DUIX.AI开源应用开发手册

简介：

本文档旨在为开发者提供基于duix.ai开源项目完整对接流程示例。在开发过程中，您可以参考各个环节的配置、优化及注意事项。希望这份文档能为您的开发工作带来便捷，共同推动duix.ai项目的应用与发展。

一、开发环境

项目	描述
系统	MacOS 13.6.6
CPU架构	M2 ARM64
开发 IDE	Android Studio Preview 2024.1

二、Android环境

1、构建工具

项目	版本
Gradle	8.7

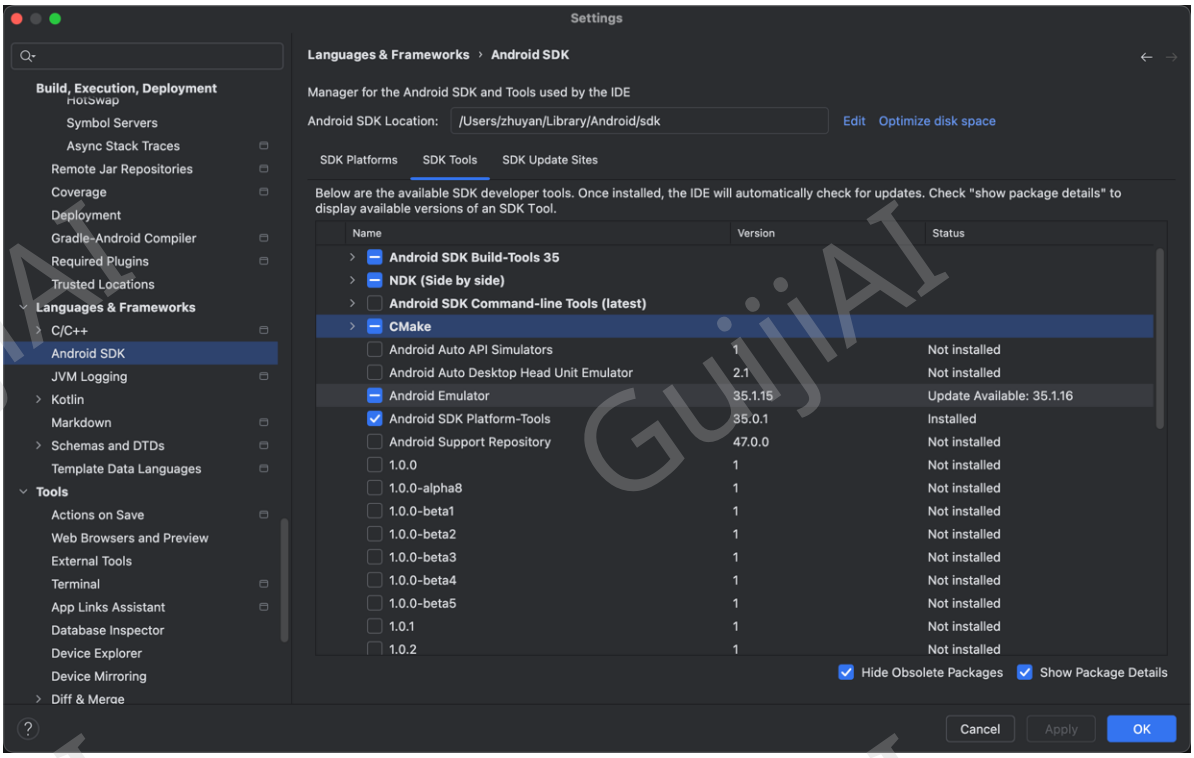
2、SDK platforms

项目	版本
Android API 35	35.0.0
Android 9.0 ("Pie")	9.0

3、SDK Tools

项目	版本
Android SDK Build-Tools 35	35.0.0
NDK (side by side)	27.0.11902837 rc2
CMake	3.22.1
Android Emulator	35.1.15
Android SDK Platform-Tools	35.0.1

