Ambiente de desarrollo y sus elementos de configuración

Elementos de un proyecto Android (Parte II)

{desafío} latam_

Implementar una interfaz de usuario utilizando buenas prácticas en el manejo de estilos para brindar un aspecto visual e interacciones acordes a lo requerido

- **Unidad 1**: Ambiente de desarrollo y sus elementos de configuración.
- **Unidad 2:** Elementos de la interfaz, navegación e interacción.
- **Unidad 3**: Fundamentos de GIT y GitHub.

Te encuentras aquí

{desafío} latam_

¿Qué aprenderás en esta sesión?

Identifica los elementos de configuración principales de un proyecto Android

{desafío} latam_

¿Qué vimos en la clase anterior?

{desafío} latam_

/* Principales funciones de los scripts de compilación */

Scripts de compilación

¿Qué son?

Son archivos activos, programables que ejecutan tareas repetitivas como compilar el código fuente y los recursos, para luego empaquetar en un archivo distribuible.

- En Android, estos scripts de compilación permiten compilar y empaquetar en un archivo APK para probar, implementar, firmar y distribuir.
- En Android, se ocupa **Gradle** como **sistema de automatización de construcción**.
- Gradle permite programar tareas para su automatización mediante scripts escritos en un lenguaje de programación.

/* Gradle*/

Gradle

Características generales

- Es un sistema de automatización de construcción.
- Diseñado para construcciones multi-proyecto.
- Utiliza un lenguaje especializado para su propósito (Groovy).
    - En el caso de Android también se puede programar en Kotlin, lo que entrega gran versatilidad.
- Es de código abierto.

/* Gradle en Android*/

Android Gradle Plugin

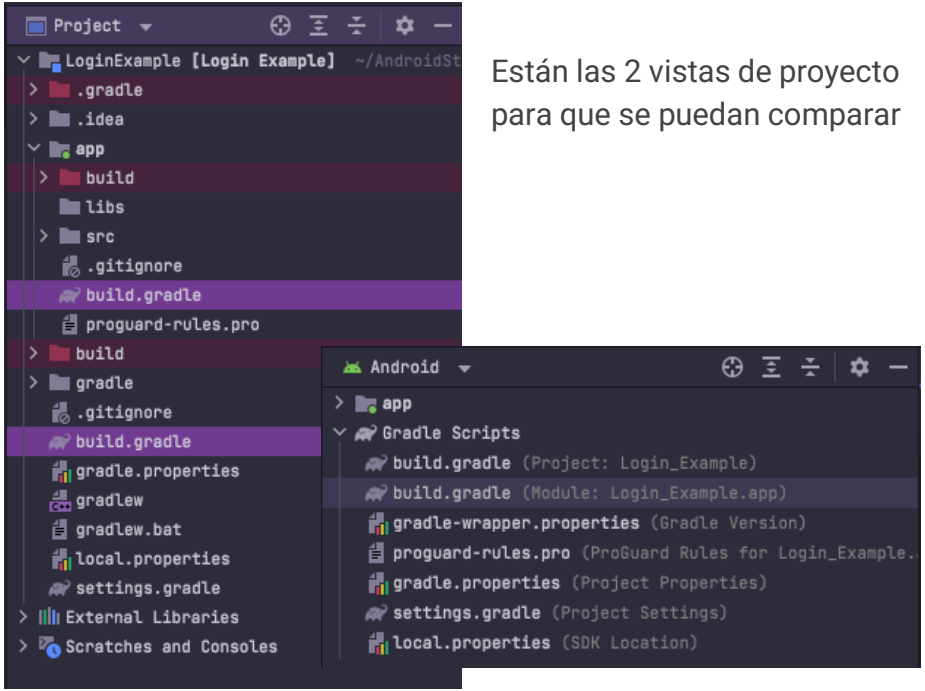
- El Android Gradle Plugin agrega funcionalidades propias del desarrollo Android.
- Se actualiza en conjunto con Android Studio.
- Incluye tareas automatizadas para, por ejemplo, ejecutar test unitarios, o construir un APK de depuración.
- Su estructura es con un proyecto general que contiene subproyectos o módulos.
- Para eso, tiene el archivo **build.gradle** en la carpeta raíz del proyecto.

