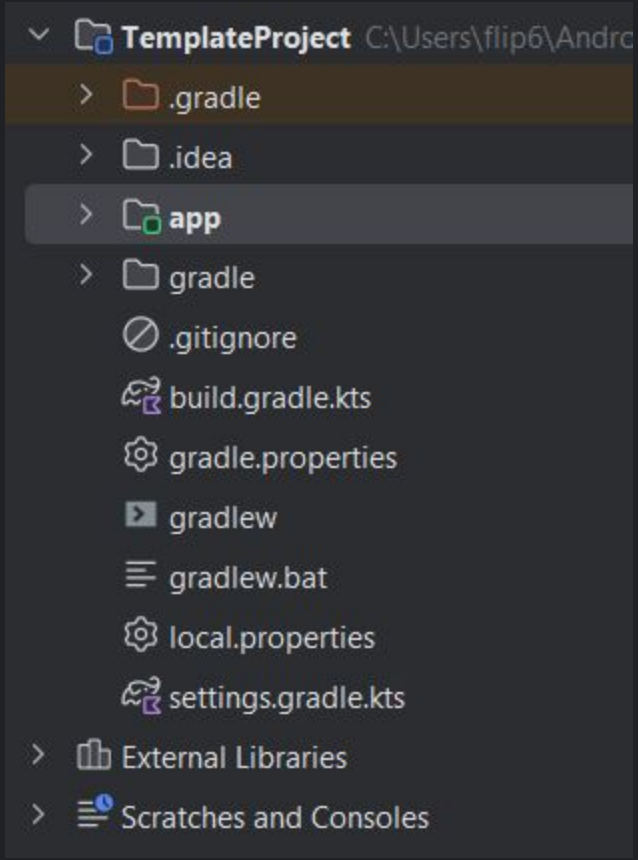
Read release notes



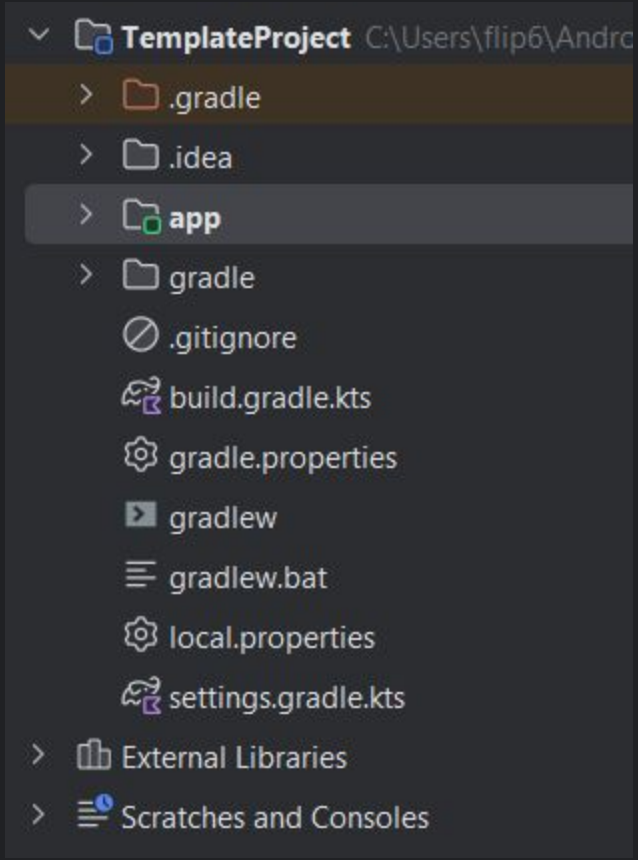
GDG

Project Structure

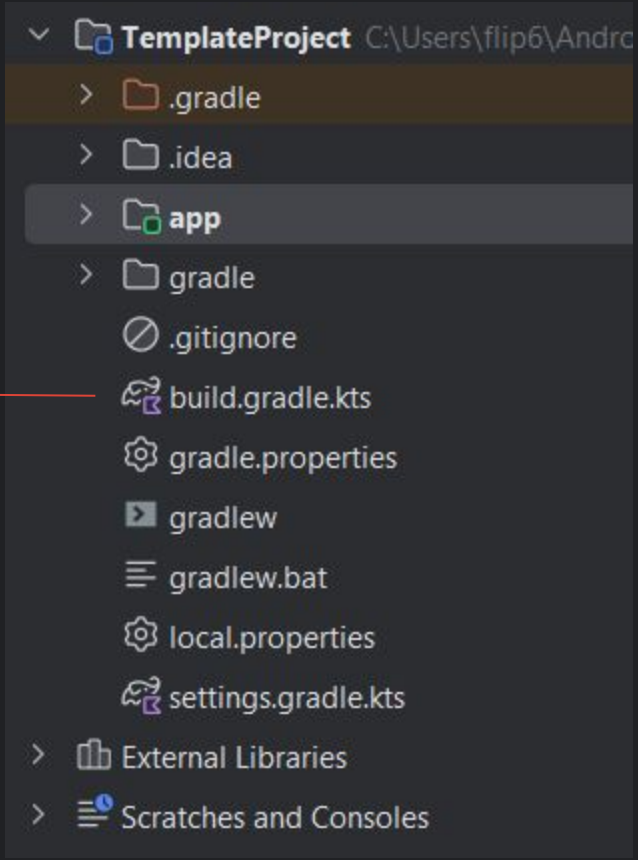


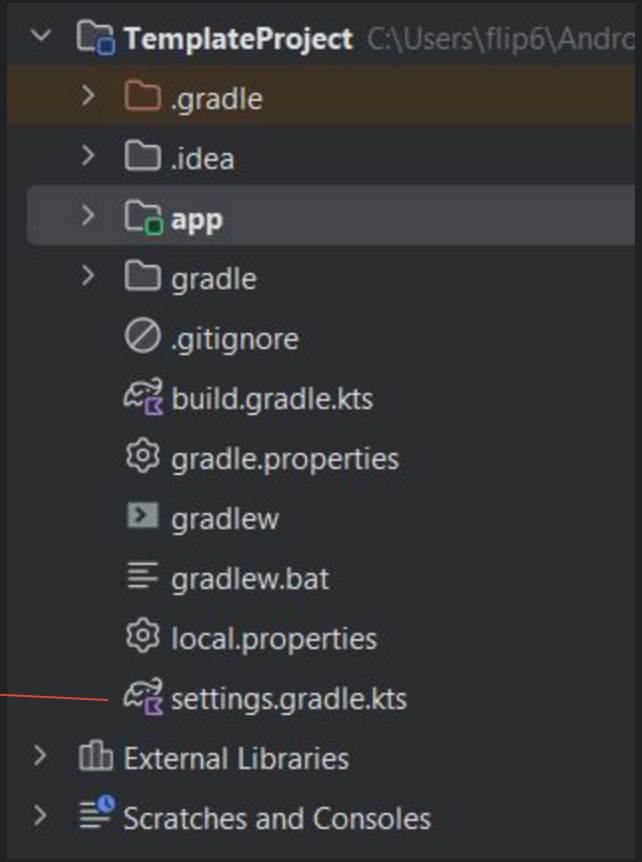

TemplateProject C:\Users\flip6\Andro
.gradle
.idea
app
gradle
.gitignore
build.gradle.kts
gradle.properties
gradlew
gradlew.bat
local.properties
settings.gradle.kts
External Libraries
Scratches and Consoles