环境安装如下：



附上Android环境安装配置教程

https://blog.csdn.net/a910247/article/details/138012201

三、项目准备

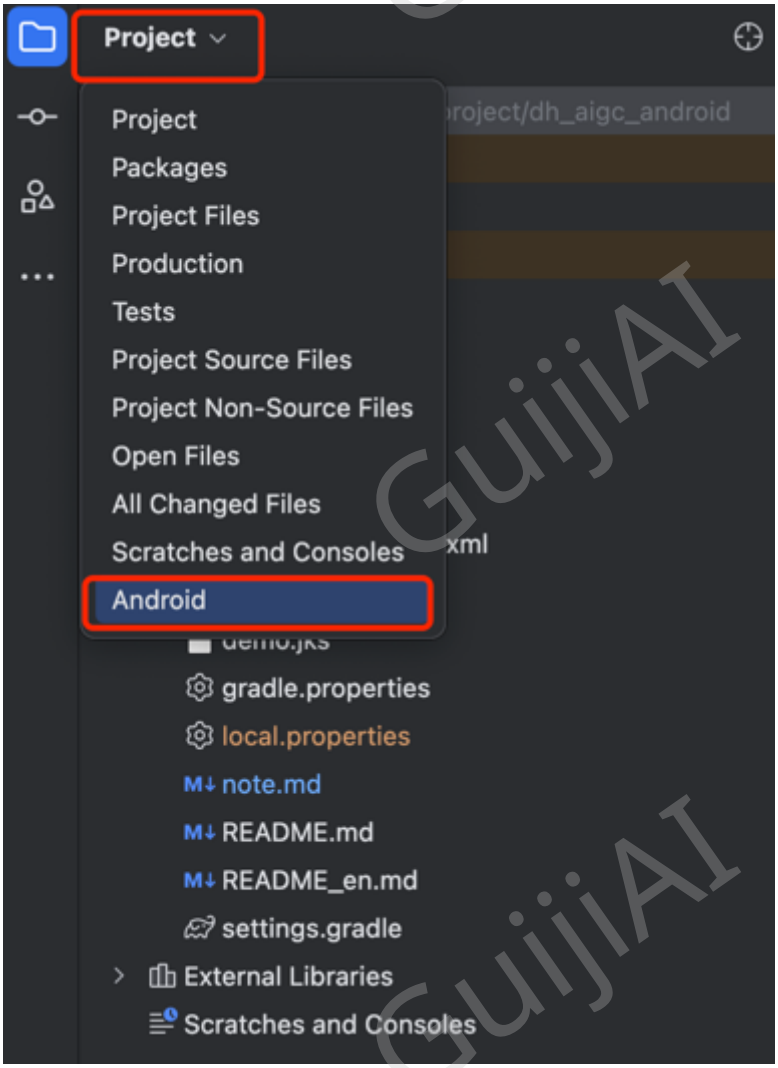
下载地址：https://github.com/GuijiAI/duix.ai

Git 克隆地址：https://github.com/GuijiAI/duix.ai.git

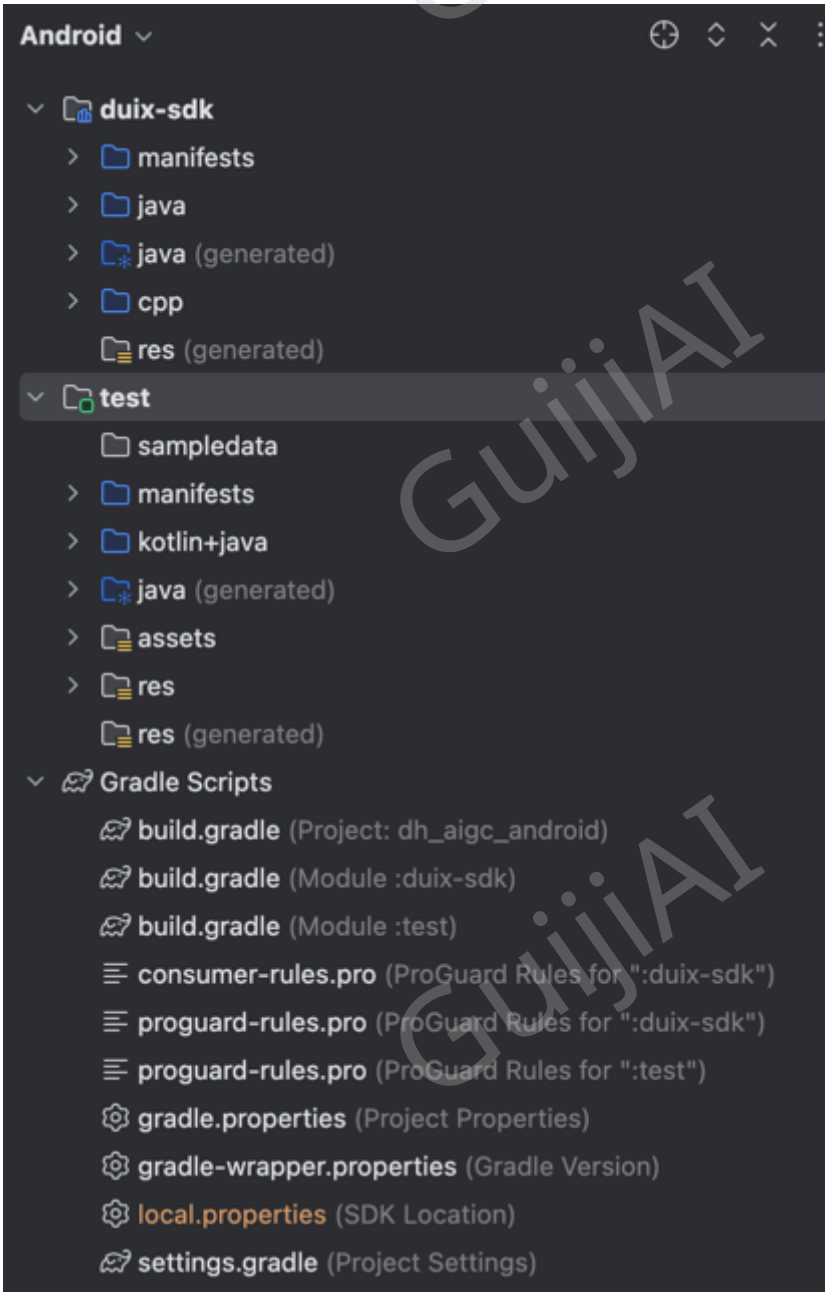
使用Android Studio导入项目dh_aigc_android

等待gradle环境和依赖包下载...

