ROTEIRO DE INSTALAÇÃO – REACT NATIVE

Siga os passos abaixo para instalar os aplicativos necessários para execução das aplicações criadas em React Native - Learn the Basics · React Native**.**

1º) O que vamos instalar:

- Node + NPM;
  - https://nodejs.org/en/
- Yarn; (`npm install -g yarn`)
  - https://classic.yarnpkg.com/en/docs/install/#windows-stable
- Expo; (`npm install -g expo-cli`)
  - https://docs.expo.io/workflow/expo-cli/
  - https://reactnative.dev/docs/environment-setup
- Visual Studio Code e configurações.
  - https://code.visualstudio.com/

Opcional **(se você tiver espaço em disco e processamento):**

- **Android Studio** https://developer.android.com/
  - **SDK´s 28, 29 e 30**
  - **Android Virtual Device (Emulador)**
  - **JDK >= 8 (Java)** https://openjdk.java.net/install/

Como opção a arquitetura acima, pode-se instalar o Chocolatey, ele já instala o Node e JDK:

Execute o comando abaixo para baixar o Chocolatey https://chocolatey.org/install#individual :

```
Set-ExecutionPolicy Bypass -Scope Process -Force;
[System.Net.ServicePointManager]::SecurityProtocol =
[System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex
((New-Object
System.Net.WebClient).DownloadString('https://chocolatey.org/insta
ll.ps1'))
```

Em seguida execute:

- `choco install -y nodejs.install openjdk8`

Ver também https://codepen.io/ReactDallas/pen/gwOAzK

2º) Baixe o NODE utilizando o link https://nodejs.org/en/download/ .

Instale a versão LTS (recomendada).

3º) Abra o Prompt de Comando ou o Windows Power Shell, em modo Administrador e execute os comandos abaixo para instalar as aplicações indicadas acima:

```
npm install -g yarn

npm install -g expo-cli
```

4º) Se não instalou o Chocolatey, então será necessário instalar o OpenJDK 8 ou superior: https://openjdk.java.net/install/

5º) Se seu computador possui no mínimo 4Gb de RAM e disco rígido com espaço, instale o Android Studio e seus SDK´s 28, 29 e 30, assim como seu Emulador (Android Virtual Device – AVD).

6º) Instale o Visual Studio Code e seus pacotes para utilizarmos a linguagem JavaScript, TypeScript, HTML e CSS.

Adicione o Material Icon Theme, GitHub Pull Requests and Issues e o React Native Tools.

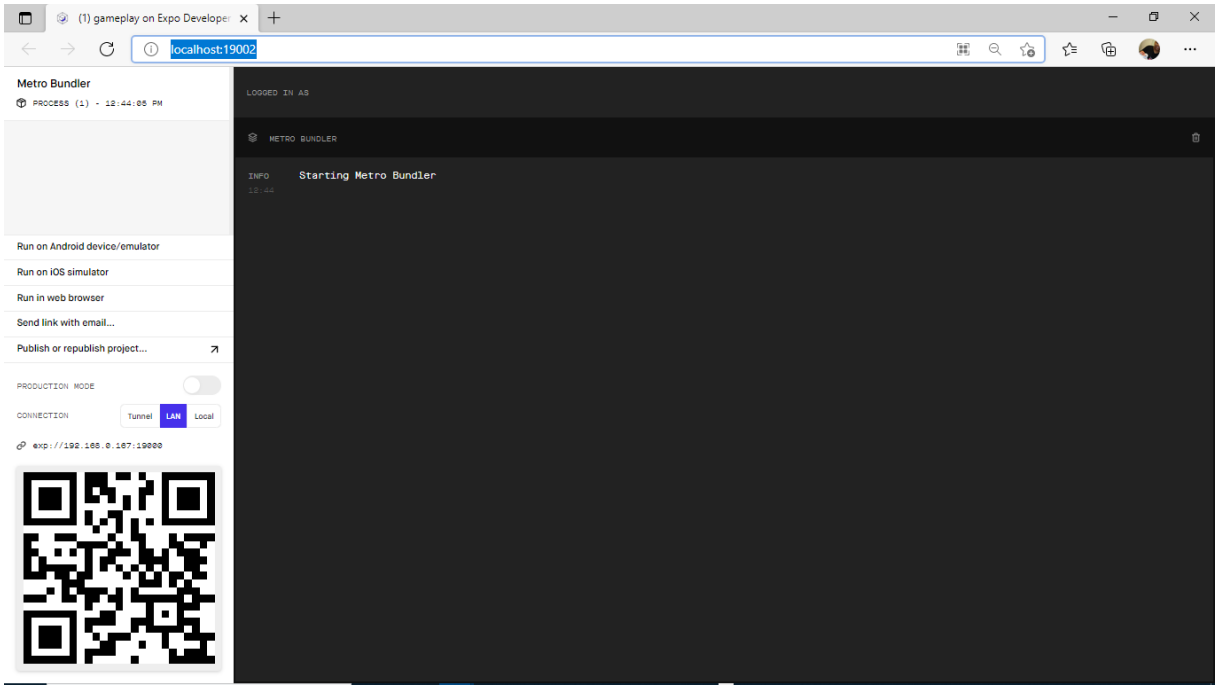
7º) Volte para o Prompt de Comando no modo Administrador e execute os comandos abaixo:

```
CD \

MD PAM1

CD PAM1

expo init gameplay

CD GAMEPLAY

CODE .
```