Android Code



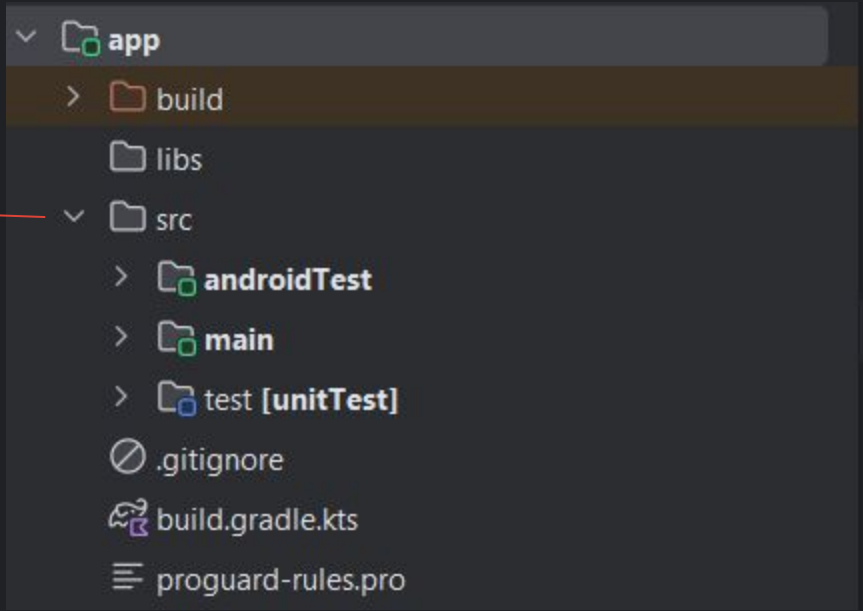
Project plugins



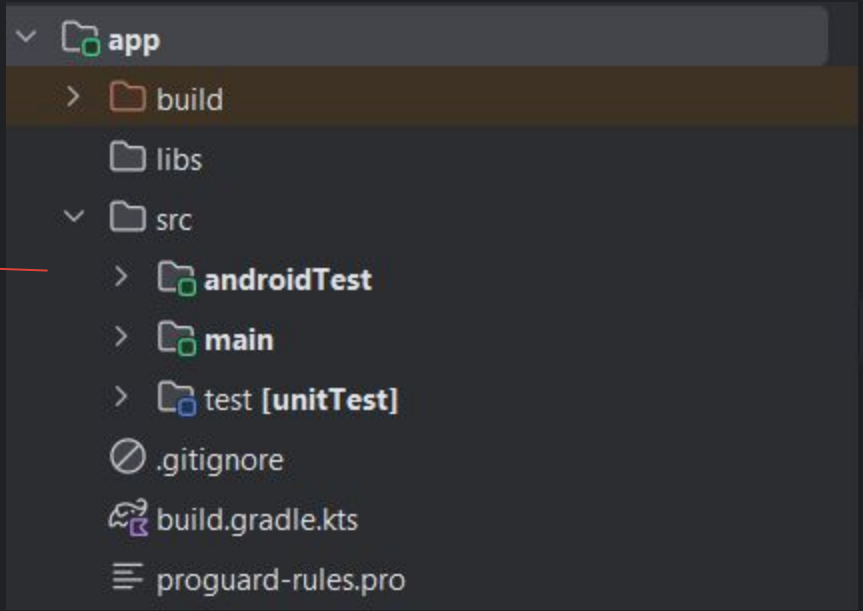


Project settings

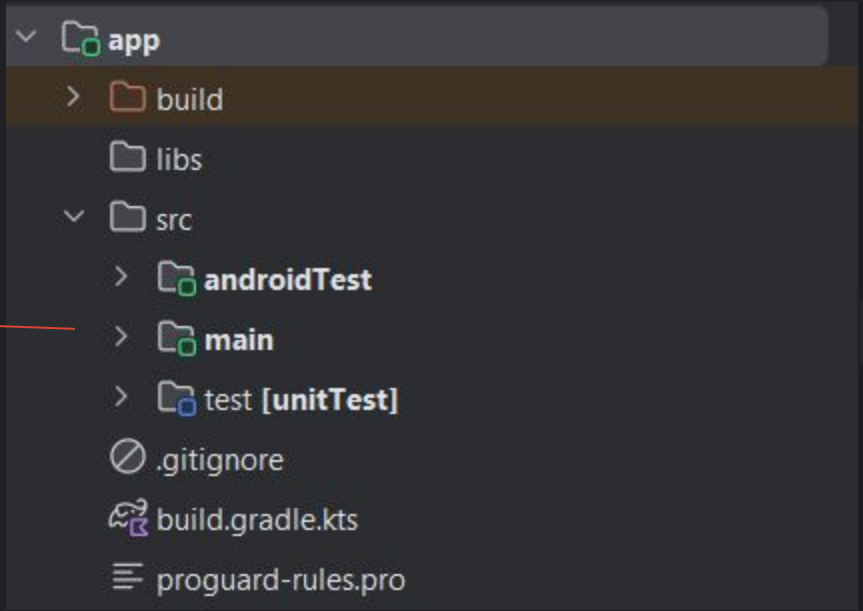
App code


app
build
libs
src
androidTest
main
test [unitTest]
.gitignore
build.gradle.kts
proguard-rules.pro

