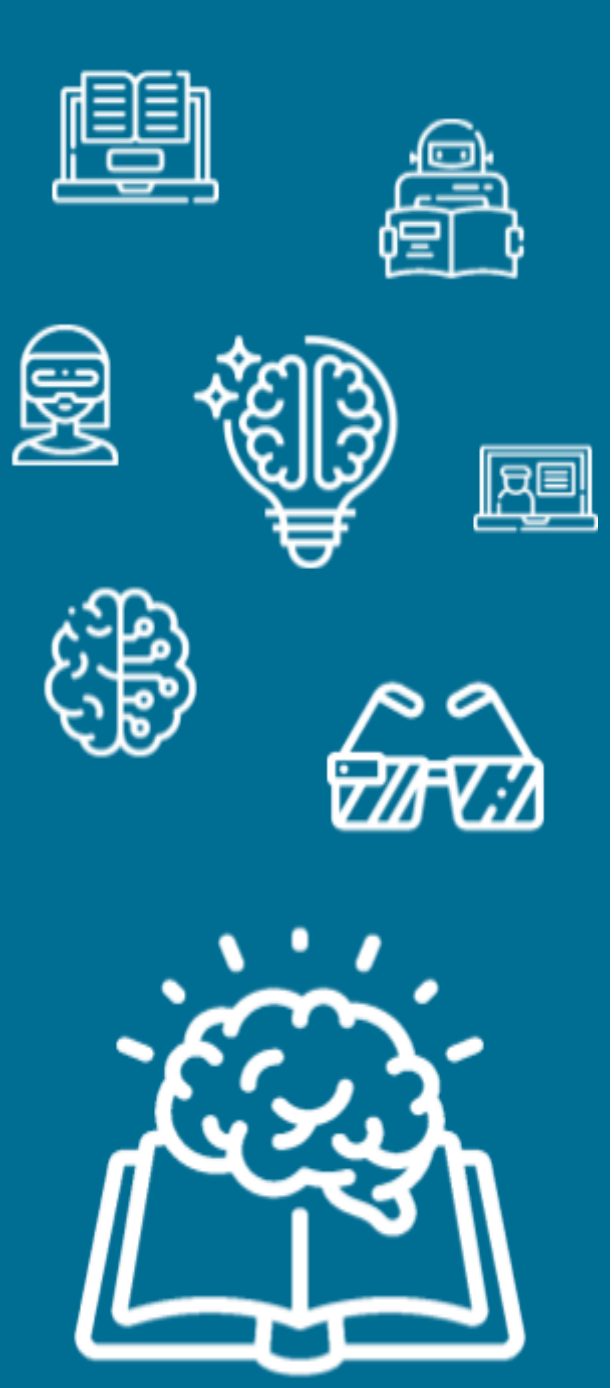


Introdução ao Desenvolvimento para Dispositivos Móveis

[Conceitos Básicos]

Apostila do aluno

cepedi

Me. Ciro Daniel Gurgel de Moura
Vitor Rafael Queiroz Ferreira
Autor da apostila

Me. Ciro Daniel Gurgel de Moura
Vitor Rafael Queiroz Ferreira
Instrutor do curso

Autor

Ciro Daniel Gurgel de Moura

Titulação: Mestre em Ciência da Computação

Áreas de conhecimento: Experiência na área de Sistemas de Informação, com ênfase em Banco de Dados, Mineração de Dados, Geoinformática e Processamento de Imagens, atuando principalmente no Desenvolvimento de Sistemas Web e Aplicações Móveis.

Vitor Rafael Queiroz Ferreira

Titulação: Graduando em Análise e Desenvolvimento de Sistemas

Áreas de conhecimento: Conhecimento na área de Desenvolvimento de Softwares, com ênfase em Desenvolvimento Fullstack, atuando principalmente nos seguintes temas: Typescript; React; NextJS; VueJS; RestAPI. Portfolio: https://vitorrafael.com.br/

APRESENTAÇÃO

Olá! Boas-vindas ao curso de **Introdução ao Desenvolvimento para Dispositivos Móveis** oferecido pela FIT TECH Academy!

O desenvolvimento de aplicativos para dispositivos móveis é um campo tecnológico em constante evolução, que cresce exponencialmente, apresentando constantemente novas soluções e desafios. Além disso, a comunidade de profissionais nesse setor expande-se diariamente, reunindo programadores, designers e entusiastas.

Neste curso, você mergulhará no fascinante mundo dos dispositivos móveis, abrangendo todos os aspectos, desde a concepção inicial das interfaces de aplicativos e seu design visual até a implementação das funcionalidades que tornam os aplicativos úteis e interativos. Além disso, irá explorar a programação para diferentes plataformas, como Android e iOS.

O conteúdo deste curso é fundamentado em diversos materiais elaborados por profissionais especializados e respeitados na área. Para uma compreensão mais profunda, é importante que você leia atentamente este conteúdo e realize as atividades propostas ao longo de cada seção, além de explorar os materiais recomendados nas referências bibliográficas.

A apostila está organizada em sete seções distintas: **Introdução ao desenvolvimento de aplicações** - Onde serão explorados os conceitos de desenvolvimento móvel, oferecendo uma compreensão abrangente de como são pensadas as aplicações; **Entendendo o React Native** - Aqui, será tratado como funciona e o que é o React Native; **Ambiente e Build** - Nesta seção serão dadas dicas e passos de como preparar o ambiente para desenvolver em React Native; **Estilização** - Serão apresentadas soluções iniciais e alternativas para a estilização de aplicações; **Hooks e Context:** Você entenderá como funcionam algumas das funcionalidades padrões e primordiais do React Native; **Navegação:** Nesta seção, você irá compreender como é feita a navegação entre telas; e **Armazenamento local:** Onde irá compreender como guardar informações persistentes.

Desejo a você, estudante, que tenha um excelente curso!!

Boa Leitura !!

Indicação de ícones (opcional)

Saiba mais: *oferece novas informações que enriquecem o assunto ou "curiosidades" e notícias recentes relacionadas ao tema estudado.*

Exemplos: *descreve exemplos sobre o assunto que está sendo abordado.*

Atenção: *indica pontos de maior relevância no texto.*

Avisos: *oferece avisos referente ao assunto..*

Sumário

1 Introdução ao desenvolvimento de aplicações

Hoje em dia, é comum, para a maior parte das pessoas, ter um smartphone em mãos, pronto para ser usado em quase todas as situações. À medida que os dias passam, cada vez mais aplicativos estão disponíveis, e, ao mesmo tempo, mais problemas surgem, estimulando a criação de novas soluções. E é nesse momento que entra em cena o desenvolvedor de aplicativos para dispositivos móveis. Eles são os responsáveis por programar soluções para resolver problemas do dia a dia e também desafios maiores.

1.1 Como desenvolver para diversos dispositivos?

Esta é uma ótima primeira pergunta e o pontapé necessário para entrar no mundo do React Native. A maior parte deste curso vai focar no desenvolvimento de aplicativos híbridos. Isso quer dizer que será desenvolvido um único código e ele será utilizado em aplicativos tanto para iOS quanto para Android. No decorrer da apostila, será possível entender como isso ocorre, e ainda será apresentado o React Native, que é uma ferramenta poderosa para ajudar a resolver esse desafio.

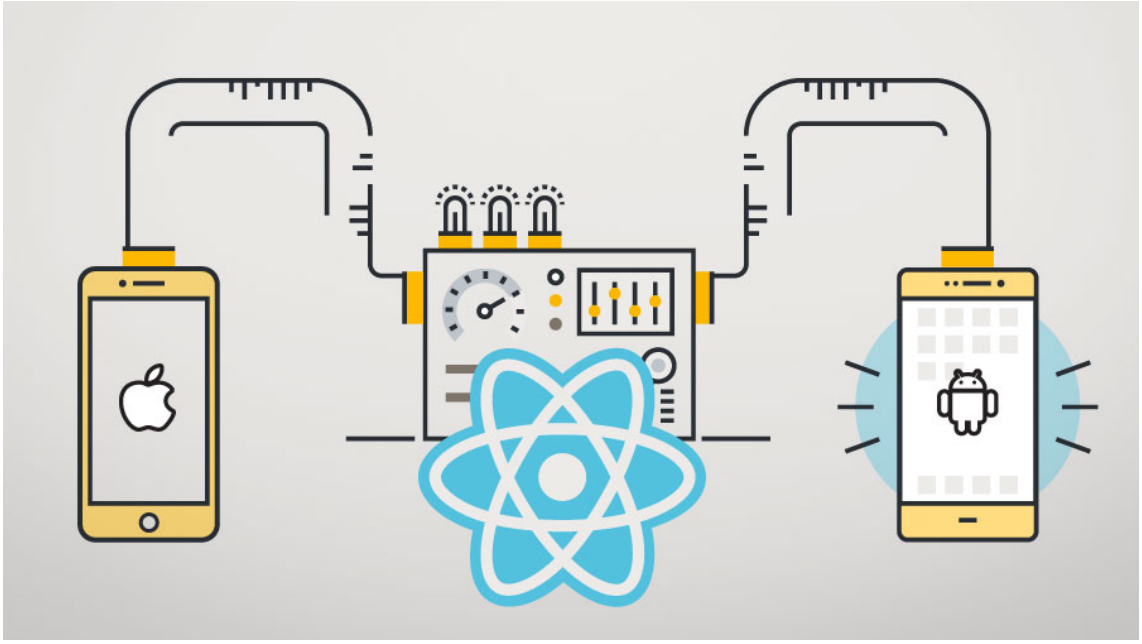


Figura 1 - Ilustração do funcionamento híbrido do React Native. Fonte: https://www.bounteous.com/insights/2016/07/19/react-native-does-universal-code-really-work

2 Entendendo o React Native

O React Native é uma ferramenta de desenvolvimento de aplicativos móveis lançada pelo Facebook em 2015 (React Native, 2023). Ela usa JavaScript e React para criar aplicativos nativos para iOS e Android. A grande vantagem é que você pode compartilhar boa parte do código entre as duas plataformas, economizando tempo e esforço.

Ele surgiu porque os desenvolvedores queriam simplificar o processo de criar aplicativos para diferentes sistemas operacionais móveis. Com o framework, você pode usar o mesmo código para iOS e Android, mantendo a flexibilidade e a eficiência dos aplicativos nativos. Desde seu lançamento, ele se tornou muito popular por acelerar o desenvolvimento de aplicativos e criar interfaces de usuário modernas e eficientes.