依赖下载完毕后切换Android视图


Project
Project
Packages
Project Files
Production
Tests
Project Source Files
Project Non-Source Files
Open Files
All Changed Files
Scratches and Consoles
Android
gradle.properties
local.properties
note.md
README.md
README_en.md
settings.gradle
External Libraries
Scratches and Consoles

切换之后项目结构如下



项目包含两个子工程

1、duix-sdk工程

duix数字人渲染源码，作为test项目的依赖包

build.gradle(Module:duix-sdk) 配置如下

```
plugins {
    id 'com.android.library'
}
```

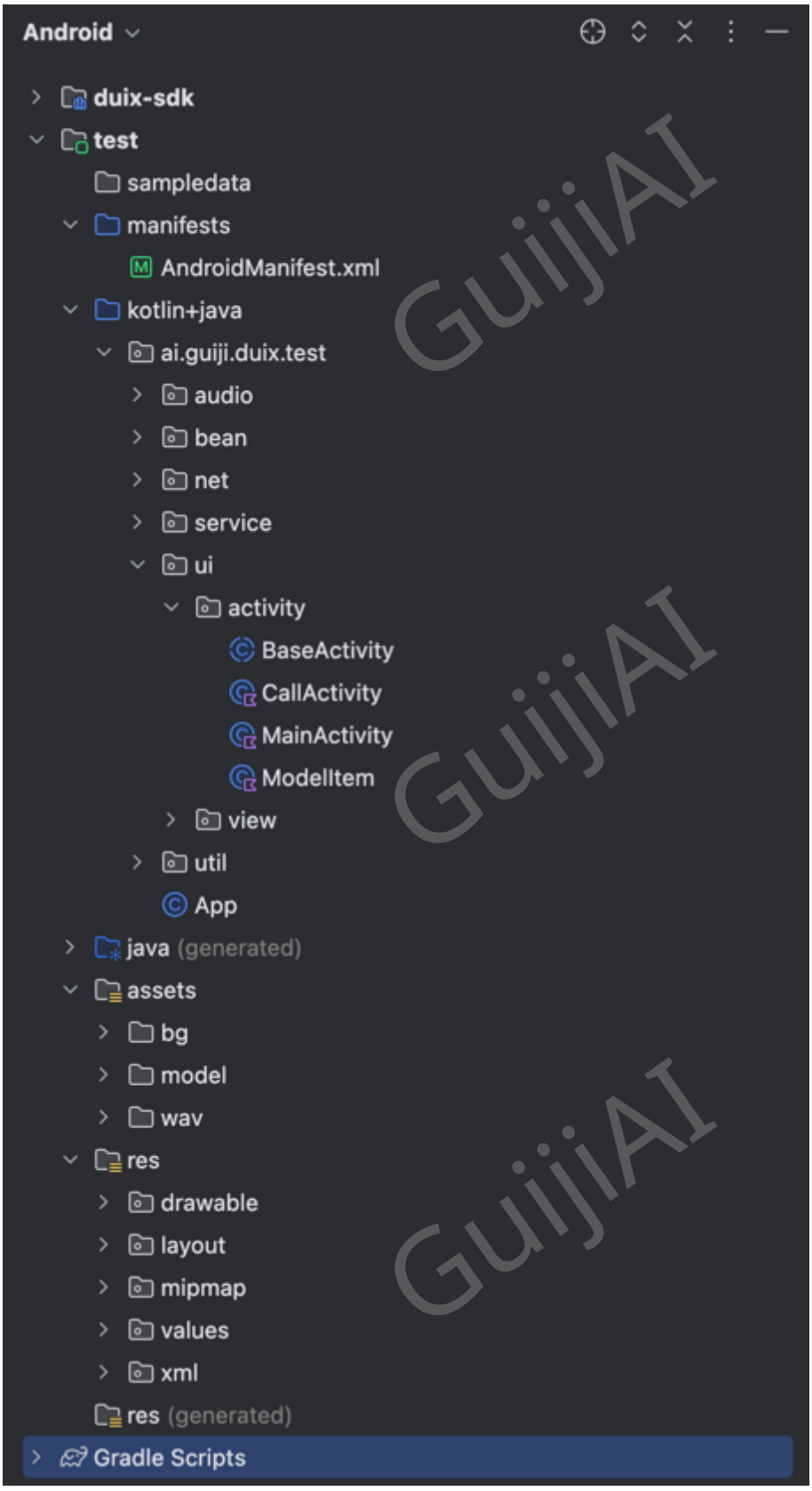
这里的duix-sdk源码我们不用动，demo项目依赖已经帮我们引用好了

可以在test.build.gradle中查看引用

```
implementation project(":duix-sdk")
```

2、test工程

android应用demo工程，这里也是我们需要开发的app工程，工程结构如下：



build.gradle(Module:test) 配置如下

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

在 build.gradle(Module:test) 中增加依赖

```
dependencies {
    // 引用SDK项目依赖
    implementation project(":duix-sdk")
    // sdk 中使用到 exoplayer 处理音频(必选)
    implementation 'com.google.android.exoplayer:exoplayer:2.14.2'

    // http请求依赖
    implementation 'com.squareup.okhttp3:okhttp:4.10.0'
    implementation 'com.squareup.okhttp3:okhttp-sse:4.10.0'
    ...
}
```

android权限要求, AndroidManifest.xml中,增加如下配置

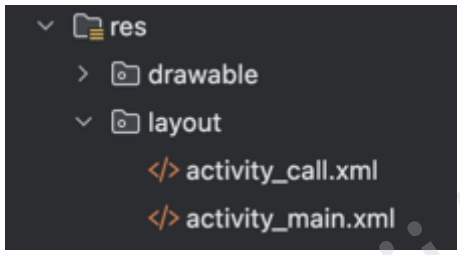
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- 录音权限 -->
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
    <!-- 存储权限 -->
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

