Índice general

¿Cómo leer este libro? 15
CAPÍTULO 1. Diseño avanzado de la interfaz de usuario 19
1.1. Acceder a objetos globales de la aplicación 21
1.1.1. La clase Application 21
1.2. Material Design 25
1.2.1. Definición de la paleta de colores de la aplicación 27
1.3. RecyclerView 29
1.4. *Fragments* 36
1.4.1. Insertar *fragments* desde XML 39
1.4.2. Insertar *fragments* desde código 46
1.4.3. Comunicación e intercambio de *fragments* 49
1.5. Otros elementos de la interfaz de usuario 50
1.5.1. Menús contextuales 50
1.5.2. La barra de acciones (*ActionBar / Toolbar*) 52
1.5.2.1. Barra de acciones con *ActionBar* 53
1.5.2.2. Barra de acciones con Toolbar 56
1.5.2.3. Configurar la barra de acciones desde un *fragment* 57
1.5.3. Vistas animadas: CoordinationLayout, AppBarLayout, FloatingActionButton y SnackBar 59
1.5.4. Pestañas con TabLayout 67
1.6. Navigation Drawer 72
1.6.1. Ocultar elementos de la interfaz de usuario 78
1.7. Servicios de búsquedas 83
1.7.1. Añadiendo preferencias con *PreferenceFragment* 86
1.8. Animaciones 90
1.8.1. Animaciones de vistas: transiciones entre actividades 91
1.8.1.1. Aplicando animaciones de vistas en Audiolibros 94
1.8.2. Animaciones de propiedades 96
1.8.2.1. El motor básico de animación: *ValueAnimator* 97
1.8.2.2. Automatizando las animaciones: *ObjectAnimator* 99
1.8.2.3. Combinando animaciones: *AnimatorSet* 101

1.8.2.4. Definiendo animaciones en XML 101
1.8.2.5. Nuevas propiedades de la clase View 104
1.8.2.6. Aplicando animaciones de propiedades en Audiolibros 106
1.8.3. Animaciones en RecyclerView 108
1.9. Otros aspectos introducidos en la versión 5.0 110
1.9.1. Extraer paleta de colores desde imágenes 110
1.9.2. Uso de gráficos vectoriales 112
CAPÍTULO 2. Diseño personalizado de vistas 119
2.1. Algunos conceptos básicos 120
2.2. Una vista como la composición de varias vistas 122
2.2.1. Creación de escuchadores de eventos 125
2.3. Modificación de vistas existentes 128
2.3.1. Algo más de información sobre TextView 131
2.4. Creación de nuevos atributos XML 133
2.5. Una vista creada desde cero 137
2.5.1. Diseño y dibujo de la vista 137
2.5.2. Gestión de eventos 142
2.5.3. Cómo Android dibuja las vistas y obtiene sus tamaños 144
2.5.4. Interactuando con otros objetos 147
2.6. Creación de *widgets* de escritorio 149
2.6.1. Pasos a seguir para crear un *widget* 149
2.6.1.1. Definir las características del *widget* 149
2.6.1.2. Diseñar el *layout* del *widget* 151
2.6.1.3. Crear una clase descendiente de AppWidgetProvider 151
2.6.1.4. Declarar el *widget* en AndroidManifest 152
2.6.1.5. Crear una actividad para configurarlo 152
2.6.2. Creación de un *widget* de escritorio sencillo 153
2.6.3. Actualizando el *widget* de escritorio 155
2.6.4. Actuando ante el evento onClick 157
2.6.5. Añadiendo una actividad de configuración 159
2.7. Notificaciones con *widgets* personalizados 162
CAPÍTULO 3. Hilos de ejecución 167
3.1. Hilos de ejecución en Android 168

