

# Ambiente de desenvolvimento Android

---

Roteiro de configuração e preparação do ambiente de trabalho Android em uma máquina com o sistema operacional Linux.

## Objetivo

- Descrever a arquitetura do sistema operacional Android;
- Configurar o ambiente Linux para sincronizar o Android Open Source Project (AOSP);
- Validar a funcionalidade básica do ambiente configurado.

Nas seções a seguir serão descritos os passos para cada configuração.

## Sumário

1. [Estação de Trabalho](#)
  - 1.1. [Requisitos de Hardware](#)
  - 1.2. [Requisitos de Software](#)
  - 1.3. [DualBoot](#)
  - 1.4. [Máquinas Utilizadas](#)
2. [Ferramentas do Sistema](#)
  - 2.1. [Atualizando Pacotes do Sistema](#)
  - 2.2. [Instalação das Ferramentas Necessárias](#)
    - 2.2.1. [Git](#)
    - 2.2.2. [Java](#)
    - 2.2.3. [Python](#)
    - 2.2.4. [Curl](#)
3. [Instalação do VS Code](#)
4. [Configuração de Espaço Adicional de Memória \(Swap\)](#)
5. [Android Open Source Project](#)
  - 5.1. [Pacotes Necessários](#)
  - 5.2. [repo](#)
  - 5.3. [Sincronização do Código-Fonte](#)
  - 5.4. [Compilação do Sistema Operacional](#)
    - 5.4.1. [Configuração das variáveis de ambiente](#)
    - 5.4.2. [Definição da Plataforma Alvo](#)
    - 5.4.3. [Build](#)
    - 5.4.4. [Teste do Build](#)
6. [Android Studio](#)
  - 6.1. [Instalação](#)
  - 6.2. [Emulador](#)
  - 6.3. [Extra](#)
    - 6.3.1. [Adicionando Play Store](#)
    - 6.3.2. [Alterando o tamanho da RAM](#)
  - 6.4. [Instalando APP no Dispositivo Virtual](#)
7. [Referências](#)

# 1. Estação de Trabalho

Nesta etapa, iremos descrever as configurações de hardware recomendadas, além da preparação para a instalação do sistema operacional.

## 1.1. Requisitos de Hardware

A [documentação\[1\]](#) oficial do [Android Open Source Project\(AOSP\)](#) recomenda uma máquina com pelo menos [400GB](#) de espaço livre em disco, sendo 250 GB para verificação + 150 GB para criação. Com o mínimo de [64GB](#) de RAM e um processador com pelo menos [6 núcleos](#).

Estes seriam valores ideias para se ter uma performance satisfatória durante o desenvolvimento com o [Android OSP](#), mas vimos que é possível trabalhar normalmente utilizando uma máquina com um pouco menos de recursos computacionais, sendo que o que vai influência de fato, seria o tempo de build e a excursão do emulador(Com o uso da virtualização).

-  Requisitos mínimos de hardware:
  - Processador Intel ou AMD de 64-bits (4+ CPUs).
  - 16GB+ de RAM.
  - 256GB+ de armazenamento disponível.

## 1.2. Requisitos de Software

Com relação aos requisitos de software, a [documentação\[1\]](#) do [Android OSP](#) pede que seja um sistema operacional [Linux x86\\_64 bits](#). O sistema operacional escolhido para acompanhar o curso foi o [Ubuntu 22.04.5 LTS](#) por já trabalharmos com ele. Neste [link\[2\]](#) pode ser baixada a imagem de instalação do sistema operacional Ubuntu.

As etapas de [instalação do sistema operacional](#) podem ser encontradas neste [link\[3\]](#), está descrito desde a preparação da mídia inicializável até as etapas de particionamento do sistema operacional.

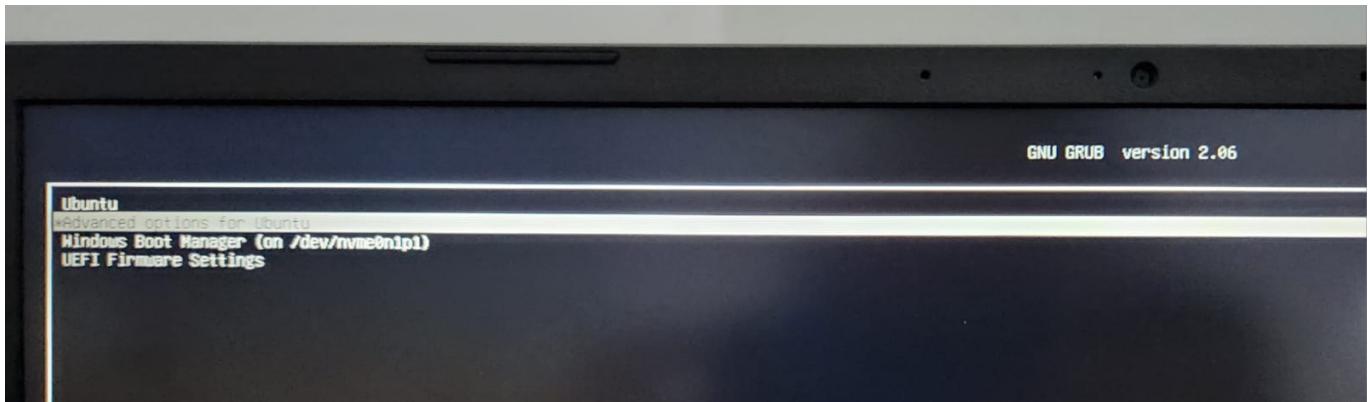
 **Nota** Para máquinas com recursos computacionais limitados, a alternativa encontrada foi a instalação e utilização do [Lubuntu 22.04\[4\]](#), uma variante mais leve que o Ubuntu.

 **Atenção:** Ter a instalação nativa do Linux é obrigatória, já que o uso de uma máquina virtual impacta significativamente os builds do Android e a execução do emulador.

## 1.3. DualBoot

Nossas estações de trabalho estão configuradas em [DualBoot](#) onde o sistema operacional [Ubuntu](#) e [Windows](#) foram instalados lado a lado, a escolha de qual sistema operacional será inicializado ocorre no momento de boot com a ferramenta [GRUB\[5\]](#) que é responsável por gerenciar a inicialização dos sistemas operacionais.

A Figura 1 abaixo mostra como é a tela do GRUP, a foto foi tirada no momento de boot da máquina.



**Figura 1:** Foto da tela de gerenciamento do GRUP

[link\[6\]](#) do processo de instalação do *dual boot* com **Windows e o Ubuntu**.

**Nota:** Para a montagem da mídia inicializável foi utilizado o software [Rufus\[7\]](#).

**Dica:** Este vídeo [\[8\]](#) descreve bem o passo a passo do processo de instalação, achamos bem didático e objetivo.

#### 1.4. Máquinas utilizadas

No Linux existem várias formas de obter informações do sistema, segue abaixo os comandos utilizados e suas respectivas saídas para cada máquina.

Máquina do Leandro:

- Sistema Operacional

```
# O comando lsb_release imprime certas informações de LSB (Linux Standard Base) e distribuição.  
$ lsb_release -a  
No LSB modules are available.  
Distributor ID: Ubuntu  
Description: Ubuntu 22.04.5 LTS  
Release: 22.04  
Codename: jammy
```

- Hardware

```
# Informações da CPU  
$ lscpu  
Architecture:           x86_64  
CPU op-mode(s):        32-bit, 64-bit  
Address sizes:         39 bits physical, 48 bits virtual  
Byte Order:            Little Endian  
CPU(s):                12  
On-line CPU(s) list:   0-11  
Vendor ID:             GenuineIntel  
Model name:            12th Gen Intel(R) Core(TM) i5-12450H
```

```

CPU family:          6
Model:              154
Thread(s) per core: 2
Core(s) per socket: 8
Socket(s):          1
Stepping:           3
CPU max MHz:       4400,0000
CPU min MHz:       400,0000
...
# Informações da RAM
$ free -h
              total        used        free      shared  buff/cache available
Mem:        23Gi       4,8Gi      12Gi      929Mi      5,4Gi   17Gi
Swap:       14Gi          0B      14Gi
# Informações do Armazenamento
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
...
/dev/nvme1n1p2  457G  413G   22G  96% /
...

```

Máquina do Jorge:

- Sistema Operacional

```

# O comando lsb_release imprime certas informações de LSB (Linux Standard
Base) e distribuição.
$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description: Ubuntu 22.04.5 LTS
Release: 22.04
Codename: jammy

```

- Hardware

```

# Informações da CPU
$ lscpu
Architecture:          x86_64
CPU op-mode(s):         32-bit, 64-bit
Address sizes:          39 bits physical, 48 bits virtual
Byte Order:             Little Endian
CPU(s):                8
On-line CPU(s) list:   0-7
Vendor ID:              GenuineIntel

```

```

Model name:           Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz
CPU family:          6
Model:               158
Thread(s) per core: 2
Core(s) per socket: 4
Socket(s):          1
Stepping:            9
CPU max MHz:        3800,0000
CPU min MHz:        800,0000
...
.

# Informações da RAM
$ free -h
              total        used        free      compart.  buff/cache
available
Mem:         15Gi       6,0Gi      763Mi      1,1Gi       8,7Gi
8,1Gi
Swap:        15Gi       6,3B      9,7Gi
.

# Informações do Armazenamento
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
...
/dev/nvme0n1p5  571G  346G   197G  64% /
...

```

## 2. Ferramentas do Sistema

Antes de baixar e compilar o código-fonte do [Android OSP](#) é necessário que o sistema contenha algumas ferramentas para o correto funcionamento. Seguindo todos os passos abaixo garantimos que o sistema funcionará corretamente



**Dica:** Atalho para abrir o terminal no Ubuntu **Ctrl + Alt + T**.

### 2.1. Atualizando Pacotes do Sistema

```

# Comando
$ sudo apt update && sudo apt upgrade -y

```

```

[sudo] password for lmendes:
Hit:1 http://br.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://br.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Hit:3 https://dl.winehq.org/wine-builds/ubuntu jammy InRelease
Hit:4 https://packages.microsoft.com/repos/edge stable InRelease
Get:5 http://br.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:6 https://packages.microsoft.com/repos/code stable InRelease [3.590 B]
Hit:7 https://storage.googleapis.com/bazel-apt stable InRelease
Get:8 https://dl.google.com/linux/chrome/deb stable InRelease [1.825 B]
Get:9 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:10 http://br.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [731 kB]
Get:11 http://br.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2.193 kB]
Get:12 https://packages.microsoft.com/repos/code stable/main arm64 Packages [18,4 kB]
Get:13 http://br.archive.ubuntu.com/ubuntu jammy-updates/main amd64 DEP-11 Metadata [103 kB]
Get:14 http://br.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 DEP-11 Metadata [212 kB]
Get:15 http://br.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1.180 kB]
Get:16 https://packages.microsoft.com/repos/code stable/main amd64 Packages [18,3 kB]
Get:17 https://packages.microsoft.com/repos/code stable/main armhf Packages [18,5 kB]
Get:18 http://br.archive.ubuntu.com/ubuntu jammy-updates/universe i386 Packages [753 kB]
Get:19 http://br.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 DEP-11 Metadata [356 kB]
Get:20 http://br.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 DEP-11 Metadata [940 B]
Get:21 http://br.archive.ubuntu.com/ubuntu jammy-backports/main amd64 DEP-11 Metadata [5.332 kB]
Get:22 http://br.archive.ubuntu.com/ubuntu jammy-backports/restricted amd64 DEP-11 Metadata [212 kB]
Get:23 http://br.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 DEP-11 Metadata [17,7 kB]
Get:24 http://br.archive.ubuntu.com/ubuntu jammy-backports/multiverse amd64 DEP-11 Metadata [212 kB]
Get:25 https://dl.google.com/linux/chrome/deb stable/main amd64 Packages [1.225 B]
Hit:26 https://ppa.launchpadcontent.net/flatpak/stable/ubuntu jammy InRelease
Get:27 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [43,1 kB]
Hit:28 https://ppa.launchpadcontent.net/fossfreedom/indicator-sysmonitor/ubuntu jammy InRelease
Get:29 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 DEP-11 Metadata [208 B]
Get:30 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 Metadata [126 kB]
Get:31 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 DEP-11 Metadata [208 B]
Hit:32 https://ppa.launchpadcontent.net/graphics-drivers/ppa/ubuntu jammy InRelease
Hit:33 https://ppa.launchpadcontent.net/obsproject/obs-studio/ubuntu jammy InRelease
Hit:34 https://ppa.launchpadcontent.net/slimbook/slimbook/ubuntu jammy InRelease
Fetched 5.955 kB in 2s (3.290 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done

```

**Figura 2:** Captura da tela após a execução do comando

## 2.2. Instalação das Ferramentas Necessárias

### 2.2.1. Git

Verificando se o Git já está instalado.

```
# Retorna a versão do Git caso esteja instalado
$ git --version
```

```

git --version
git version 2.34.1

```

**Figura 3:** Captura da tela após a execução do comando, neste caso ferramenta está instalada corretamente

Caso precise instalar ou atualizar, basta rodar o seguinte comando:

```
sudo apt-get install git-all
```

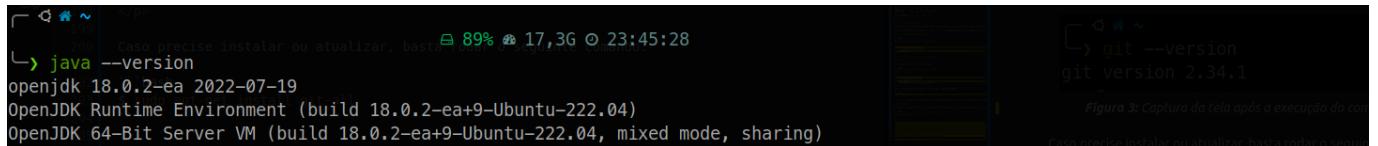


**Nota:** Para mais detalhes sobre o processo de instalação ou sobre o próprio Git, recomendamos este link [Git\[9\]](#).

## 2.2.2. Java(JDK)

O processo se repete. Primeiro verifica-se se a ferramenta já está instalada, caso contrário realiza-se o processo de instalação.

```
# Retorna informações do Java caso esteja instalado  
$ java --version
```



**Figura 4:** Captura da tela após a execução do comando, ferramenta instalada corretamente

Caso necessário, os passos abaixo orientam o processo de instalação do JDK para Linux:

- [Download](#)
- Vá para o diretório no qual deseja instalar o arquivo. Digite:

```
$ cd directory_path_name  
# Por exemplo, para instalar o software no diretório /usr/java/, digite:  
$ cd /usr/java/
```

- Mova o arquivo binário .tar.gz para o diretório atual.
- Descompacte o tarball (arquivo compactado TAR) e instale o Java

```
tar zxvf jre-8u73-linux-x64.tar.gz
```



**Dica:** Os arquivos Java são instalados em um diretório chamado `jre1.8.0_73` no diretório atual. Neste exemplo, ele é instalado no diretório `/usr/java/jre1.8.0_73`. Quando a instalação for concluída, você verá a palavra `Done`.

- Delete o arquivo .tar.gz para economizar espaço em disco.



**Nota:** Todos os passos foram tirados da documentação oficial.[10]

### 2.2.3. Python

A biblioteca padrão[11] do Python já vem instalada nos sistemas GNU/Linux mais recente ([Python on Ubuntu\[12\]](#)), com o comando abaixo podemos encontrar a versão instalada no sistema.

```
# Encontra binario do Python
$ which python3

...
# Versão do Python
$ python3 --version
```

```
which python3
/usr/bin/python3
python3 --version
Python 3.10.12
```

**Figura 5:** Captura da tela da saída da execução dos comandos citados anteriormente.

O processo de instalação do Python no Ubuntu é relativamente simples, podendo ser feito através do gerenciador de pacotes do Ubuntu. O comando abaixo demonstra o processo de instalação.

```
# Atualiza pacotes
$ sudo apt update

...
# Instala o Python (O X deve ser substituído pela a versão específica de
# instalação)
$ sudo apt install python3.x
```



**Dica:** Neste [Link\[13\]](#) temos mais detalhes sobre o processo de instalação, por exemplo, o processo de instalação a parti do código-fonte (**Method 2**).

### 2.2.4. Curl

O processo de instalação da ferramenta curl é relativamente simples, antes, verifiquemos se a ferramenta já está instalada. Os comandos abaixo foram utilizados para esse procedimento.

```

# Verifica versão, se nada for executado basta seguir o processo de
instalação
$ curl --version

...
# Atualiza pacotes
$ sudo apt update

...
# Instalação
$ sudo apt install curl
...