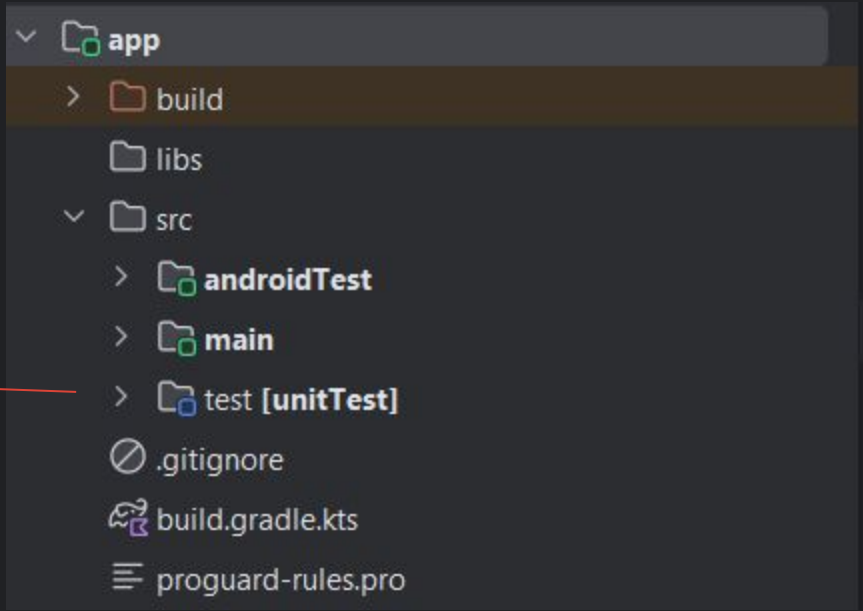
UI Tests


app
build
libs
src
androidTest
main
test [unitTest]
.gitignore
build.gradle.kts
proguard-rules.pro

