Troubleshooting Común en PAE

Compilación

Error: "Gradle sync failed"

Síntomas: Android Studio no puede sincronizar proyecto

Soluciones:

```
# 1. Limpiar caches
./gradlew clean

# 2. Actualizar gradle wrapper
./gradlew wrapper --gradle-version=8.0

# 3. Actualizar dependencies
./gradlew --refresh-dependencies build

# 4. Regenerar gradle files
rm -rf .gradle
rm build.gradle.kts.bak
./gradlew build
```

Error: "org.gradle.api.GradleException"

Síntomas: Excepción genérica de Gradle

```
org.gradle.api.GradleException:
  org.gradle.workers.internal.DefaultWorkerExecutor$WorkerExecutionException
```

Deamoscrático:

```
# Compilar sin daemon
./gradlew --no-daemon clean build

# O deshabilitar daemon
echo "org.gradle.daemon=false" >> gradle.properties
```

Error: "Could not resolve dependency"

Síntomas: No encuentra librería

```
Caused by: org.gradle.api.artifacts.ResolveException:
```

```
  Could not resolve: com.example:library:1.0.0
```

Soluciones:

```
// Verificar todas las fuentes están disponibles
repositories {
    google()
    mavenCentral()
    jitpack { url "https://jitpack.io" }
}

// Comprobar versión existe
./gradlew :dependencies | grep library:

// Si es jitpack, verificar repo público
```

Error: "Compilation failed: Duplicate class kotlin.concurrent.LockedWrapper"

Causa: Múltiples versiones de stdlib

Solución:

```
configurations {
    all {
        exclude module: 'kotlin-stdlib-jdk7'
        exclude module: 'kotlin-stdlib'
    }
}

// O forzar versión
dependencies {
    implementation("org.jetbrains.kotlin:kotlin-stdlib:2.3.10") {
        force = true
    }
}
```

Runtime

Error: "Manifest does not include the internet permission"

Síntomas: App no puede conectarse a internet

Causa: Falta permiso en AndroidManifest

Solución:

```
<!-- AndroidManifest.xml -->
<uses-permission android:name="android.permission.INTERNET" />
```

Error: "Permission Denial: reading..."

Síntomas: Acceso a datos denegado

```
PermissionError: Permission Denial: reading ContentProvider requires
android.permission.CAMERA
```

Solución: Solicitar en runtime

```
val permission = Manifest.permission.CAMERA

if (ContextCompat.checkSelfPermission(context, permission)
    != PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions(activity, arrayOf(permission), 100)
}
```

Error: "Hardware not available: Camera2"

Síntomas: Camera2 API no disponible

Causa: Dispositivo antigua o emulador sin soporte

Solución:

```
// Comprobar disponibilidad
val cameraManager = context.getSystemService(Context.CAMERA_SERVICE) as
CameraManager
val cameraIds = cameraManager.cameraIdList

if (cameraIds.isEmpty()) {
    throw Exception("No cameras available")
}
```

Error: "Bluetooth connection refused"

Síntomas: No puede conectar a balanza Bluetooth

```
BluetoothSocket connect failed: BluetoothSocket.connect() called
```

Soluciones:

```
// 1. Verificar permisos
val requiredPermissions = arrayOf(
    Manifest.permission.BLUETOOTH,
```

```
    Manifest.permission.BLUETOOTH_ADMIN,
    Manifest.permission.BLUETOOTH_SCAN,
    Manifest.permission.BLUETOOTH_CONNECT
)

// 2. Verificar dispositivo pareado
val device = bluetoothAdapter.getRemoteDevice(macAddress)

// 3. Usar UUID correcto (SPP)
val uuid = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB")
device.createRfcommSocketToServiceRecord(uuid)

// 4. Reintentar con backoff
for (attempt in 1..3) {
    try {
        socket.connect()
        break
    } catch (e: IOException) {
        if (attempt < 3) {
            delay(1000 * attempt)
        } else throw e
    }
}
```

Error: "OutOfMemory - Java heap space"

Síntomas: App crashes por falta de memoria

```
java.lang.OutOfMemoryError: Java heap space
```

