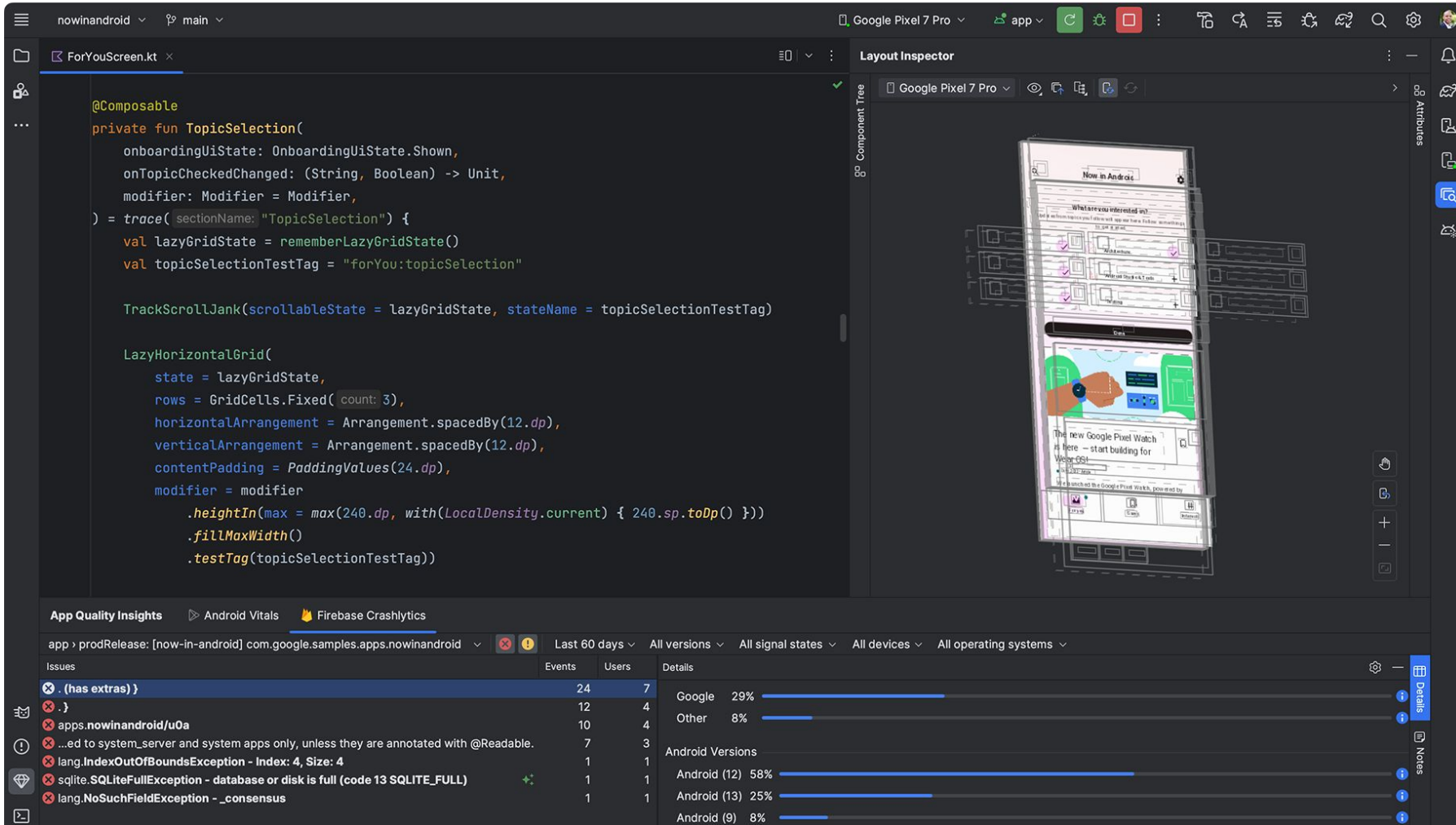
Android

XML Interface

Mark Joselli

mark.joselli@pucpr.br

Android Studio



Android Studio

- Android Studio is the official integrated development environment (IDE) for developing Android applications.
- Developed by JetBrains and maintained by Google, Android Studio offers a complete set of tools that simplify and accelerate application development for the Android operating system.
- Android Studio provides an advanced code editor with features such as auto-completion, static code analysis, and intelligent refactoring.
- It supports multiple programming languages, with Java and Kotlin being the main ones for Android development.