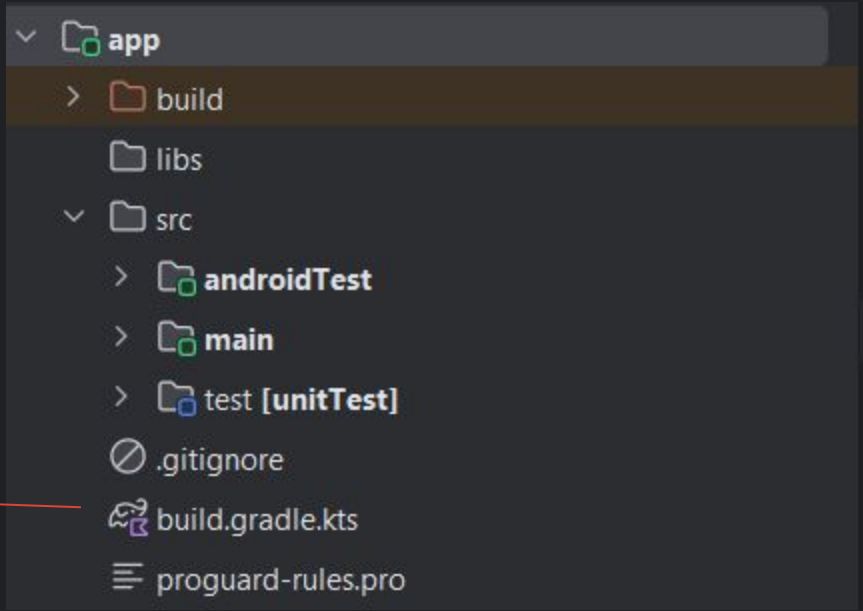
Application



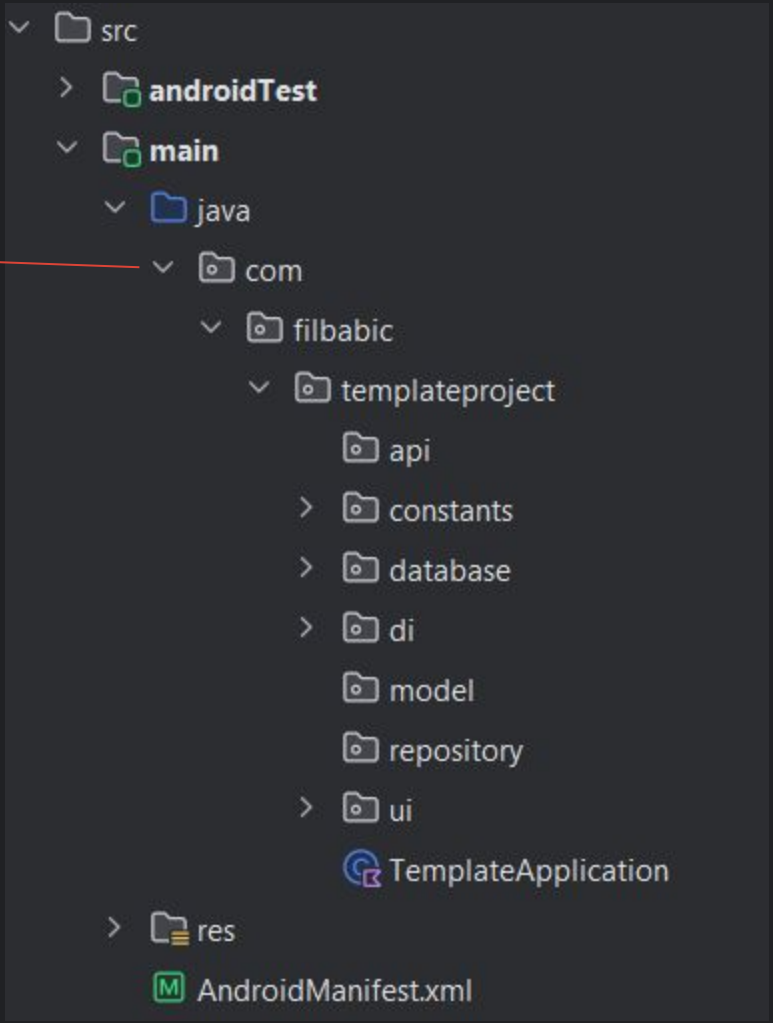
Unit tests

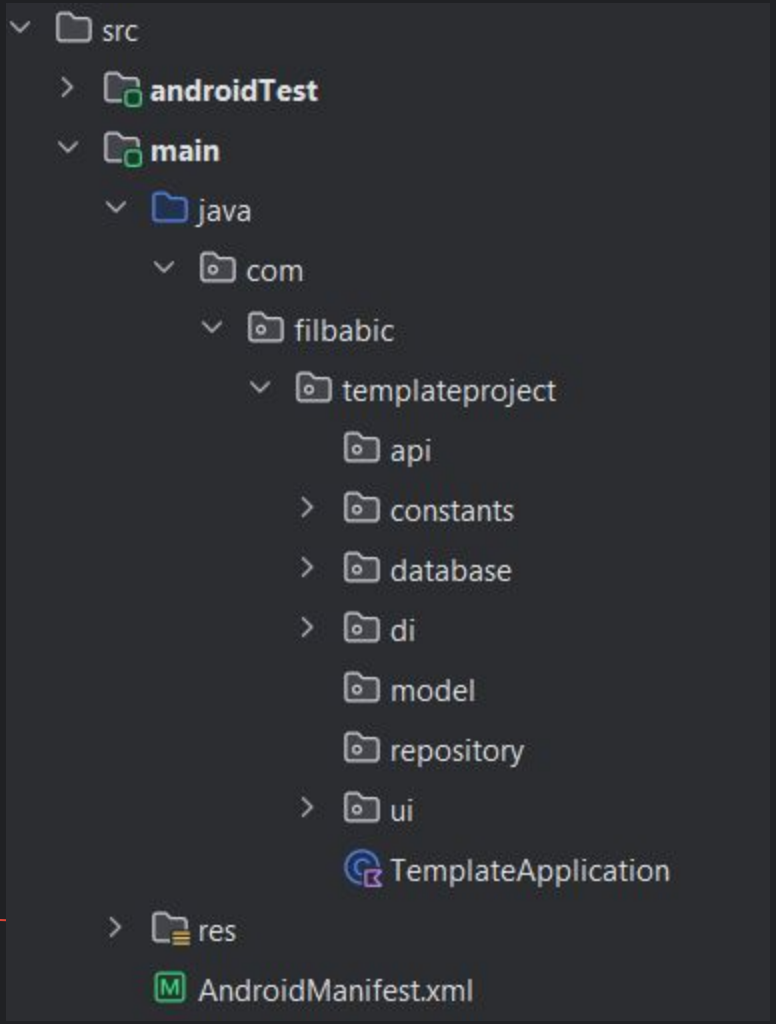

app
build
libs
src
androidTest
main
test [unitTest]
.gitignore
build.gradle.kts
proguard-rules.pro

Module configuration

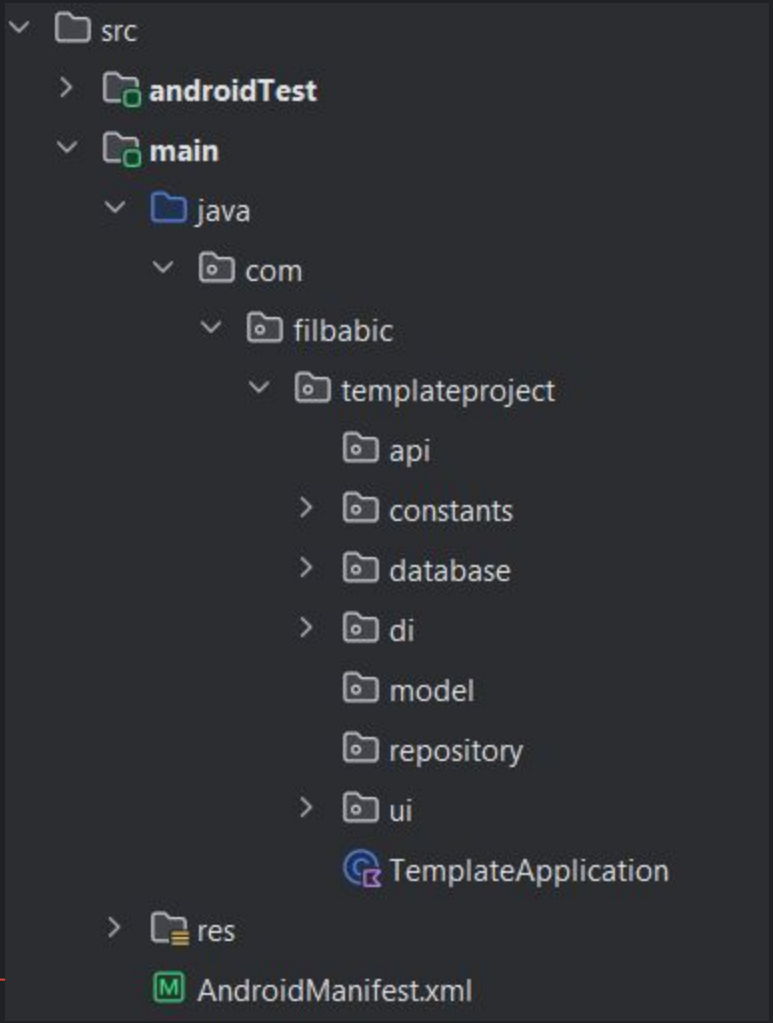


Codebase


src
androidTest
main
java
com
filbabic
templateproject
api
constants
database
di
model
repository
ui
TemplateApplication
res
AndroidManifest.xml


src
androidTest
main
java
com
filbabic
templateproject
api
constants
database
di
model
repository
ui
TemplateApplication
res
AndroidManifest.xml