Causa típica: Cargando imágenes grandes sin comprimir

Soluciones:

```
// Comprimir Bitmap antes guardar
fun saveBitmap(bitmap: Bitmap, file: File) {
    FileOutputStream(file).use { fos ->
        // Reducir calidad
        bitmap.compress(Bitmap.CompressFormat.JPEG, 80, fos)
    }
}

// Reducir tamaño de imagen
fun downscaleBitmap(original: Bitmap, maxWidth: Int = 1280): Bitmap {
    val scale = maxWidth.toFloat() / original.width
    val newHeight = (original.height * scale).toInt()
    return Bitmap.createScaledBitmap(original, maxWidth, newHeight, true)
}
```

```
// Forzar garbage collection (último recurso)
System.gc()
```

Error: "Database is locked"

Síntomas: SQLite no puede escribir

```
android.database.sqlite.SQLiteDatabaseLockedException:
  database is locked
```

Causa: Múltiples threads escribiendo

Soluciones:

```
// Usar WAL (Write-Ahead Logging)
db.enableWriteAheadLogging()

// O serializar acceso
synchronized(dbLock) {
    repository.save(entity)
}

// O usar single-threaded scope
withContext(Dispatchers.IO) {  // serializa en Single thread
    repository.save(entity)
}
```

Error: "State transition invalid"

Síntomas: StateManager rechaza transición

```
IllegalStateException: Cannot transition from WaitingForWeight to
SaveDelivery
```

Causa: Flujo de estados incorrecto

Solución: Verificar que estados sean consecutivos

```
// ✅ Correcto: estados lineales
WaitingForWeight → CaptureImages → CaptureFace → ... → SaveDelivery

// ❌ Incorrecto: saltos aleatorios
WaitingForWeight → SaveDelivery  // Imposible
```

Networking

Error: "UnknownHostException: backend.api.com"

Síntomas: No puede resolver DNS

```
java.net.UnknownHostException: Unable to resolve host "backend.api.com"
```

Causa: Internet no disponible o DNS fallido

Soluciones:

```
// Verificar conectividad
val connectivityManager =
context.getSystemService(Context.CONNECTIVITY_SERVICE) as
ConnectivityManager
val network = connectivityManager.activeNetwork
val capabilities = connectivityManager.getNetworkCapabilities(network)

val isConnected =
capabilities?.hasCapability(NetworkCapabilities.NET_CAPABILITY_INTERNET) ==
true

if (!isConnected) {
    showError("No hay conexión a internet")
    return
}

// Retry con backoff exponencial
suspend fun <T> retryWithBackoff(
    maxRetries: Int = 3,
    delayMs: Long = 1000,
    block: suspend () -> T
): T {
    repeat(maxRetries - 1) {
        try {
            return block()
        } catch (e: Exception) {
            delay(delayMs * (2 pow it))
        }
    }
    return block()  // última vez, dejar fallar
}
```

Error: "Certificate verification failed"

Síntomas: HTTPS no confía en certificado

```
javax.net.ssl.SSLHandshakeException:
  Certificate verification failed
```

Causa: Certificado self-signed o expirado

Soluciones:

```
// Para testing (NO producción):
val trustAllCerts = arrayOf<TrustManager>(
    object : X509TrustManager {
        override fun checkClientTrusted(chain: Array<X509Certificate>,
authType: String) {}
        override fun checkServerTrusted(chain: Array<X509Certificate>,
authType: String) {}
        override fun getAcceptedIssuers(): Array<X509Certificate>? = null
    }
)

// En producción: use certificate pinning (ver security-and-privacy.md)
```

Error: "P2P connection timeout"

Síntomas: Timeout conectando a máquina vía P2P

```
java.net.SocketTimeoutException: timeout after 20000ms
```

Causa: Máquina no responde o está fuera de rango

Soluciones:

```
// Aumentar timeout
val okHttpClient = OkHttpClient.Builder()
    .connectTimeout(30, TimeUnit.SECONDS)
    .readTimeout(30, TimeUnit.SECONDS)
    .writeTimeout(30, TimeUnit.SECONDS)
    .build()

// Verificar conexión Wi-Fi Direct
val wifiP2pManager = context.getSystemService(Context.WIFI_P2P_SERVICE) as
WifiP2pManager

// Buscar dispositivos disponibles primero
wifiP2pManager.discoverPeers(channel, object : WifiP2pManager.ActionListener
{
    override fun onSuccess() {
        // Pedir lista de peers
    }
    override fun onFailure(reason: Int) {
        // No hay peers
```

```
    }
})
```

P2P / DirectLink

Error: "Wi-Fi Direct not available on this device"