Gradle de módulo

Por defecto, módulo app

Al crear un nuevo proyecto, Android Studio crea por defecto un módulo llamado **app**. Este módulo contiene la aplicación Android y su propio archivo de configuración build.gradle.

Solo se debe manipular el archivo **`build.gradle` del módulo**. NO se debe manipular el archivo **`build.gradle` del proyecto**.



Están las 2 vistas de proyecto para que se puedan comparar

build.gradle

build.gradle del proyecto

Este archivo **build.gradle** está escrito en Groovy, que es un metalenguaje y permite definir tareas y sus configuraciones, que se organiza por tareas o secciones con objetivos específicos.

La definición de los valores se hace indicando la configuración que se quiere y el valor, separado por espacio, donde el valor acepta un texto (con comillas simples o dobles) y números (sin comillas).

Por ejemplo:

```
id 'com.android.application'
compileSdk 32
```

```
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
}

android {
    compileSdk 32

    defaultConfig {
        …
    }

    buildTypes {
        …
    }
    compileOptions {
        …
    }
    …
}

dependencies {
    …
}
```

{desafío} latam_

Secciones de build.gradle del módulo

- plugins
- android
  - defaultConfig
  - buildTypes
  - compileOptions
- dependencies

```
plugins {
   id 'com.android.application'
   id 'org.jetbrains.kotlin.android'
}

android {

   defaultConfig {
        …
   }

   buildTypes {
        …
   }
   compileOptions {
        …
   }
   …
}

dependencies {
   …
}
```

{desafío} latam_

Android
plugins

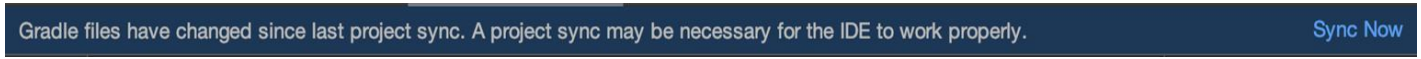
Incluye plugins con funcionalidades para la configuración de Gradle, por ejemplo, para integrar Kotlin con Gradle.

En general se mantiene de esta forma.

```
plugins {
   id 'com.android.application'
   id 'org.jetbrains.kotlin.android'
}

android { … }
```

NOTA: Cualquier modificación en el archivo build.gradle requiere volver a sincronizar, donde aparece una notificación en la parte superior que nos permite hacerlo.



Android

SDK version

- **compileSdk:** Es la versión que utiliza gradle para compilar el código. No afecta el comportamiento en tiempo de ejecución y no se empaqueta con la app.
- **minSdk:** Es la versión mínima que necesita la app para funcionar. Es uno de los criterios que utiliza la Google Play para discriminar qué apps mostrar a un usuario
- **targetSdk:** La app fue probada usando esta versión. Si el dispositivo donde se instala tiene una versión más nueva al targetSdk, Android se encarga de no utilizar características nuevas para mantener la compatibilidad

```
android {
    compileSdk 32

    defaultConfig {
        applicationId "cl.dal.loginexample"
        minSdk 27
        targetSdk 32
        versionCode 1
        versionName "1.0"
    }

    buildTypes {
        …
    }
    …
}
```

Android
defaultConfig

- **applicationId:** Es un texto que se agrega antes del nombre de la app para formar un nombre único de la app. Una vez que publiques tu app, no debes cambiar el ID de aplicación.

- **versionCode:** Número entero positivo utilizado como versión interna. Solo se usa para determinar si una versión es más reciente que otra.

- **versionName:** Es un texto que se emplea como número de versión y que se utiliza exclusivamente para mostrarselo a los usuarios.

```
plugins { }

android {
   compileSdk 32

   defaultConfig {
       applicationId "cl.dal.loginexample"
       minSdk 27
       targetSdk 32
       versionCode 1
       versionName "1.0"
   }
   …
}
```

/* Conocer las build variants por defecto */

Variables de compilación

¿Qué son?