```

```

curl 7.81.0 (x86_64-pc-linux-gnu) libcurl/7.81.0 OpenSSL/3.0.2 zlib/1.2.11 brotli/1.0.9 zstd/1.4.8 libidn2/2.3.2 libpsl/0.21.0 (+libidn2/2.3.2) libssh/0.9.6/openssl/zlib ngtcp2/1.43.0 librtmp/2.3 OpenLDAP/2.5.18
Release-Date: 2022-01-05
Protocols: dict file ftp ftps gopher gophers http https imap imaps ldap ldaps mqtt pop3 pop3s rtmp rtsp scp sftp smb smbs smtp smtps telnet tftp
Features: alt-svc AsynchDNS GSS-API HSTS HTTP2 HTTPS-proxy IDN IPv6 Kerberos Largefile libz NTLM NTLM_WB PSL SPNEGO SSL TLS-SRP UnixSockets zstd

```

**Figura 6:** Saída da chamada de comando curl no terminal.

### 3. Instalação do VS Code

O **VS Code** é um excelente editor de texto, o seu processo de instalação ocorre através da instalação de um pacote **.deb**. Seguem os comandos abaixo para a instalação.

- O primeiro passo é realizar o download do pacote de instalação através do link: [Download Visual Studio Code\[14\]](#).
- Abra o terminal no mesmo local do arquivo baixado e digite o comando abaixo:

```

$ sudo apt install ./<file>.deb

# Se estiver em uma distribuição Linux mais antiga, precisará executar os
# comandos abaixo:
# sudo dpkg -i <file>.deb
# sudo apt-get install -f # Install dependencies

```

As figuras abaixo demonstram o processo de instalação.

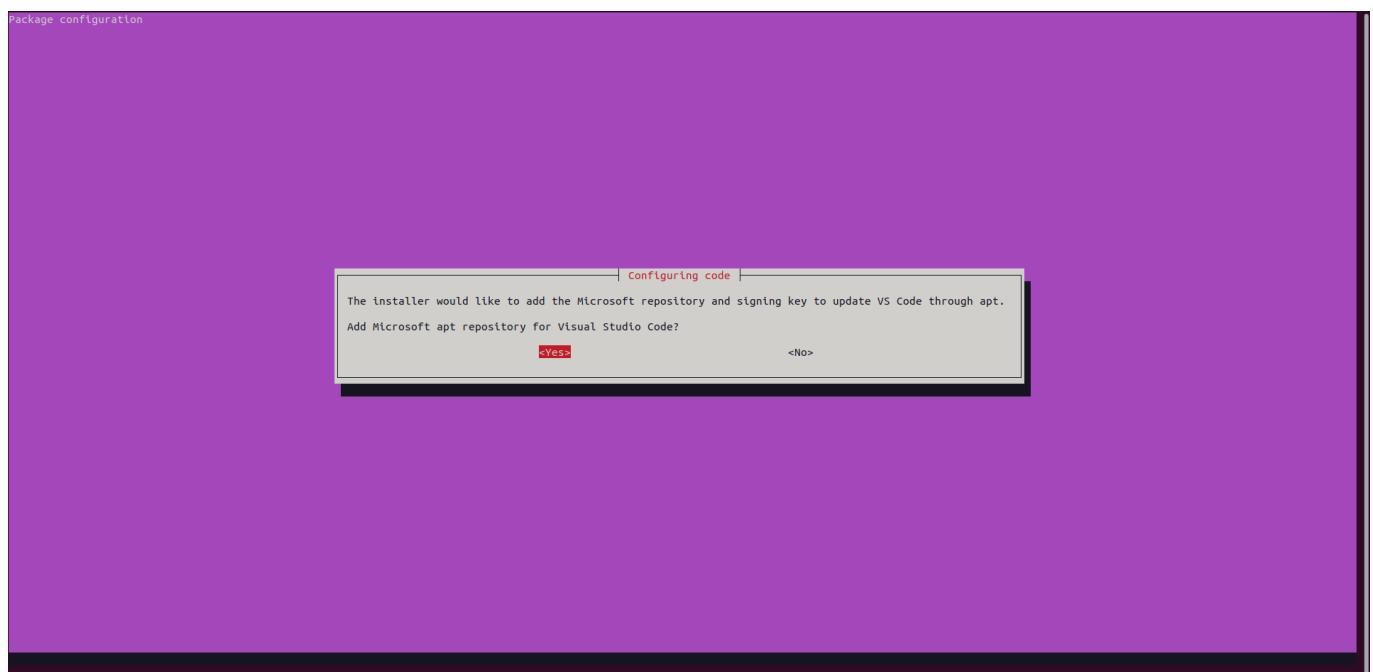
```

leandrosantos@leandro-OptiPlex-5070: ~/Downloads$ ls
code_1.96.0-1733888194_amd64.deb
leandrosantos@leandro-OptiPlex-5070: ~/Downloads$ sudo apt install ./code_1.96.0-1733888194_amd64.deb
[sudo] password for leandrosantos:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note: selecting 'code' instead of './code_1.96.0-1733888194_amd64.deb'.
The following packages were automatically installed and are no longer required:
  apport-symptoms python3-systemd
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  code
0 upgraded, 1 newly installed, 0 to remove and 30 not upgraded.
Need to get 0 B/105 MB of archives.
After this operation, 424 MB of additional disk space will be used.
Get: /home/leandrosantos/Downloads/code_1.96.0-1733888194_amd64.deb code amd64 1.96.0-1733888194 [105 MB]
Selecting previously unselected package code.
(Reading database... 215547 files and directories currently installed.)
Preparing to unpack .../code_1.96.0-1733888194_amd64.deb ...
Unpacking code (1.96.0-1733888194) ...

Progress: [ 20%] [#####

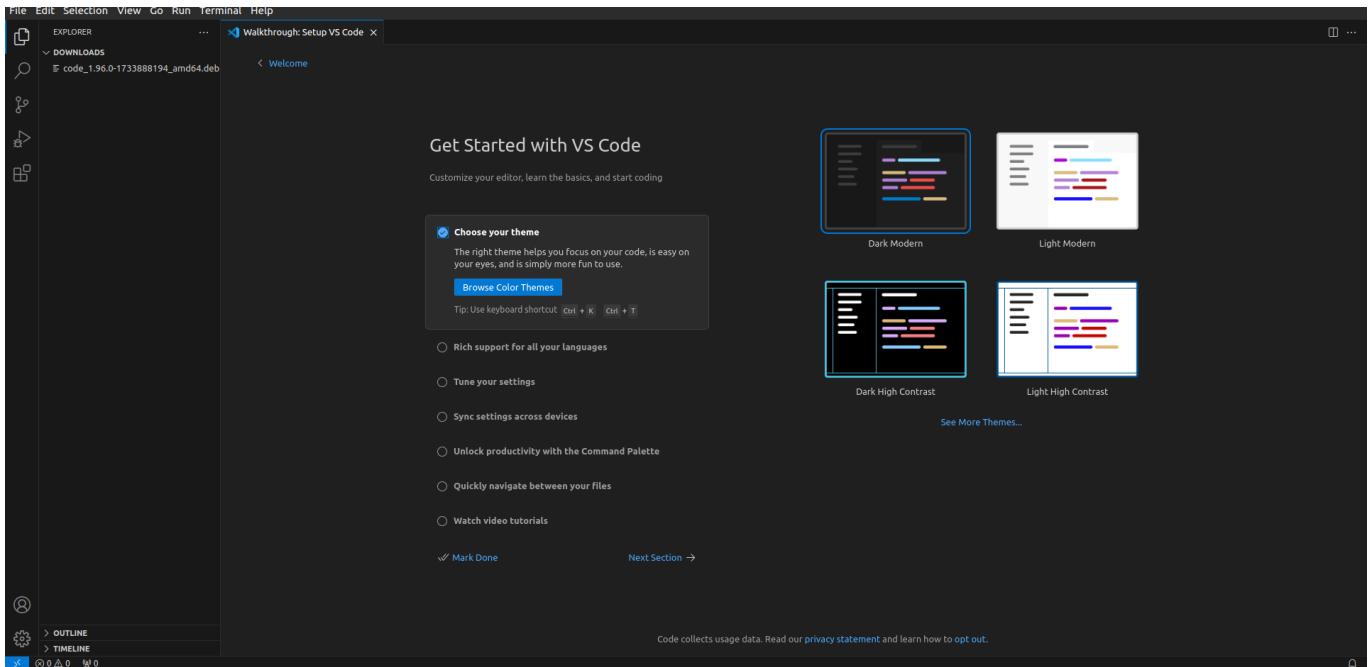
```

**Figura 7:** Instalando VS code pelo terminal.



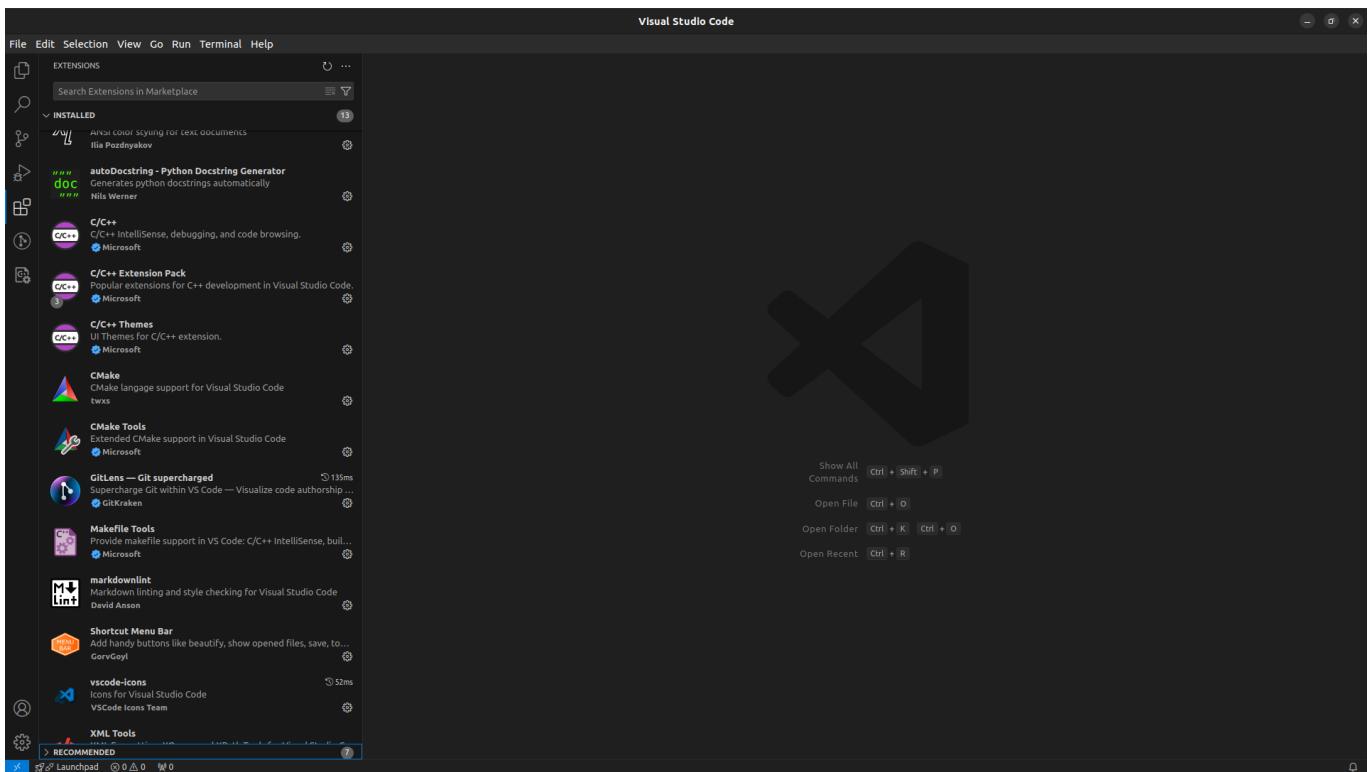
**Figura 8:** Permite atualizar o VS Code automaticamente.

Processo de instalação concluído com sucesso, agora basta abrir o **VS Code pelo** menu de **App**, ou abrir o terminal e digitar **code**. A tela da figura abaixo irá aparecer, basta confirmar o tema e pronto, o vs code está pronto para o uso.



**Figura 9:** configuração inicial do tempo do VS Code.

Meu VS Code e algumas das extensões instaladas.



**Figura 10:** Extensões do VS Code.

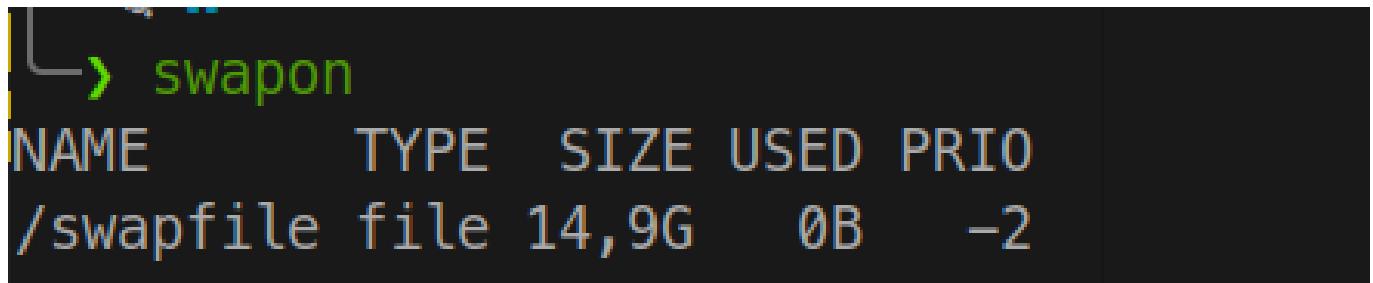
## 4. Configuração de Espaço Adicional de Memória (Swap)

Em uma máquina que contenha menos de 32GB de RAM é recomendado a criação da região de SWAP para evitar problemas de compilação do Android OSP, como foi o caso de nossas máquinas. Uma máquina possui 24gb de RAM e foi dedicada 15GB de Swap. A outra máquina tem 16GB de RAM e foi dedicada aos mesmos 15GB de Swap.

Abaixo, seguem os comandos utilizados para a criação/aumento da região do Swap:

- Habilitando ou alterando região de **SWAP**

```
# Retorna arquivo de Swap caso exista
$ swapon
NAME      TYPE  SIZE USED PRI
/swapfile file 14,9G   0B   -2
```



```
swapon
NAME      TYPE  SIZE USED PRI
/swapfile file 14,9G   0B   -2
```

**Figura 11:** Região de SWAP.

 **Nota:** Valor atual da configuração, antes era pouco mais de 2gb

- Verificando a região de Swap atual

```
# Pré-aloca espaço em disco (16gb no arquivo swapfile)
$ sudo fallocate -l 16G /swapfile
# Da permissão de leitura e escrita para o arquivo swapfile
$ sudo chmod 600 /swapfile
# Cria área de swap no arquivo swapfile
$ sudo mkswap /swapfile
# Habilita região de Swap
$ sudo swapon /swapfile
# "joga" as informações da região de Swap no arquivo de informações do
# sistema de arquivos
# Esta etapa garante que a região de Swap será habilitada na inicialização
# da máquina
$ sudo sh -c "echo '/swapfile none swap sw 0 0' >> /etc/fstab"
```

## 5. Android Open Source Project (AOSP)

Algumas etapas e ferramentas são necessárias para o correto funcionamento do Build do Android. A seguir, serão mostrados todos os passos executados desde a instalação dos pré-requisitos até a sincronização do repositório.

### 5.1. Pacotes Necessários

Para o Ubuntu 18 ou superior o [Google recomenda\[15\]](#) os seguintes pacotes:

```
sudo apt-get install git-core gnupg flex bison build-essential zip curl  
zlib1g-dev libc6-dev-i386 x11proto-core-dev libx11-dev lib32z1-dev libgl1-  
mesa-dev libxml2-utils xsltproc unzip fontconfig
```

**⚠️ Atenção:** Durante a etapa de build, ocorreu erro de falta de pacote (**libncurses5**) para isso,  
precisou-se apenas instalar o seguinte pacote

```
sudo apt install libncurses5-dev
```

## 5.2. repo

Segundo sua [documentação](#)[16], "Repo é uma ferramenta construída sobre o Git. O Repo ajuda a gerenciar muitos repositórios do Git, faz os uploads para sistemas de controle de revisão e automatiza partes do fluxo de trabalho de desenvolvimento."