Será criada uma pasta com o nome *gameplay* dentro da pasta *PAM1*, se o VSCode não estiver configurado para rodar do prompt de comando, abra o VSCode e arraste a pasta gameplay para dentro dele, e vamos codar.

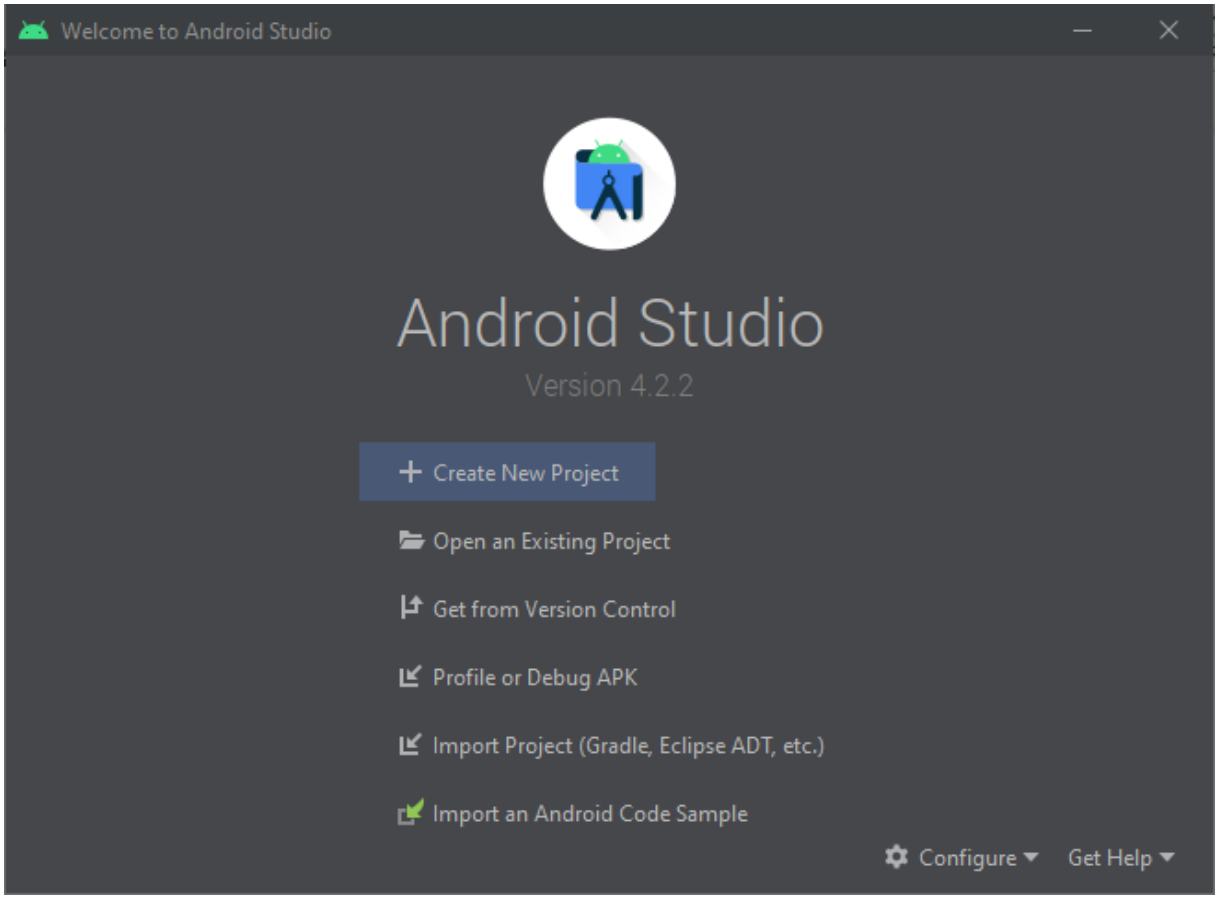
Antes de começar a codificação, teste o código de exemplo criado no arquivo App.tsx. Abra o Terminal do VSCode (ALT+F12) e digite: `expo start`