Permiten crear diferentes versiones de una app usando el mismo proyecto.

Cada una representa una versión diferente de una app. Por ejemplo, para tener una versión gratuita y otra versión pagada que incluye más funcionalidades.

Además, se pueden compilar versiones distintas para dispositivos según el nivel de API, por ejemplo, una versión específica para los dispositivos desde Android 11 hacia arriba.

Build Variants

defaultConfig

La configuración por defecto **(defaultConfig)** se aplica a todas las variantes de compilación.

Por ejemplo, todas las variantes de compilación utilizan como minSdk el especificado en 27.

```
android {
    compileSdk 32

    defaultConfig {
        applicationId "cl.dal.loginexample"
        minSdk 27
        targetSdk 32
        versionCode 1
        versionName "1.0"
    }

    buildTypes {
        …
    }
    …
}
```

Debug

Variable de compilación para depuración

Es la variable de compilación que se incluye por defecto y no viene incluida en el archivo build.gradle.

Se puede definir y agregar características especiales, como por ejemplo, agregarle el **.debug** al nombre del archivo APK usando **applicationIdSuffix ".debug"**

```
android {
   compileSdk 32
   …

   defaultConfig {
       applicationId "cl.dal.loginexample"
       minSdk 27
       targetSdk 32
       versionCode 1
       versionName "1.0"
   }

   buildTypes {
      release { }

      debug {
         minifyEnabled false
         debuggable true
         applicationIdSuffix ".debug"
      }
   }
}
```

Release

Variable de compilación para producción

Esta variable de compilación se utiliza para compilar el código para producción.

En general, a las aplicaciones de producción no se les permite que sean debuggeables por temas de seguridad.

```
android {
   compileSdk 32
   …

   defaultConfig {
       applicationId "cl.dal.loginexample"
       minSdk 27
       targetSdk 32
       versionCode 1
       versionName "1.0"
   }

   buildTypes {
      release {
         minifyEnabled true
         debuggable false
      }

      debug { }
   }
}
```

Personalizadas

Se pueden crear variables de compilación personalizadas, por ejemplo, para staging que corresponde a la etapa de pruebas previa al lanzamiento a producción.

```
android {
   compileSdk 32
   …

   defaultConfig {
       applicationId "cl.dal.loginexample"
       minSdk 27
       targetSdk 32
       versionCode 1
       versionName "1.0"
   }

   buildTypes {
      release { }

      debug { }

      staging {
        initWith debug
        applicationIdSuffix ".debugStaging"
      }
   }
}
```