O Google utiliza o **repo** para gerenciar o código fonte do **AOSP** e seu processo de instalação foi relativamente simples. Abaixo, segue os comandos utilizados para a sua instalação

```
# Debian/Ubuntu.  
$ sudo apt-get install repo
```

A documentação também da a opção de instalação manual:

```
# Instalação manual  
$ mkdir -p ~/.bin  
$ PATH="${HOME}/.bin:$PATH"  
$ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/.bin/repo  
$ chmod a+r ~/bin/repo
```

```
↳ repo version  
repo version v2.50.1  
  (from https://gerrit.googlesource.com/git-repo)  
  (tracking refs/heads/stable) tempo dependendo da capacidade da máquina e conexão com a internet  
  (Wed, 18 Dec 2024 09:23:49 -0800) to no servidor  
repo launcher version 2.50  
  (from /usr/bin/repo)  
  (currently at 2.50.1)  
repo User-Agent git-repo/2.50.1 (Linux) git/2.34.1 Python/3.10.12  
git 2.34.1  
git User-Agent git/2.34.1 (Linux) git-repo/2.50.1  
Python 3.10.12 (main, Nov 6 2024, 20:22:13) [GCC 11.4.0]  
OS Linux 6.8.0-49-generic (#49~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Wed Nov 6 17:42:15 UTC 2)  
CPU x86_64 (x86_64)  
Bug reports: https://issues.gerritcodereview.com/issues/new?component=1370071
```

**Figura 12:** Saída do comando repo version.



**Nota:** A saída precisa indicar a versão 2.4 ou mais recente.

### 5.3 Sincronização do Código-Fonte

- Primeiro, foi criada uma pasta local para a sincronização do repositório.

```
mkdir -p /home/$USER/Documents/Courses/Embedded_Systems/android-osp
```

- Agora, foi escolhida a branch da versão do Android que iremos trabalhar e foi realizada a sincronização.

A branch escolhida foi a android-13.0.0\_r83 do Android 13

```
# Inicializando repo apontando para a branch android-13.0.0_r83
$ repo init --partial-clone -b android-13.0.0_r83 -u
<https://android.googlesource.com/platform/manifest>

# Iniciando sincronização do repositório
# Este processo pode demorar bastante tempo dependendo da capacidade da
# máquina e conexão com a internet
# -c: Busca apenas o branch atual do manifesto no servidor
# -j$(nproc): Divide a sincronização entre as threads de execução para que
# ela seja concluída mais rapidamente
$ repo sync -c -j$(nproc)
```

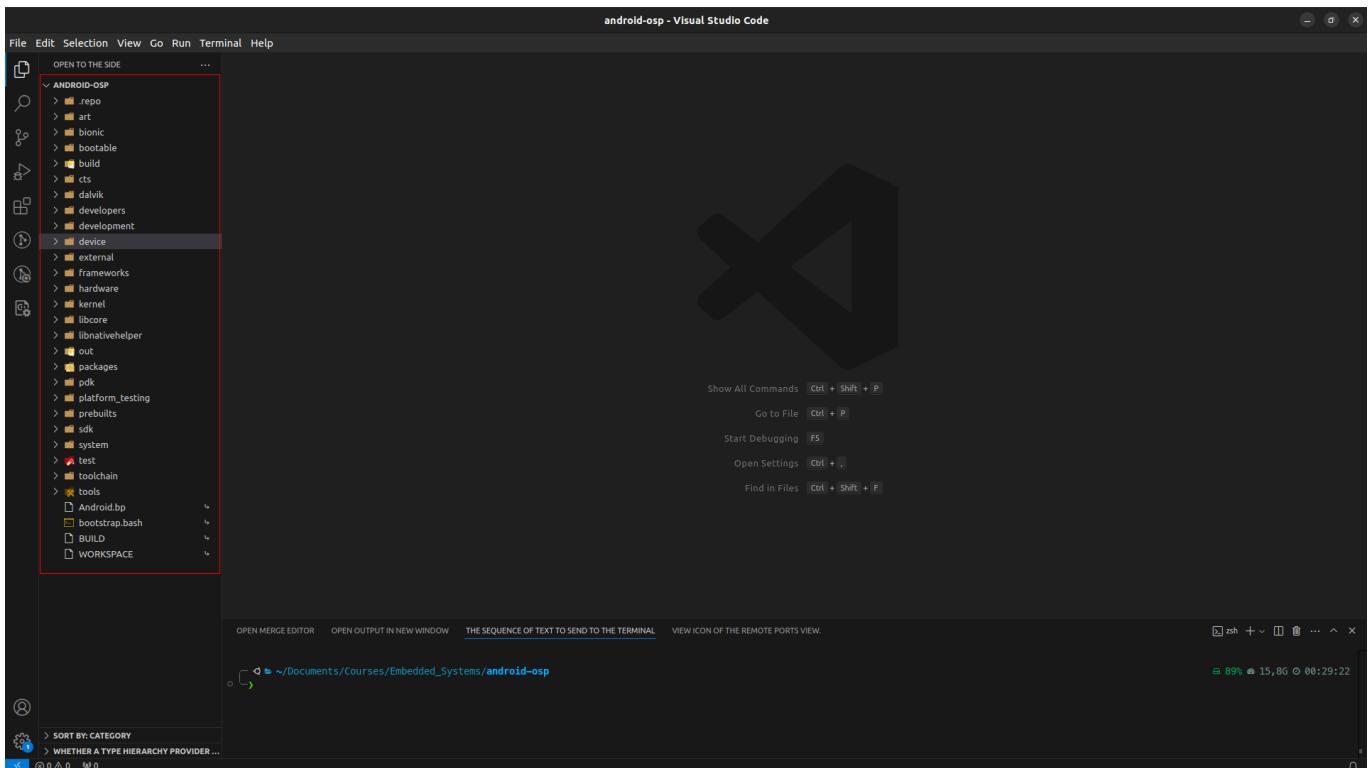
```
└─> ~/Documents/Courses/Embedded_Systems/android-osp
    > repo sync -j$(nproc)
Fetching: 100% (1146/1146), done in 1m53.413s
Checking out: 100% (1146/1146), done in 3.313s
repo sync has finished successfully.
```

**Figura 13:** Saída do comando repo sync.



**Dica:** O comando nproc --all retorna o número de threads disponível no sistema para execução.

Abrindo o código-fonte do Android no VSCode, ainda dentro da pasta após sincronizar, basta digitar `code .` e o VS Code será executado a parti do caminho local.



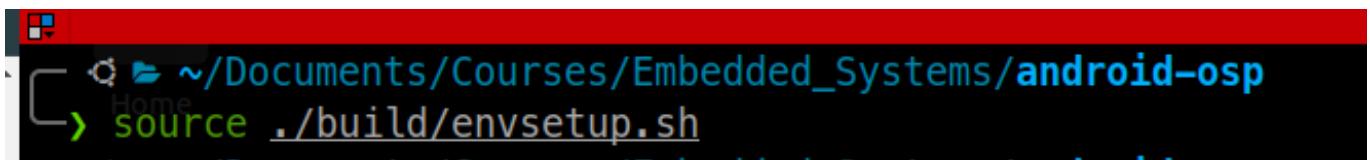
**Figura 14:** Visualização do código-fonte do Android.

## 5.4. Compilação do Sistema Operacional

### 5.4.1. Configuração das variáveis de ambiente

Para habilitar as variáveis de ambiente segundo a [documentação do Google\[17\]](#), basta entrar na pasta raiz do projeto e rodar o comando abaixo. Se tudo der certo, nenhum erro será gerado. Esse script além de configurar variáveis de ambiente, importa vários comandos que permitem trabalhar com o código-fonte do Android.

```
# Cria as variáveis de ambiente
$ source ./build/envsetup.sh
```



**Figura 15:** Saída da execução do script.

### 5.4.2. Definição da Plataforma Alvo

O comando [lunch\[18\]](#) é o responsável por fazer a definição das configurações de build para a respectiva plataforma, o alvo abaixo foi o escolhido:

```
lunch sdk_phone_x86_64
```

```

lunch sdk_phone_x86_64
=====
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=13
TARGET_PRODUCT=sdk_phone_x86_64
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release
TARGET_ARCH=x86_64
TARGET_ARCH_VARIANT=x86_64
TARGET_2ND_ARCH=x86
TARGET_2ND_ARCH_VARIANT=x86_64
HOST_ARCH=x86_64
HOST_2ND_ARCH=x86
HOST_OS=linux
HOST_OS_EXTRA=Linux-6.8.0-50-generic-x86_64-Ubuntu-22.04.5-LTS
HOST_CROSS_OS=windows
HOST_CROSS_ARCH=x86
HOST_CROSS_2ND_ARCH=x86_64
HOST_BUILD_TYPE=release
BUILD_ID=TQ3A.230805.001.S1
OUT_DIR=out
PRODUCT_SOONG_NAMESPACES=device/generic/goldfish device/generic/goldfish-opengl hardware/google/camera hardware/google/camera/devices/EmulatedCamera device/generic/goldfish device/generic/goldfish-opengl
=====
```

**Figura 16:** Saída do comando lunch

#### 5.4.3. Build

A partir do Android 7.0, o Google trouxe o sistema de [build Soong](#)[19] para substituir o make. Segundo o Google o make estava se tornando lento e não escalonável, isso fez com que eles trocassem seu sistema de build do Android para o Soong.

Escolhemos utilizar o sistema de build Soong para nosso projeto, abaixo está a saída da execução do comando além do comando utilizado.

```
m -j$(nproc)
```

```

build/make/core/soong_config.mk:209: warning: BOARD_PLAT_PUBLIC_SEPOLICY_DIR has been deprecated. Use SYSTEM_EXT_PUBLIC_SEPOLICY_DIRS instead.
build/make/core/soong_config.mk:210: warning: BOARD_PLAT_PRIVATE_SEPOLICY_DIR has been deprecated. Use SYSTEM_EXT_PRIVATE_SEPOLICY_DIRS instead.
=====
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=13
TARGET_PRODUCT=sdk_phone_x86_64
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release
TARGET_ARCH=x86_64
TARGET_ARCH_VARIANT=x86_64
TARGET_2ND_ARCH=x86
TARGET_2ND_ARCH_VARIANT=x86_64
HOST_ARCH=x86_64
HOST_2ND_ARCH=x86
HOST_OS=linux
HOST_OS_EXTRA=Linux-6.8.0-50-generic-x86_64-Ubuntu-22.04.5-LTS
HOST_CROSS_OS=windows
HOST_CROSS_ARCH=x86
HOST_CROSS_2ND_ARCH=x86_64
HOST_BUILD_TYPE=release
BUILD_ID=TQ3A.230805.001.S1
OUT_DIR=out
PRODUCT_SOONG_NAMESPACES=device/generic/goldfish device/generic/goldfish-opengl hardware/google/camera hardware/google/camera/devices/EmulatedCamera device/generic/goldfish device/generic/goldfish-opengl
=====

[ 0% 0/1] analyzing Android.bp files and generating ninja file at out/soong/build.ninja
 0:21 analyzing Android.bp files and generating ninja file at out/soong/build.ninja
    Consuma as opções de personalização do build de configuração para mais detalhes sobre a execução do emulador. A Figura 2
```

Se você quiser criar outras versões do Android, os nomes de ramificação delas podem ser encontrados no [repositório público do Android](#). Eles correspondem aos da página [Codenames, tags e números de build do Android](#).

1. Crie uma imagem do sistema no AVD. É o mesmo processo que o de [criar uma imagem do sistema no dispositivo Android](#). Por exemplo, para criar um AVD x86 de 64 bits:

Nesta página  
Arquitetura do Android Emulator.  
Criar imagens no AVD  
Compartilhar imagens do sistema no AVD para outras pessoas usarem com o Android Studio  
Adicionar suporte para várias telas

**Figura 17:** Início do build

```

[100% 30615/30615] Create system-qemu.img now
removing out/target/product/emulator_x86_64/system-qemu.img.qcow2
out/host/linux-x86/bin/sgdisk --clear out/target/product/emulator_x86_64/system-qemu.img

#### build completed successfully (48:32 (mm:ss)) ####
```

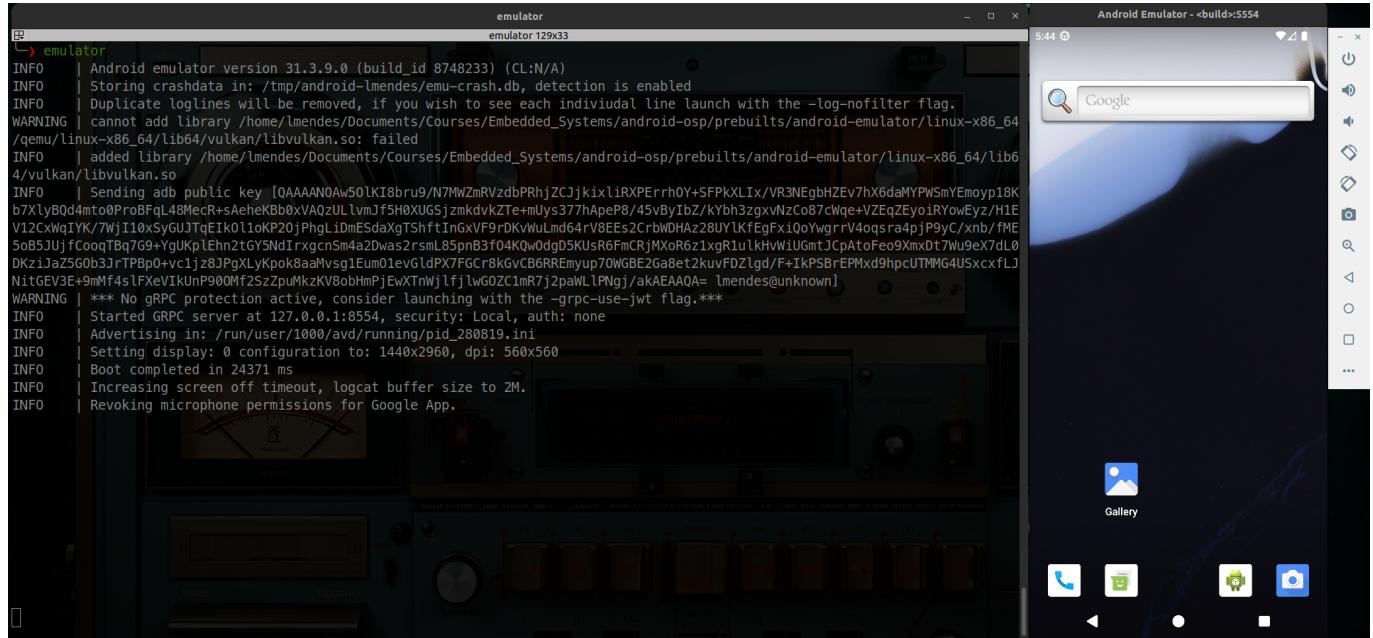
**Figura 18:** Fim do Build



**Nota:** Esse build não foi o primeiro, por isso o tempo de build foi relativamente pequeno se comparado ao inicial que demorou cerca de 3h

#### 5.4.4. Teste do Build

Um teste simples foi realizado após a build que foi a execução do emulador conforme a figura abaixo:



**Figura 19:** Emulator

## 6. Android Studio

### 6.1. Instalação

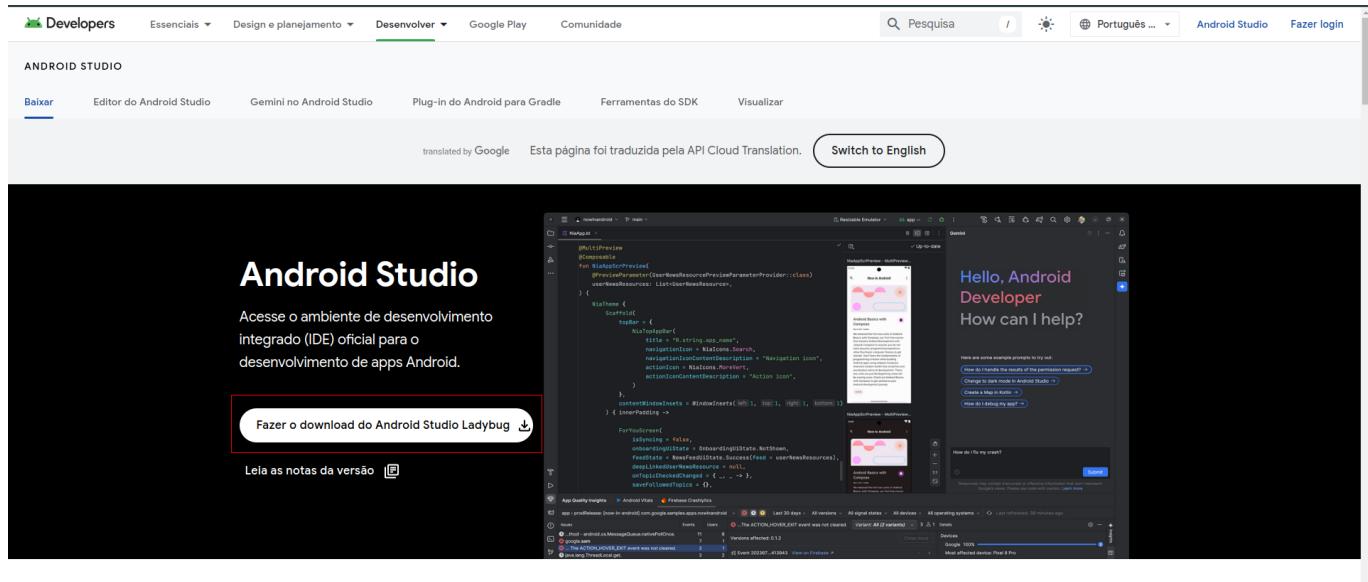
O processo de instalação do Android Studio escolhido foi o manual[\[20\]](#), abaixo seguem os passos utilizados:

- Bibliotecas necessárias para máquinas de 64 bits

```
sudo apt-get install libc6:i386 libncurses5:i386 libstdc++6:i386 lib32z1  
libbz2-1.0:i386
```

- Download

Para realizar o download do Android Studio, basta acessar o [site oficial\[21\]](#) da IDE e fazer o download da versão que escolher. Nós escolhemos a versão mais recente.