Step by step

Project Configuration

- **Install Android Studio:**
  - **Make sure you have Android Studio installed. If not, download and install from https://developer.android.com/studio?utm_source=android-studio&hl=pt-br**
  - **Create a New Project:**
- **Open Android Studio.**
  - **Click "Start a new Android Studio project".**
  - **Select the “Empty Views Activity” activity template and click “Next”.**
  - **Configure the application name, company domain, and project location.**
  - **Select the minimum SDK versions (I'll use 27 for now).**
  - **Choose the default Build Configuration language.**
  - **Click "Finish".**

Device Manager

- Android Studio's Device Manager is an integrated tool that lets you create, manage, and test virtual and physical Android devices (AVDs).
- It provides a flexible and convenient environment for testing applications on different hardware configurations and Android versions without the need for additional physical devices.
- How to Access Device Manager
  - Quick Access: Click the "Device Manager" icon in the Android Studio toolbar.
  - Menu: Go to Tools > Device Manager in the top menu.

Step-by-step guide to enable the physical device

1. **Open Device Settings: On your Android device, go to Settings.**
2. **Access "About Phone":: Scroll down and tap About phone. On some devices, this may be under the System section.**
3. **Find the Version Number: Within About Phone, find the Build number option. On some devices, it may be under Software information.**
4. **Enable Developer Options: Tap Build Number repeatedly (usually 7 times) until a message appears saying you are now a developer. You may be asked to enter your PIN or password.**
5. **Return to Main Settings: After enabling Developer Options, return to the main Settings screen.**
6. **Access Developer Options: You will now see a new option called Developer options in the settings (usually in the System section).**
7. **Enable USB Debugging: Within Developer Options, scroll down until you find USB Debugging and enable it.**
8. **Install USB Drivers (Windows): For devices that are not automatically recognized in Windows, you may need to install manufacturer-specific drivers. Android Studio and the Android SDK offer tools like Google USB Driver, but you may need manufacturer-specific drivers (Samsung, LG, etc.).**

Step by step to create an Android Virtual Device (AVD) in Android Studio

- **Start Creation: In the Device Manager window, click the Create Device button.**
- **Select Profile: A list of predefined devices will appear, such as Pixel, Nexus, and other common devices. Select the device that most closely matches the specifications you want to emulate and click Next.**
- **Select System Image: On the next screen, you will see a list of system images available for different versions of Android. Choose the desired version and click Next. If necessary, you can download other system images by clicking the Download link next to the desired version.**
- **Configure AVD Properties**
- **Give your AVD a name in the AVD Name section.**
- **Adjust additional settings as needed, such as RAM, internal storage, screen orientation, etc.**
- **If desired, check the Enable Device Frame option to show the device frame.**
- **Review Settings:: Review all AVD settings and click Finish to create the device.**
- **Launch the AVD: In the Device Manager AVD list, find the newly created device and click the Play icon (green triangle) to launch the AVD.**
- **Test the Application: With the AVD running, you can deploy and test your application on the virtual device directly from Android Studio by clicking Run in your project.**

Debugging and Performance Tools

- Android Studio offers a wide range of debugging and performance analysis tools that are essential for developing high-quality Android applications.
- These tools help you identify and fix bugs, optimize performance, and ensure that your application functions correctly in a variety of conditions.

Debugging and Performance Tools - Logcat

- Logcat is a tool built into Android Studio that captures and displays system and application message logs. It is essential for real-time debugging and provides detailed information about errors, warnings, and other log messages that you insert into your code.
- Basic Usage:
  - Log.d("TAG", "Debug message"): Debug message.
  - Log.e("TAG", "Error message"): Error message.
- Filtering and Search:
  - You can filter the logs by severity level, process name, or search text.
- ADB command:
  - adb logcat: Displays logs directly in the terminal.