Saiba mais: Acesse o site oficial do React Native para conhecer mais sobre o assunto, ver a documentação oficial, obter novidades e também ficar por dentro de quais empresas estão utilizando-o. Visite agora mesmo: https://reactnative.dev/

2.1 Desenvolvimento Híbrido

A criação de uma única base de código para dois sistemas (iOS e Android) é possível graças à abordagem de desenvolvimento multiplataforma. Os desenvolvedores escrevem o código em JavaScript usando a biblioteca React e, em seguida, o React Native traduz esse código para JavaScript, que passa pelo processo de *bundling*, ou empacotamento, e é enviado para as plataformas nativas, onde será mapeado o código empacotado para ser transformado em elementos nativos específicos de cada plataforma. Ou seja, em vez de desenvolver duas versões separadas do aplicativo, os desenvolvedores podem compartilhar a maioria do código entre iOS e Android, economizando tempo e esforço. Essa abordagem permite uma manutenção mais eficiente e uma experiência de usuário consistente em ambas as plataformas. Veja a figura 2, que ilustra como isso é feito por baixo dos panos.

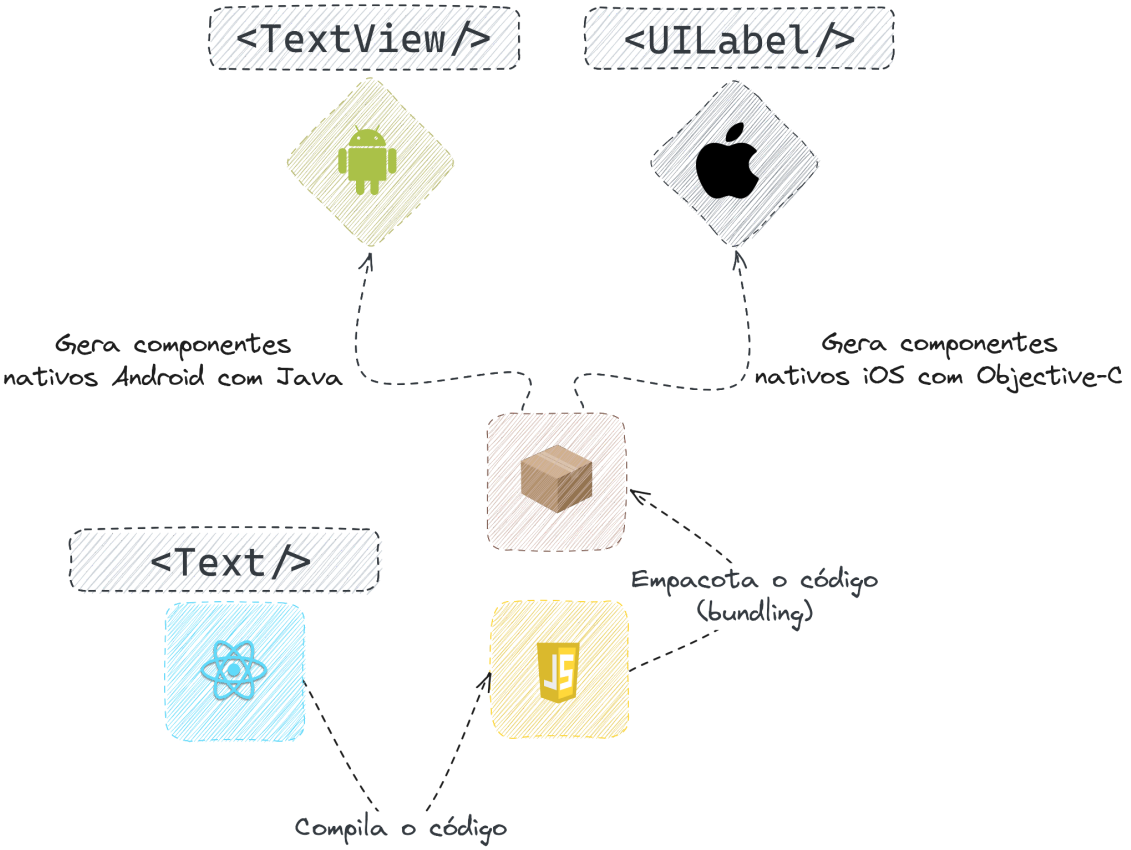


Figura 2 - Um exemplo de como funciona o processo de compilação do React Native para iOS e Android. Fonte: Reprodução do autor.

2.2 Expo: A ferramenta de desenvolvimento em React Native

O Expo é um conjunto de ferramentas e serviços construído em cima do React Native que simplifica ainda mais o processo de desenvolvimento de aplicativos móveis (Expo, 2023). Ele fornece um ambiente completo para criar, testar e implantar aplicativos sem a necessidade de configurar e gerenciar nativamente a infraestrutura específica de cada plataforma.

Com este, os desenvolvedores podem se concentrar no código do aplicativo, enquanto ele cuida de tarefas complexas, como a compilação nativa, a gestão de certificados de assinatura de aplicativos e a configuração do ambiente de desenvolvimento. Além disso, o Expo oferece uma variedade de APIs prontas para uso, como câmera, geolocalização e notificações, facilitando a incorporação de funcionalidades comuns nos aplicativos.

Para desenvolvedores iniciantes ou para projetos mais simples, o mesmo pode ser uma escolha conveniente, permitindo que eles aproveitem os benefícios do React Native de maneira mais simplificada.

2.3 Sintaxe JSX

Assim como no React, o React Native emprega interfaces declarativas através da sintaxe JSX (JavaScript Syntax Extension), proporcionando uma abordagem intuitiva para a construção de interfaces em diversos dispositivos. Este tipo de interface apresenta uma sintaxe otimizada para elementos móveis, permitindo a tradução eficiente para elementos nativos. Similar ao HTML, utiliza tags para delimitar inícios e fins de informações.

O JSX no React Native segue o paradigma funcional, onde cada componente é uma função. Por exemplo, o componente fundamental, chamado App, é uma função que retorna os elementos que compõem a interface do usuário. Esta abordagem funcional facilita a criação, reutilização e compreensão de componentes.

Ao criar novos arquivos de componente, é recomendado utilizar a extensão .tsx para a versão TypeScript do JSX. Vale notar que a extensão .ts não será reconhecida como JSX, sendo reservada para abstrações, como funcionalidades globais. Essa prática proporciona um ambiente de desenvolvimento organizado e orientado a tipos, melhorando a manutenção e escalabilidade do projeto.

2.4 Ciclo de Renderização

Compreender o fluxo de processos no React Native é essencial para desenvolvedores que buscam construir interfaces eficientes e dinâmicas. A analogia abaixo utiliza uma confeitaria para ilustrar como o React, funcionando como gerente, coordena a criação e atualização de componentes. Essa abordagem facilita a compreensão do ciclo de vida dos componentes e do papel crucial do React na renderização eficiente da interface do usuário. A seguir, será explorado como esse fluxo ocorre na prática, destacando o acionar, renderizar e entregar, além de como o React realiza pedidos de renderização em duas situações específicas.

Imagine uma aplicação chamada "Confeitaria". Os componentes correspondem aos confeiteiros, sendo cada um especializado em criar um tipo específico de bolo, como cenoura, chocolate ou amanteigado. O React, neste contexto, representa o gerente da confeitaria, coordenando os pedidos dos clientes.

2.4.1 Acionar

Nesta etapa, o gerente (React) chama os confeiteiros (componentes) e repassa os pedidos dos clientes (renderização).

2.4.2 Renderizar

Aqui, os confeiteiros (componentes) confeccionam os bolos (renderização) conforme solicitado. Cada confeiteiro utiliza os ingredientes necessários (props) para criar o bolo específico.

2.4.3 Entregar

O gerente (React) recebe os bolos dos confeiteiros e os entrega aos clientes (interface do usuário).

O React realiza esse ciclo de processos em duas situações:

- **Inicialização da Aplicação:**

  Quando a aplicação é iniciada, o React realiza a primeira renderização.

- **Atualização de Estado:**

  Sempre que um estado (state) é atualizado, o React solicita uma renderização para ajustar a interface às mudanças. Esse processo envolve:

- Atualizar o componente com as mudanças.
- Comparar e identificar quais componentes precisam ser re-renderizados, percorrendo a árvore de componentes de forma eficiente.
- Modificar na DOM apenas o necessário, otimizando a performance da aplicação.

3 Ambiente e Build

O objetivo deste conteúdo é auxiliar no entendimento de pontos básicos para desenvolver em React Native, desde a configuração do ambiente até a criação do primeiro projeto e também um repositório.

Ao longo das próximas seções serão abordadas as configurações necessárias para preparar o ambiente em dois sistemas operacionais amplamente utilizados pela comunidade: Linux e Windows.

3.1 Windows

Para iniciar o desenvolvimento mobile no Windows, é necessário atender a alguns requisitos mínimos. Abaixo, será explicado como instalar as dependências necessárias para executar o primeiro projeto em React Native utilizando o Expo.

Fique ligado!
Atualmente, recomenda-se executar o Expo usando o comando **npx expo**, sem a necessidade de instalar um pacote adicional.

3.1.1 Git

Comece pela instalação do Git. Vá até a página de Downloads do Git https://git-scm.com/download/win. Selecione o instalador de acordo com a arquitetura do seu sistema: 32 ou 64 bits.

Saiba mais: Acesse o site oficial do Git para conhecer mais sobre essa ferramenta, seu uso e documentação. Visite agora mesmo: https://git-scm.com/doc

No primeiro passo, serão apresentados os termos de uso da aplicação.

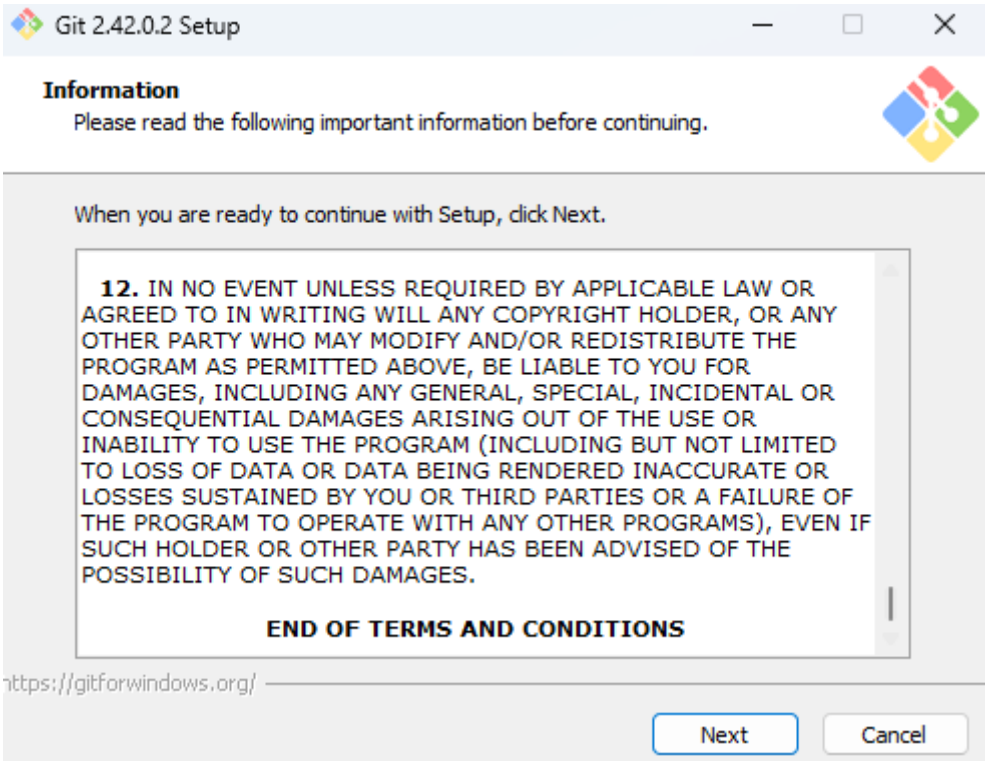


Figura 3 - Tela de Termos do Git. Fonte: Reprodução do autor.