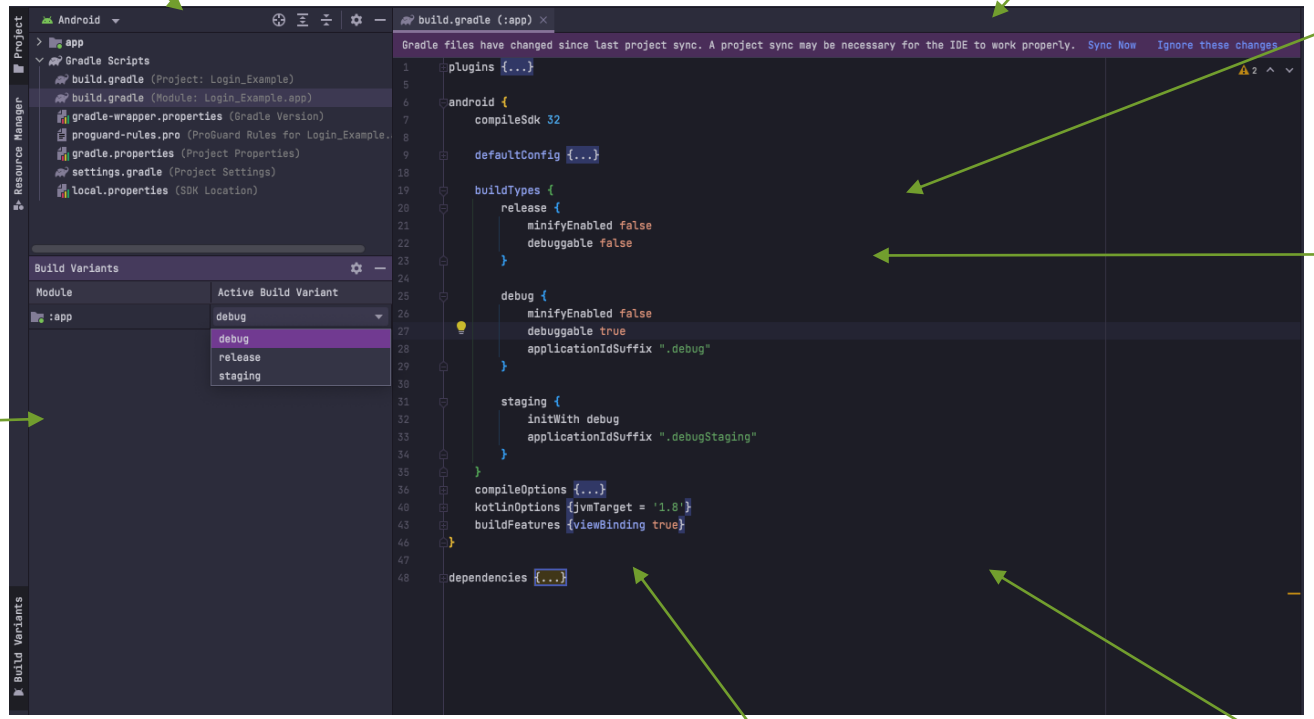
Vista de proyecto tipo Android

Luego de modificar el archivo build.gradle se debe sincronizar el proyecto

Configuración común para las variables de compilación

Variables de compilación definidas en el archivo build.gradle del módulo

Elegir variable de compilación usando el IDE



Variables de compilación definidas en el archivo build.gradle

Dependencias externas

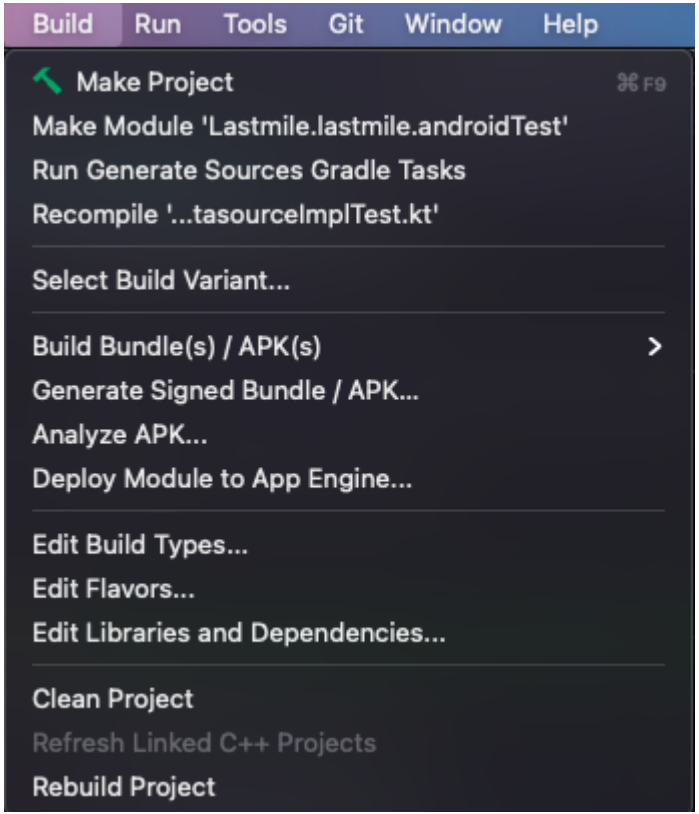
{desafío} latam_

Menú Build

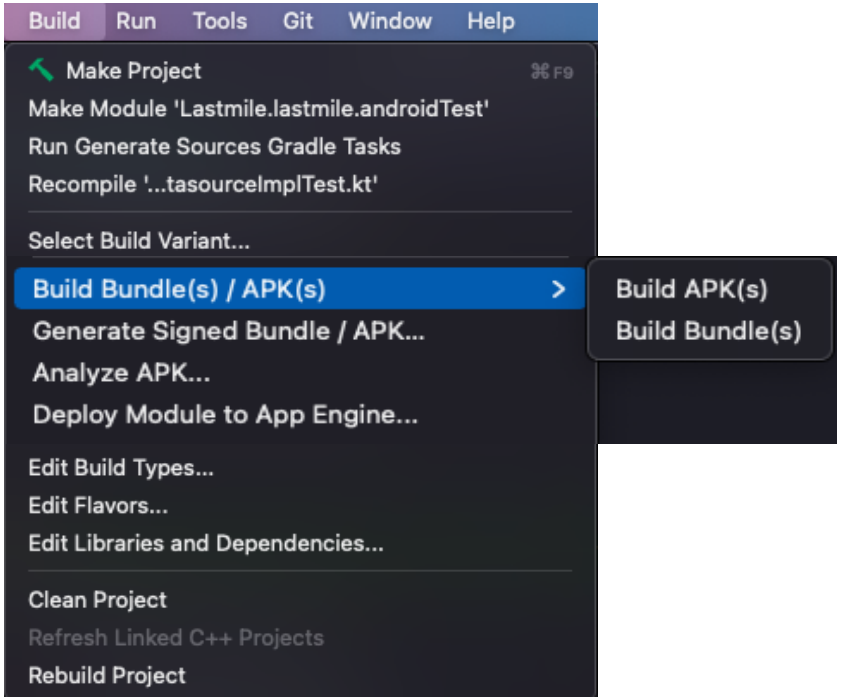
El menú Build agrupa la construcción del proyecto.
Make Project: Construye el proyecto

Clean Project: Elimina todo lo generado en la compilación, para empezar de cero.

Rebuild Project: Aplica Clean Project y Make Project.



Menú Build



Se puede construir manualmente un APK o Bundle dependiendo del Build Variant seleccionado.

También se puede construir un APK/Bundle firmado, que se utiliza para producción.

Luego de generado el APK, Android Studio muestra una notificación con la acción.