Debugging and Performance Tools - Debugger

- Android Studio Debugger allows you to pause code execution, examine variables, evaluate expressions, and follow the program flow.
- Breakpoints:
  - Click in the left margin of the code editor to set breakpoints.
  - Breakpoints can be conditional, that is, they only interrupt execution when a certain condition is true.
- Variable Inspection:
  - When code stops at a breakpoint, you can inspect the state of variables and objects in the "Debugger" panel.
- Step Over, Step Into, Step Out:
  - Allows you to advance in code execution line by line (Step Over), enter functions (Step Into), or exit them (Step Out).

Debugging and Performance Tools - Android Profiler

- Android Profiler is a powerful tool for monitoring CPU, memory, network and battery usage while running your app. It offers real-time graphs and allows you to identify performance bottlenecks.
- CPU Profiler:
  - Allows you to record code execution traces to identify parts that consume the most CPU.
  - Visualizes method calls and the time spent on each one.
- Memory Profiler:
  - Monitors memory allocation in real time.
  - Identifies memory leaks (Memory Leaks) and monitors the use of allocated objects.
- Network Profiler:
  - Monitors network calls in real time, including size of data sent/received and response time.
  - Useful for optimizing network usage and identifying bottlenecks.
- Energy Profiler:
  - Monitors the application's energy consumption, identifying processes that impact the battery.

Debugging and Performance Tools - Layout Inspector

- The Layout Inspector allows you to inspect the View Hierarchy of a screen in real time, viewing the attributes and properties of each interface element.
- Layout Hierarchy:
  - Visualizes the view hierarchy in a tree structure.
  - Useful for identifying layout issues such as unwanted overlaps.
- Attributes:
  - Inspects the attributes of each view, such as size, position, margins, and padding.
- Snapshot:
  - Capture screen snapshots for offline analysis.

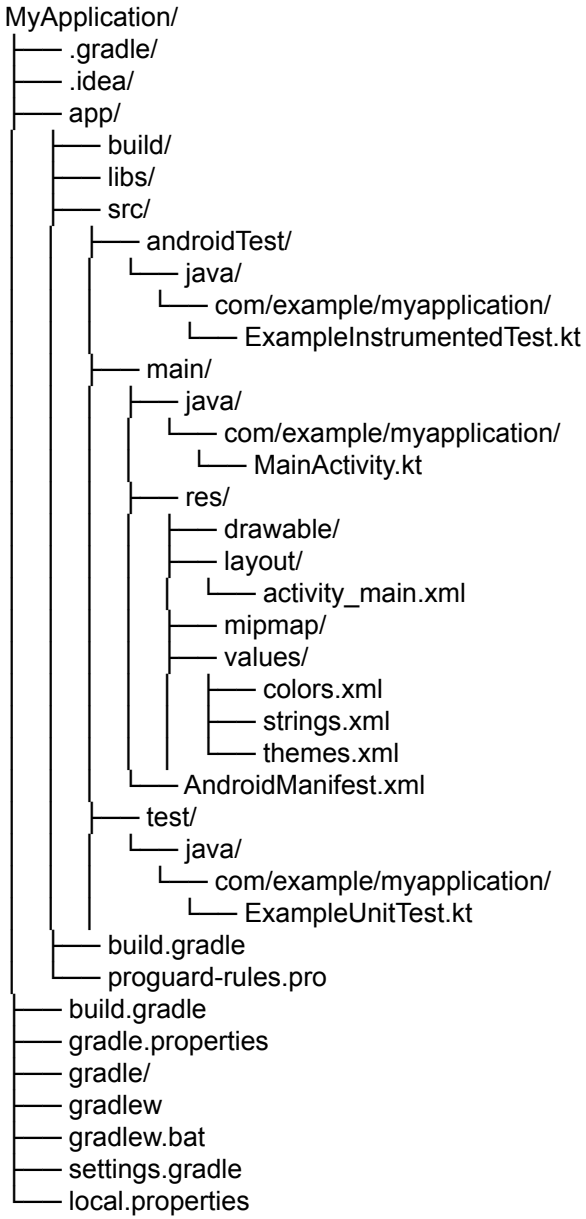
Debugging and Performance Tools - Database Inspector

- Database Inspector allows you to inspect and modify your application's SQLite database while it is running.
- Database Query:
  - Executes SQL queries directly against the running application's database.
  - View and edit stored data.
- Real-Time Monitoring:
  - Monitors changes to the database as the application runs.
  - Useful for debugging database operations and validating data.