Em seguida, será solicitado o diretório de instalação. É recomendável que você utilize o padrão para facilitar configurações futuras.

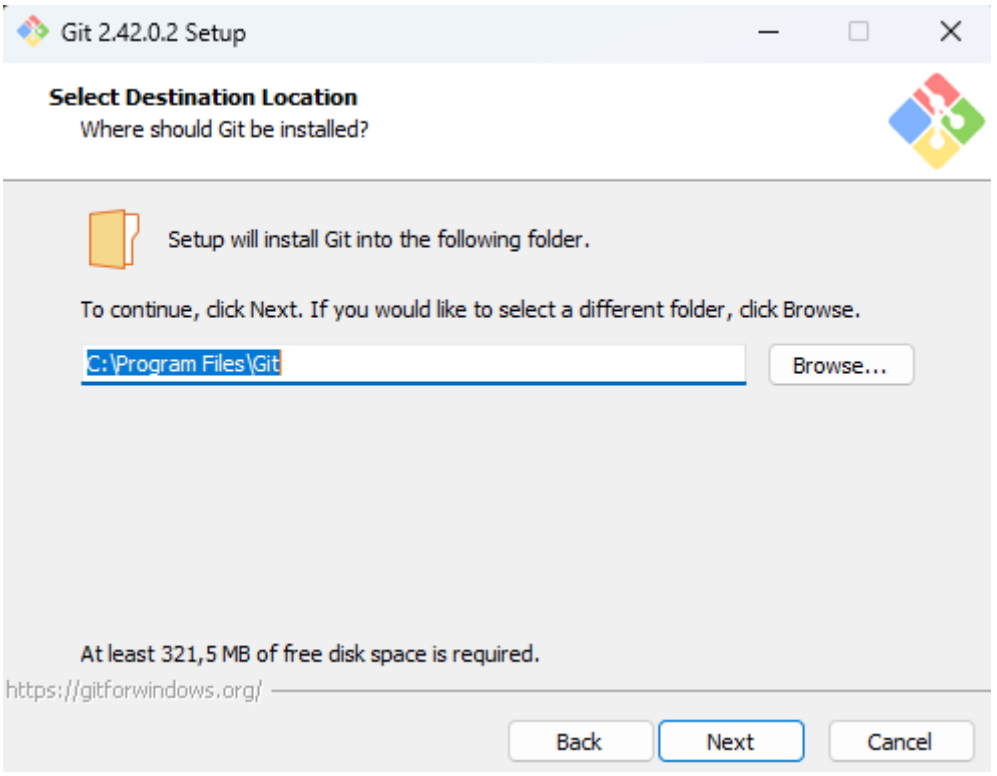


Figura 4 - Tela de seleção do local de instalação do Git. Fonte: Reprodução do autor.

Aqui pode ser utilizada a configuração padrão. Se desejar, marque a opção "Additional Icons - On the Desktop" para criar um atalho na área de trabalho do sistema.

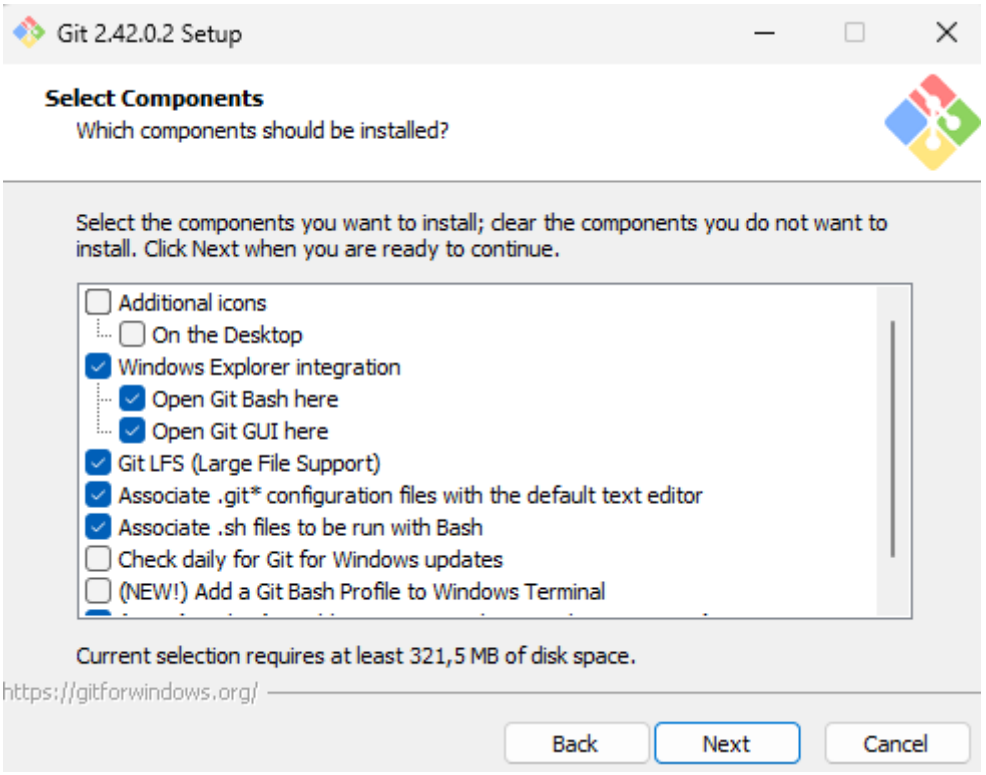


Figura 5 - Tela com opções de componentes do Git. Fonte: Reprodução do autor.

Você pode optar por não criar um atalho no Menu Iniciar do Windows, marcando "Don't create a Start Menu folder" e também pode escolher o nome do atalho.

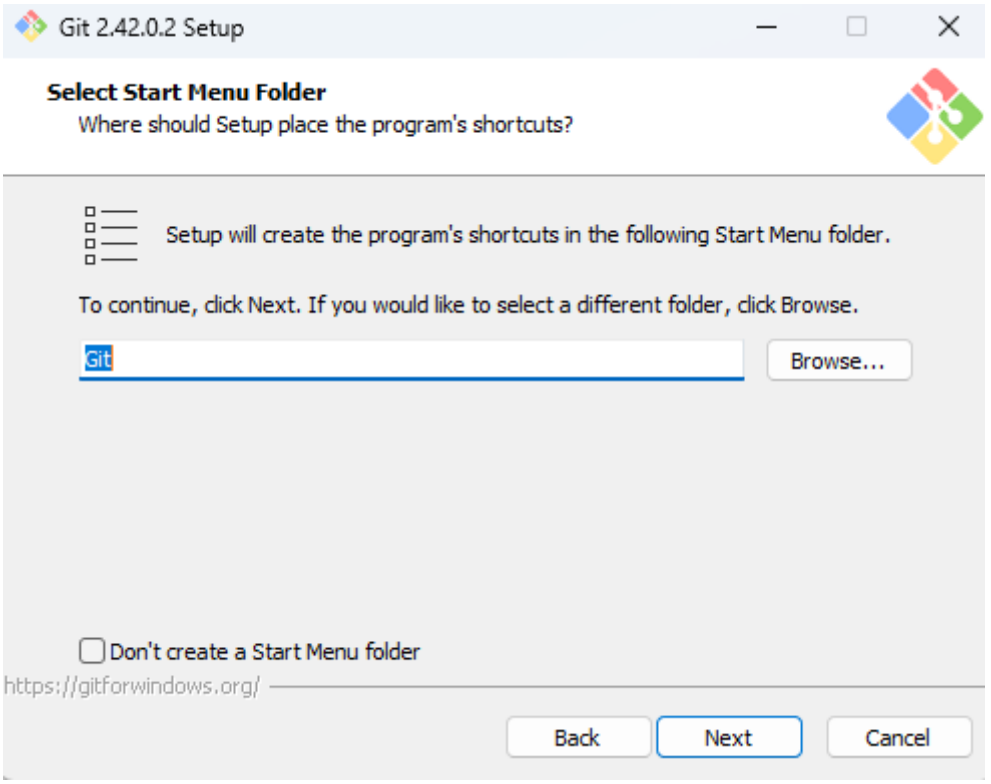


Figura 6 - Tela com opção de ícone do Git no menu iniciar. Fonte: Reprodução do autor.

Aqui você pode escolher o editor de texto padrão para comandos Git. O Vim é leve, mas não é muito intuitivo. Pode ser interessante utilizar o Notepad ou o VSCode.

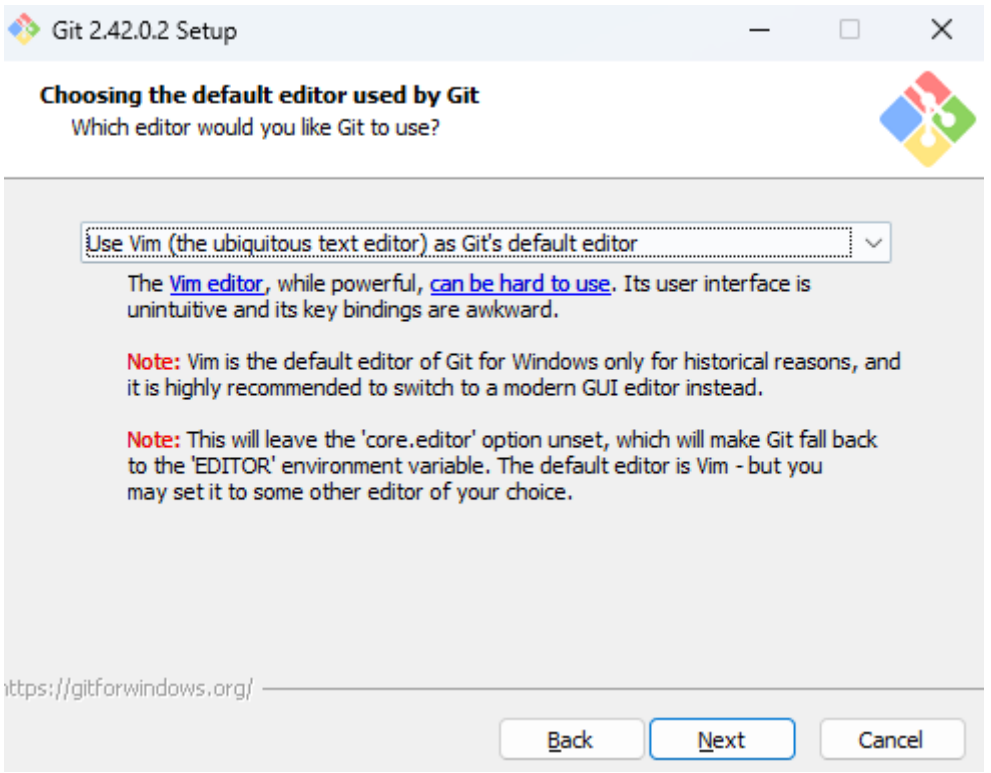


Figura 7 - Tela de seleção do editor de texto padrão do Git. Fonte: Reprodução do autor.