{desafío} latam_

/* Dependencias */

Dependencias

¿Qué son?

Una dependencia es una biblioteca o módulo externo en nuestro proyecto.
Las dependencias nos permiten incluir funcionalidades externas, que se desarrollan de forma separada a nuestro código.

Hay bibliotecas para distintos objetivos. A continuación, algunos ejemplos:

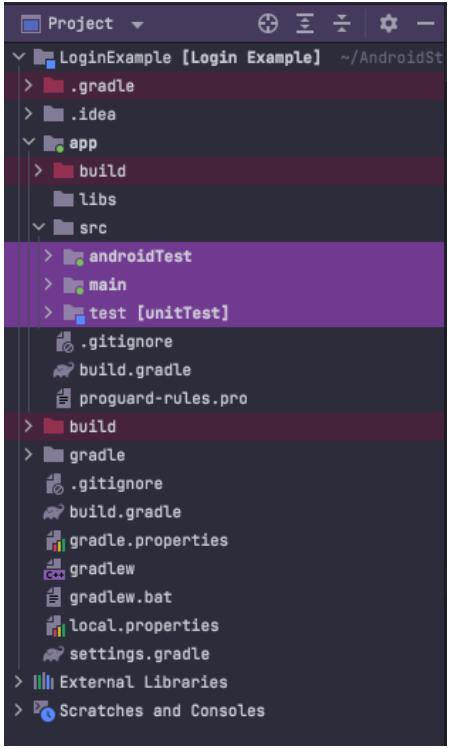
- Retrofit: Para facilitar el consumo de información por internet.
- Timber: Para facilitar y mejorar los Logs.
- Coil: Para cargar y mostrar imágenes.

Dependencias

Source set

Se utilizan 3 Source Set, o carpetas que contienen código.

- **main:** Contiene el código de la app, el que es compartido para todas las diferentes versiones de la app (build variants).
- **androidTest:** Contiene tests que se ejecutan en un dispositivo, conocidos como *instrumented tests*.
- **tests:** Contiene tests unitarios o locales. No necesitan un dispositivo para ejecutarse, por lo que no dependen directamente de Android para ejecutarse.