## Novos recursos

RECURSO RECURSO

**Figura 20:** Android Studio download

- Instalação

Abrindo o terminal dentro do diretório onde foi realizado o download, basta digitar os comandos abaixo:

```
# Extraia os dados
$ tar -zxvf android-studio-2024.2.1.12-linux.tar.gz android-studio/
# Move a pasta para o diretório de instalação
$ sudo mv android-studio /opt/
# Cria link simbólico do binário do android studio
$ sudo ln -sf /opt/android-studio/bin/studio /bin/android-studio
```

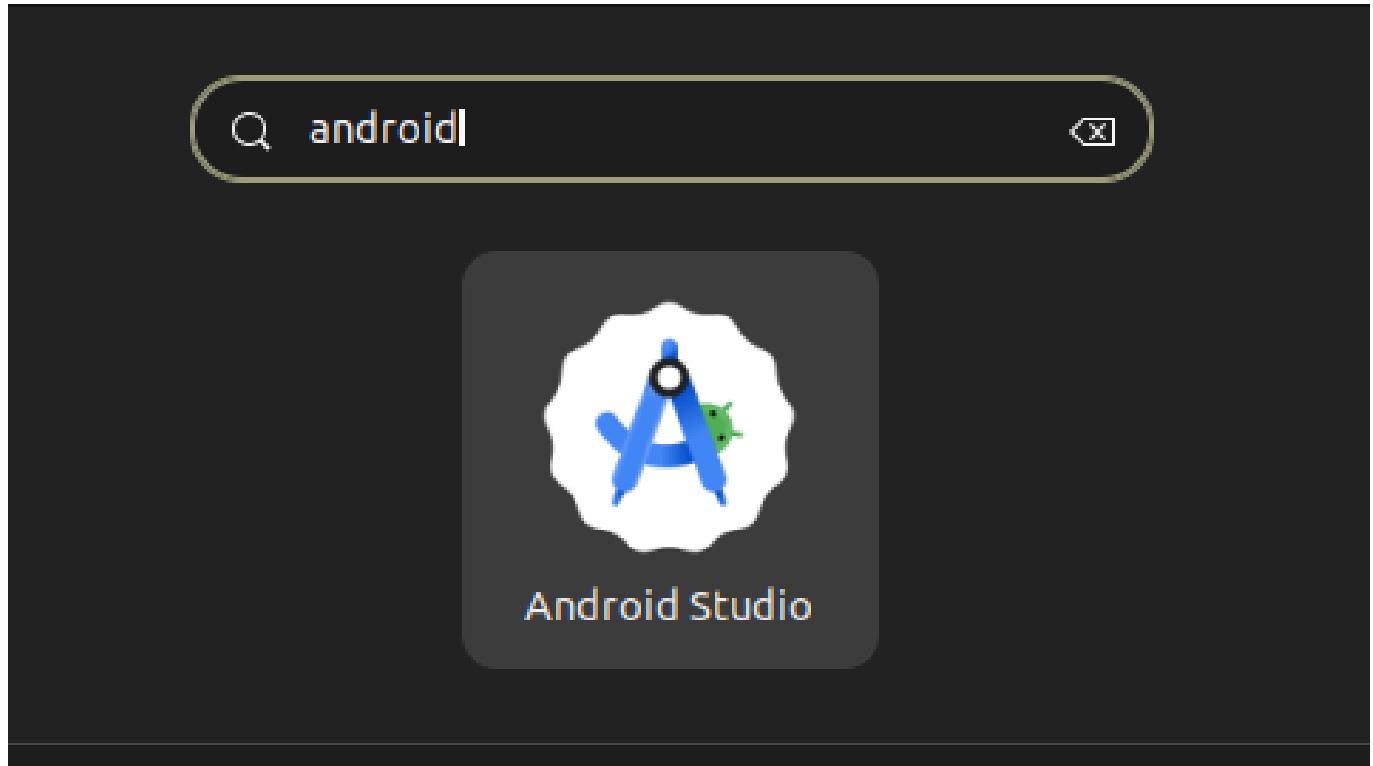
- Criação de atalho da aplicação

```
# Cria arquivo de inicialização do android studio via interface gráfica
sudo nano /usr/share/applications/android-studio.desktop
```

Basta copiar e colar as informações abaixo dentro do arquivo criado e pronto. O ícone irá aparecer no menu inicial.

```
[Desktop Entry]
Version=1.0
Type=Application
Name=Android Studio
Comment=Android Studio
Exec=/opt/android-studio/bin/studio
Icon=/opt/android-studio/bin/studio.png
Categories=Development;IDE;
```

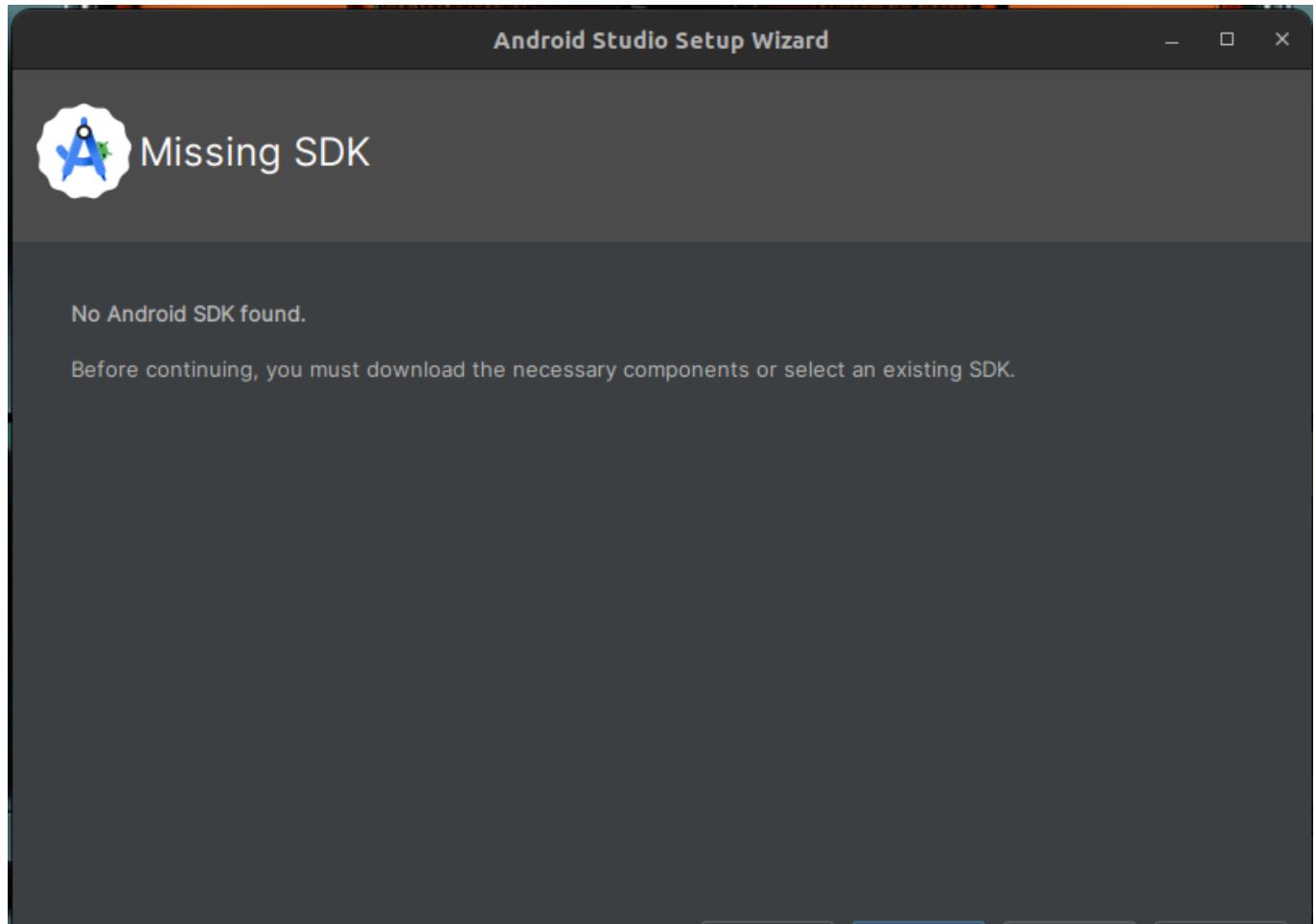
```
Terminal=false  
StartupNotify=true  
StartupWMClass=jetbrains-android-studio  
Name[en_GB]=android-studio.desktop
```



**Figura 21:** Menu de Aplicativos

Para continuar o processo de instalação, basta clicar no ícone do Android Studio e, se tudo estiver correto, as telas abaixo aparecerão.

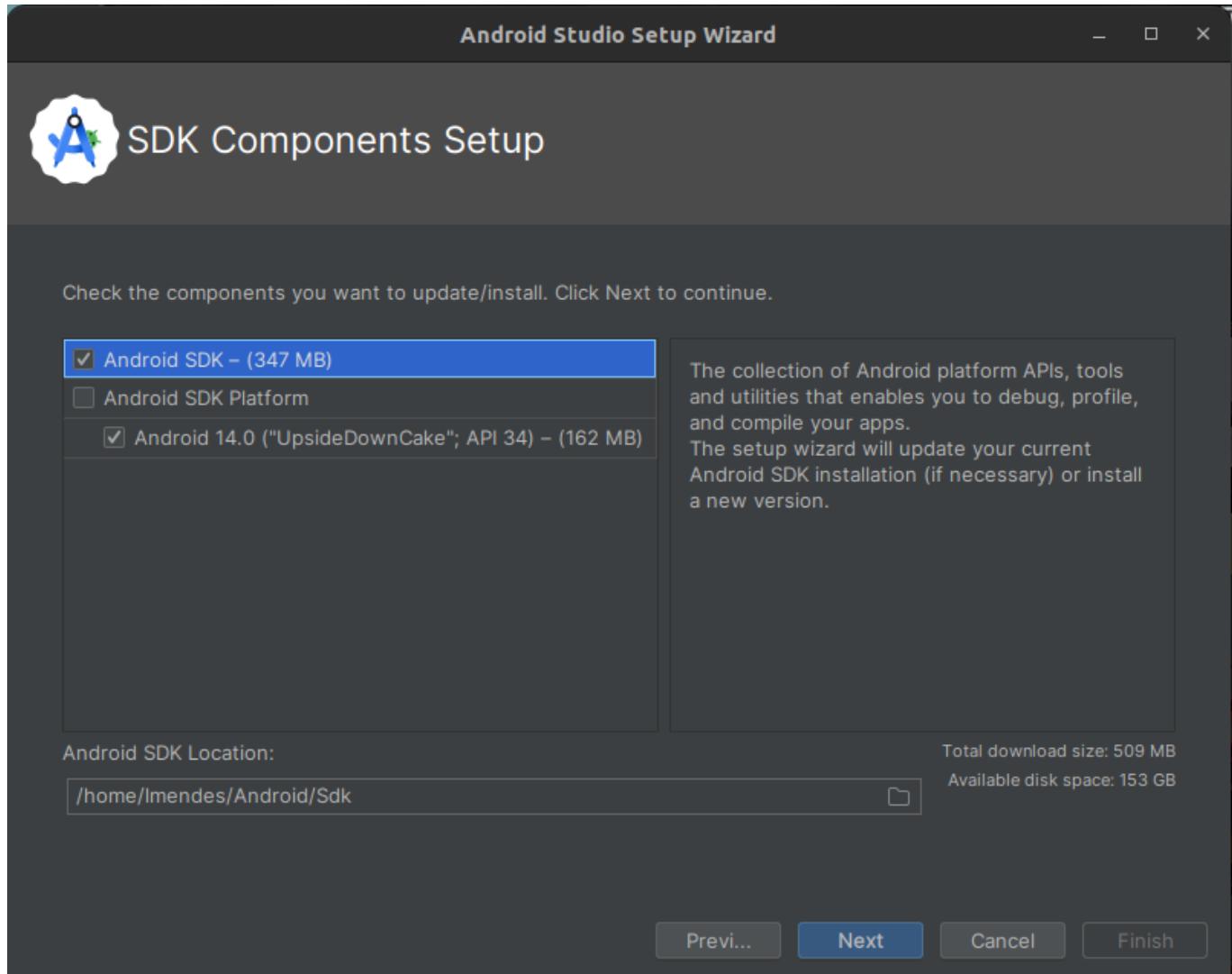
Nesta tela, o Android Studio não achou nenhum SDK instalado, basta apertar em **Next** e seguir para a tela de instalação.



**Figura 22:** Tela inicial de instalação

---

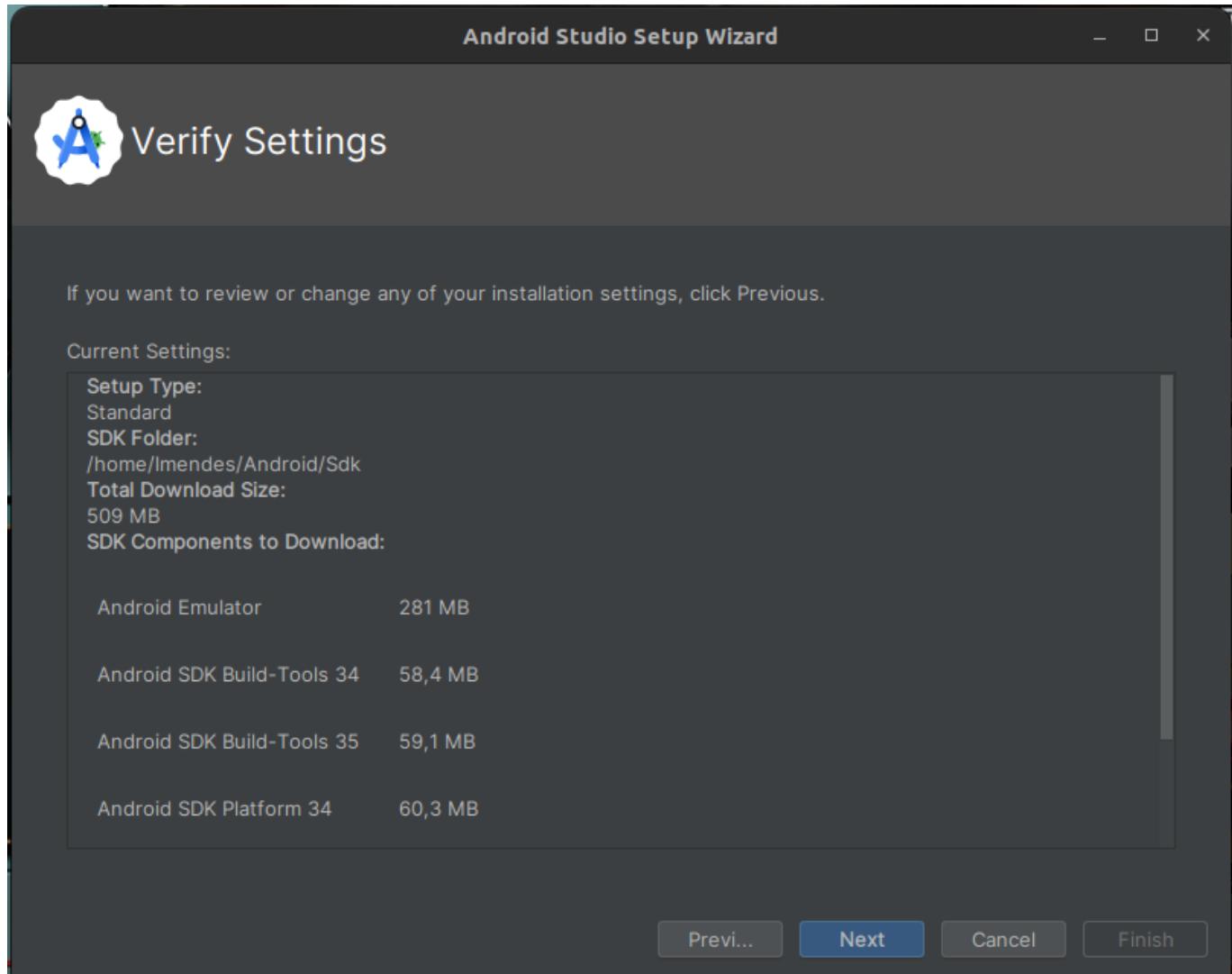
Na próxima tela, pode deixar as configurações padrão, nesta tela é feita a escolha da versão do SDK do Android. [Next](#)