Ao iniciar um repositório Git, é criada automaticamente uma branch padrão chamada de "master". É possível alterar esse nome para qualquer outro utilizando a segunda opção, sendo "main" a opção mais comumente utilizada (Git, 2023).

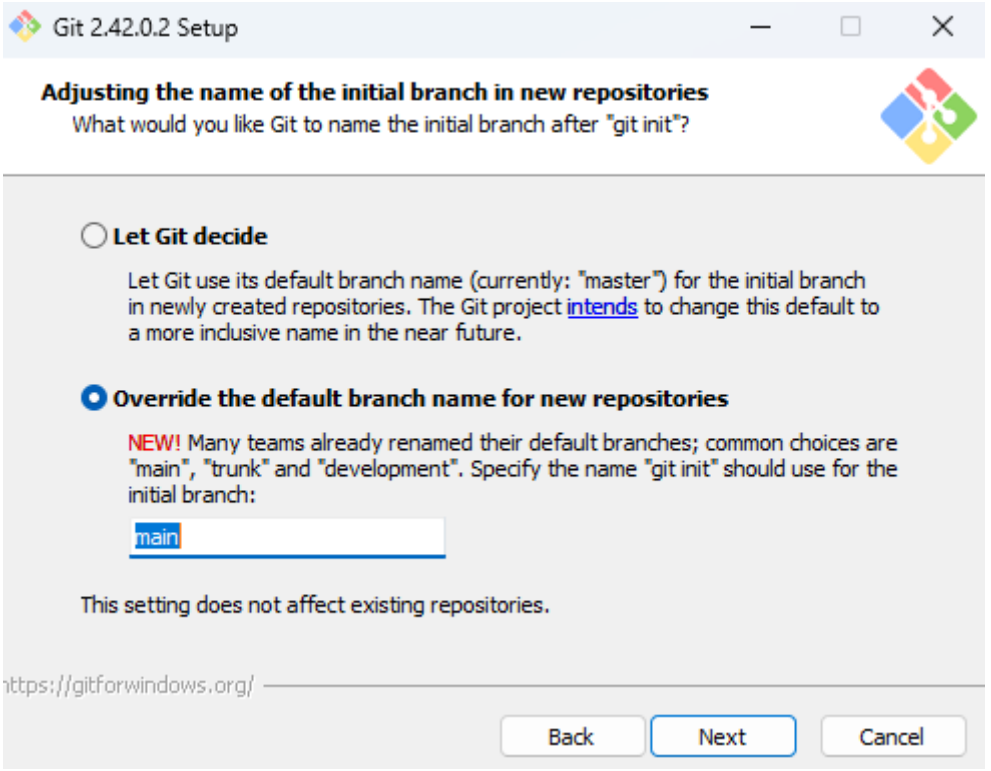


Figura 8 - Tela de ajuste do nome da branch inicial do Git. Fonte: Reprodução do autor.

Você pode escolher como o Git será executado. ATENÇÃO: A opção padrão irá alterar as variáveis de ambiente do seu sistema, mas é a opção mais recomendada, pois você poderá usar o Git em outros terminais além do Git Bash.

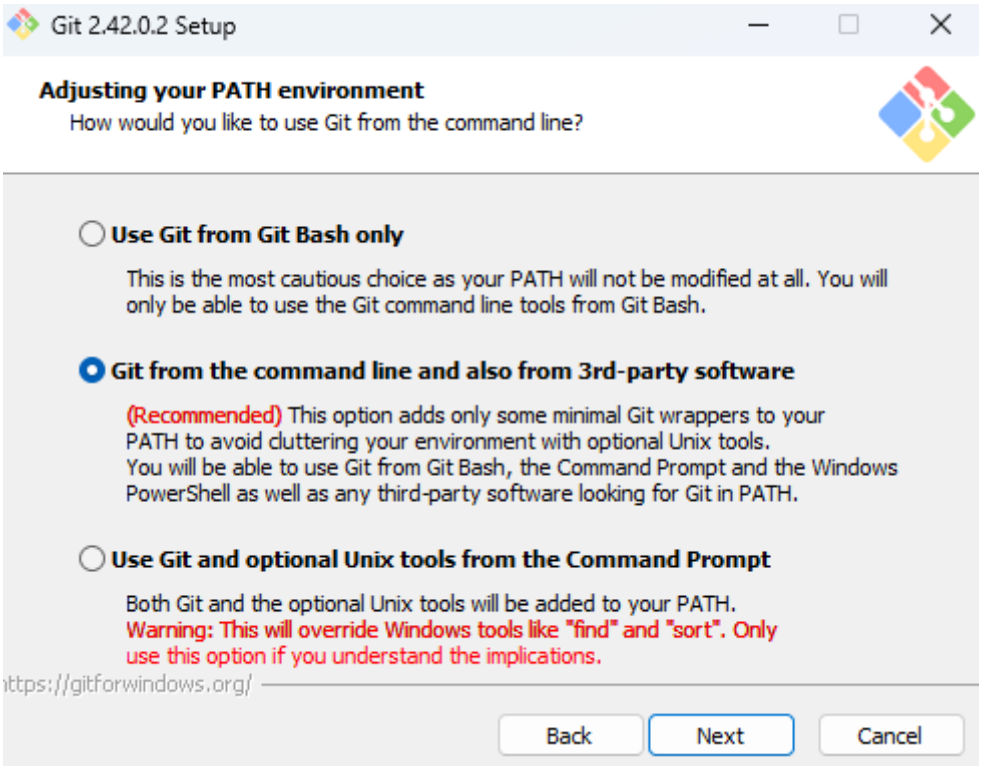


Figura 9 - Tela de variáveis de ambiente do Git. Fonte: Reprodução do autor.

Você pode escolher entre usar um algoritmo de conexão SSH já embutido no Git ou utilizar um externo. É recomendado que seja utilizado o integrado.

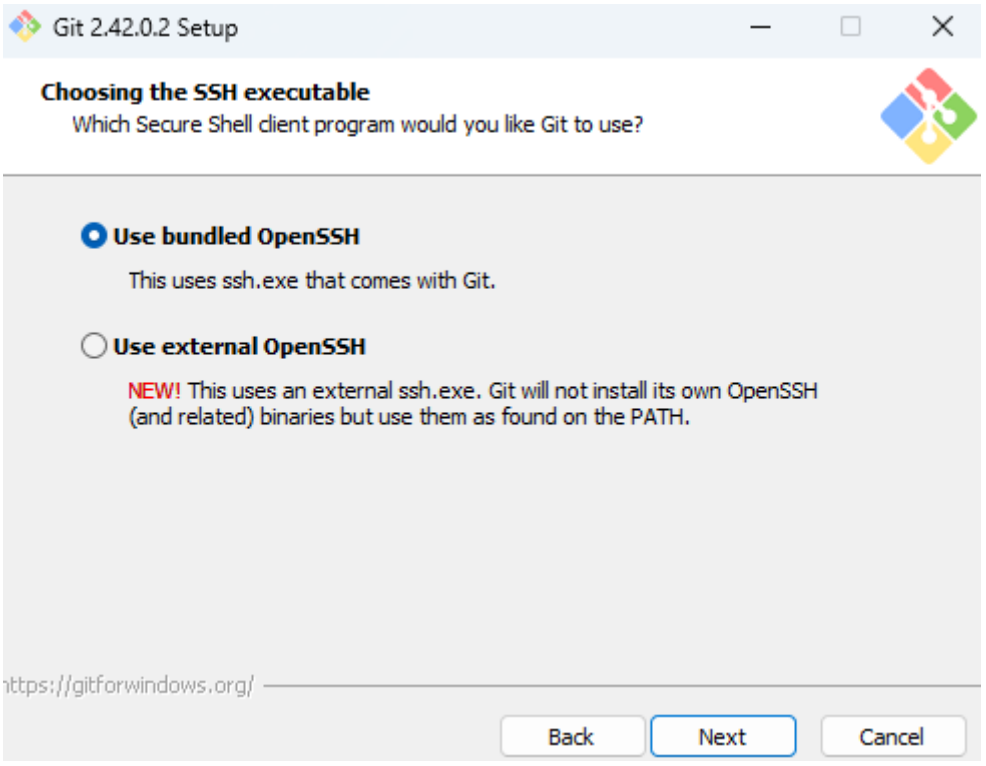


Figura 10 - Tela de configuração do SSH do Git. Fonte: Reprodução do autor.

Você poderá escolher o tipo de validação de certificados que será utilizado. É recomendado utilizar a opção padrão (OpenSSL).

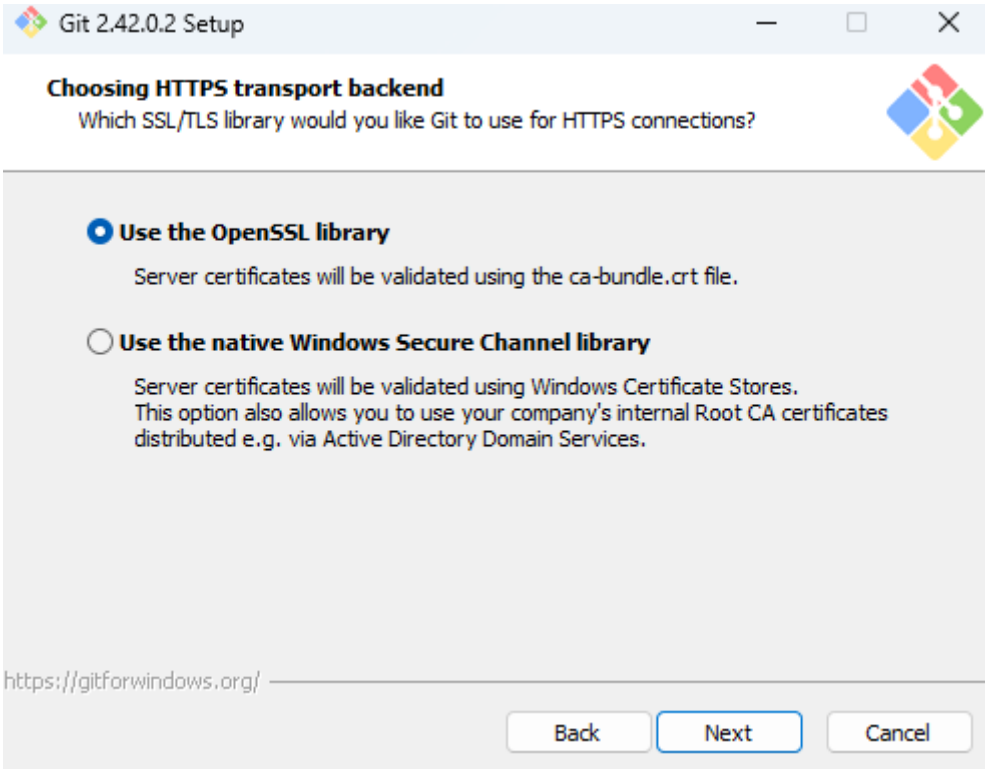


Figura 11 - Tela de configuração do HTTPS do Git. Fonte: Reprodução do autor.

