

MIGRATION_KOTLIN_2.0

Migración a Kotlin 2.0 y Stack Moderno (AGP 9)

Fecha: 2026-01-23

Stack objetivo: Gradle 8.10.2 + AGP 8.7.3 + Kotlin 2.0.21 + minSdk 24

Cambios Implementados

1. Toolchain (Gradle/AGP/Kotlin)

- ✓ Gradle wrapper: 8.7 → 8.10.2
- ✓ Android Gradle Plugin (AGP): 8.5.2 → 8.7.3
- ✓ Kotlin: 1.9.24 → 2.0.21
- ✓ JDK: mantenido en 17 (requisito de AGP 9)
- ✓ Compose plugin: migrado a org.jetbrains.kotlin.plugin.compose (Kotlin 2+ oficial)

2. SDK Android

- ✓ minSdk: 26 → 24 (ampliada compatibilidad)
- ✓ compileSdk/targetSdk: 34 → 35 (alineado con Play Store 2026)

3. Migración KAPT → KSP (velocidad de build)

- ✓ KSP plugin: añadido 2.0.21-1.0.28
- ✓ Room: migrado de kapt → ksp (compiler)
- ✓ Hilt: migrado de kapt → ksp (compiler)
- ⚡ Ganancia esperada: 20-40% reducción en tiempos de compilación incremental

4. Optimizaciones de Build Speed

Configurado en gradle.properties: - ✓ Memory: 4GB → 6GB JVM heap (-Xmx6144m) - ✓ Configuration cache: activado (AGP 9 lo soporta mejor) - ✓ KSP incremental: activado - ✓ Parallel GC: optimizado para builds rápidos - ✓ R8 full mode: activado para releases más pequeños

5. Actualización Masiva de Dependencias

AndroidX Core

Dependencia	Antes	Ahora	Notas
core-ktx	1.12.0	1.15.0	Stable
appcompat	1.6.1	1.7.0	Stable
material	1.11.0	1.12.0	Stable
lifecycle	2.7.0	2.8.7	Stable

Dependencia	Antes	Ahora	Notas
-------------	-------	-------	-------

activity-compose	1.8.2	1.9.3	Stable
------------------	-------	-------	--------

Compose

Dependencia	Antes	Ahora	Notas
-------------	-------	-------	-------

Compose BOM	2024.02.00	2024.12.01	Compatible Kotlin 2.0
-------------	------------	------------	-----------------------

navigation-compose	2.7.7	2.8.5	Stable
--------------------	-------	-------	--------

Persistencia & Estado

Dependencia	Antes	Ahora	Notas
-------------	-------	-------	-------

room	2.6.1	2.6.1	Sin cambios (última estable)
------	-------	-------	------------------------------

datastore-preferences	1.0.0	1.1.1	Stable
-----------------------	-------	-------	--------

Networking

Dependencia	Antes	Ahora	Notas
-------------	-------	-------	-------

retrofit	2.9.0	2.11.0	⚠️ Revisa breaking changes
----------	-------	--------	----------------------------

okhttp	4.12.0	4.12.0	Sin cambios
--------	--------	--------	-------------

gson	2.10.1	2.11.0	Stable
------	--------	--------	--------

DI & Async

Dependencia	Antes	Ahora	Notas
-------------	-------	-------	-------

hilt	2.51.1	2.54	Con KSP
------	--------	------	---------

coroutines	1.7.3	1.9.0	Stable
------------	-------	-------	--------

Seguridad & Auth

Dependencia	Antes	Ahora	Notas
-------------	-------	-------	-------

credentials	1.2.2	1.5.0	Stable
-------------	-------	-------	--------

biometric	1.1.0	1.2.0	Stable
-----------	-------	-------	--------

security-crypto	1.1.0-alpha06	❌ ELIMINADO	⚠️ Ver migración abajo
-----------------	---------------	-------------	------------------------

Imagen & UI

Dependencia Antes Ahora Notas

coil-compose	2.5.0	2.7.0	Stable
--------------	-------	-------	--------

Testing

Dependencia Antes Ahora Notas

mockito	5.10.0	5.14.2	Stable
---------	--------	--------	--------

androidx.test.ext:junit	1.1.5	1.2.1	Stable
-------------------------	-------	-------	--------

espresso	3.5.1	3.6.1	Stable
----------	-------	-------	--------

ACCIÓN REQUERIDA: Migración de security-crypto (DEPRECADO)

Estado: androidx.security:security-crypto está **oficialmente deprecado** por Google y no recibirá más actualizaciones.