**Figura 23:** Escolha do SDK

---

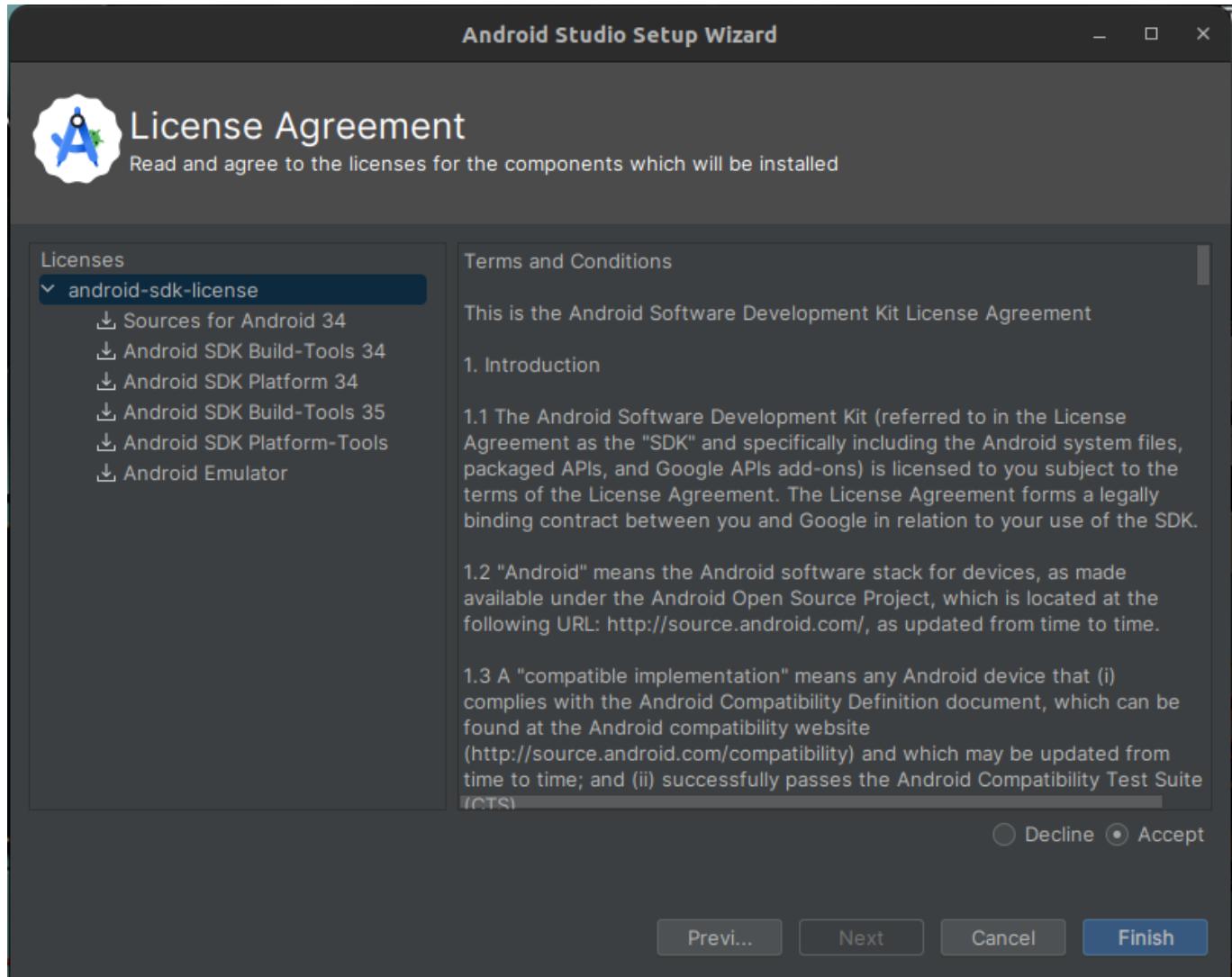
Tela de verificação das configurações, estando tudo certo, basta apertar em **Next**.



**Figura 24:** Validação das configurações

---

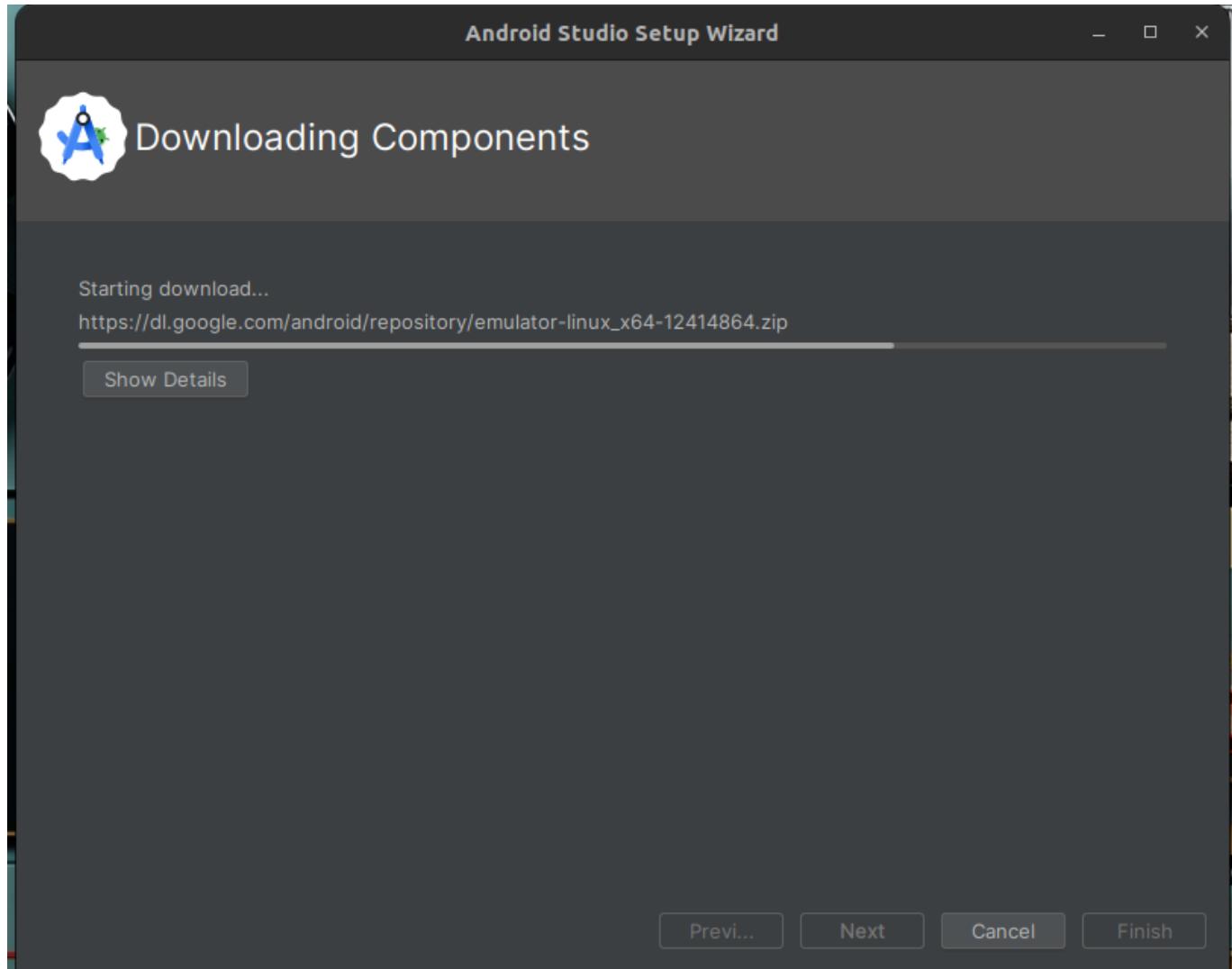
Tela de aceite dos termos de licenciamento. Se concorda, basta apertar **Finish**.