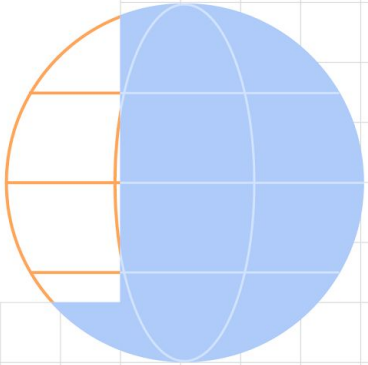
Resources



Manifest
Definitions

GDG

Gradle Build Process



Gradle Build Process

Set of scripts used to run a project

- Connects **3rd party dependencies**
- **Loads resources**
- **Packages** different files together
- Builds an **executable** file*
- Applies **signing**

- app
  - src
  - .gitignore
  - build.gradle
  - proguard-rules.pro
- configs
- gradle
- .gitignore
- build.gradle
- gradle.properties
- gradlew
- gradlew.bat
- local.properties
- settings.gradle

- androidApp
  - src
  - build.gradle.kts
- core
- gradle
- iosApp
- .gitignore
- build.gradle.kts
- gradle.properties
- gradlew
- gradlew.bat
- local.properties
- settings.gradle.kts

Gradle Script Types

Each type affects different parts of the build

- **Project-level** build.gradle script
- **Module-level** build.gradle script
- **Settings** gradle script

Project-level Script

Applied throughout the project

- **Ext (external/exported)** properties - useful for versions* (moved to **.toml**)
- **Repositories** used for dependencies
- Project level **tasks**
- Project level plugins

Module-level Script

Applied inside a single module

- **Plugins** for that module
- **Android based configuration & build options**
- Module dependencies & inclusions (other modules)
- Repositories required for the module

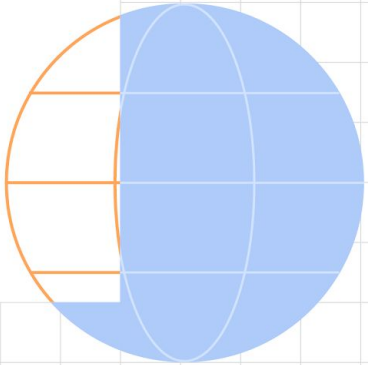
Settings Script

Common project configuration*

- Plugins! (**KTS only**)
- Dependency resolution (**KTS only**)
- Config like project name, modules to include in build loading...

GDG

Android Key Components



Android Key Components

These things pretty much make up an Android app

- **Activities**, Views and Fragments
- Intents
- **Content providers**
- **Services** and **Broadcast receivers**
- Android Manifest

Activities, Fragments, Views

Building blocks of the UI and user interaction

- **Activity** - Represents one screen
- **Fragment** - Represents *fragmented* parts of a UI and logic, usable in multiple screens
- **View** - Base building block of the UI (XML only)
- **Composable functions** - Base building blocks of the UI (Compose)

Intents

Action to open a new screen or trigger a system

- **Target Activities, Broadcast Receivers or Services**
- Start a new *system*
- Can contain small pieces of data
- Can be restricted to components or application names
- Used in notifications*

Content Providers

They provide content!

- Used to load data from the Android system database/persistence
- Extremely annoying to use
- Mostly avoided if possible, but necessary in some cases (like SMS, MMS, contacts...)

