Dominando el Desarrollo en Android: Estructura de Proyectos, Gradle y AndroidX

Para la Universidad Nacional de Colombia sede Manizales

Contenido

1. Introducción
2. Archivos del Proyecto
3. Gradle
    - Archivos de Construcción de Gradle
4. AndroidX
5. Ejercicio Práctico
6. Conclusión
7. Recursos

Introducción

En el artículo anterior, revisamos los pasos para configurar **Android Studio** y lanzar por primera vez una aplicación básica. Sin embargo, un proyecto Android típico contiene múltiples carpetas y archivos que cumplen diferentes propósitos. Conocer su estructura y comprender su funcionamiento es esencial para desarrollar aplicaciones de manera sólida y escalable.

En este documento profundizaremos en la arquitectura general de un proyecto Android, explicaremos qué es Gradle y cómo automatiza las tareas de compilación y empaquetado. Por último, abordaremos la importancia de **AndroidX**, la colección de librerías recomendada para el desarrollo moderno de aplicaciones en Android.

> Nota Académica: Comprender estos conceptos en conjunto te permitirá organizar tu aplicación y gestionarla adecuadamente,

> facilitando tareas de mantenimiento, escalabilidad y colaboración en equipo.

Archivos del Proyecto

Cuando creamos un nuevo proyecto en Android Studio, se generan automáticamente varias carpetas y archivos. A pesar de que no es necesario configurarlos todos manualmente, es crucial entender su función:

1. **Código fuente (src/main/java o src/main/kotlin)**

   Contiene las clases de la aplicación escritas en Java o Kotlin. Aquí se encuentra la **lógica de negocio** y las distintas **actividades**, **fragmentos** y **componentes** que dan vida a la aplicación.

2. **Recursos (src/main/res)**

   Incluye archivos de diseño de interfaz (*layouts*), cadenas de texto (*strings*), íconos, imágenes y otros elementos gráficos. Organizar adecuadamente estos recursos garantiza un desarrollo ordenado y facilita la localización (traducciones) y el cambio de temas.

3. **AndroidManifest.xml**

   Archivo fundamental que describe la estructura de la aplicación ante el sistema operativo Android. Aquí se declaran **actividades**, **permisos**, **intenciones (intents)**, y otros aspectos que definen el comportamiento del aplicativo.