**Figura 25:** Termos e condições

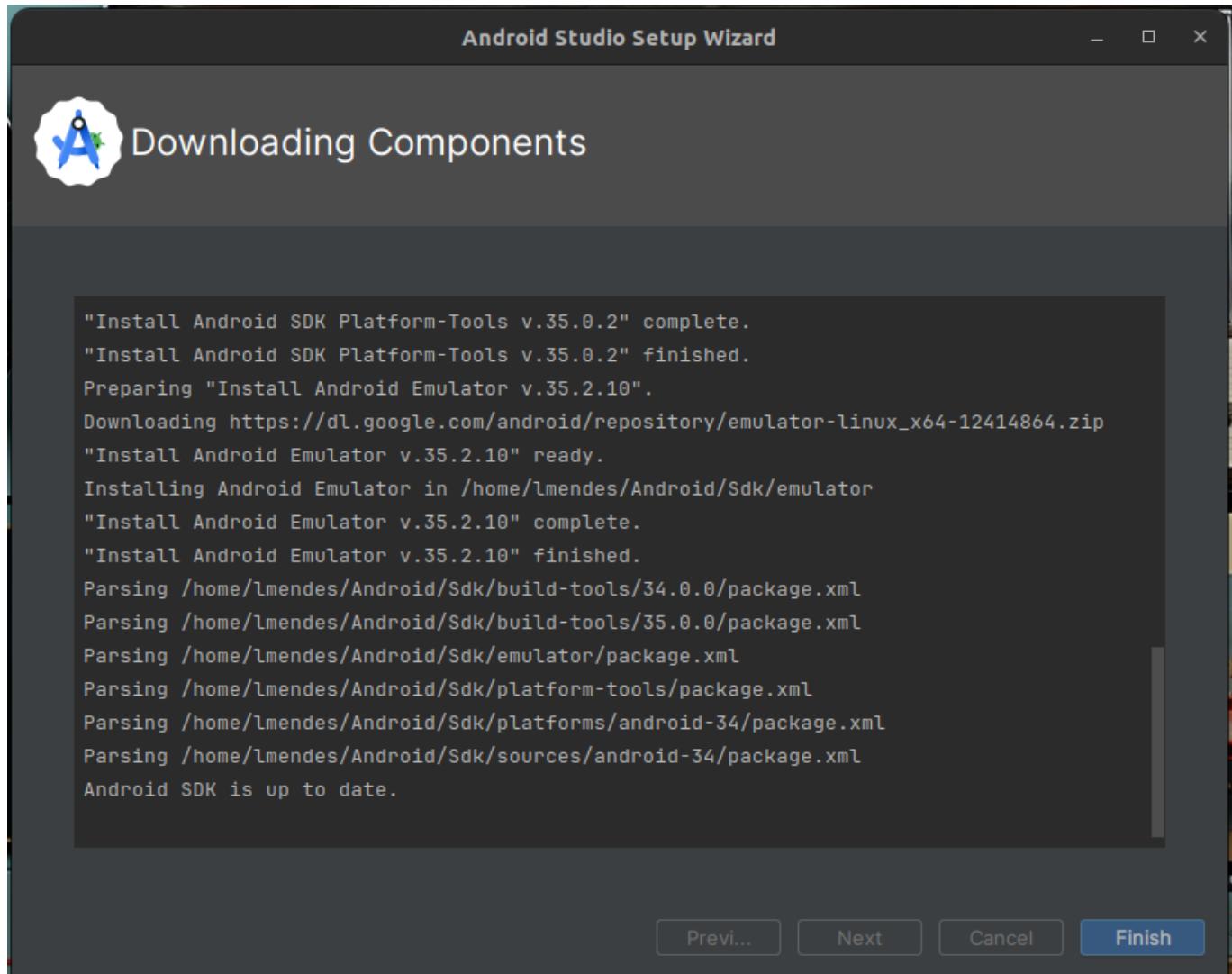
---

Tela de Download



**Figura 26:** Donwloads

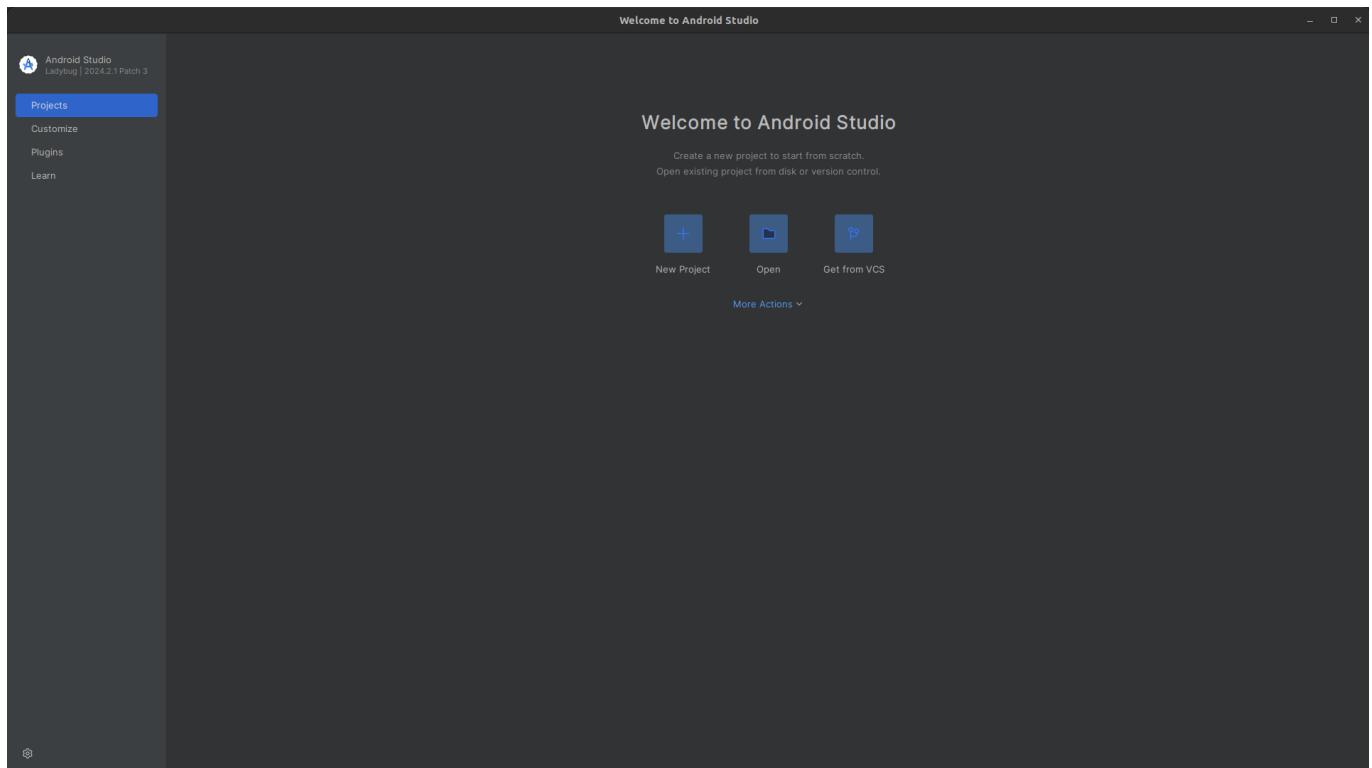
Tela de instalação



**Figura 27:** Instalação

---

Instalação finalizada

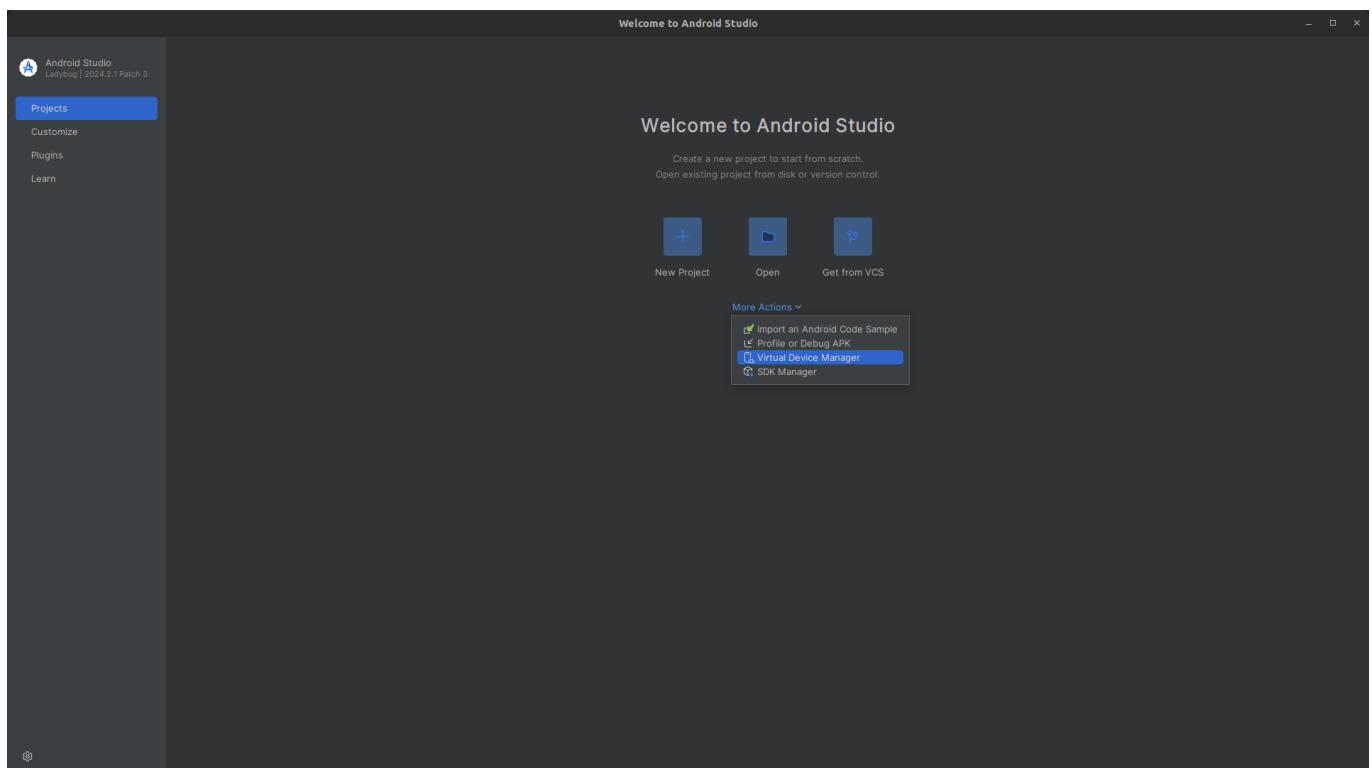


**Figura 28:** Tela inicial do Android Studio

## 6.2. Emulador

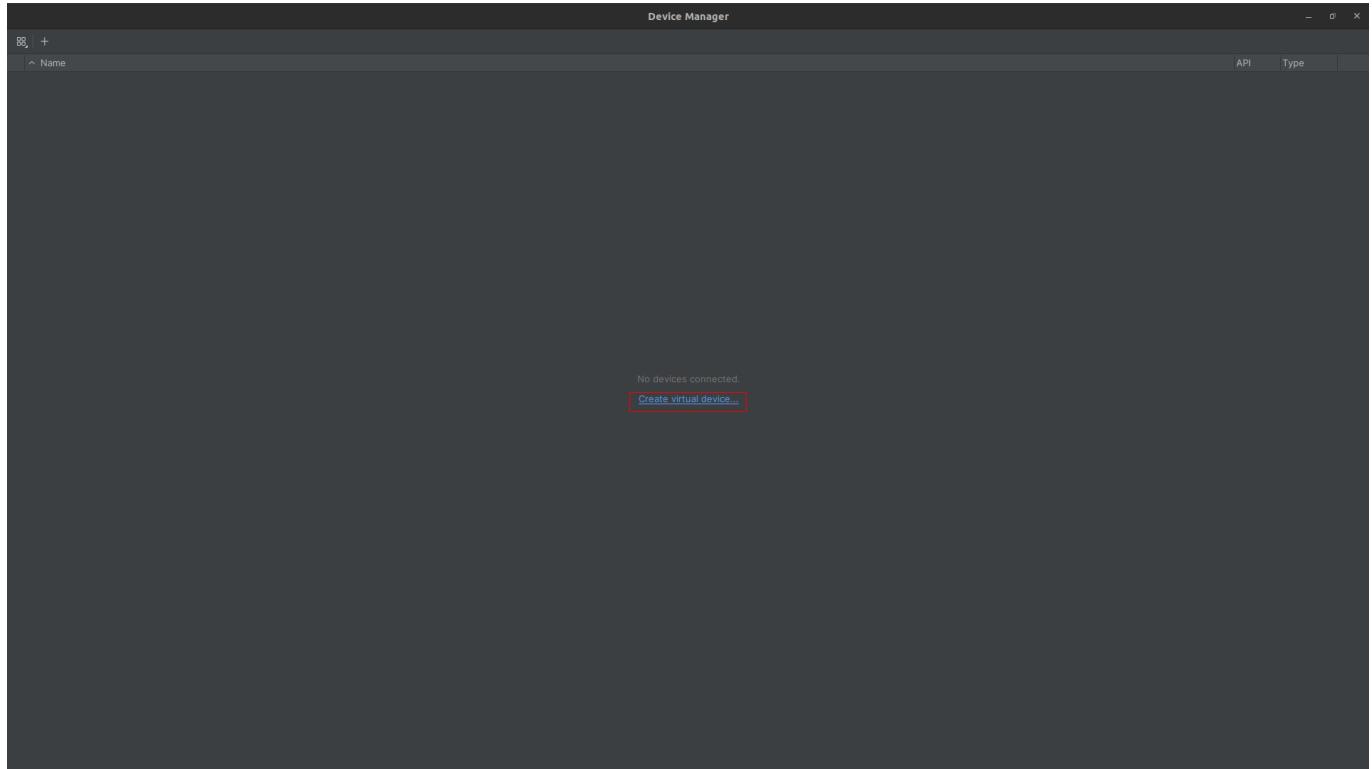
- Criando um dispositivo virtual.

Abrindo o gerenciador de dispositivos virtuais.



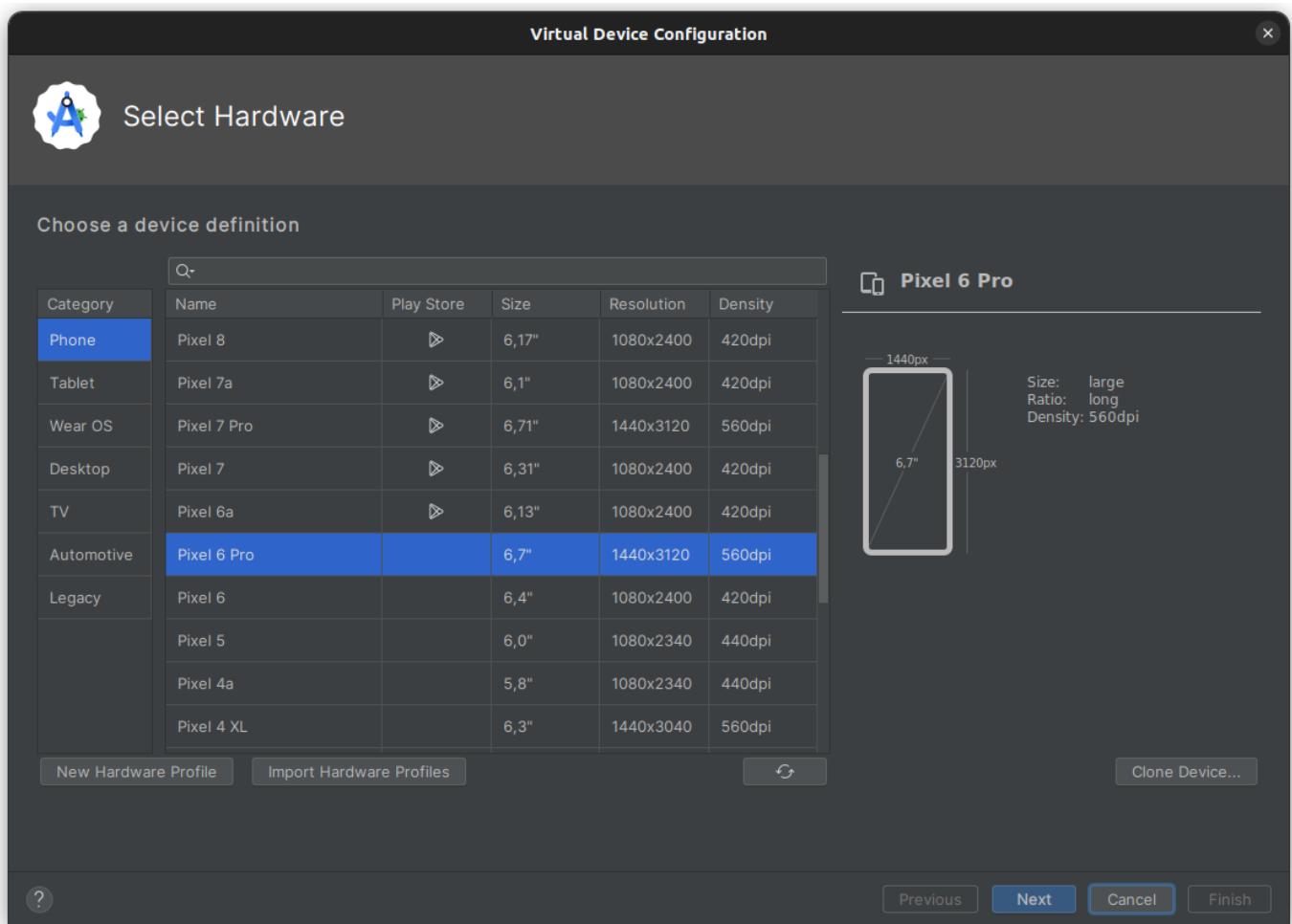
**Figura 29:** Configuração: Virtual Devices Manager

Nesta tela, basta apertar em **Criar dispositivo virtual**.



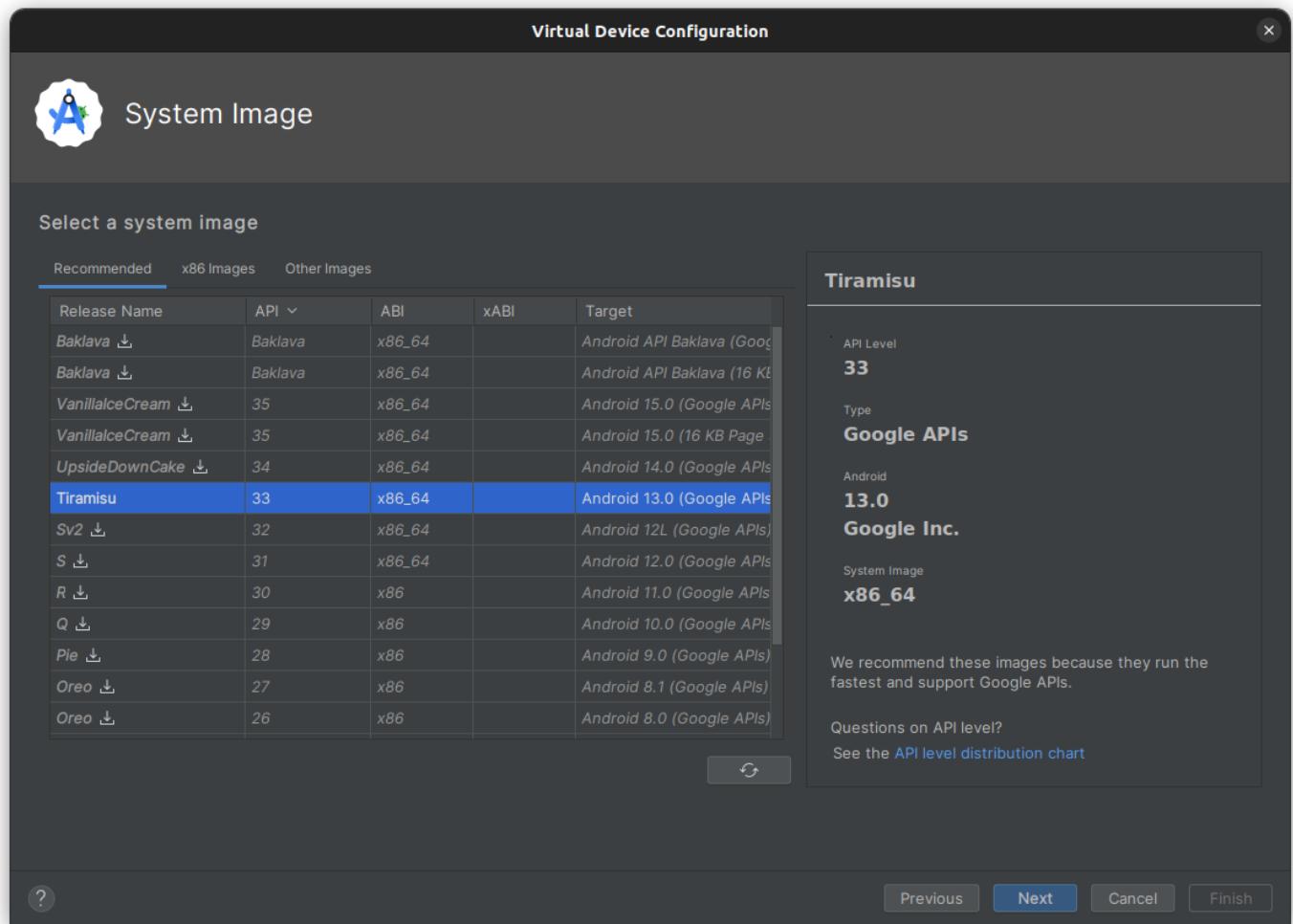
**Figura 30:** Virtual Devices Manager

Escolhendo a categoria e o modelo do dispositivo a ser criado.



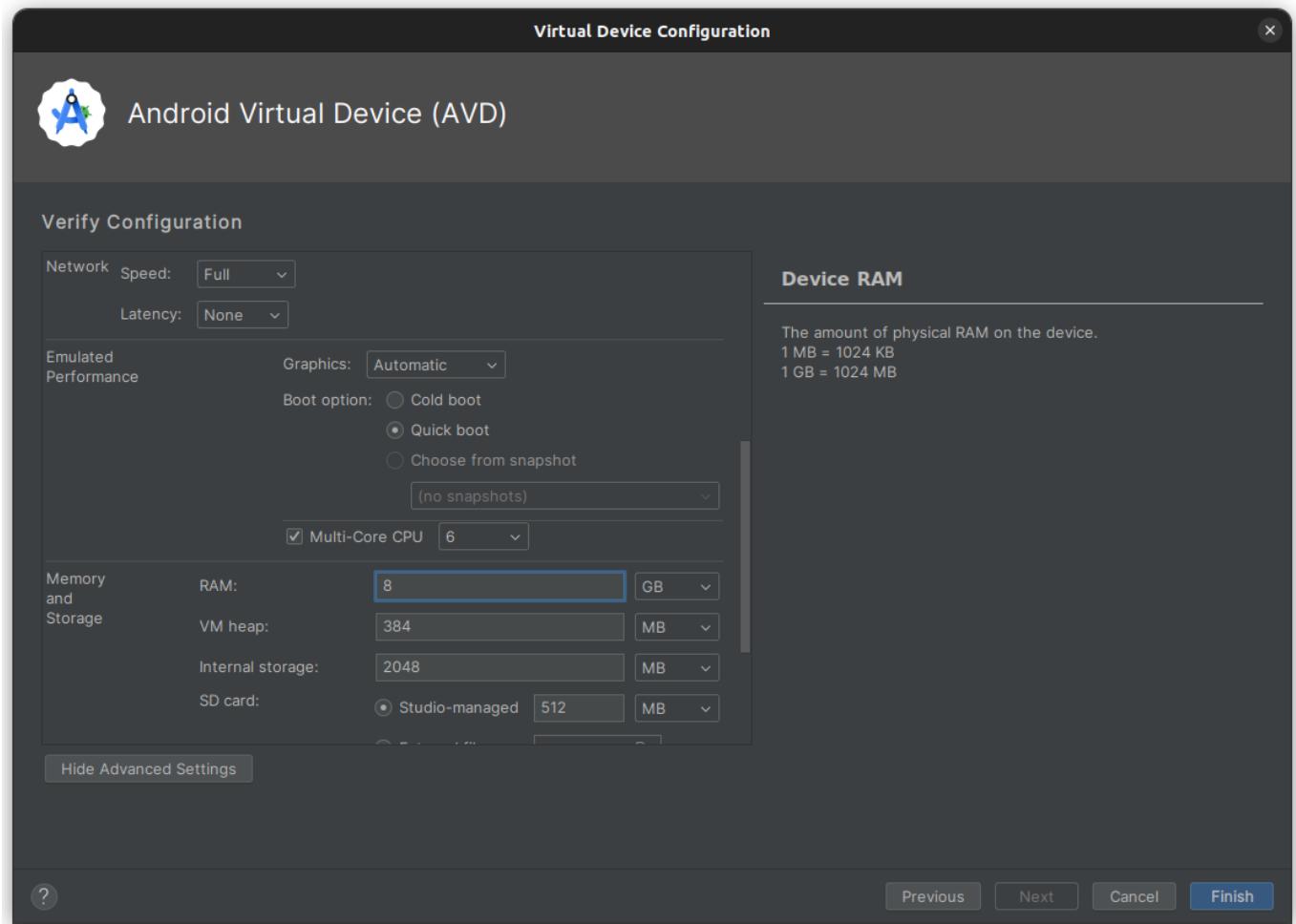
**Figura 31:** Seleção do Hardware

Escolhendo a imagem do sistema a ser baixada, a versão escolhida foi a mesma do AOSP baixado na seção [Android Open Source Project](#).