Gradle

- Gradle is the build automation tool used by Android Studio to manage the build of Android projects.
  - It is highly flexible and powerful, allowing you to define how your project should be built, including configuring and managing library and plugin dependencies.
- In a typical Android project, you will find the following build.gradle files:
  - build.gradle at project level (root):
    - This file typically configures plugins and dependencies that affect the entire project, such as Gradle versions and build class (buildscript) dependencies.
  - build.gradle at module (app) level:
    - This file configures module-specific dependencies (usually the "app" module) and defines how the application will be built.
- After editing the build.gradle file, Android Studio usually prompts you to sync the project to apply the changes.
- You can do this manually by clicking Sync Now or selecting File > Sync Project with Gradle Files.

Basic Structure of a Project



app/ directory

This is the main module of your Android project. It contains all the source code, resources and configuration files for the application.

app/build/: Automatically generated directory containing the build files.

app/libs/: Directory to include external libraries (.jar or .aar files) that are not available in Maven repositories.

app/src/: Main directory containing the application source code and resources.

androidTest/: Contains code for instrumentation tests, which are tests run on Android devices or emulators.

java/: Source code for instrumentation tests.

com/example/myapplication/: The default package where instrumentation tests are stored.

ExampleInstrumentedTest.kt: Automatically created instrumentation test example.

- **app/src/main/:** Main directory of the application's source code and resources.
  - **java/:** Contains the main source code of the application.
    - **com/example/myapplication/:** The default package for Kotlin/Java code.
    - **MainActivity.kt:** The automatically created MainActivity file.
  - **res/:** Contains the application resources such as layouts, images and strings.
  - **drawable/:** Contains graphic resources (images, vectors, etc.).
  - **layout/:** Contains the XML layout files.
    - **activity_main.xml:** The main layout of the activity (MainActivity).
  - **mipmap/:** Contains the application's launcher icons.
  - **values/:** Contains value resources such as colors, strings, and themes.
    - **colors.xml:** Defines the colors used in the application.
    - **strings.xml:** Contains the strings used in the application.
    - **themes.xml:** Defines application themes, such as interface styles.
- **AndroidManifest.xml:** The manifest file that defines the application structure, including activities, permissions, and application settings. https://developer.android.com/guide/topics/manifest/manifest-intro

- **App/src/test/:** Contains the unit test code.
  - **java/:** Source code for the unit tests.
    - **com/example/myapplication/:** The default package where unit tests are stored.
    - **ExampleUnitTest.kt:** Automatically created unit test example.
- **build.gradle (app module):** Gradle configuration file specific to the app module. Here you define dependencies, build configurations, and other module-specific options.
- **proguard-rules.pro:** Configuration file for ProGuard, used to optimize, minify, and obfuscate code during release APK creation.

- **gradle/ directory**
  - This directory contains files related to the Gradle wrapper, which ensures that the project uses a specific version of Gradle without the need for manual installation.
- **build.gradle file (Project Level)**
  - The project-level build.gradle file is used to configure dependencies and plugins that affect the entire project.
- **gradle.properties file**
  - This file can be used to set Gradle-specific properties such as memory settings and custom settings for the build.
- **settings.gradle file**
  - Defines the modules that are part of your project. In a simple project, there is usually only one module (app), but larger projects can have multiple modules.
- **local.properties file**
  - Contains machine-specific local settings, such as the path to the Android SDK. This file is generally not included in version control.
- **gradlew and gradlew.bat**
  - Scripts to run Gradle using the Gradle wrapper (gradlew). gradlew is for Unix/Linux/Mac, while gradlew.bat is for Windows.

Activity

- https://developer.android.com/guide/components/activities/intro-activities
- https://developer.android.com/guide/components/activities/activity-lifecycle