Aqui, você pode escolher como o Git irá tratar os fins de linha em arquivos de texto. A primeira opção, que é a padrão, é recomendada para cross-platform.

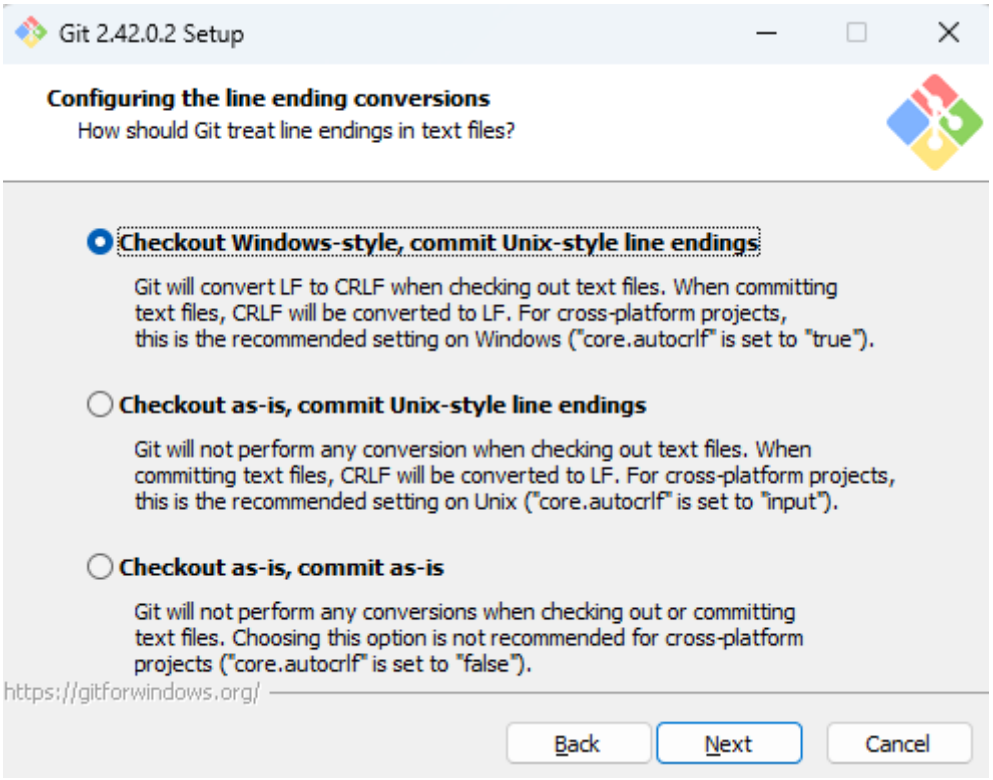


Figura 12 - Tela de configuração do final das linhas de arquivo do Git. Fonte: Reprodução do autor.

É possível escolher qual emulador de terminal você deseja utilizar com o Git Bash. No entanto, é recomendado que você use o MinTTY devido a limitações com caracteres unicode e execução de scripts do CMD.

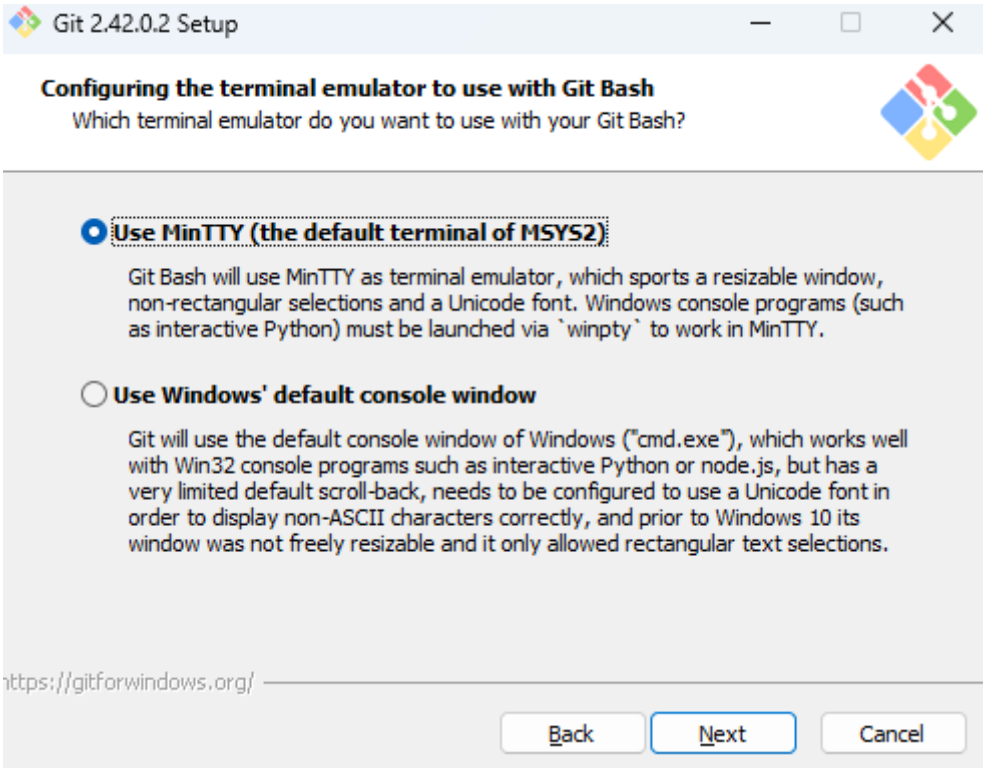


Figura 13 - Tela de configuração de terminal do Git. Fonte: Reprodução do autor.

Abaixo, você irá decidir como tratar o comando git pull. É recomendado que se utilize a opção padrão.

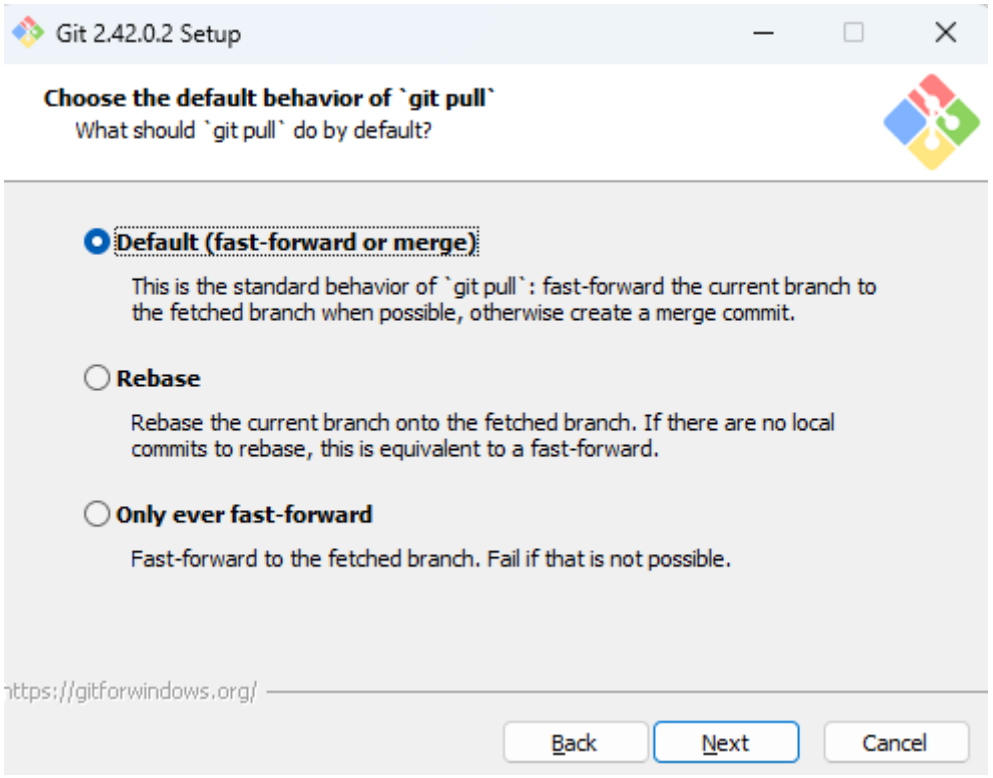


Figura 14 - Tela de configuração do git pull. Fonte: Reprodução do autor.

Se desejar, você pode utilizar o gerenciador de credenciais do Git.

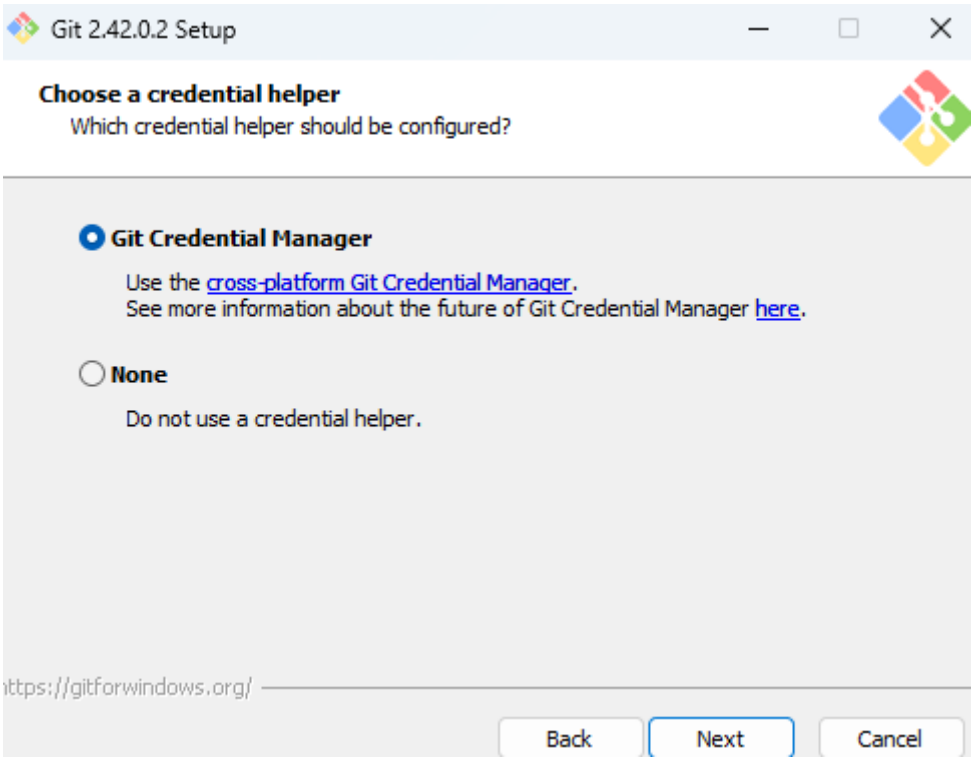


Figura 15 - Tela de configuração de credenciais do Git. Fonte: Reprodução do autor.