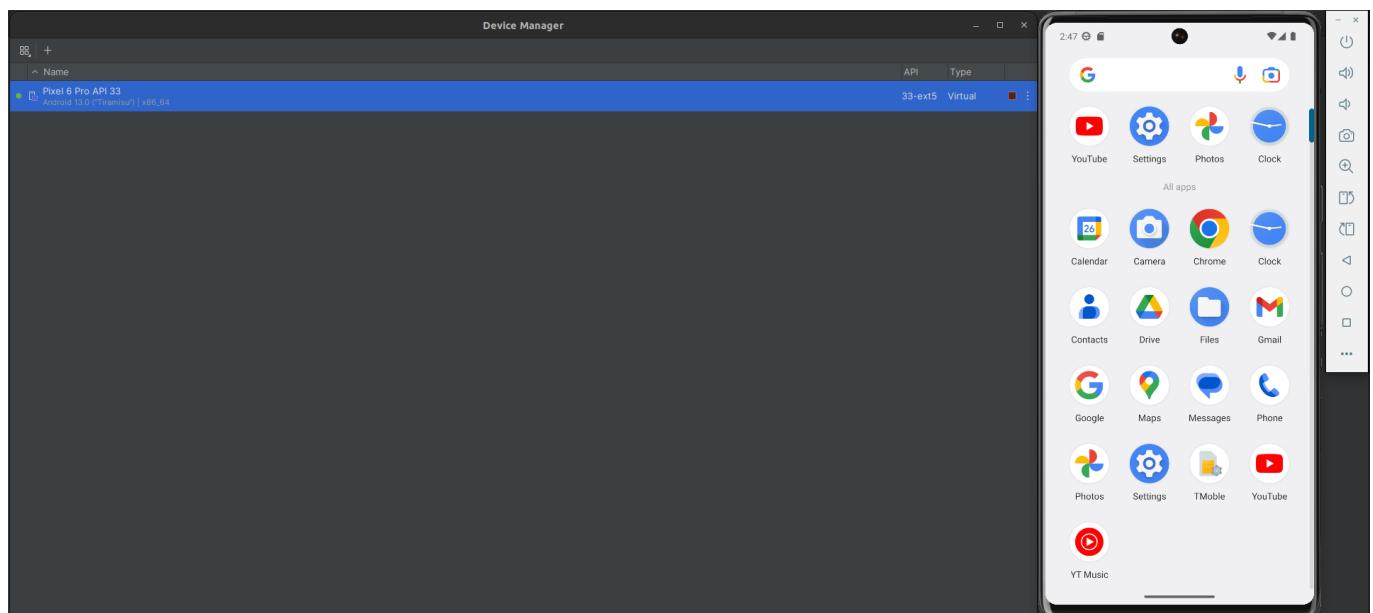
**Figura 32:** Seleção da imagem do sistema

Nesta parte é feito o ajuste das configurações do dispositivo virtual, foi realizada a alteração do tamanho da RAM e da quantidade de núcleos do processador.



**Figura 33:** Configuração do AVD

Dispositivo criado, dando play e verificando o funcionamento.



**Figura 34:** Teste do dispositivo Virtual



**Dica:** [Documentação oficial\[22\]](#) com mais detalhes sobre os dispositivos virtuais.

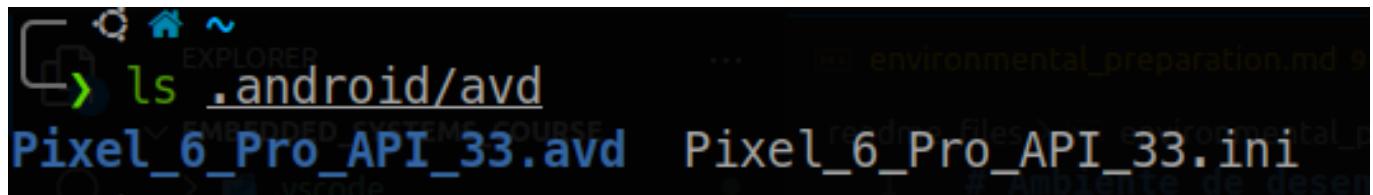
## 6.3. Extra

### 6.3.1. Adicionando PlayStore

Como é possível notar, alguns dispositivos virtuais não têm a Play Store instalada na imagem do sistema, uma maneira simples de realizar esse procedimento é executar os passos a seguir:

- Liste as configurações do dispositivo no conforme a figura abaixo.

Esta pasta foi criada a partir do dispositivo criado na seção anterior.



**Figura 35:** Configurações do dispositivo virtual

Entre na pasta e liste todos os arquivos conforme a figura a seguir.

```
└> cd .android/avd/Pixel_6_Pro_API_33.avd
└─> ls -la
total 1526456
drwxrwxr-x 5 lmendes lmendes 4096 dez 26 14:48 .
drwxrwxr-x 3 lmendes lmendes 4096 dez 26 14:45 ..
-rw-rw-r-- 1 lmendes lmendes 96 dez 26 14:45 AVD.conf
-rw----- 1 lmendes lmendes 0 dez 26 14:46 bootcompleted.ini
-rw-r--r-- 1 lmendes lmendes 69206016 dez 26 14:45 cache.img
-rw-r--r-- 1 lmendes lmendes 196616 dez 26 14:45 cache.img.qcow2
-rw-rw-r-- 1 lmendes lmendes 1270 dez 26 14:45 config.ini
-rw-rw-r-- 1 lmendes lmendes 113 dez 26 14:48 emulator-user.ini
-rw-rw-r-- 1 lmendes lmendes 158 dez 26 14:45 emu-launch-params.txt
-rw----- 1 lmendes lmendes 18874368 dez 26 14:45 encryptionkey.img
-rw-r--r-- 1 lmendes lmendes 2031616 dez 26 14:48 encryptionkey.img.qcow2
-rw-rw-r-- 1 lmendes lmendes 4246 dez 26 14:45 hardware-qemu.ini
-rw-rw-r-- 1 lmendes lmendes 1720908 dez 26 14:45 initrd
drwxrwxr-x 3 lmendes lmendes 4096 dez 26 14:46 modem_simulator
-rw-rw-r-- 1 lmendes lmendes 0 dez 26 14:45 multiinstance.lock
-rw-rw-r-- 1 lmendes lmendes 0 dez 26 14:45 read-snapshot.txt
-rw-rw-r-- 1 lmendes lmendes 536870912 dez 26 14:45 sdcard.img
-rw-r--r-- 1 lmendes lmendes 589824 dez 26 14:48 sdcard.img.qcow2
drwxr--r-- 2 lmendes lmendes 4096 dez 26 14:48 snapshots
drwxr-xr-x 2 lmendes lmendes 4096 dez 26 14:47 tmpAdbCmds
-rw-rw-r-- 1 lmendes lmendes 1048576 dez 25 11:44 userdata.img
-rw----- 1 lmendes lmendes 6488064 dez 26 14:45 userdata-qemu.img
-rw-r--r-- 1 lmendes lmendes 990052352 dez 26 14:48 userdata-qemu.img.qcow2
-rw-rw-r-- 1 lmendes lmendes 23 dez 26 14:45 user-settings.ini
-rw-rw-r-- 1 lmendes lmendes 9 dez 26 14:45 version_num.cache
```

**Figura 36:** Arquivos de configurações do dispositivo virtual

Faça as seguintes alterações nos respectivos arquivos:

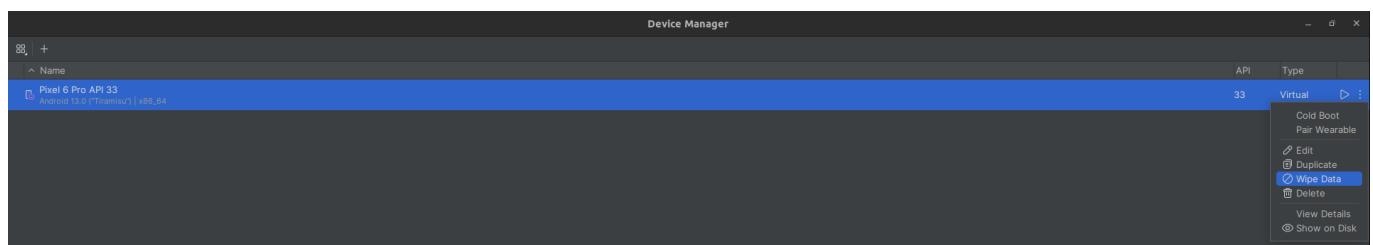
#### config.ini

```
...
# Troce de false para true
PlayStore.enabled = true
# Troque de google_apis para google_apis_playstore
# Android/Sdk/system-images/android-33/
image.sysdir.1 = system-images/android-33/google_apis_playstore/x86_64/
...
```

#### hardware-qemu.ini

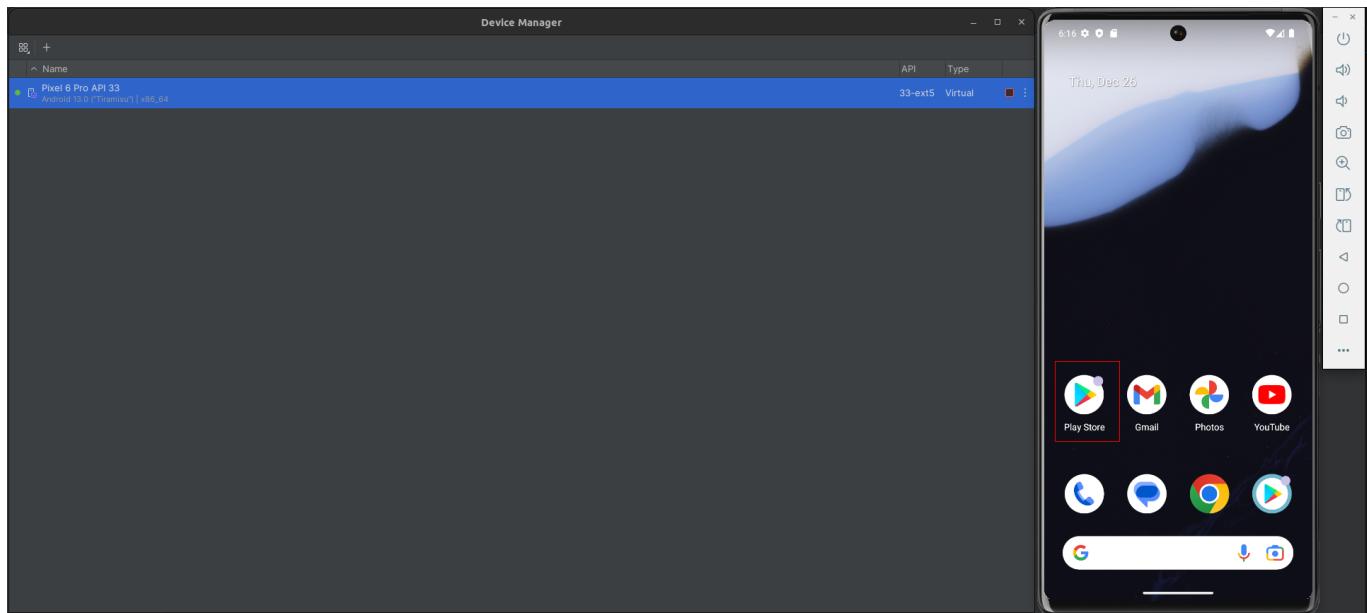
```
...
# Troque de google_apis para google_apis_playstore
kernel.path = /home/lmendes/Android/Sdk/system-images/android-
33/google_apis_playstore/x86_64//kernel-ranchu
# Troque de google_apis para google_apis_playstore
disk.ramdisk.path = /home/lmendes/Android/Sdk/system-images/android-
33/google_apis_playstore/x86_64//ramdisk.img
# Troque de google_apis para google_apis_playstore
disk.systemPartition.initPath = /home/lmendes/Android/Sdk/system-
images/android-33/google_apis_playstore/x86_64//system.img
# Troque de google_apis para google_apis_playstore
disk.vendorPartition.initPath = /home/lmendes/Android/Sdk/system-
images/android-33/google_apis_playstore/x86_64//vendor.img
# Troce de false para true
PlayStore.enabled = false
...
```

Antes de reiniciar o emulador, apague os dados do dispositivo conforme a imagem abaixo:



**Figura 37:** Configuração do dispositivo

- Teste no dispositivo

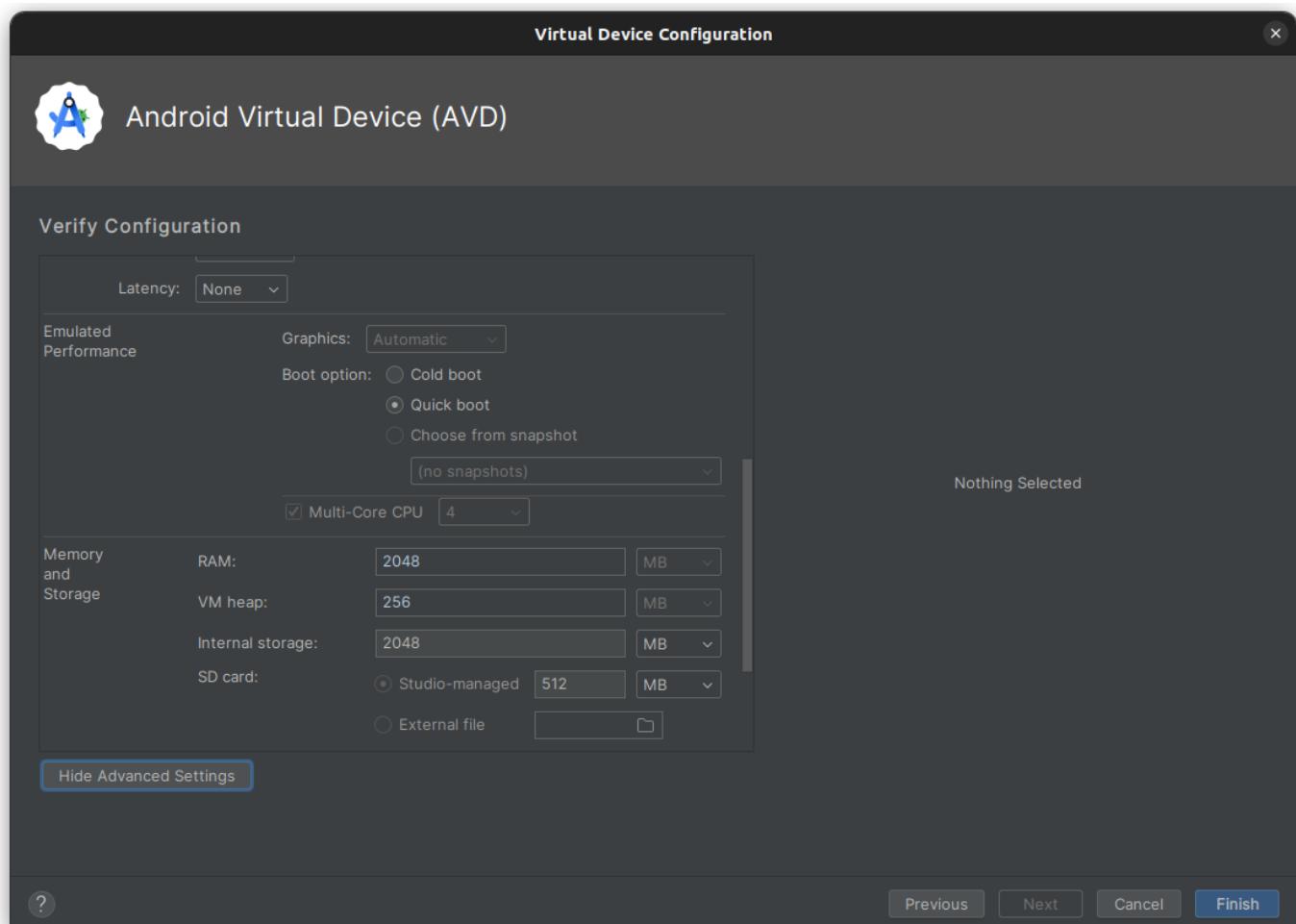


**Figura 38:** Configuração do dispositivo

### 6.3.2. Alterando a RAM

Alguns perfis de dispositivos, principalmente os que têm a Play Store na pré-instalada, não permitem a alteração das configurações de desempenho. Conforme a figura abaixo.