3.1.1. Programación basada en eventos y el hilo de ejecución de usuario 169
3.1.1.1. Cola de eventos y bucle de eventos 170
3.1.2. Concurrencia en programación orientada a eventos 173
3.1.2.1. Hilos para el manejo de eventos 173
3.1.2.2. Hilos en segundo plano sin acceso a la GUI 175
3.1.2.3. Utilizando runOnUiThread para actualizar la interfaz gráfica 176
3.1.2.4. Utilizando post() para actualizar la interfaz gráfica 177
3.1.2.5. Utilizando Handler para actualizar la interfaz gráfica 178
3.1.2.6. Receptores de anuncios para actualizar la interfaz gráfica 180
3.2. La clase AsyncTask 182
3.2.1. Extendiendo AsyncTask 183
3.2.2. Secuencia de operaciones 185
3.2.3. Inicialización y ejecución de AsyncTask 188
3.2.4. Cancelar una tarea asíncrona 192
3.2.5. Estados de una tarea asíncrona 195
3.2.6. Proteger la tarea asíncrona del cambio de orientación 198
3.2.7. Ejecución concurrente de tareas asíncronas 206
3.2.8. Gestionando las excepciones 212
3.2.9. Resumen de errores comunes con AsyncTask 214
3.2.10. Aplicaciones comunes de AsyncTask 215
3.3. Servicios en Android 216
3.3.1. Introduciendo los Servicios 216
3.3.1.1. Creación y ejecución 217
3.3.1.2. Ciclo de vida 218
3.3.2. Servicios enlazados 221
3.3.2.1. Administración del ciclo de vida de un servicio enlazado 222
3.3.2.2. Comunicándonos con un servicio enlazado 224
3.3.2.3. Comunicándonos con un servicio local (*Local Binding*) 224
3.3.2.4. Comunicándonos con un servicio remoto 232
3.3.3. Devolviendo el resultado de la ejecución del servicio 237
3.3.3.1. Comunicación mediante una interfaz 237
3.3.3.2. Comunicación de resultados mediante Intents 242
3.3.4. Detectando comunicaciones sin gestionar 246

3.3.5. ¿Por qué utilizar servicios para ejecuciones asíncronas?....248
3.3.5.1. Eligiendo una técnica asíncrona....248
3.3.6. Aplicaciones de los servicios....249
3.3.7. Nuevas restricciones a la ejecución en segundo plano....250
3.3.7.1. Consideraciones genéricas....250
3.3.7.2. Restricciones sobre los servicios....251
3.3.7.3. Restricciones sobre receptores de anuncios....254
3.3.7.4. Restricciones sobre ubicación en segundo plano....255
3.3.7.5. Guía de migración....256
3.3.7.6. Poner servicios en primer plano....257
3.4. Animaciones con SurfaceView....262
3.4.1. Arquitectura de gráficos en Android....262
3.4.2. ¿Qué es la clase SurfaceView?....263
3.4.3. Llamando al método onDraw()....264
3.4.4. Programación con SurfaceViews....265
3.4.5. Estructura de la aplicación....266
CAPÍTULO 4. Introducción al Testing en Android....275
4.1. Introducción al desarrollo guiado por test....277
4.1.1. Realización de test en Android Studio....279
4.2. Test unitarios....280
4.2.1. Test unitario con parámetros....285
4.2.2. Test que captura excepciones....287
4.2.3. Creación de las clases MathExpression y MathCalculator....296
4.2.4. Test doble....309
4.2.5. Creación de la clase CalculatorPresenterImp....315
4.3. Test de interfaz de usuario....318
4.3.1. Test de interfaz de usuario con Espresso....318
4.3.2. Test de instrumentación con Firebase Test Lab....333
CAPÍTULO 5. Introducción a Kotlin....341
5.1. Introducción....342
5.1.1. Un poco de historia....342
5.1.2. Características de Kotlin....343
5.1.2.1. Conciso....343
5.1.2.2. Legible y expresivo....344

5.1.2.3. Código más seguro 344
5.1.2.4. Interoperable con Java 344
5.1.3. Uso de Kotlin en Android Studio 345
5.2. Variables y constantes 346
5.3. Estructuras de control 347
5.4. Funciones 349
5.4.1. Lambdas y funciones anónimas 350
5.4.2. Funciones inline 352
5.5. Clases 355
5.5.1. Clases de datos 358
5.5.2. Declaraciones desestructuradas 359
5.5.3. Extensiones 361
5.5.4. Type Alias 362
5.6. Anko 362
5.6.1. Funciones comunes 364
5.6.2. Layouts 365
5.7. Tratamiento de null 366
5.7.1. Añadiendo una comprobación previa 366
5.7.2. Llamada segura mediante operador ? 366
5.7.3. El operador Elvis ?: 367
5.7.4. El operador !! 367
5.7.5. El operador let 368
5.7.6. Interoperabilidad con Java 370
5.7.7. Nulidad y colecciones 371
5.7.8. Lateinit vs lazy 372
5.7.9. Cómo evitar el operador !! 372
5.7.9.1. Utiliza val en lugar de var 373
5.7.9.2. Utiliza lateinit 373
5.7.9.3. Utiliza el operador let 374
5.7.9.4. Crear funciones globales para gestionar casos complejos 374
5.7.9.5. Utilizar el operador Elvis 375
5.7.9.6. Reafirmarte en tu solución 375
5.8. Chequeos de tipo y Smart Cast 376
5.8.1. Chequeo de tipos 376