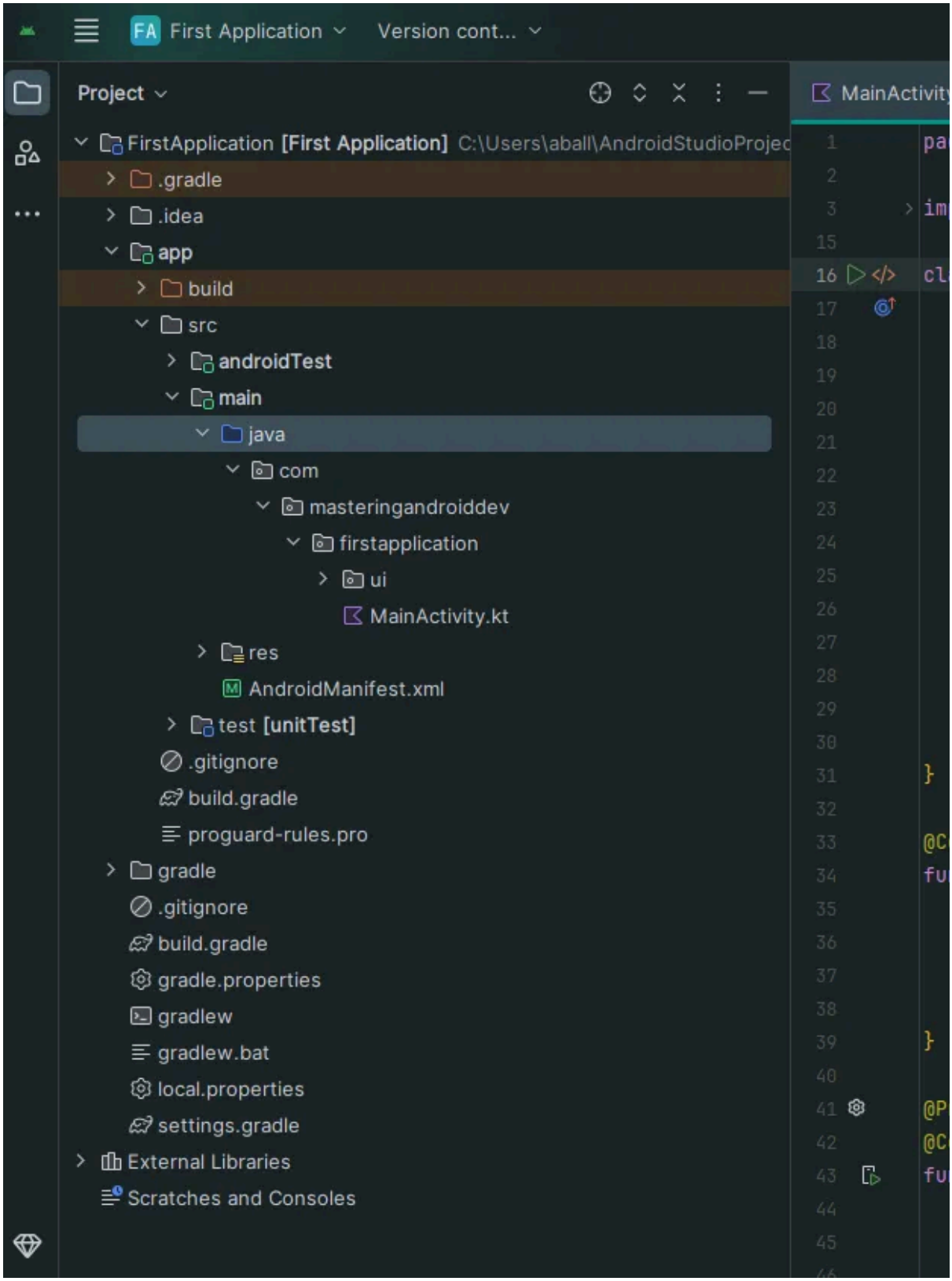
4. **Carpetas de prueba (androidTest y test)**

   - **androidTest**: Pruebas instrumentadas que se ejecutan en un dispositivo o emulador Android real, permitiendo verificar aspectos de la UI o interacciones con componentes del sistema.
   - **test**: Pruebas unitarias que no requieren un entorno Android completo; se ejecutan en la JVM local (del computador).

5. **Módulo (app)**

   El módulo principal se encuentra en la carpeta denominada **“app”**. Cada proyecto puede contener uno o varios módulos. Por ejemplo, podrías tener un módulo de **biblioteca** (library) y otro para la **aplicación** en sí. Esta estructura modular facilita la separación de responsabilidades y la reutilización de código.

En la **vista "Android"** del panel **Project** (ubicada usualmente en la parte izquierda de Android Studio), se simplifica la estructura a fin de mostrar las carpetas relevantes para el desarrollo. Sin embargo, si cambias a la **vista "Project"**, observarás todas las carpetas tal como se encuentran en el disco:



La vista **Project** expone carpetas como **.gradle** o **.idea**, que por el momento no requieren modificación directa. En la carpeta **app**, encontrarás:

```
└── app/
    ├── build: Contiene los archivos resultantes de la compilaci
ón.
    └── src: Carpeta principal con código y recursos.
        ├── androidTest: Pruebas instrumentadas (en un dispositi
vo o emulador).
        ├── main: Código fuente y recursos esenciales.
        │   ├── java (o kotlin): Clases del proyecto.
        │   ├── res: Archivos de diseño, imágenes, textos, etc.
        │   └── AndroidManifest.xml: Define información clave de
la aplicación.
        └── test: Pruebas unitarias en la JVM local.
```

Gradle

Gradle es un sistema de automatización de compilaciones que facilita la creación, empaquetado y prueba de proyectos de software. En el contexto de Android, Gradle se encarga de:

- **Compilar** el código fuente.
- **Gestionar dependencias** (librerías externas que tu proyecto requiere).
- **Ejecutar pruebas** (unitarias e instrumentadas).
- **Empaquetar** la aplicación en formatos como **APK** o **AAB** para su distribución.

