Configuración del Entorno de Desarrollo

Requisitos del sistema

Hardware mínimo

- **CPU**: Intel i5 / AMD Ryzen 5 o equivalente
- **RAM**: 8 GB (recomendado 16 GB)
- **Almacenamiento**: 50 GB SSD libre (Android SDK + emulador)
- **GPU**: GPU integrada suficiente

Software

- **OS**: Windows 10+, macOS 10.15+, o Linux (Ubuntu 20.04+)
- **JDK**: OpenJDK 17 (requerido por Android)

Instalación

1. Instalar JDK 17

Windows

```
# Descargar desde adoptopenjdk.net o usar chocolatey
choco install openjdk17

# Verificar
java -version
# openjdk version "17.0.x"
```

macOS

```
brew install openjdk@17

# Configurar JAVA_HOME
echo 'export PATH="/usr/local/opt/openjdk@17/bin:$PATH"' >> ~/.zshrc
source ~/.zshrc
```

Linux

```
sudo apt-get install openjdk-17-jdk
```

```
java -version
```

2. Instalar Android Studio

Descargar desde developer.android.com/studio

Primera ejecución:

```
1. Welcome → Next
2. Standard Installation (recomendado)
3. Seleccionar path de Android SDK
   (Típicamente: ~/Android/sdk o %APPDATA%\Android\sdk)
4. Descargar componentes (Android 15, emulador, etc.)
```

3. Configurar Android SDK

Abrir SDK Manager

```
Android Studio → Settings → Appearance & Behavior → System Settings →
Android SDK
```

Instalar componentes necesarios

- **API 27-36**: Compilación + testing
- **Emulador**: si se va a usar
- **Build-tools 36.0.1**: para compilar
- **Git**: para control de versiones

Instalación vía CLI:

```
# Aceptar licencias
yes | sdkmanager --licenses

# Instalar específicos
sdkmanager "platforms;android-36"
sdkmanager "build-tools;36.0.1"
sdkmanager "system-images;android-36;google_apis;arm64-v8a"
```

4. Instalar Git

Windows

Descargar de git-scm.com

```
git --version
```

macOS

```
brew install git

git --version
```

Linux

```
sudo apt-get install git

git --version
```

Configuración de Android Studio

Configurar JDK

```
Settings → Build, Execution, Deployment → Gradle
  → Gradle JDK: Configurado por IDE (17)

O especificar manualmente:
GRADLE_JDK=17
```

Gradle wrapper

El proyecto incluye gradle wrapper:

```
# Windows
gradlew.bat --version

# macOS / Linux
./gradlew --version
# Gradle 8.0 (recomendado)
```

Memory para Gradle

```
gradle.properties (crear si no existe):

org.gradle.jvmargs=-Xmx4g
org.gradle.parallel=true
org.gradle.workers.max=8
```

Plugins recomendados en Android Studio

```
Settings → Plugins:

- Kotlin
- Git
- GitHub Copilot (opcional)
- Detekt (linting)
- ktlint (formatting)
```

Clonar repositorio

SSH (recomendado)

```
# Generar clave SSH (si no existe)
ssh-keygen -t ed25519 -C "tu_email@example.com"

# Agregar a GitHub
cat ~/.ssh/id_ed25519.pub
# Copiar a GitHub → Settings → SSH Keys

# Clonar
git clone git@github.com:AdaCompanyTeam/PAE.git
cd PAE
```

HTTPS (alternativa)

```
git clone https://github.com/AdaCompanyTeam/PAE.git
cd PAE
```

Configurar proyecto en Android Studio

1. Abrir proyecto

```
File → Open → seleccionar directorio PAE
```

Android Studio sincronizará automáticamente.

2. Sync y build inicial

```
# Manualmente
```

```
./gradlew clean build

# O desde Android Studio
Build → Clean Project
Build → Rebuild Project
```

3. Crear local.properties

```
# local.properties (en raíz del proyecto)
sdk.dir=/path/to/Android/sdk

# Ejemplos:
# Windows:   C:/Users/USERNAME/AppData/Local/Android/sdk
# macOS:     /Users/USERNAME/Library/Android/sdk
# Linux:     /home/USERNAME/Android/sdk
```

Emulador Android

Crear AVD (Android Virtual Device)

Terminal

```
# Listar imágenes disponibles
sdkmanager --list

# Crear AVD
avdmanager create avd \
    --name "Nexus_5_API_36" \
    --package "system-images;android-36;google_apis;arm64-v8a" \
    --device "Nexus 5"

# Listar AVDs creadas
avdmanager list avd
```

Android Studio

```
Tools → AVD Manager
  → Create Virtual Device
    → Nexus 5 (compatibilidad)
    → API 36 (lat versión)
    → Finish
```

Ejecutar emulador

```
# Listar emuladores
emulator -list-avds

# Ejecutar
emulator -avd Nexus_5_API_36

# Con opciones
emulator -avd Nexus_5_API_36 -no-audio -gpu auto
```

Acelerar emulador

```
# Habilitar KVM (Linux)
sudo apt-get install qemu-kvm

# Hardware acceleration
emulator -avd Nexus_5_API_36 -accel on -gpu auto

# En gradle.properties
android.enableEmulatorDisplay1=true
android.qemuid.display_type=FRAME
```

Conectar dispositivo físico

Vía USB

Habilitar Developer Mode (Dispositivo Android)

```
Settings → About phone
  → Tap "Build number" 7 veces
  → Developer options aparecen
  → Settings → Developer options
    → USB debugging: ON
```

Conectar

```
# Windows / macOS / Linux
adb devices

# Si pide permiso, aceptar en dispositivo

adb devices
```

```
  # Listar conectados
  # emulator-5554 device
  # 192.168.x.x:5555 device
```