Aqui, você pode utilizar funcionalidades do Git, como Symbolic Links e Caching. Para este guia, iremos utilizar apenas o caching do sistema.

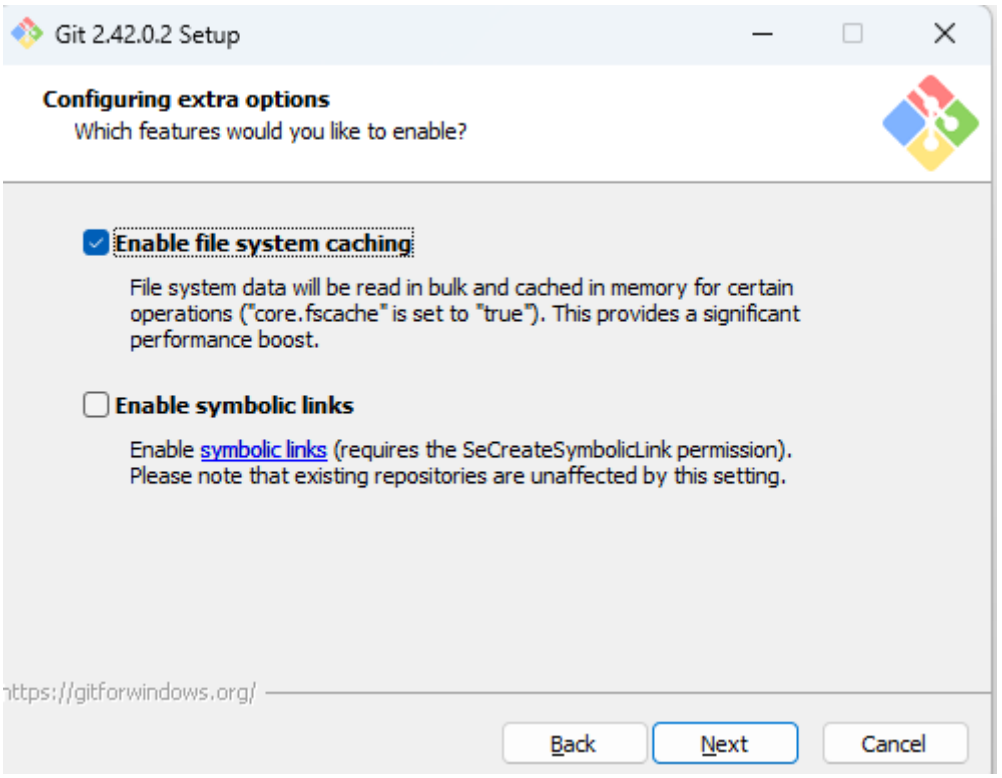


Figura 16 - Tela de configuração de funcionalidades do Git. Fonte: Reprodução do autor.

Essas funcionalidades são experimentais no momento. Utilize por sua conta e risco.

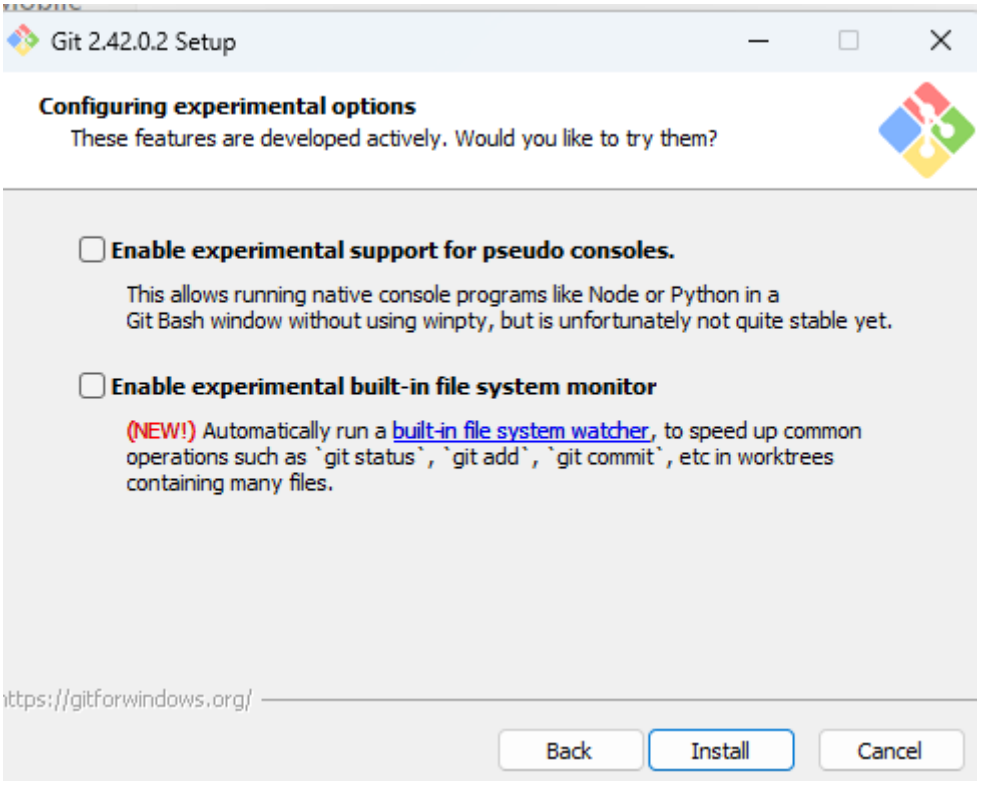


Figura 17 - Tela de configuração das funcionalidades experimentais do Git. Fonte: Reprodução do autor.

Com isso, o Git já será instalado e estará pronto para uso. Agora iremos partir para o próximo passo.

3.1.2 Node

Para utilizar JavaScript, é essencial instalar o Node.js (Node.js, 2023). Abaixo, há um guia sobre como fazer isso. Acesse a página de Download do Node.js em Download | Node.js. É recomendado instalar a versão LTS (Long-Term Support).

Termos de uso

Figura 18 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Escolha o diretório de instalação. É recomendado deixar o padrão.

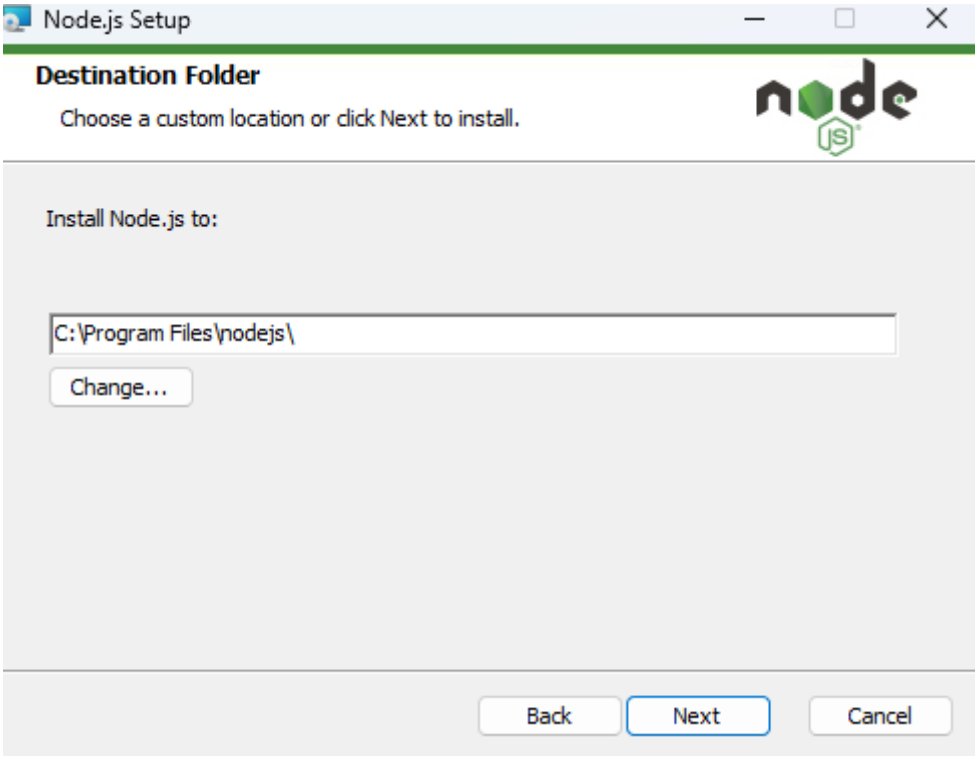


Figura 19 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Esta é uma parte crucial da instalação e é recomendado utilizar a configuração padrão. O instalador cuidará da configuração do ambiente do Node, incluindo as variáveis de ambiente.