Dependencias

En esta sección se agregan las dependencias externas o bibliotecas (libraries).

Se agrupan en 3 tipos:

- implementation
- testImplementation
- androidTestImplementation

Pueden existir más de una dependencia por tipo.

```
android {
    …
}

dependencies {

    implementation 'androidx.core:core-ktx:1.7.0'
    implementation 'androidx.appcompat:appcompat:1.4.2'
    implementation 'com.google.android.material:material:1.6.1'
    implementation 'androidx.annotation:annotation:1.4.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    implementation 'androidx.lifecycle:lifecycle-livedata-ktx:2.5.1'
    implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.5.1'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-
core:3.4.0'
}
```

Dependencias

Al indicar un tipo de dependencia se indica el alcance, o sea, en cuáles source set estará disponible.

	main	android	tests
implementation	SI	SI	SI
testImplementation	NO	NO	SI
androidTestImplementation	NO	SI	NO

Dependencias

Agregar una dependencia directamente a build.gradle

Se puede agregar las dependencias directamente en esta sección, editando el archivo **`build.gradle del módulo`**.

```
android {
   …
}

dependencies {

   implementation 'androidx.core:core-ktx:1.7.0'
   implementation 'androidx.appcompat:appcompat:1.4.2'
   implementation 'com.google.android.material:material:1.6.1'
   implementation 'androidx.annotation:annotation:1.4.0'
   implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
   implementation 'androidx.lifecycle:lifecycle-livedata-ktx:2.5.1'
   implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.5.1'
   testImplementation 'junit:junit:4.13.2'
   androidTestImplementation 'androidx.test.ext:junit:1.1.3'
   androidTestImplementation 'androidx.test.espresso:espresso-
core:3.4.0'
}
```

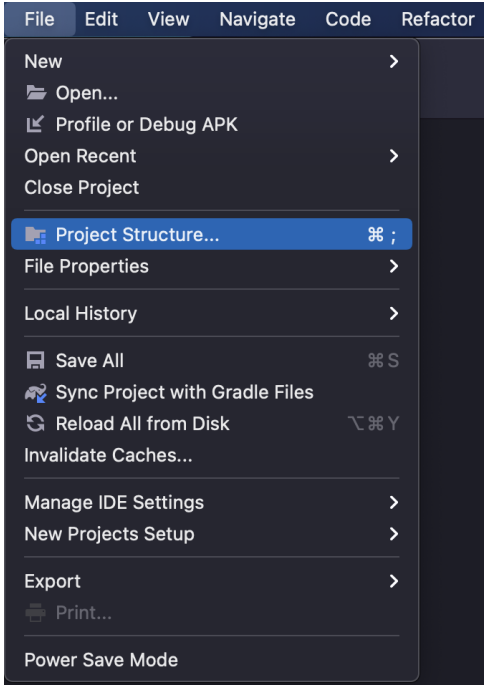
Dependencias

Agregar una dependencia usando Android Studio

Otra forma de gestionar las dependencias es utilizando Android Studio.

File → Project Structure...

No es más que una interfaz gráfica que modifica el archivo **build.gradle del módulo**.