Qué hacer

Opción 1: Android Keystore + EncryptedSharedPreferences replacement

Si usabais EncryptedSharedPreferences o EncryptedFile:

```
// ANTES (con security-crypto)
val masterKey = MasterKey.Builder(context)
    .setKeyScheme(MasterKey.KeyScheme.AES256_GCM)
    .build()

val encryptedPrefs = EncryptedSharedPreferences.create(
    context,
    "secret_prefs",
    masterKey,
    EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,
    EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM
)

// DESPUÉS (con Keystore nativo)
import android.security.keystore.KeyGenParameterSpec
import android.security.keystore.KeyProperties
import javax.crypto.Cipher
import javax.crypto.KeyGenerator
import javax.crypto.SecretKey
import javax.crypto.spec.GCMParameterSpec

object KeystoreHelper {
    private const val KEY_ALIAS = "zabala_master_key"
    private const val ANDROID_KEYSTORE = "AndroidKeyStore"
    private const val TRANSFORMATION = "AES/GCM/NoPadding"

    fun getOrCreateKey(): SecretKey {
        val keyStore = KeyStore.getInstance(ANDROID_KEYSTORE).apply { load(null) }

        if (!keyStore.containsAlias(KEY_ALIAS)) {
```

```

        val keyGenerator = KeyGenerator.getInstance(
            KeyProperties.KEY_ALGORITHM_AES,
            ANDROID_KEYSTORE
        )
        keyGenerator.init(
            KeyGenParameterSpec.Builder(
                KEY_ALIAS,
                KeyProperties.PURPOSE_ENCRYPT or KeyProperties.PURPOSE_DECRYPT
            )
                .setBlockModes(KeyProperties.BLOCK_MODE_GCM)
                .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_NONE)
                .setKeySize(256)
                .build()
        )
        keyGenerator.generateKey()
    }

    return keyStore.getKey(KEY_ALIAS, null) as SecretKey
}

fun encrypt(data: ByteArray): Pair<ByteArray, ByteArray> {
    val cipher = Cipher.getInstance(TRANSFORMATION)
    cipher.init(Cipher.ENCRYPT_MODE, getOrCreateKey())
    val encrypted = cipher.doFinal(data)
    val iv = cipher.iv
    return Pair(encrypted, iv)
}

fun decrypt(encrypted: ByteArray, iv: ByteArray): ByteArray {
    val cipher = Cipher.getInstance(TRANSFORMATION)
    cipher.init(Cipher.DECRYPT_MODE, getOrCreateKey(), GCMParameterSpec(128, iv))
    return cipher.doFinal(encrypted)
}
}

// Uso en DataStore (recomendado para persistencia segura)
class SecurePreferencesRepository(context: Context) {
    private val datastore = context.dataStore

    suspend fun saveSecureString(key: String, value: String) {
        val (encrypted, iv) = KeystoreHelper.encrypt(value.toByteArray())
        datastore.edit { prefs ->
            prefs[stringPreferencesKey("${key}_data")] =
                Base64.encodeToString(encrypted, Base64.NO_WRAP)
            prefs[stringPreferencesKey("${key}_iv")] =
                Base64.encodeToString(iv, Base64.NO_WRAP)
        }
    }

    suspend fun getSecureString(key: String): String? {
        val prefs = datastore.data.first()
        val encryptedStr = prefs[stringPreferencesKey("${key}_data")] ?: return null
        val ivStr = prefs[stringPreferencesKey("${key}_iv")] ?: return null

        val encrypted = Base64.decode(encryptedStr, Base64.NO_WRAP)
        val iv = Base64.decode(ivStr, Base64.NO_WRAP)

        return String(KeystoreHelper.decrypt(encrypted, iv))
    }
}

```

Opción 2: Si solo necesitabais almacenar tokens/credenciales

Usar `androidx.credentials:credentials` (ya actualizado a 1.5.0):

```

// Para tokens de autenticación
val credentialManager = CredentialManager.create(context)

```

```
// Guardar
val credential = PasswordCredential("username", "token")
credentialManager.saveCredential(SavePasswordRequest(credential))

// Recuperar
val getCredRequest = GetCredentialRequest(
    listOf(GetPasswordOption())
)
val result = credentialManager.getCredential(context, getCredRequest)
```

Archivos a revisar

Buscar en el código fuente:

```
cd /home/kalista/erronkak/erronka4/Zabala\ Gailetak/android-app
grep -r "security-crypto" app/src/
grep -r "EncryptedSharedPreferences" app/src/
grep -r "EncryptedFile" app/src/
grep -r "MasterKey" app/src/
```