¿Por qué se necesita un sistema de compilación?

En proyectos de gran envergadura, mantener manualmente los pasos de compilación, enlazado y empaquetado resulta complejo. Gradle automatiza estos procesos mediante scripts de configuración, permitiendo que con un simple **clic en "Run"** (o un comando en la terminal) se construya y despliegue la aplicación.

> Dato Académico: Gradle se ha vuelto estándar en la comunidad de Android debido a su flexibilidad y a la integración nativa que ofrece con Android Studio y el plugin oficial de Android.

Archivos de Construcción de Gradle

Observa que en tu proyecto pueden existir varios archivos `build.gradle`. Los más comunes son:

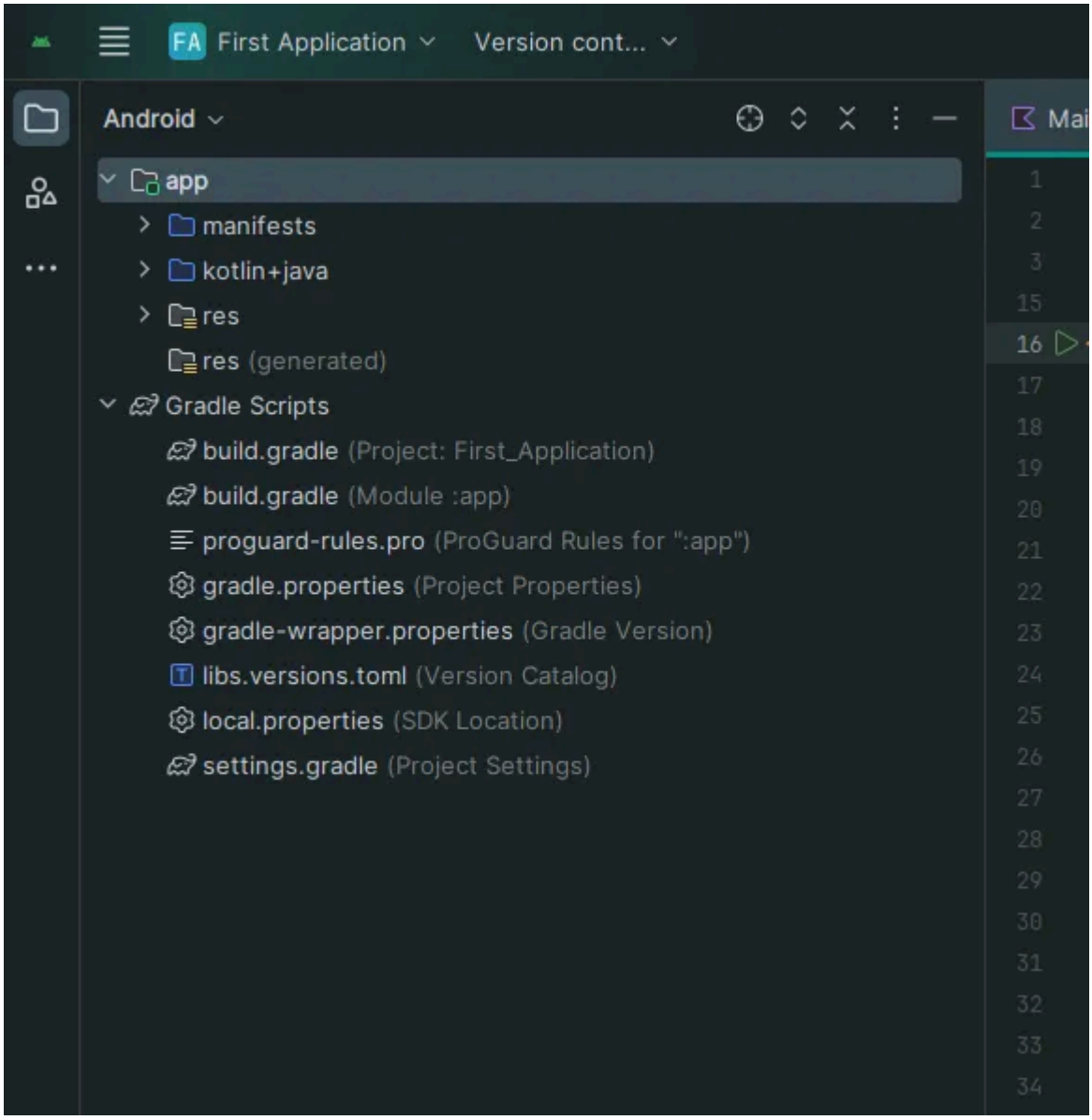
1. **Root Build File** (`<PROJECT_ROOT>/build.gradle`)