</manifest>
```

3、 Activity

这里我们只介绍最基本的Activity，其他Activity请自行实现。

每个Activity对应一个xml的ui配置，可以定义ui界面控件



BaseActivity

定义了一个基础的抽象Activity，作为渲染的基类，主要实现了onCreate()、onDestroy()、onPause()、onResume()等生命周期方法。

MainActivity

app的主Activity，主要实现了数字人形象资源下载和加载已经加载的完成性检测。

CallActivity

数字人形象的加载和渲染，主要实现了数字人形象的加载、渲染、播放、暂停、销毁等操作。

四、SDK调用及API说明

1、 数字人形象构建

使用RenderSink接口接受渲染帧数据，SDK中提供了该接口实现DUIXRenderer.java。也可以自己实现该接口自定义渲染。
其中RenderSink的定义如下:

使用DUIXRenderer及DUIXTextureView控件简单实现渲染展示,该控件支持透明通道可以自由设置背景及前景:

```
    // ...
    mDUIXRender =
        DUIXRenderer(
            mContext,
            binding.glTextureView
        )

    binding.glTextureView.setEGLContextClientVersion(GL_CONTEXT_VERSION)
    binding.glTextureView.setEGLConfigChooser(8, 8, 8, 8, 16, 0) // 透明
    binding.glTextureView.isOpaque = false           // 透明
    binding.glTextureView.setRenderer(mDUIXRender)
    binding.glTextureView.renderMode = GLSurfaceView.RENDERMODE_WHEN_DIRTY
// 一定要在设置完Render之后再调用
```

2. duix对象构建

在渲染页onCreate()阶段构建DUIX对象并添加回调事件

```
duix = DUIX(mContext, baseDir, modelDir, mDUIXRender) { event, msg, info ->
    when (event) {
        ai.guiji.duix.sdk.client.Constant.CALLBACK_EVENT_INIT_READY -> {
            initOK()
        }

        ai.guiji.duix.sdk.client.Constant.CALLBACK_EVENT_INIT_ERROR -> {

        }
        // ...

    }
}
// 异步回调结果
duix?.init()
```

DUIX对象构建说明:

参数	类型	描述
context	Context	系统上下文
baseDir	String	存放模型驱动的配置文件,需要自行管理. 可将压缩文件解压到外部存储并提供文件夹路径

参数	类型	描述
modelDir	String	存放模型文件的文件夹,需要自行管理. 可将压缩文件解压到外部存储并提供文件夹路径
render	RenderSink	渲染数据接口，sdk提供了默认的渲染组件继承自该接口，也可以自己实现
callback	Callback	SDK处理的各种回调事件

参考test中 ai.guiji.duix.test.ui.activity.CallActivity示例

3. 启动数字人播报

在初始化成功后，可以播放音频以驱动形象