A página abaixo será aberta:

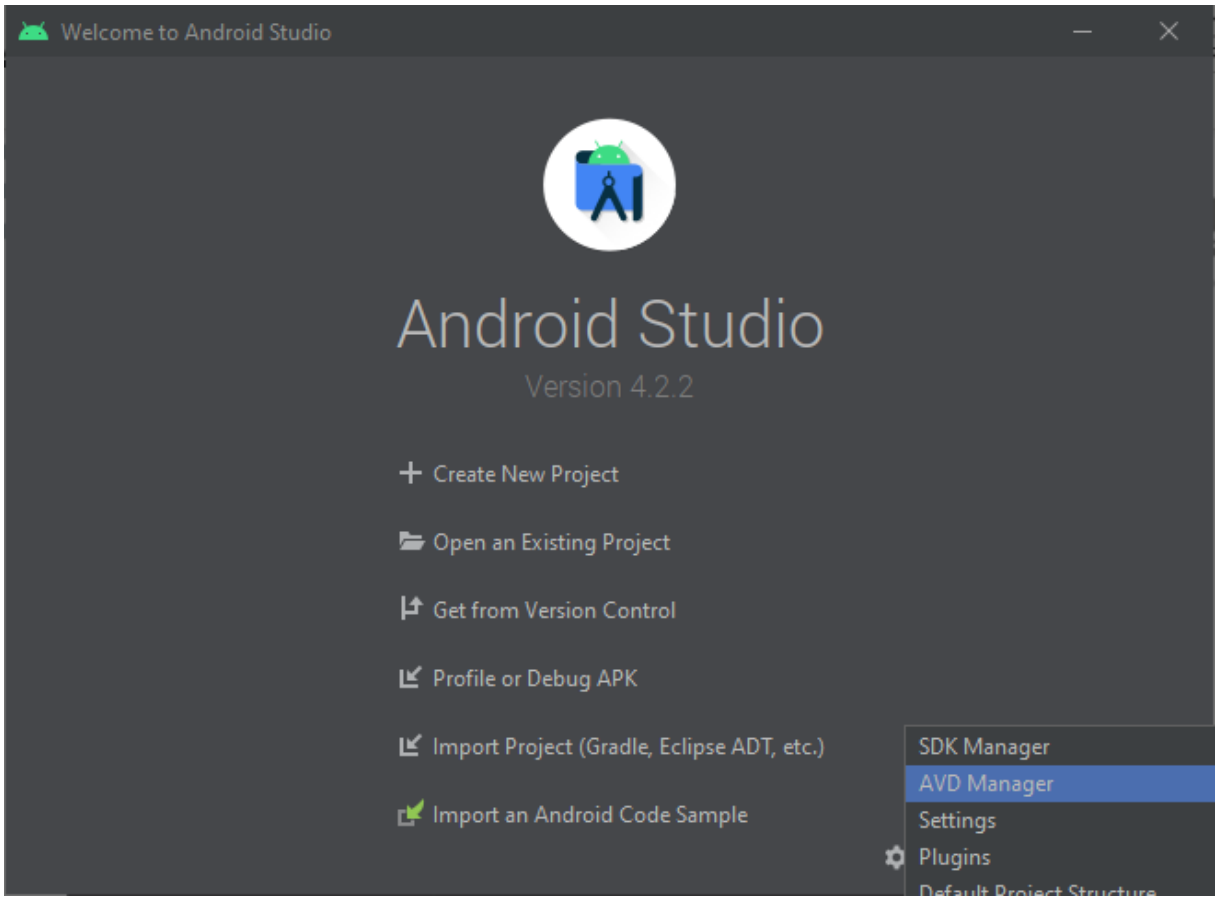


Baixe o aplicativo Expo no seu celular Android, ou Expo Go no celular iPhone, abra o aplicativo e leia o QRCode para rodar a aplicação no seu aparelho.