Vía WiFi (Network ADB)

```
# En dispositivo: Settings → Developer options
#   → TCP/IP debugging: puerto 5555

# En host
adb connect 192.168.x.x:5555

# Verificar
adb devices
```

Ejecutar app

Debug en emulador

```
# Via Android Studio
Run → Run 'Machine' (o RutaPAE)
  → Seleccionar emulador/dispositivo

# Via terminal
./gradlew :Machine:installDebug
adb shell am start co.ada.paemachine/.MainActivity
```

Debug en dispositivo físico

```
# Mismo proceso: detecta dispositivo automáticamente
./gradlew :Machine:installDebug
```

Run con logs

```
# Terminal 1: seguir logs
adb logcat

# Terminal 2: run app
./gradlew :Machine:installDebug
adb shell am start co.ada.paemachine/.MainActivity
```

Herramientas útiles

adb (Android Debug Bridge)

```
# Información dispositivo
adb shell getprop ro.build.version.release  # Android version

# Instalar APK
adb install app-debug.apk

# Desinstalar
adb uninstall co.ada.paemachine

# Shell interactivo
adb shell
  > pm list packages
  > ls /data/data/co.ada.paemachine/

# Forward puerto
adb forward tcp:8080 tcp:8080

# Pull archive
adb pull /data/data/co.ada.paemachine/files/photo.jpg

# Push archivo
adb push photo.jpg /data/data/co.ada.paemachine/files/
```

Android Device Monitor

```
# File system explorer
Device File Explorer (Android Studio)
  Tools → Device File Explorer

# Screenshots
adb shell screencap -p /sdcard/screenshot.png
adb pull /sdcard/screenshot.png
```

Logcat

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

# Filtrar por nivel
adb logcat *:E  # Solo errores
```

```
# Con timestamp
adb logcat -v threadtime

# Exportar a archivo
adb logcat > logs.txt 2>&1

# Limpiar logs
adb logcat -c
```

Variables de entorno

Windows

```
REM Agregar a Environment Variables:
JAVA_HOME=C:\Program Files\openjdk-17
ANDROID_SDK_ROOT=%APPDATA%\Android\sdk
ANDROID_HOME=%APPDATA%\Android\sdk
PATH=...;%JAVA_HOME%\bin;%ANDROID_SDK_ROOT%\platform-
tools;%ANDROID_SDK_ROOT%\emulator

REM Verificar
echo %JAVA_HOME%
echo %ANDROID_HOME%
```

macOS / Linux

```
# En ~/.bashrc, ~/.zshrc, o ~/.profile:

export JAVA_HOME=/usr/libexec/java_home -v 17)  # macOS
export JAVA_HOME=/usr/lib/jvm/java-17-openjdk   # Linux
export ANDROID_SDK_ROOT=$HOME/Android/sdk
export ANDROID_HOME=$ANDROID_SDK_ROOT

export PATH=$PATH:$JAVA_HOME/bin
export PATH=$PATH:$ANDROID_SDK_ROOT/platform-tools
export PATH=$PATH:$ANDROID_SDK_ROOT/emulator
export PATH=$PATH:$ANDROID_SDK_ROOT/cmdline-tools/latest/bin

# Aplicar cambios
source ~/.bashrc
# o
source ~/.zshrc
```

Verificación de setup

```
#!/bin/bash
# setup-check.sh

echo "=== Verificar Setup PAE ==="

echo -n "Java version: "
java -version 2>&1 | head -1

echo -n "Git version: "
git --version

echo -n "Gradle version: "
./gradlew --version

echo -n "Android SDK: "
[ -d "$ANDROID_SDK_ROOT" ] && echo "OK" || echo "NOT FOUND"

echo -n "JAVA_HOME: "
echo $JAVA_HOME

echo -n "Emulator: "
emulator -version | head -1

echo "=== All set! ==="
```

Ejecutar:

```
chmod +x setup-check.sh
./setup-check.sh
```

Troubleshooting setup

"gradle daemon stopped"

```
./gradlew --stop

# Limpiar caches
rm -rf ~/.gradle/caches
rm -rf .gradle
```

"SDK path corrupted"

```
Android Studio → Settings → Appearance & Behavior → System Settings →
Android SDK
  → Edit → Seleccionar ruta correcta
```

```
  → Apply
```

"Java version incompatible"

```
# Forzar JDK 17
export JAVA_HOME=/usr/libexec/java_home -v 17  # macOS
export JAVA_HOME=/usr/lib/jvm/java-17-openjdk  # Linux
java -version  # Debe mostrar 17.x
```

IDE Tips

Keyboard shortcuts (Android Studio)

Comando	Windows/Linux	macOS
Run	Shift+F10	Ctrl+R
Debug	Shift+F9	Ctrl+D
Build	Ctrl+F9	Cmd+F9
Gradle sync	(manual)	(manual)
Search	Ctrl+F	Cmd+F
Replace	Ctrl+H	Cmd+H
Go to class	Ctrl+N	Cmd+O
Go to file	Ctrl+Shift+N	Cmd+Shift+O
Go to line	Ctrl+G	Cmd+G

Code generation

```
Posicionarse en clase:
  → Code → Generate...
    → Getters/Setters
    → Equals/HashCode
    → toString()
    → Constructor
```

CI/CD Local

Ejecutar todos los checks

```
# Clean + build + test
./gradlew clean build connectedCheck

# O por partes
./gradlew test                    # Unit tests
./gradlew connectedAndroidTest    # Android tests
./gradlew lint                    # Lint check
```

```
./gradlew :Machine:assembleDebug  # Build debug
```