Services & Broadcast Receivers

Background systems that trigger events or do work

- Services are **long or short running background operations**
- Broadcast receivers **receive intents** and then **trigger behavior** or **start components** (like services)
- Receivers can work across-apps
- **WorkManager*** - Essentially a way to replace most services and other background workers

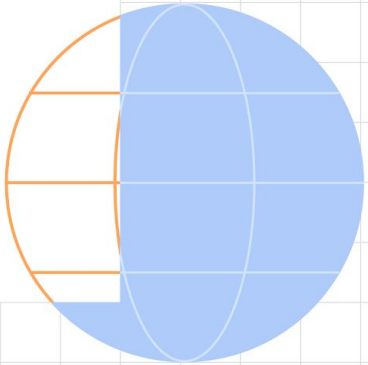
AndroidManifest

Single file describing the core functionality of an app

- Defines all Activities, receivers, services, permissions, configuration and so on
- **Each Android module requires one manifest**
- **Manifests are merged** across Android modules*

GDG

Building UI (Compose/XML)



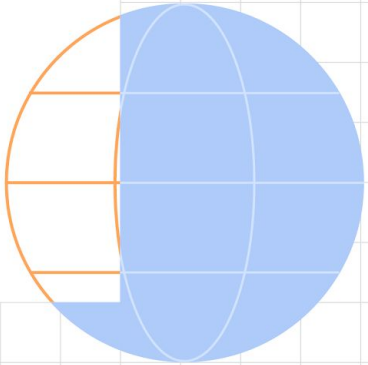
Building UI

Two ways - legacy (XML) and modern (Compose)

- XML requires of people to learn XML, learn how the View system works, lifecycle, combine XML attributes & programmatic View properties
- Compose is fully declarative, no objects, no Views, no XML, only Kotlin and functions
- **There's interop** -> Can embed XML in Compose and use Compose in XML

GDG

Dependencies, common libraries



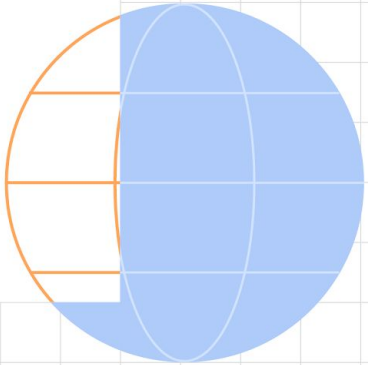
Common Libraries

Android strongly emphasizes the use of libraries

- **Database**: **Room**, SQLDelight, Realm
- **Networking**: **Retrofit**, Ktor
- **Image Loading**: **Coil**, Glide, Picasso
- **Parsing**: **Kotlin Serialization**, Moshi, Gson
- **Various Tooling**: **Firebase**

GDG

AndroidX & Jetpack



AndroidX & Jetpack

“Modern” ways of building Android apps

- Historically there weren’t **strict guidelines** for Android app development
- Google/Android didn’t suggest any architectural patterns, libraries, code styling
- Then Jetpack came and we got a new **fresh set of tools for development**

Jetpack libraries

Explore all libraries

Filter by keyword or use case

* Popular and often-used libraries are listed first

activity *	Access composable APIs built on top of Activity.
appcompat *	Allows access to new APIs on older API versions of the platform (many using Material Design).
appsearch *	Build custom in-app search capabilities for your users.
camera *	Build mobile camera apps.
compose *	Define your UI programmatically with composable functions that describe its shape and data dependencies.
databinding *	Bind UI components in your layouts to data sources in your app using a declarative format.
fragment *	Segment your app into multiple, independent screens that are hosted within an Activity.

More

What's new

Compose August 2023

Jetpack Compose 1.5.0 moves to stable and brings major performance improvements including a refactoring of high-level modifiers such as `Clickable` that can improve composition time by 80%. August'23 Compose also brings up to 70% improvement in memory allocation (especially in the graphics stack), which will reduce the memory footprint of compose on devices

Wear Compose and Tiles 1.2

Both Wear Compose and Wear Tiles have moved to stable to further enhance the experience of wearOS developers. Both are complementary to each other where Wear Compose can be used to build complex app screens on wear devices, and wear tiles can be used to create the app tiles. The new Wear Compose release contains new functionalities such as Expandable Items and Swipe to reveal. Wear Tiles release also now supports widget animation, and platform data binding (such as health data source).

Window 1.1

1.1 stabilizes activity embedding APIs, allowing apps like WhatsApp, eBay and Temu to ship large screen layouts. The API is enriched with features, as it enables developers to modify split screen behavior, check (and change) split state at runtime, implement horizontal splits, and start a modal in full window.

Release notes

For more information, visit our release notes.