Uma forma simples de realizar essas alterações é criando um clone do perfil e alterar as configurações de desempenho do clone.



**Figura 39:** Configuração do dispositivo bloqueadas

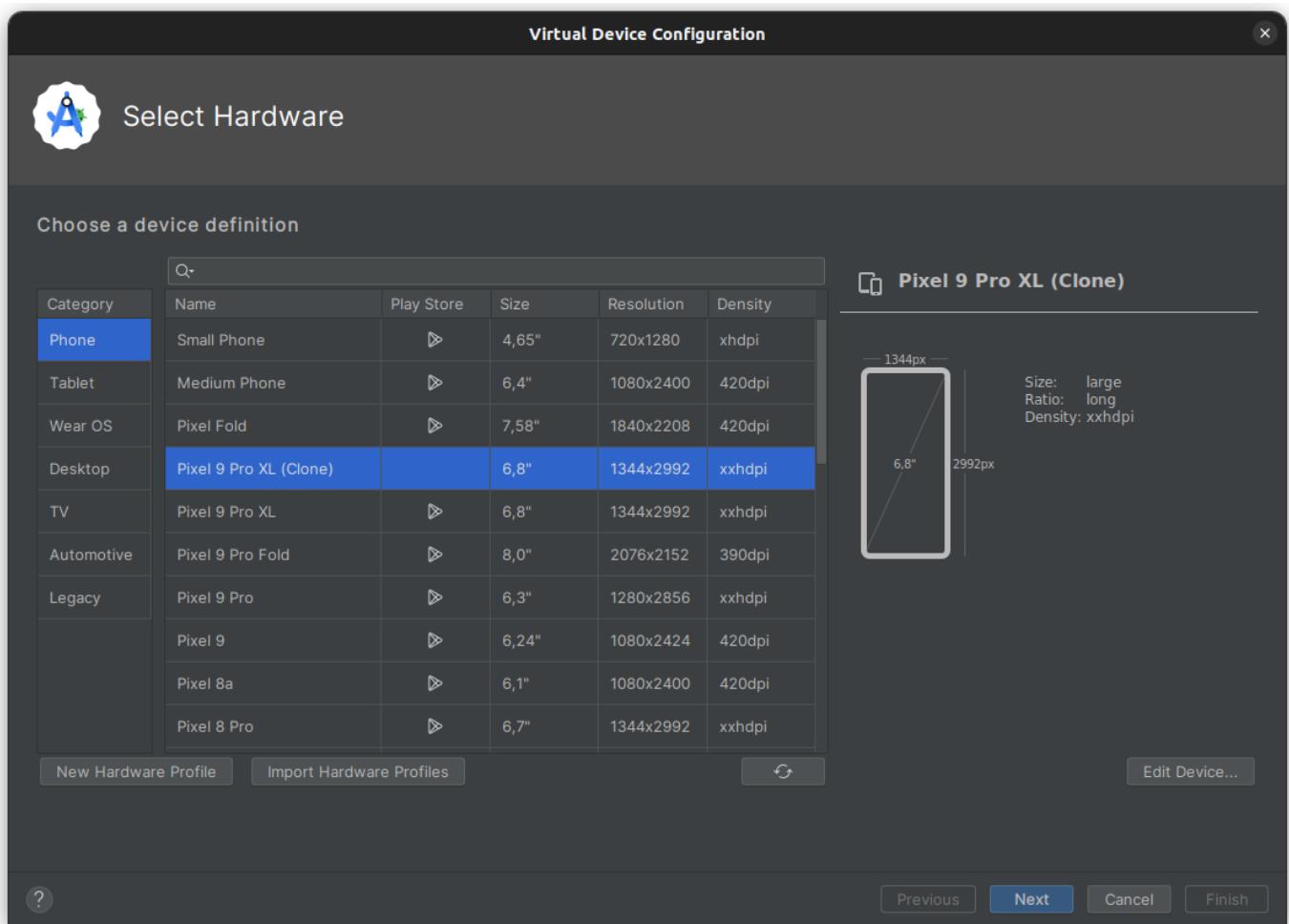
- Criando um dispositivo clone

As imagens abaixo mostram o passo a passo para o procedimento.

Para clonar o dispositivo basta escolher e aperta no botão em destaque da imagem a baixo.

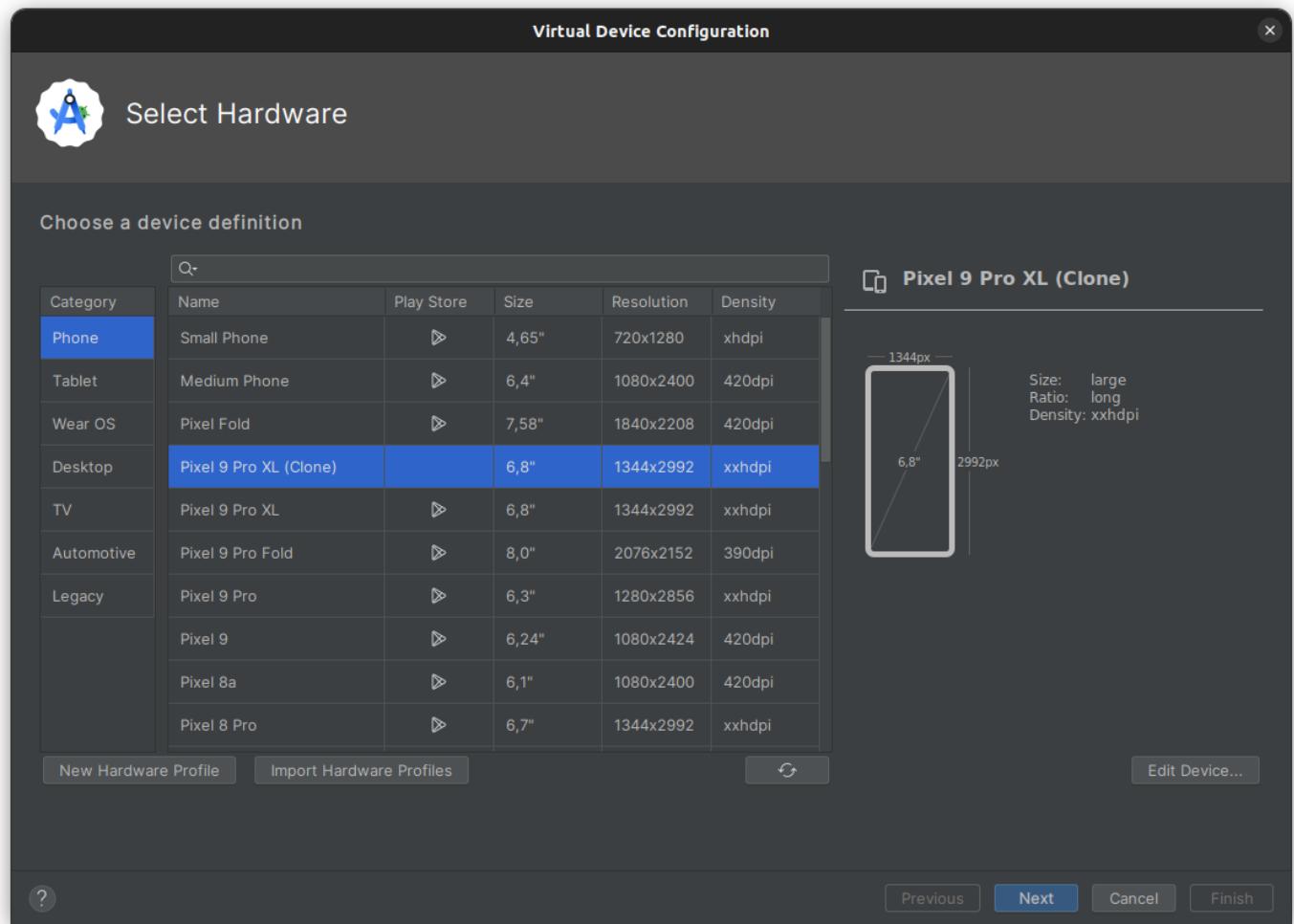
**Figura 40:** Clonando dispositivo

Dispositivo clonado aparecendo na lista de dispositivos



**Figura 41:** Dispositivo clonado

Agora as configurações podem ser alteradas, caso queira, pode adicionar novamente a Play Store no dispositivo conforme o tópico anterior.



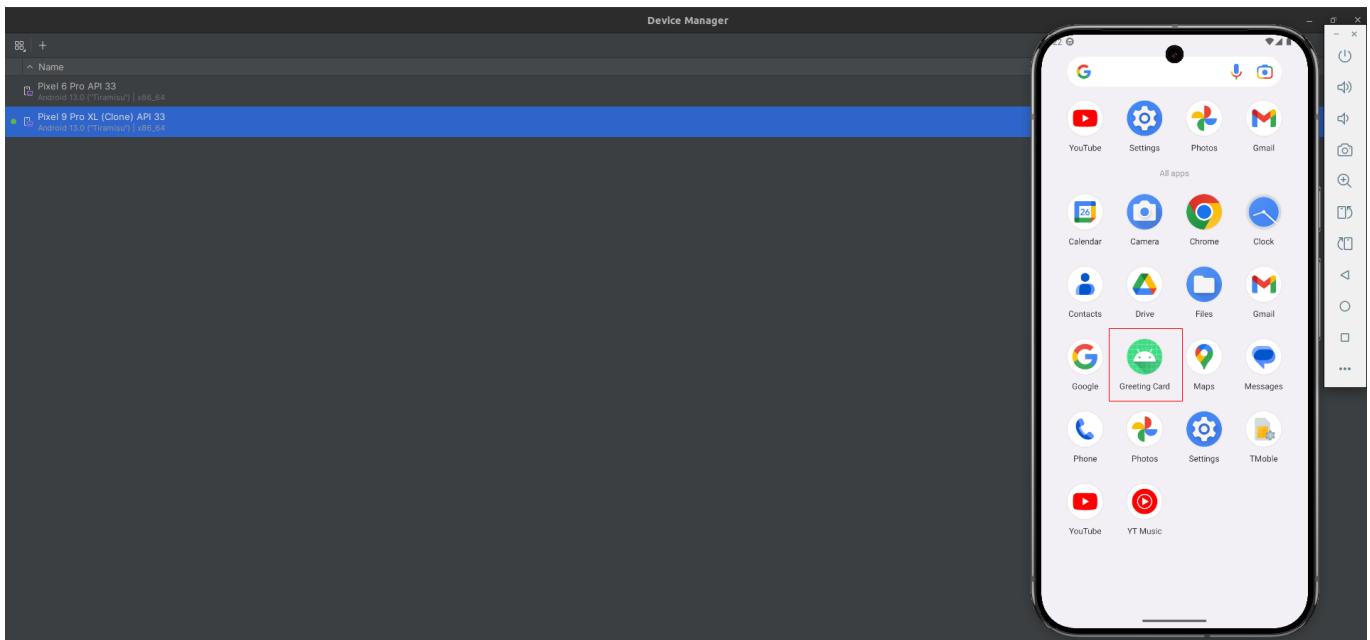
**Figura 42:** Configurando dispositivo clonado



**Dica:** A RAM também pode ser alterada direto no arquivo de configuração do dispositivo, este link[23] mostra o procedimento, muito semelhante ao de adicionar a Play Store.

## 6.4. Instalando APP no Emulador

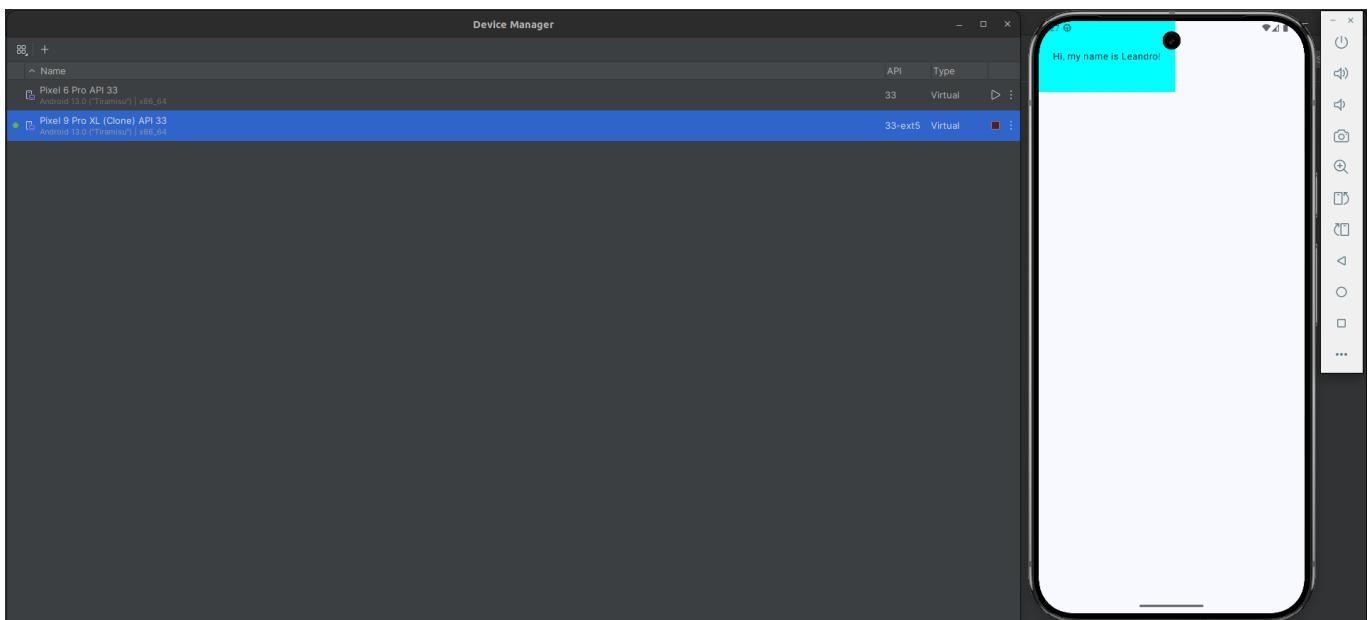
O processo de instalação escolhido para teste foi relativamente simples, apenas arrastando e soltando o arquivo[24] ".apk", gerado de um exemplo de Hello Word, na tela do emulador, depois foi clicado no arquivo e feita a visualização da execução do APP.



**Figura 43:** Aplicativo Greeting Card Instalado

---

#### Rodando aplicativo



**Figura 44:** Rodando aplicativo Greeting Card



**Nota:** [Link\[25\]](#) do exemplo utilizado para criar o APP.

---

## 7. Referências

1. [Android OSP Hardware Requirements](#)
2. [Ubuntu Desktop 22.04.5 LTS](#)
3. [Instalação do Ubuntu Desktop](#)
4. [Sistema Operacional Lubuntu 22.04](#)

5. [Software GRUB](#)
6. [Windows Dual Boot](#)
7. [Software Rufus](#)
8. [Como Instalar Linux Ubuntu Junto do Windows 11 Com Dual Boot Fácil](#)
9. [Ferramenta Git](#)
10. [Java Documentação Oficial.](#)
11. [Biblioteca Padrão do Python](#)
12. [Python on Ubuntu](#)
13. [How to Install Python 3 on Ubuntu 20.04 or 22.04](#)
14. [Download Visual Studio Code](#)
15. [Pacotes Necessários para o Android OSP](#)
16. [Repo README](#)
17. [Configurar o ambiente de build \(AOSP\)](#)
18. [Escolher um destino de Build \(AOSP\)](#)
19. [Build Soong](#)
20. [Instalar o Android Studio no Linux](#)
21. [Android Studio Download](#)
22. [Criar e gerenciar dispositivos virtuais](#)
23. [How to Increase RAM in Android Emulators with Google Play Images](#)
24. [Instalar e adicionar arquivos no AVD](#)
25. [Create your first Android app](#)