Pasos Post-Migración

1. Sync y Build Inicial

```
cd /home/kalista/erronkak/erronka4/Zabala\ Gailetak/android-app
```

```
# Actualizar wrapper si no se recoge automáticamente
./gradlew wrapper --gradle-version=8.10.2
```

```
# Limpiar build anterior
./gradlew clean
```

```
# Build inicial (puede tardar por descarga de nuevas dependencias)
./gradlew :app:assembleDebug
```

Notas: - Primera compilación será lenta (descarga AGP 9, Kotlin 2.0.21, nuevas libs) - Configuration cache puede mostrar warnings en primera ejecución (esperado) - KSP genera código en build/generated/ksp/ (nuevo path vs kapt)

2. Verificar Generación de Código KSP

```
# Debe existir código generado para Hilt y Room
ls -la app/build/generated/ksp/debug/kotlin/
```

```
# Verificar que Hilt genera componentes
find app/build/generated/ksp -name "*_HiltComponents*"
```

```
# Verificar que Room genera DAOs
find app/build/generated/ksp -name "*_Impl.kt"
```

3. Ejecutar Tests

```
# Unit tests
./gradlew :app:testDebugUnitTest
```

```
# UI tests (si hay emulador/dispositivo)
./gradlew :app:connectedDebugAndroidTest
```

4. Validar en Android Studio

1. **File** → **Invalidate Caches / Restart** (recomendado después de cambio grande)
2. **Build** → **Rebuild Project**
3. Verificar que no hay errores de sincronización en “Build” tab
4. Comprobar que auto-completion funciona en clases anotadas con Hilt/Room

5. Testing en Dispositivos minSdk 24

Como ahora soportamos **API 24 (Android 7.0)**: - Probar en emulador con API 24 para detectar problemas de compatibilidad - Si usáis APIs Java 8+ (Stream, Optional, Time, etc.), puede que necesitéis **coreLibraryDesugaring**:

```
// En app/build.gradle.kts, si aparecen errores de API Java moderna
android {
    compileOptions {
        isCoreLibraryDesugaringEnabled = true
    }
}

dependencies {
    coreLibraryDesugaring("com.android.tools:desugar_jdk_libs:2.1.4")
}
```

Breaking Changes Potenciales

Retrofit 2.9 → 2.11

- **Call adapters**: Si usabais adapters personalizados, revisar API (normalmente no hay cambios)
- **Converter factories**: Gson converter es compatible

OkHttp (sin cambios de versión)

- Mantenido en 4.12.0 (no hay breaking changes)

Coroutines 1.7 → 1.9

- **Flows**: comportamiento más estricto con cancellation (generalmente mejor)
- **TestDispatchers**: API de testing puede tener cambios menores

Compose BOM 2024.02 → 2024.12

- **Material3**: nuevos componentes disponibles (DatePicker, TimeInput, etc.)
- **Navigation**: NavBackStackEntry.savedStateHandle más robusto

Hilt con KSP

- **Generación de código**: path cambia de build/generated/source/kapt → build/generated/ksp
 - **Incremental builds**: KSP es más rápido pero puede requerir clean si hay problemas iniciales
 - **Logging**: errores de KSP aparecen en formato diferente (generalmente más claro)
-

Rollback Plan

Si hay problemas críticos, revertir a estado anterior:

```
git checkout HEAD~1 -- gradle/wrapper/gradle-wrapper.properties
git checkout HEAD~1 -- build.gradle.kts
git checkout HEAD~1 -- app/build.gradle.kts
git checkout HEAD~1 -- gradle.properties
```

```
./gradlew clean
./gradlew :app:assembleDebug
```

Métricas Esperadas

Build Times (estimación)

- **Clean build:** +10-15% más lento inicialmente (AGP 9 + Kotlin 2.0 overhead)
- **Incremental build** (con KSP): **30-40% más rápido** vs KAPT
- **Configuration cache hit:** hasta **50% más rápido** en re-builds sin cambios

APK Size

- **R8 full mode:** 5-10% reducción esperada en release
- **Compose:** sin cambios significativos

Compatibilidad

- **Dispositivos adicionales:** ~5% más usuarios alcanzables con minSdk 24
-

Recursos y Referencias

- [AGP 9.0.0 Release Notes](#)
 - [Kotlin 2.0 Release](#)
 - [Compose Kotlin Compatibility](#)
 - [KSP Documentation](#)
 - [Android Keystore System](#)
 - [security-crypto Deprecation](#)
-

Contacto y Soporte

Para dudas sobre esta migración: - Revisar issues de build en Android Studio Build Output - Consultar logs de Gradle con --info o --debug flags - Documentación del proyecto: [AGENTS.md](#)

Última actualización: 2026-01-23