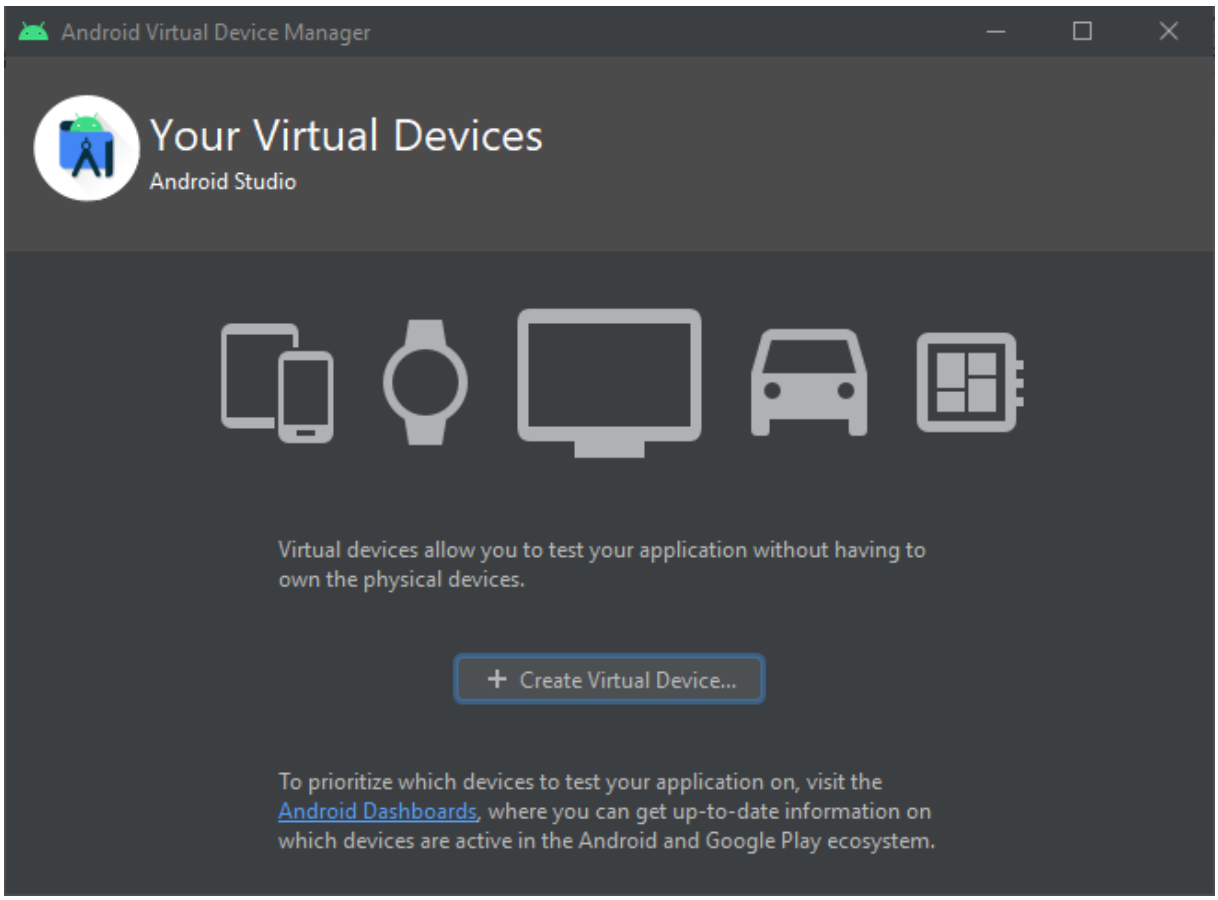
Se optar por rodar o Emulador do Android Studio, abra o Android Studio, na tela inicial procure por uma engrenagem (Configure):