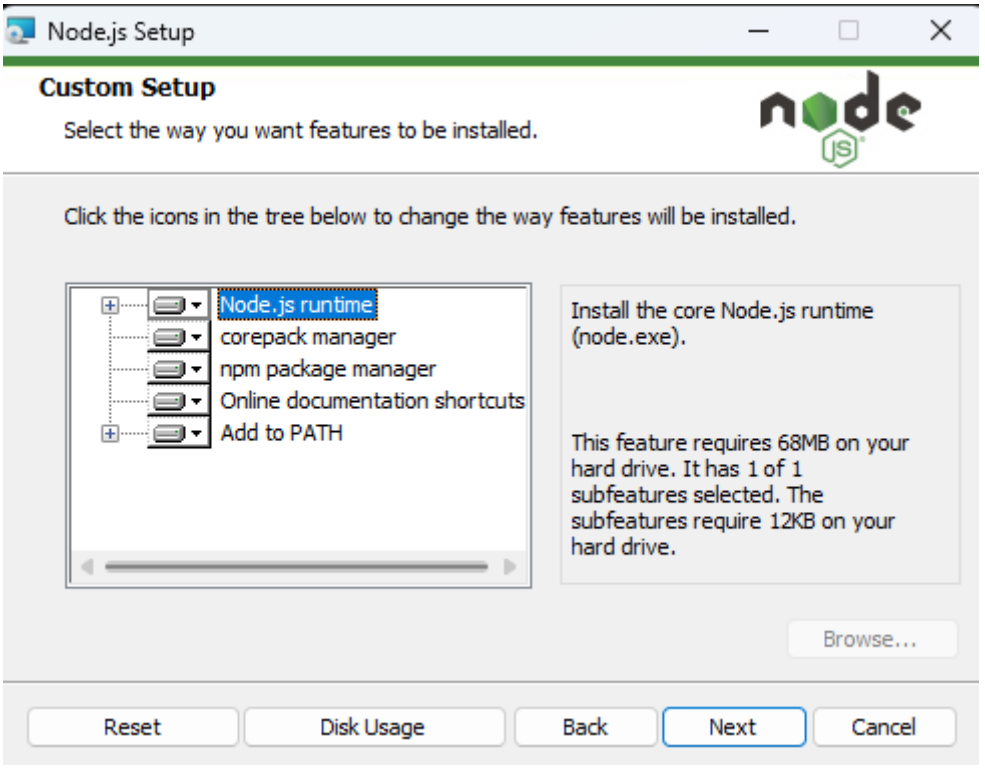


Figura 20 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Se desejar, você pode marcar essa opção para instalar ferramentas adicionais, incluindo o gerenciador de pacotes para Windows, o Chocolatey. No entanto, não é necessário.

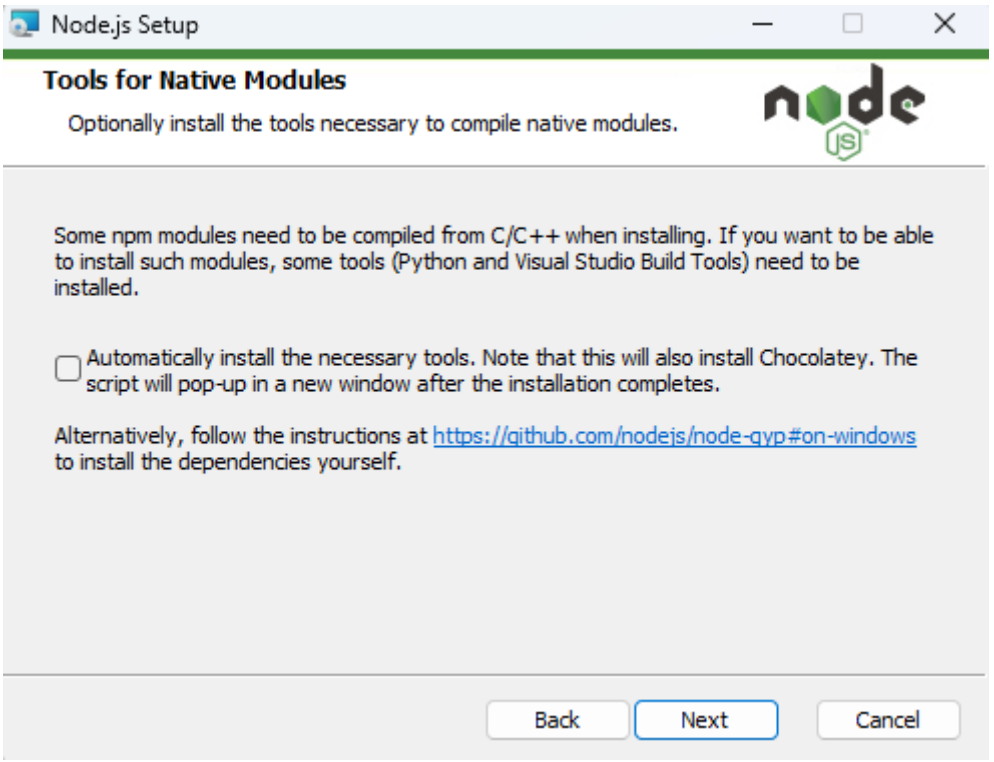


Figura 21 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Ao clicar em Install, a instalação irá continuar e logo em seguida será finalizada.

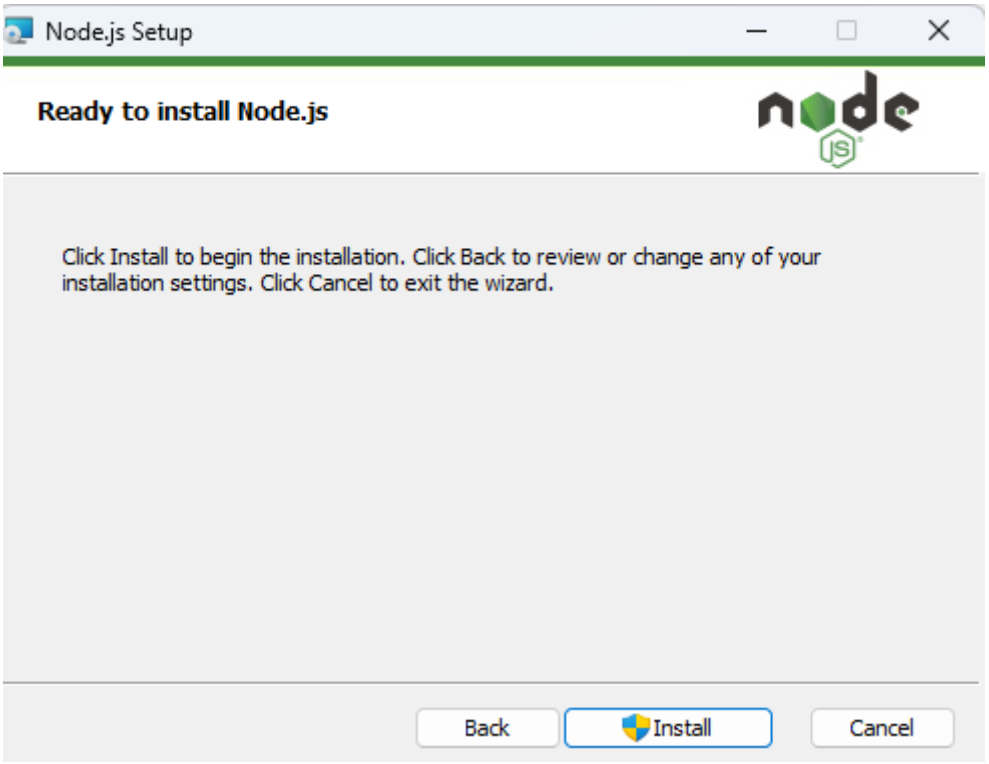


Figura 22 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Após concluir a instalação, abra um terminal e digite os seguintes comandos no seu terminal:

```
node -v
# Deve retornar algo como: v18.18.0

npm -v
# Deve retornar algo como: 9.8.1
```

Pronto, com isso o sistema estará preparado para o desenvolvimento de aplicações móveis. Siga para as próximas páginas para mais algumas dicas, como: IDE e Emulador.

3.1.3 Emulador

Dando início ao desenvolvimento, prossiga para a instalação do ambiente virtual android, o Android Studio, essa etapa servirá para sistema Windows.

O Android Studio é um ambiente de desenvolvimento integrado (IDE) utilizado para desenvolver aplicativos Android. Ele oferece uma variedade de ferramentas e recursos que facilitam a criação, teste e depuração de aplicativos Android (Android Developers, 2023).

Uma das principais funcionalidades do Android Studio é seu Emulador e Dispositivos Virtuais, onde ele integra emuladores que permitem testar aplicativos em diferentes versões do Android e em vários dispositivos virtuais.

O download pode ser feito através do link https://developer.android.com/studio, aceite os termos de uso e baixe o instalador.

Após baixar o instalador, o execute para iniciar o procedimento de instalação;

Na tela do instalador, a primeira coisa que irá ser questionado, é se desejar instalar o Android Virtual Device, que é propriamente o principal para o desenvolvimento, então, selecione 'Sim'.

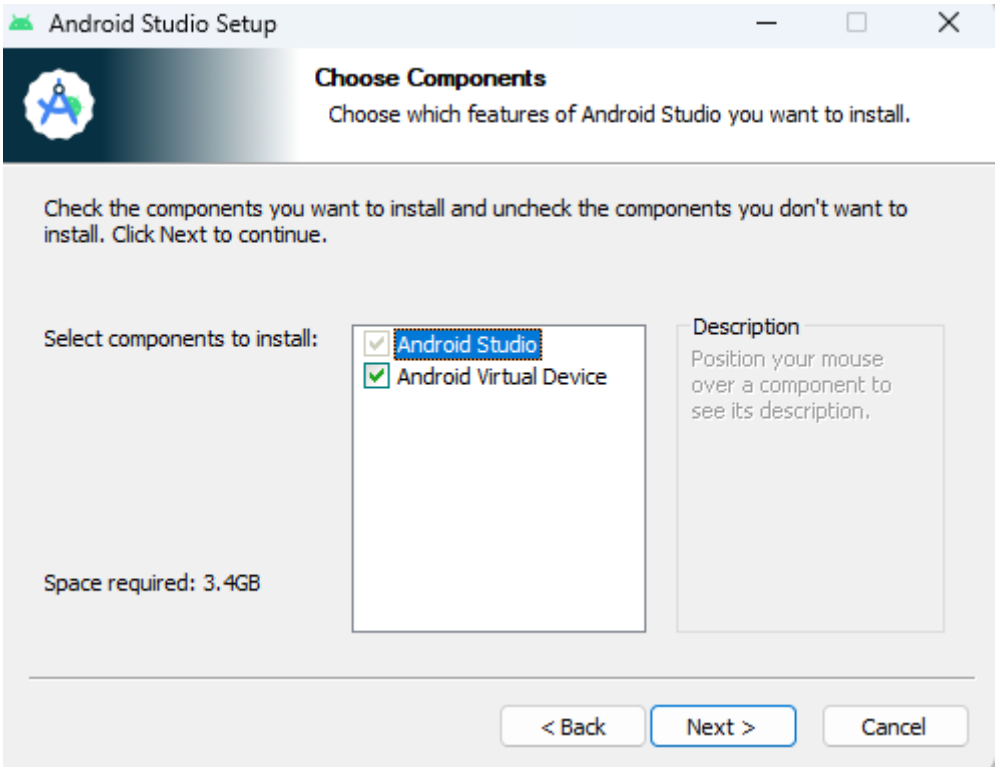


Figura 22 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Escolha o local de instalação, é recomendado que utilize o padrão.