5.8.2. Smart Cast 376
5.8.3. Casting explícitos 377
5.9. Colecciones 378
5.9.1. Principales interfaces 379
5.9.2. Listas 379
5.9.3. Conjuntos 381
5.9.4. Mapas 381
5.9.5. Funciones sobre colecciones 382
5.10. apply vs with 383
5.10.1. with 383
5.10.2. apply 384
5.10.3. Diferencias entre apply y with 385
5.11. Clases abstractas selladas y enumeradas 385
5.11.1. Clases abstractas 385
5.11.2. Clases anidadas 386
5.11.3. Clases selladas (sealed) 386
5.11.4. Clases enumeradas 389
5.12. Métodos estáticos 391
5.13. Inmutabilidad en propiedades de clases 394
5.14. Buenas prácticas de programación con Kotlin 396
5.14.1. Soporte para expresiones (idioms) y patrones 396
5.14.2. Programación funcional 396
5.14.3. Utilización de expresiones 397
5.14.4. Funciones de extensión para utilidades 398
5.14.5. Argumentos nombrados en lugar de setter fluido 398
5.14.6. Utilizar apply() para inicializar objetos de agrupamiento 398
5.14.7. No utilizar la sobrecarga en parámetros por defecto 399
5.14.8. Gestionar apropiadamente la nulidad 399
5.14.8.1. Evitar las comprobaciones if-null 399
5.14.8.2. Evitar las comprobaciones de tipo if-type 399
5.14.8.3. Evitar las reafirmaciones de not-null !! 400
5.14.8.4. Utiliza let 400
5.14.9. Mapeado conciso con funciones de expresión simple 400

5.14.10. Hacer referencia a los parámetros del constructor en la inicialización de propiedades 401
5.14.11. Desestructuración 402
5.14.12. Creación de estructuras de datos....402
5.14.13. Definir varias clases en un fichero....402
5.14.14. Value Objects....403
CAPÍTULO 6. Arquitecturas de software 409
6.1. S.T.U.P.I.D. 410
6.1.1. Singleton Invasion 410
6.1.2. Tight coupling 411
6.1.3. Untestability 411
6.1.4. Premature Optimization....412
6.1.5. Indescriptive Naming....412
6.1.6. Duplication 412
6.2. Los principios S.O.L.I.D. 414
6.2.1. Single Responsibility Principle 415
6.2.2. Open/Closed principle....416
6.2.3. Liskov substitution principle....417
6.2.4. Interface segregation principle 417
6.2.5. Dependency inversion principle 418
6.3. Patrones de diseño comunes 418
6.3.1. El patrón de delegación (Delegation) 419
6.3.2. El patrón Observer....420
6.3.2.1. El patrón Observer con un solo escuchador 420
6.3.2.2. El patrón Observer con varios escuchadores....421
6.3.3. El patrón Null Object 423
6.3.4. El patrón Command....424
6.3.5. El patrón Adapter 426
6.3.6. El patrón Decorator 428
6.3.6.1. El patrón decorador en Kotlin 430
6.3.7. El patrón Builder....430
6.3.8. El patrón Singleton 432
6.3.8.1. El patrón Singleton en Kotlin 434
6.4. Patrones de arquitectura 436

6.4.1. Model View Controller 436
6.4.2. Model View Presenter 438
6.4.3. Modelo View View-Model 441
6.5. Arquitectura CLEAN 441
6.5.1. Reglas de dependencias 442
6.5.2. Interface Adapters 443
6.5.3. Capas 443
6.5.3.1. Capa de presentación 444
6.5.3.2. Casos de uso y capa de dominio 444
6.5.3.3. Capa de datos 446
6.6. Data Binding 448