```
duix?.playAudio(wavPath)
```

参数说明:

参数	类型	描述
wavPath	String	16k采样率单通道16位深的wav本地文件

4. 终止当前播报

当数字人正在播报时调用该接口终止播报。

调用示例如下:

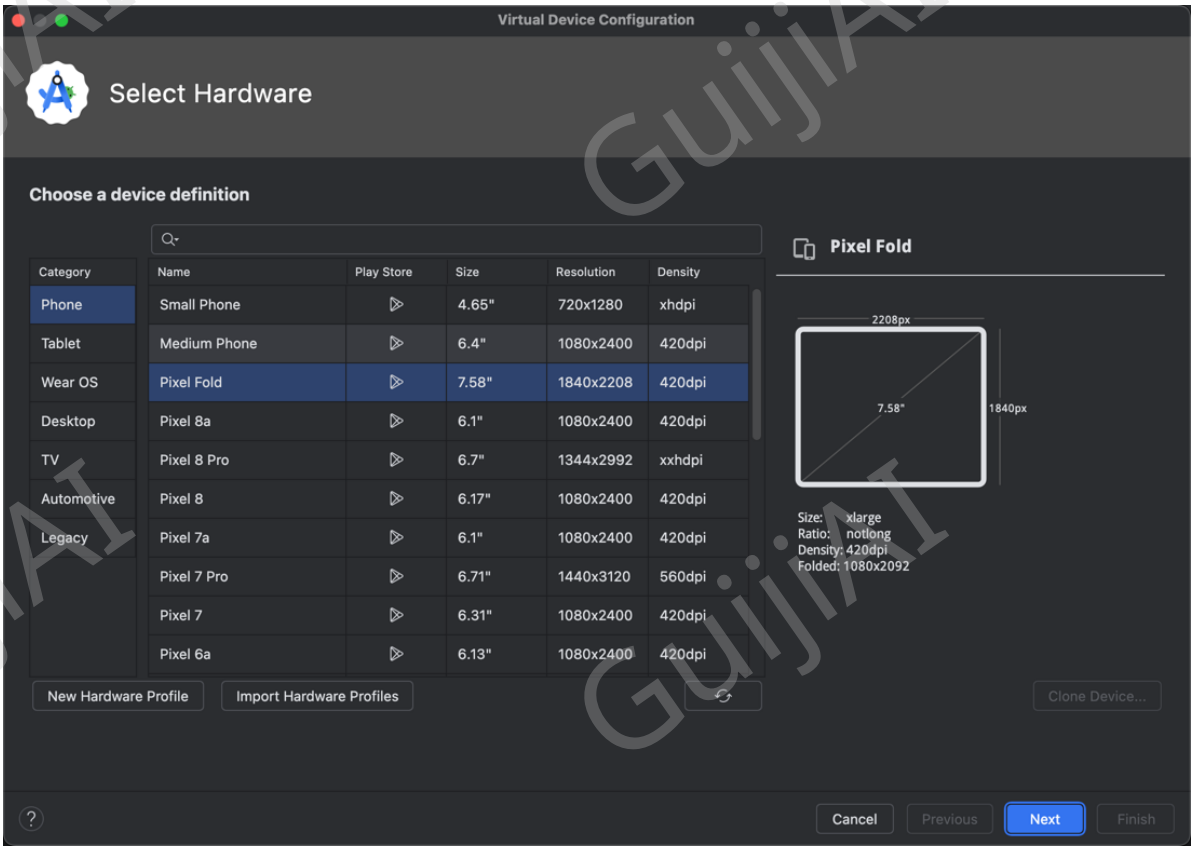
```
duix?.stopAudio()
```

五、项目运行

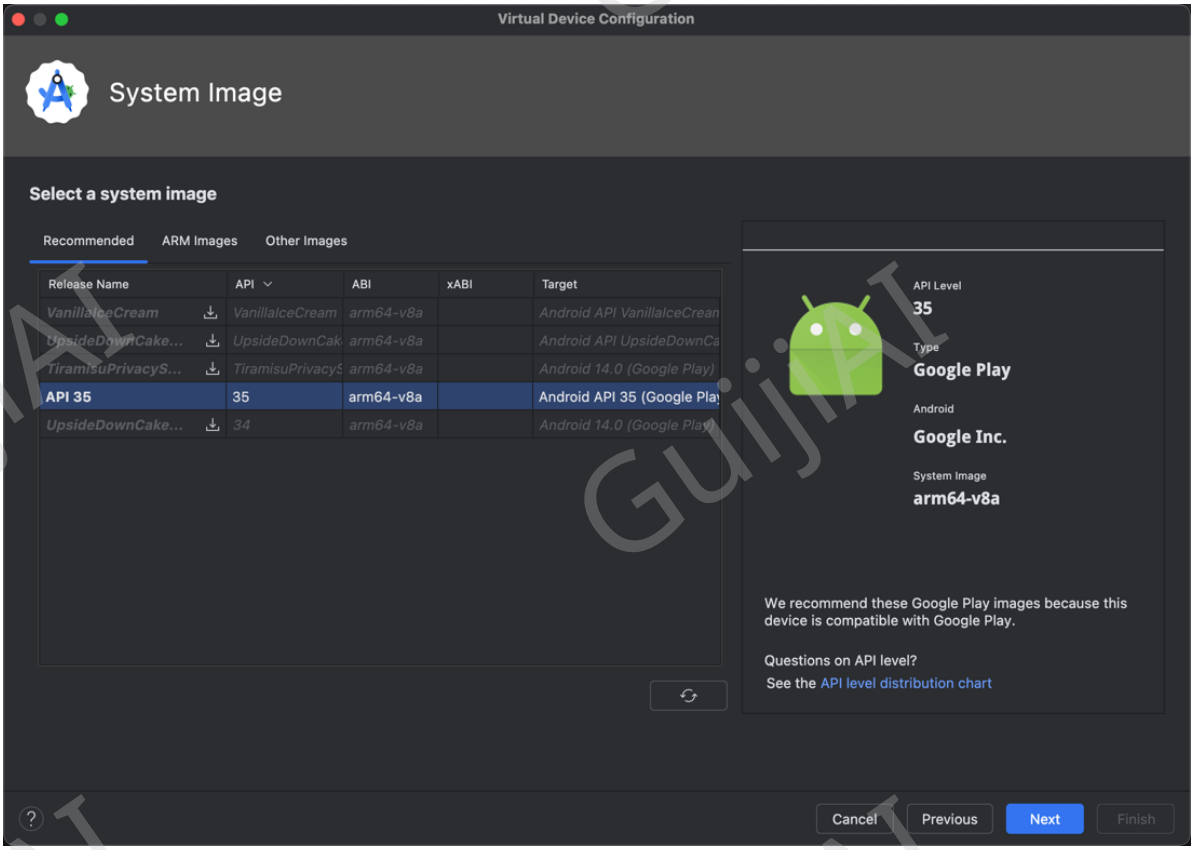
1、创建虚拟机

点击+添加一个虚拟机

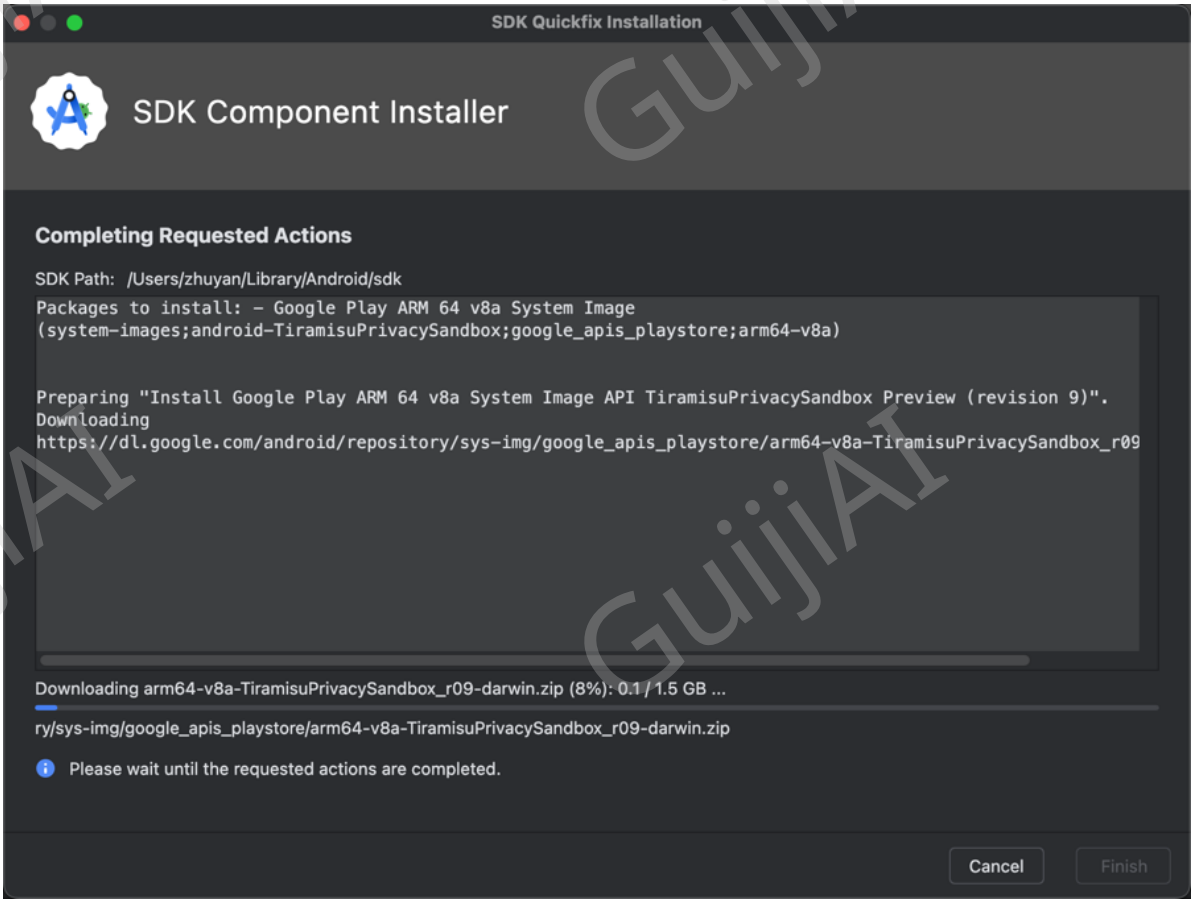
选择一个设备