   Configura parámetros de nivel global para todo el proyecto, como la versión de Gradle o repositorios generales (Maven Central, Google, etc.).

2. **Subproject-Level Build File** (`<PROJECT_ROOT>/app/build.gradle`)

   Se encarga de la configuración particular del módulo "app", donde se especifican las dependencias específicas, versiones del SDK, configuraciones de empaquetado, entre otros.

En la **vista "Android"** del panel **Project**, estos archivos suelen aparecer agrupados en una sección llamada **"Gradle Scripts"**:



AndroidX

Si revisas un archivo de código como `MainActivity.kt`, es probable que aparezcan declaraciones `import androidx.*`.

Por ejemplo:

```
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.material3.Text
...
```

AndroidX es el nombre con el que se agrupan las librerías modernas que Google recomienda para desarrollar aplicaciones Android. Reemplaza el antiguo **Support Library** y mantiene una separación clara entre el código del sistema operativo y las librerías complementarias.

> Android Jetpack es una marca que engloba a todas las librerías de AndroidX, poniéndoles un nombre más comercial y reconocible. Técnicamente, Jetpack y AndroidX hacen referencia a los mismos componentes, pero Jetpack es la denominación promocional.

Estas librerías ofrecen múltiples ventajas:

- **Compatibilidad** con versiones anteriores del sistema operativo.
- **Modularidad**: permiten incluir solo los componentes necesarios.
- **Nuevas APIs** y elementos de arquitectura (Room, ViewModel, LiveData, etc.) que facilitan el desarrollo de aplicaciones más robustas.

Ejercicio Práctico

Para afianzar lo aprendido, se sugiere el siguiente ejercicio:

1. **Elimina la carpeta** `java` dentro de `src/main` y también **el archivo** `AndroidManifest.xml` del proyecto.
2. **Intenta compilar** o **ejecutar** la aplicación.

Observarás que aparecerán **múltiples errores**. Tu reto consiste en **restaurar** ambos elementos (la carpeta con tus clases Java/Kotlin y el `AndroidManifest.xml`) para dejar el proyecto en un estado funcional.

- Crea nuevamente la carpeta `java` en la ruta exacta requerida (`src/main/java/...`).
- Restaura o crea un archivo `AndroidManifest.xml` con la etiqueta `<manifest>` y la configuración básica.

Este ejercicio te ayudará a comprender la relevancia de cada archivo y cómo la estructura del proyecto está estrechamente vinculada a la forma en que Android Studio y Gradle procesan la aplicación.

Conclusión

La estructura de un proyecto Android puede parecer compleja al principio, pero cada carpeta y archivo tiene una función específica:

- **Archivos fuente y recursos** se ubican en carpetas separadas para mantener la organización.
- **Gradle** automatiza la compilación y manejo de dependencias, siendo la base para construir aplicaciones escalables y adaptables.
- **AndroidX** (también conocido como Jetpack) brinda librerías modernas que facilitan la compatibilidad y la arquitectura de la aplicación.

Al dominar estos conceptos, estarás mejor preparado para afrontar proyectos más avanzados y emplear las mejores prácticas de desarrollo en Android.

Recursos

- **Projects Overview** (en inglés)
- **Gradle Build Overview** (en inglés)
- **Android Build Structure** (en inglés)