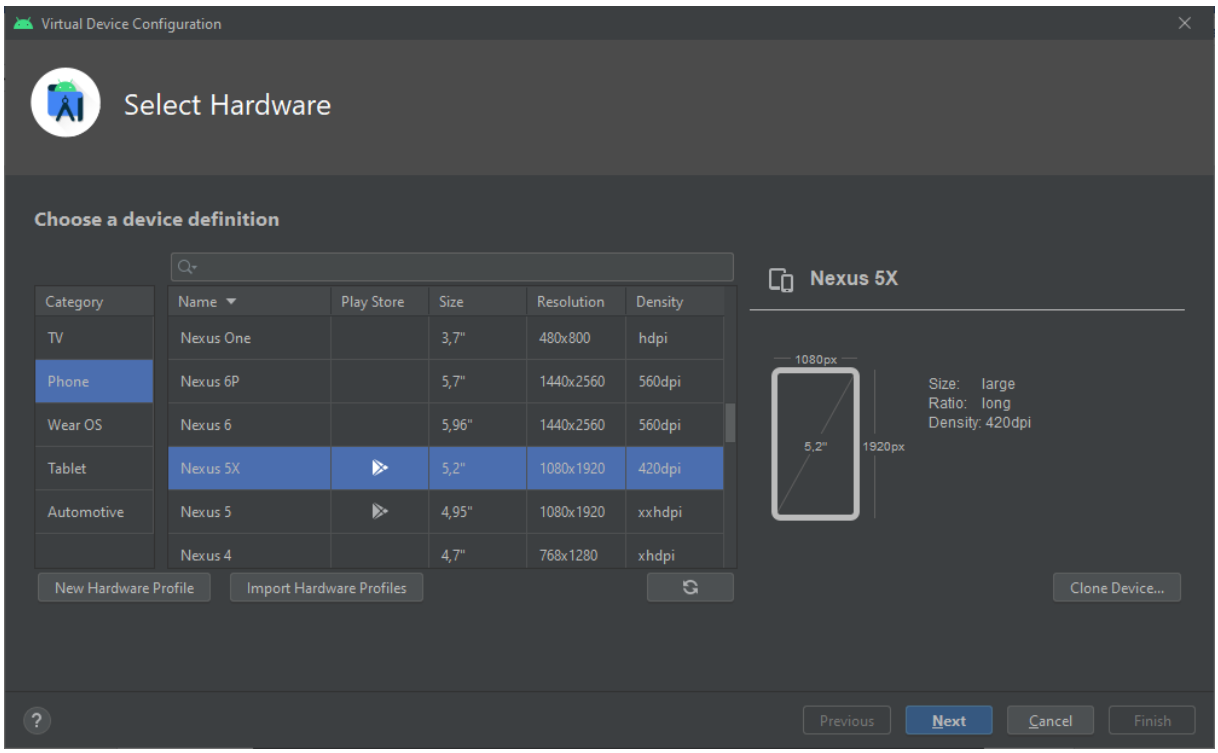
Escolha a opção AVD Manager:



Clique no botão Create Virtual Device

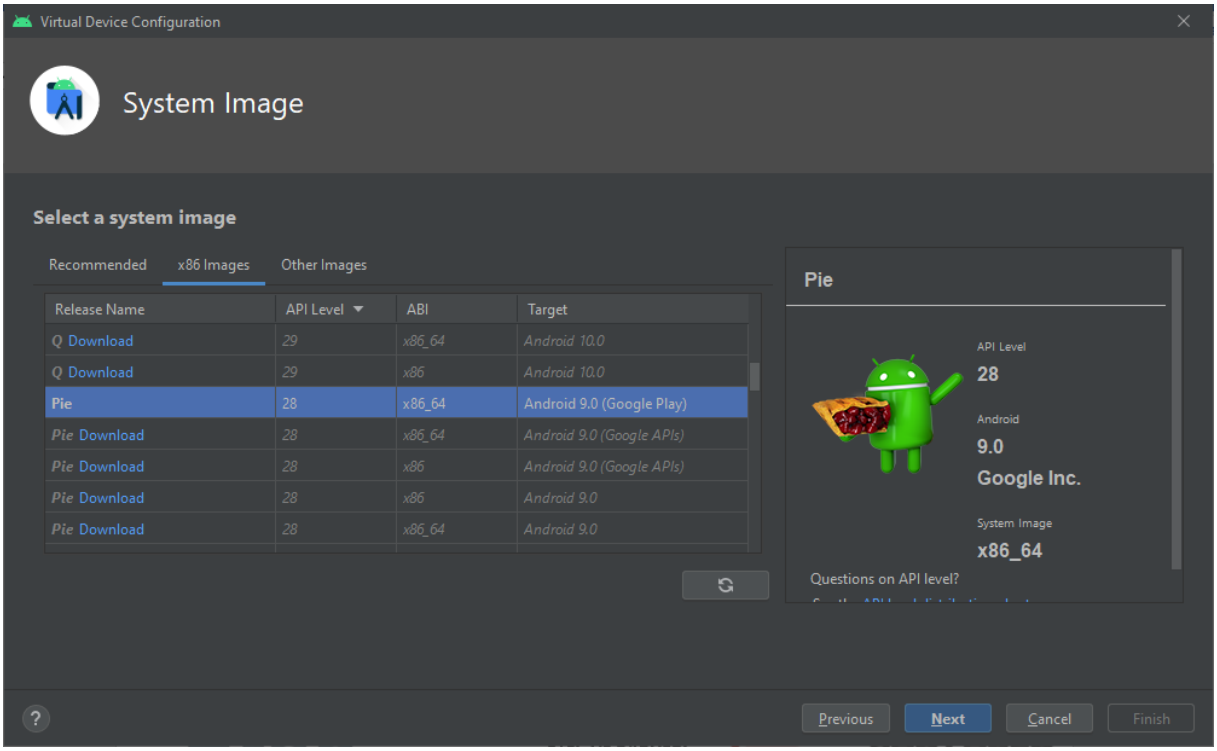


Escolha o aparelho Nexus 5X ou outro de sua preferência e clique em Next:

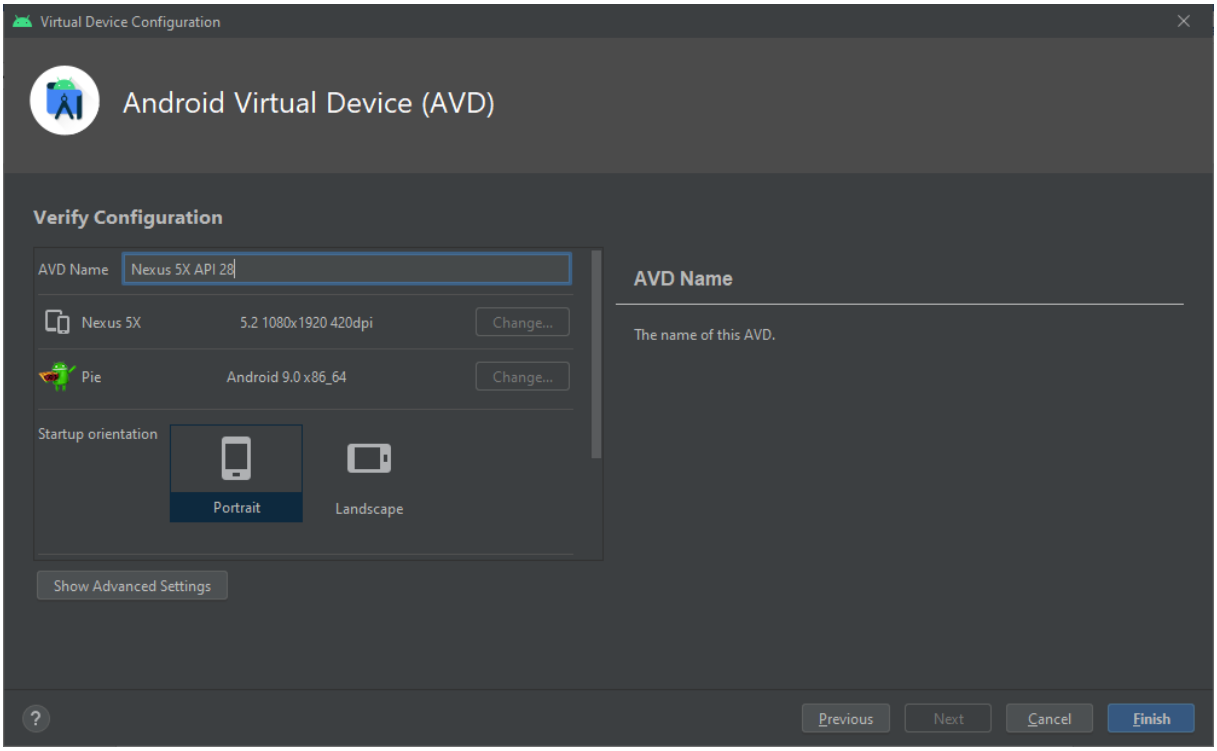


Baixe e instale uma versão do Android (S.O. do emulador) e clique em Next:

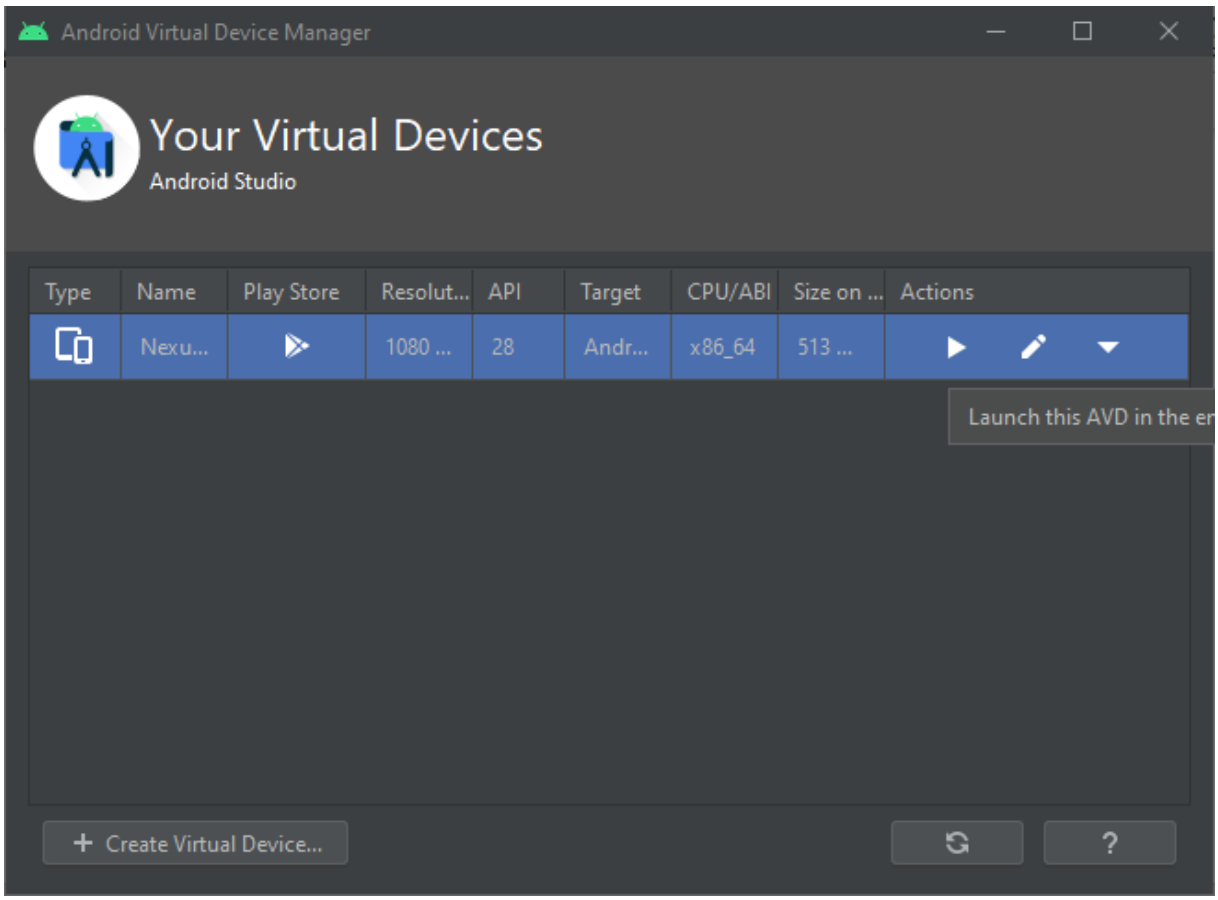
Recomendo a versão Pie SDK 28 - x86_64



Na tela final clique em FINISH, se preferir troque o nome do emulador em AVD Name:

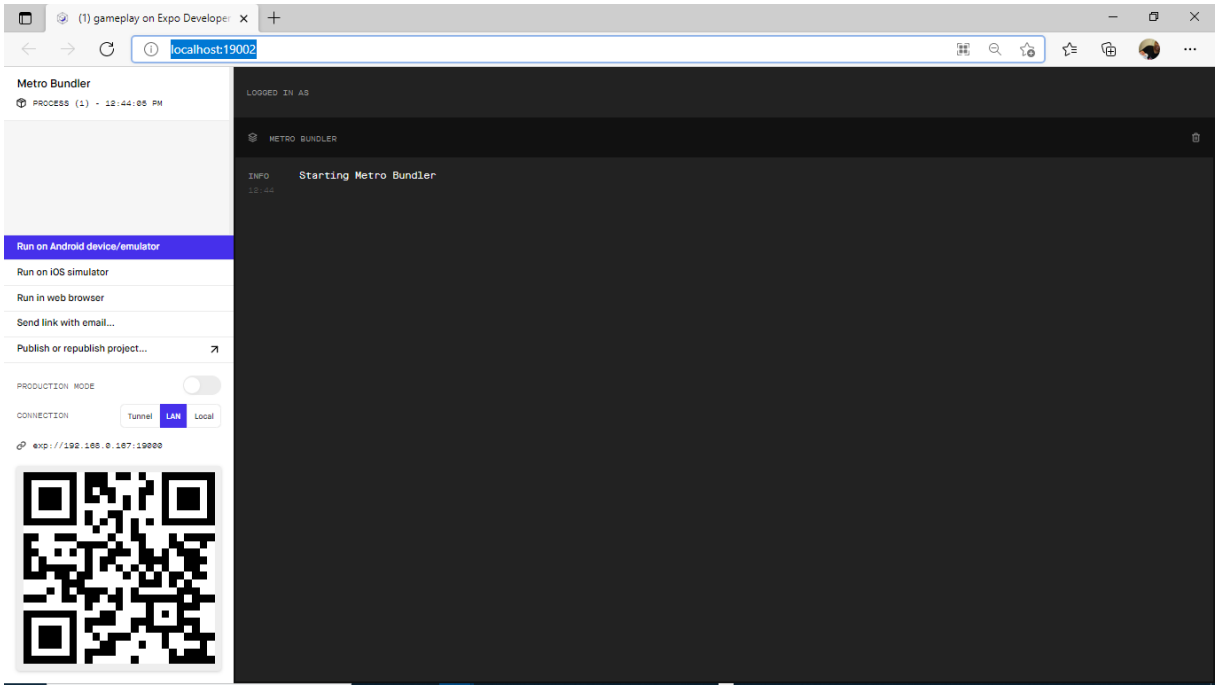


Emulador criado, cliquem no PLAY (Launch this AVD in emulator):



Um emulador do aparelho criado será executado:

Voltando para a página do Expo no computador, há a opção de executar no emulador:



Run on Android device/emulator

Agora sim, vamos codar:

Na raiz do projeto temos um arquivo chamado App.tsx

```
import React from "react";
import { Signin } from './src/screens/Signin';

export default function App(){
  return(
    <Signin />
  );
}
```

Dentro da pasta raiz do projeto gameplay crie uma pasta chamada src, em seguida dentro de src crie uma pasta chamada screens e dentro desta pasta criar outra chamada Signin, nesta pasta vamos criar dois arquivos:

index.tsx

```tsx
import React, { useState} from "react";
import {View, Text, TextInput} from 'react-native';
import { styles } from './styles';

export function Signin(){
  const [text, setText] = useState('Adriano');

  return(
    <View style = {styles.container}>
      <Text>Hello World, NLW Together</Text>

      <TextInput
        style={styles.input}
        onChangeText={setText}
      />

      <Text>
          Você digitou: {text}
      </Text>
    </View>
  );
}
```

styles.ts

```ts
import { StyleSheet } from "react-native";

export const styles = StyleSheet.create({
    container: {
        flex: 1,
        justifyContent: 'center',
        alignItems: 'center',
    },
    input: {
        height: 50,
        width: 200,
        borderBottomWidth: 2,
    }
});
```

Referência Bibliográfica

[1] https://app.rocketseat.com.br/node/mission-react-native/group/nlw-together-react-native/lesson/aula-01-liftoff-2. Rocketseat. NLW Together – julho/2021.