选择虚拟机镜像，第一次需要下载镜像（选择跟你cpu匹配的镜像）



等待镜像下载

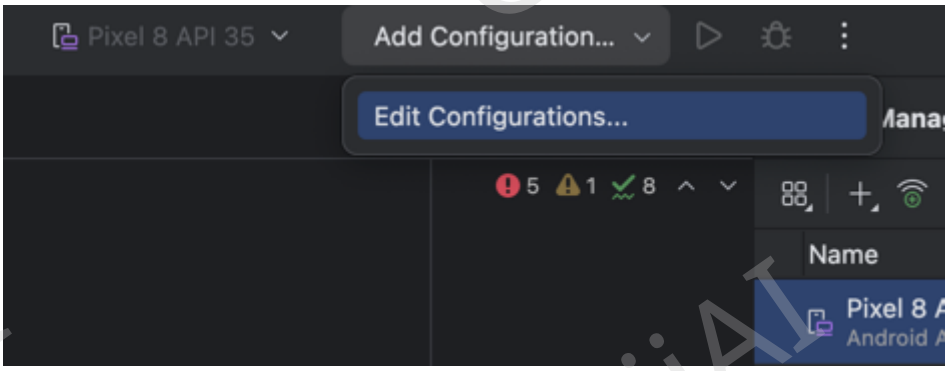


设置虚拟机参数

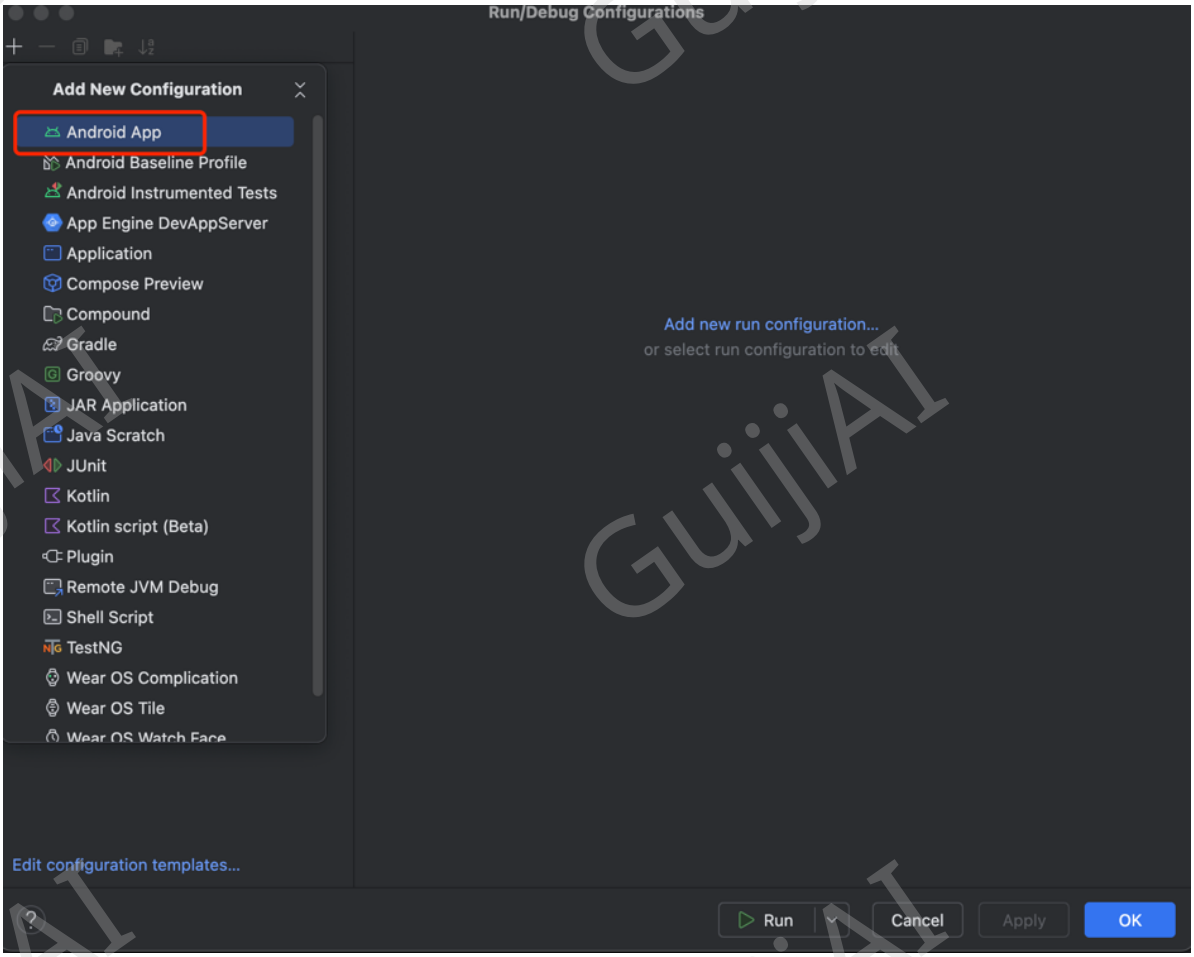
完成虚拟机配置

2、启动项目

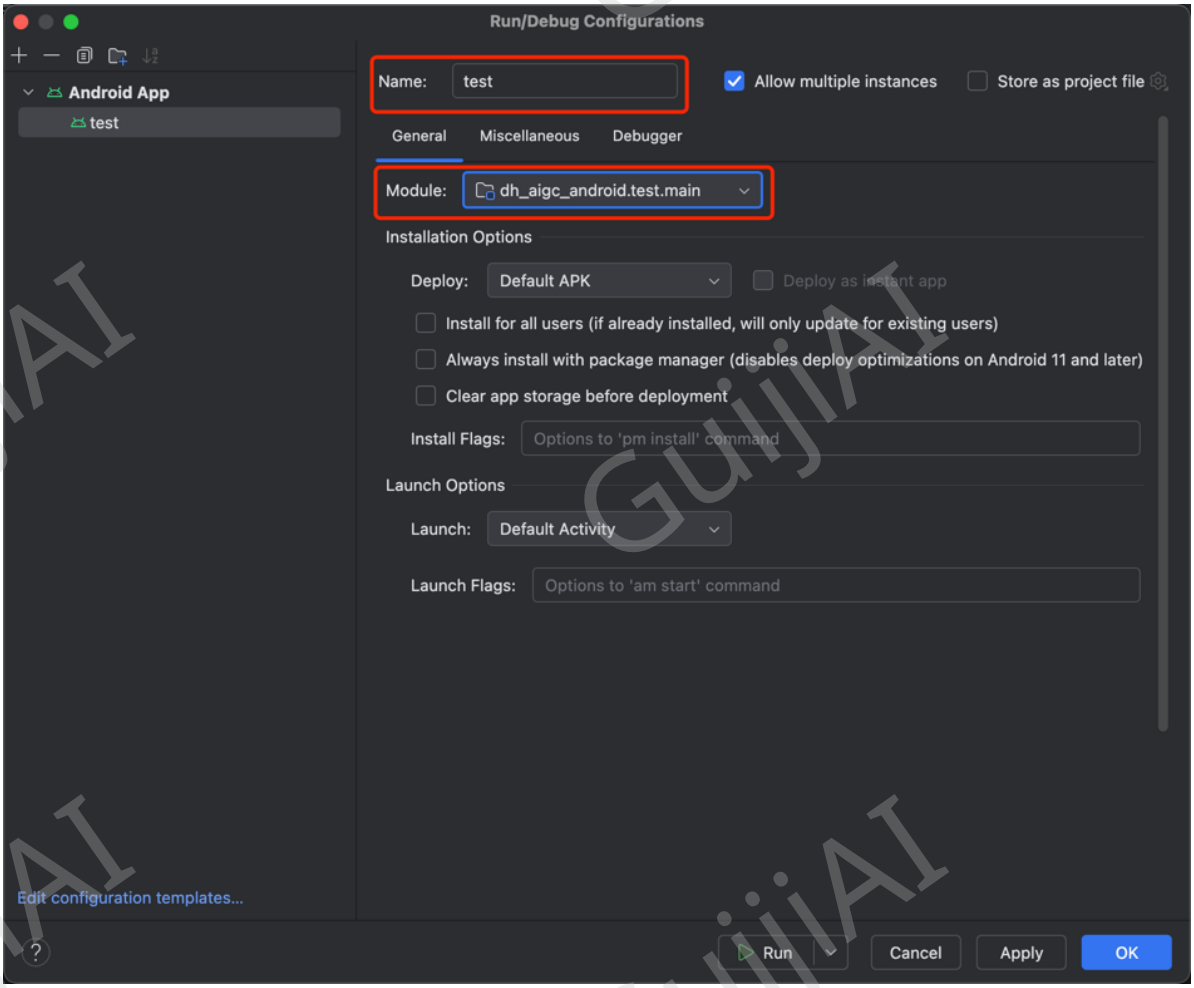
添加一个Configuration



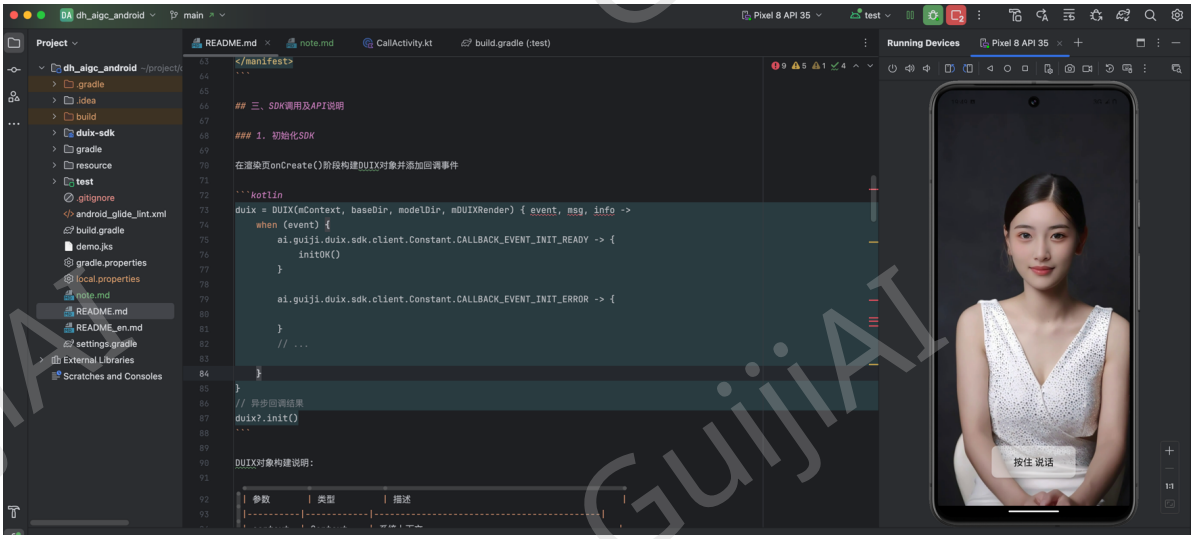
添加一个Android APP



填写项目名称、选择Module


Run/Debug Configurations
Android App
test
Name: test
Allow multiple instances
Store as project file
General
Miscellaneous
Debugger
Module: dh_aigc_android.test.main
Installation Options
Deploy: Default APK
Deploy as instant app
Install for all users (if already installed, will only update for existing users)
Always install with package manager (disables deploy optimizations on Android 11 and later)
Clear app storage before deployment
Install Flags: Options to 'pm install' command
Launch Options
Launch: Default Activity
Launch Flags: Options to 'am start' command
Edit configuration templates...
Run
Cancel
Apply
OK

启动项目


Project
README.md
note.md
CallActivity.kt
build.gradle (:test)
Running Devices
Pixel 8 API 35
按住 说话