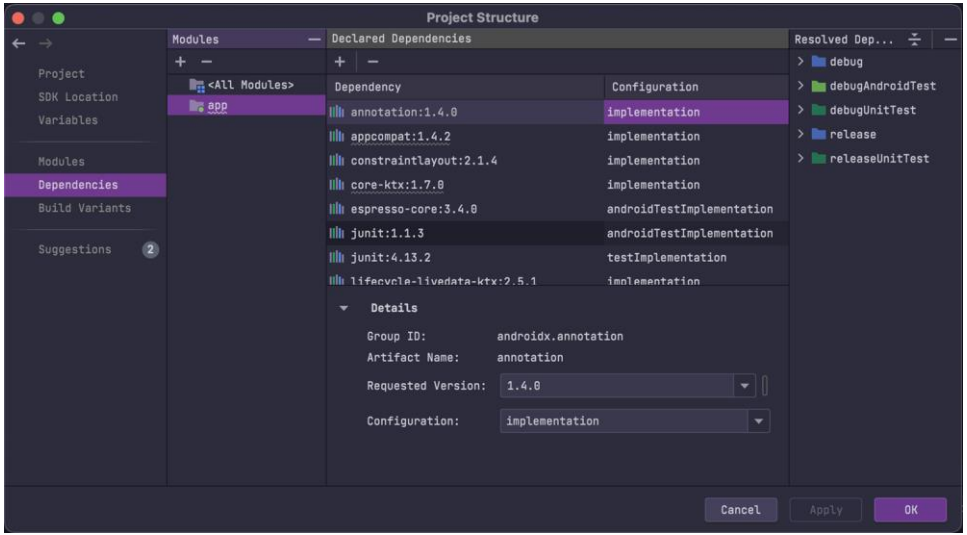


Dependencias

Usando Android Studio

Se puede ver parte de los componentes configurados para este proyecto.

En particular, nos interesa Dependencies, que permite administrar (agregar, quitar o modificar) las dependencias.

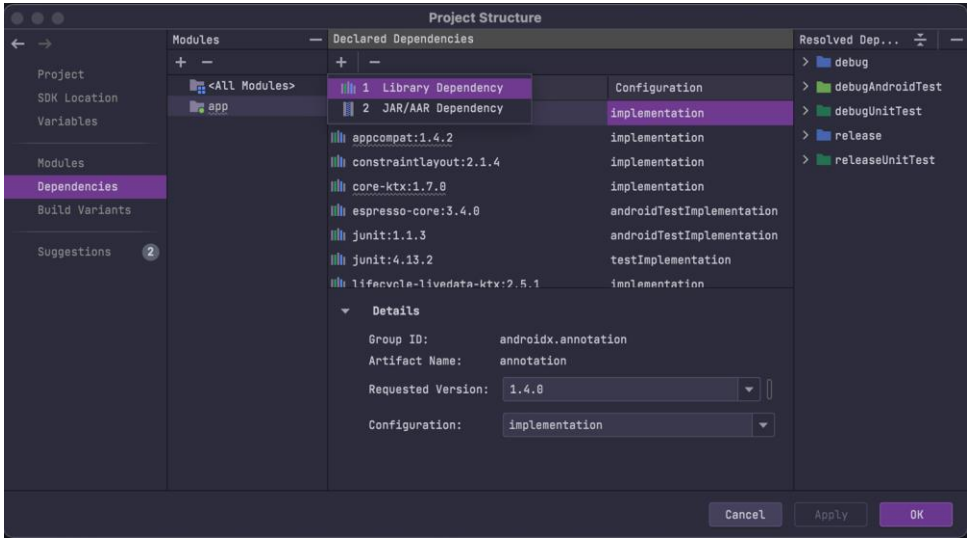


{desafío} latam_

Dependencies

Usando Android Studio

Al seleccionar el módulo app y luego presionar el botón + asociado a las dependencias declaradas (Declared Dependencies) se permite agregar una nueva dependencia directamente desde internet usando Library Dependency.



{desafío} latam_

Dependencies

Usando Android Studio

Debemos seguir los 2 pasos indicados:

1. Buscar la dependencia por nombre.

1. Indicar a qué tipo de configuración se agrega la dependencia.

{desafío} latam_

Actividad práctica

Dependencias

Timber es un módulo con un logger utilitario sobre el class Log de Android.

En el apartado Download, se puede ver como agregar la dependencia al archivo build.gradle del proyecto.

```
dependencies {
  implementation 'com.jakewharton.timber:timber:5.0.1'
}
```

La configuración de Timber (Timber.plant()) queda a criterio propio. GITHUB

¿Cuáles son los pros / contras de utilizar dependencias externas?

¿Cuáles criterios emplearías para elegir una biblioteca externa?

Próxima sesión...

Desafío evaluado - Ambiente de desarrollo y sus elementos de configuración (I)

{desafío} latam_

{desafío}
latam_

Academia de
talentos digitales