Síntomas: No se puede usar P2P

```
DirectLinkException: Wi-Fi Direct not available
```

Causa: Dispositivo viejo sin Wi-Fi Direct

Solución: Usar fallback a hotspot

```
try {
    return p2pManager.connectMachine(machineId)
} catch (e: DirectLinkException) {
    // Fallback a hotspot
    return p2pManager.connectMachineHotspot(ssid)
}
```

Error: "Peer not found in scan results"

Síntomas: No encuentra máquina en discoverable

Soluciones:

```
// 1. Verificar que máquina está en modo descubrible
// (ejecutar en Machine: directlink.startAdvertising())

// 2. Re-escanear
p2pManager.discoverableMachineIds()

// 3. verificar que ambos están en mismo rango
// Wi-Fi Direct: debe estar en rango de ~100m o menos

// 4. Retry
for (attempt in 1..5) {
    val peers = p2pManager.discoveredMachineCandidates()
    if (peers.isNotEmpty()) return
    delay(500)
}
```

Base de datos

Error: "VectorDB index corruption"

Síntomas: Búsqueda vectorial falla

```
VectorDBException: Index corrupted - index dimensions mismatch
```

Solución:

```
// Reconstruir índice
repository.rebuildVectorIndex()

// O truncar si es necesario
repository.beneficiaries.deleteAll()
```

Error: "No space left on device"

Síntomas: No puede guardar fotos o entregas

```
java.io.IOException: No space left on device
```

Soluciones:

```
// Verificar espacio disponible
val stat = StatFs(Environment.getDataDirectory().path)
val availableBytes = stat.availableBytes

if (availableBytes < 100_000_000) {  // < 100 MB
    throw Exception("Espacio insuficiente")
}

// Limpiar fotos antiguas
val photosDir = File(context.filesDir, "photos")
photosDir.listFiles()?.forEach { file ->
    if (System.currentTimeMillis() - file.lastModified() >
30.days.inMilliseconds) {
        file.delete()
    }
}
```

Testing

Error: "Test timed out after 10000ms"

Causa: Corrutina no completa

Solución:

```
@Test
fun testDeliverySync() = runBlocking {  // Usar runBlocking
    val result = repository.syncDeliveries()
    assertEquals(true, result)
}

// O alargar timeout
@Test(timeout = 30000)
fun slowTest() { }
```

Error: "MockedObject(derivedFrom(...))"

Causa: Mock no configurado correctamente

Solución:

```
// Configure el mock antes de usar
`when`(mockRepository.getDelivery(1))
    .thenReturn(testDelivery)

// Luego usar
val delivery = mockRepository.getDelivery(1)
```

Logs y Debugging

Buscar error específico

```
# Filtrar por tag
adb logcat | grep "P2P_MANAGER"

# Filtrar por nivel
adb logcat "*:E"  # solo errores

# Con timestamp
adb logcat -v threadtime | grep StateManager

# Caché en archivo
adb logcat > logcat.txt 2>&1
```

Breakpoint en Android Studio

```
1. Click en número de línea (marcar breakpoint)
2. Run → Debug 'AppName'
3. Pausará cuando llegue a línea
4. F9 para continuar
5. F10 for step over
6. F11 for step into
```

Performance

App muy lenta

Causas comunes:

1. **Main thread blocking**: Operations IO/DB en UI thread
2. **Memory leak**: Retaining objetos grandes
3. **Excessive allocations**: Crear muchos objetos
4. **Unoptimized queries**: O(n²) en loops

Debug:

```
# Profiler CPU
adb shell am profile start --sampling 1000 com.package

# Profiler Memory
adb shell dumpsys meminfo com.package

# Traces
adb shell am trace-ipc start
adb shell am trace-ipc stop
```

Checklist de debugging

- [ ] ¿Está el dispositivo/emulador conectado?
- [ ] ¿Android Studio está sincronizado?
- [ ] ¿Build es reciente?
- [ ] ¿Logcat limpio (sin warnings críticos)?
- [ ] ¿Permisos solicitados en runtime?
- [ ] ¿Conectado a internet/Bluetooth?
- [ ] ¿Versión de Android compatible?
- [ ] ¿No hay hardcoded paths incorrectos?

Recursos de ayuda

- Android Developers Troubleshooting
- Gradle troubleshooting
- Common Kotlin issues
- Stack Overflow: tag `android` + descripción del error