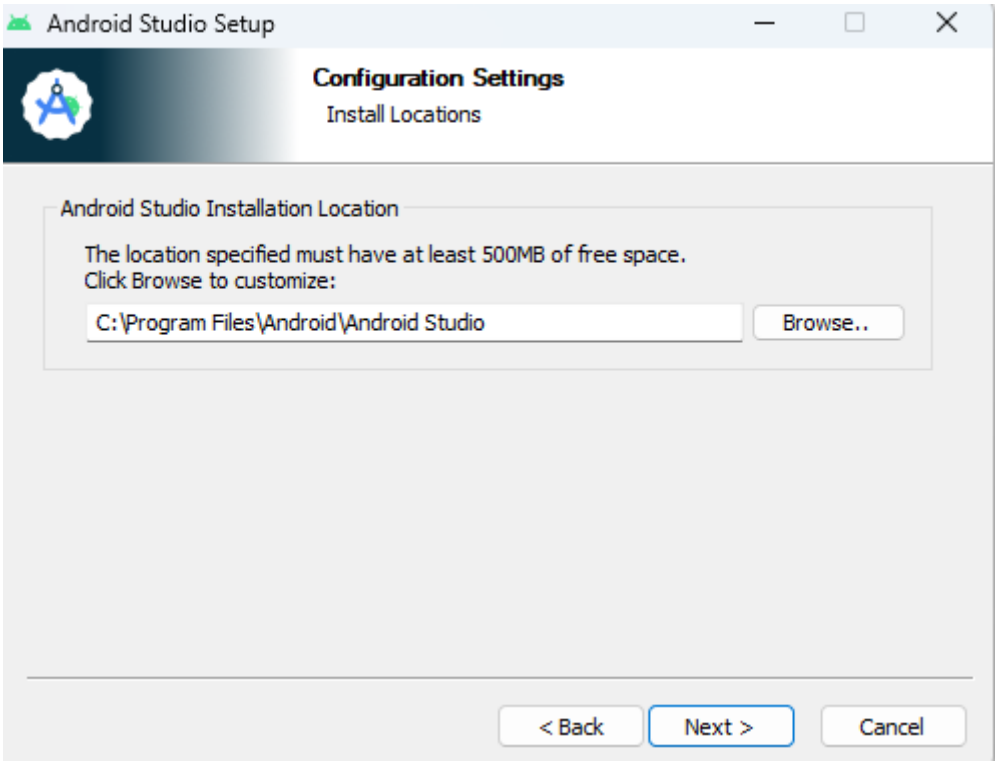


Figura 23 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Se não desejar que seja criado um atalho na área de trabalho, marque a caixa de seleção.

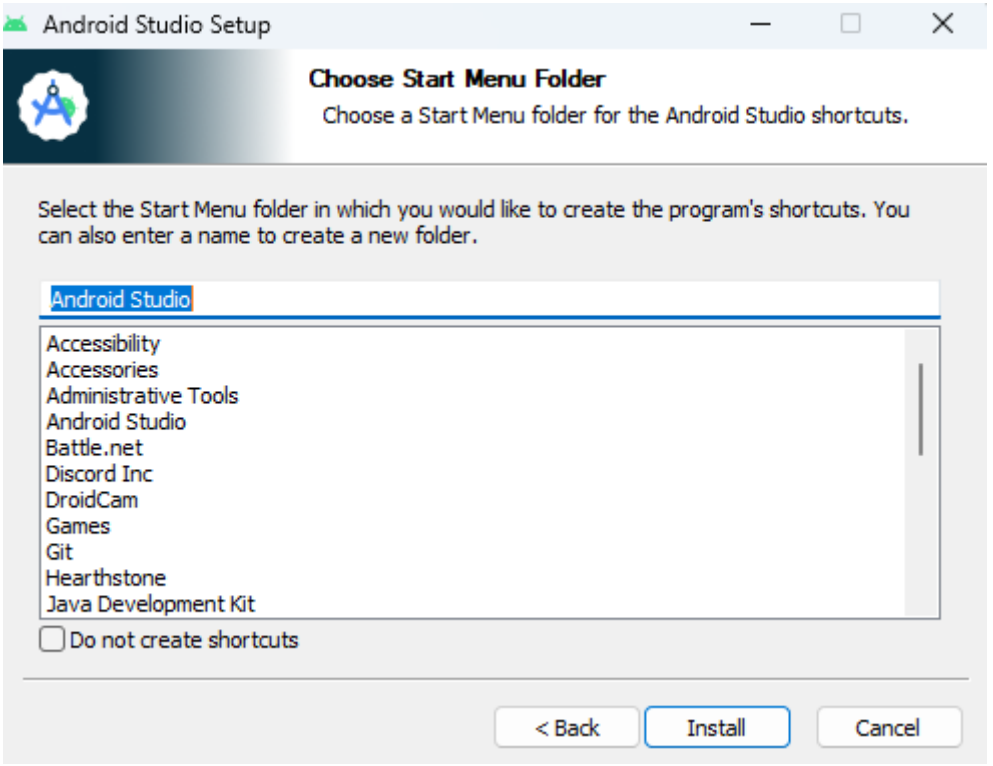


Figura 24 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Ao entrar pela primeira vez no Android Studio, será perguntado se deseja enviar estatísticas de desenvolvimento para o Google, você pode optar por sim ou não, isso não irá afetar o desenvolvimento.

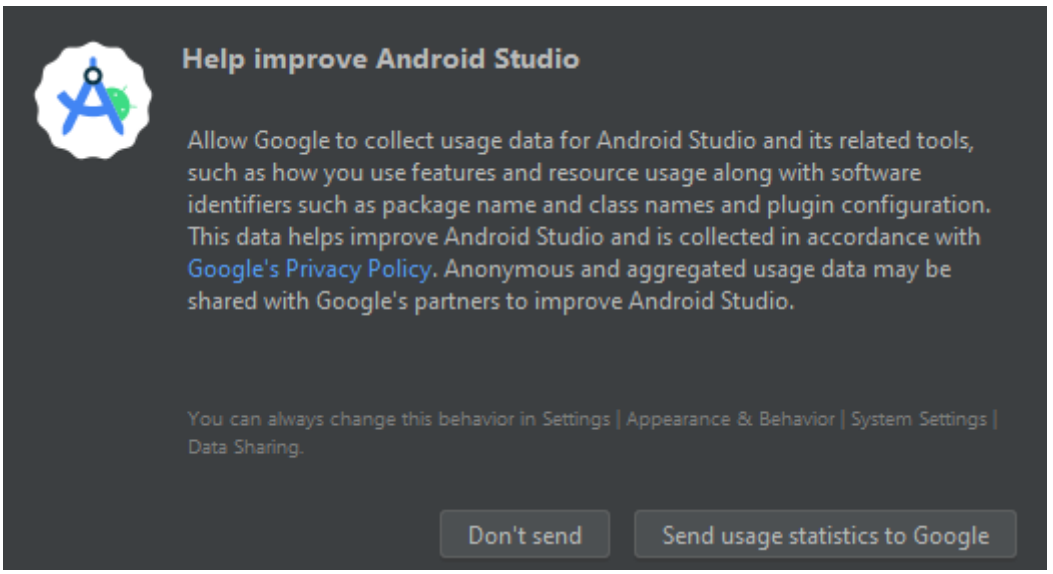


Figura 25 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Com o Android Studio aberto, clique em More Actions e depois em Virtual Device Manager

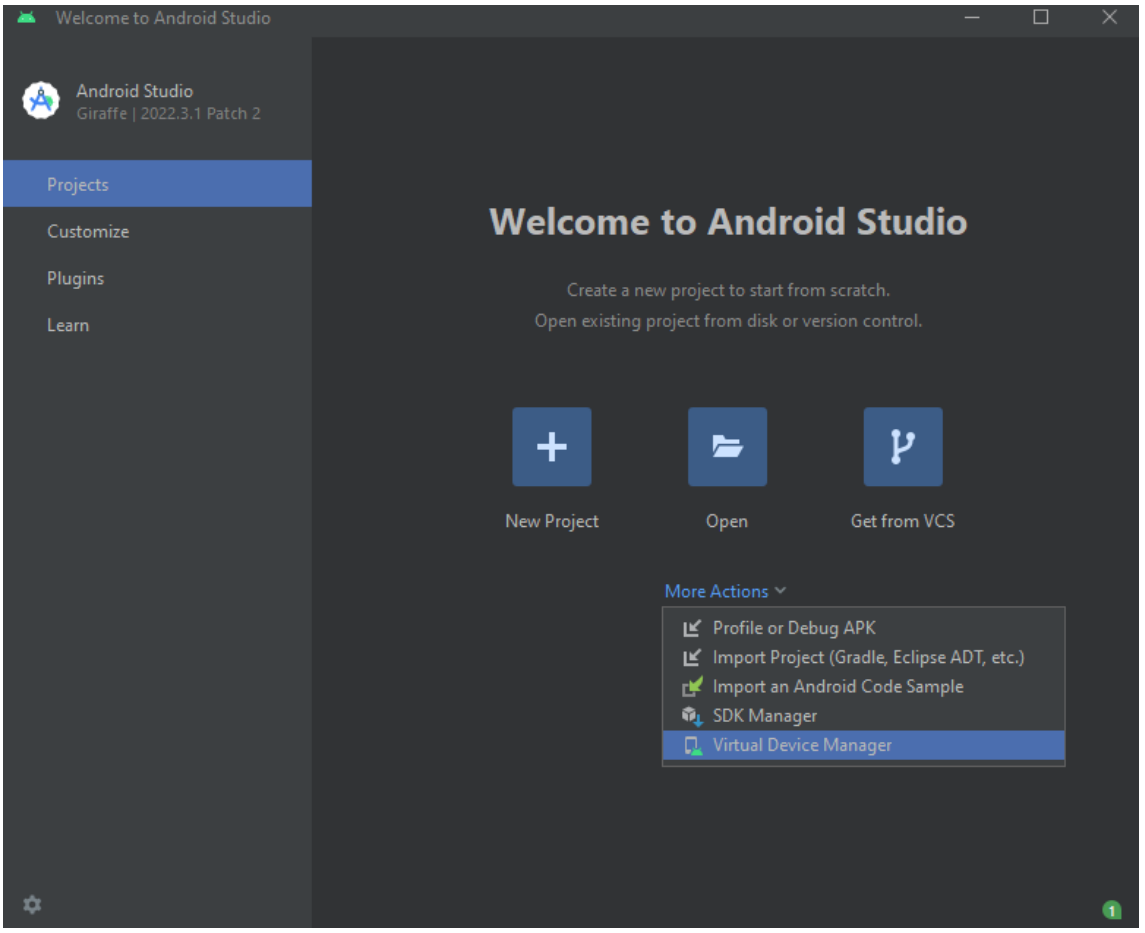


Figura 26 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Caso seja sua primeira vez no Android Studio, a tela seguinte terá uma lista vazia. De toda forma, clique em Create Device

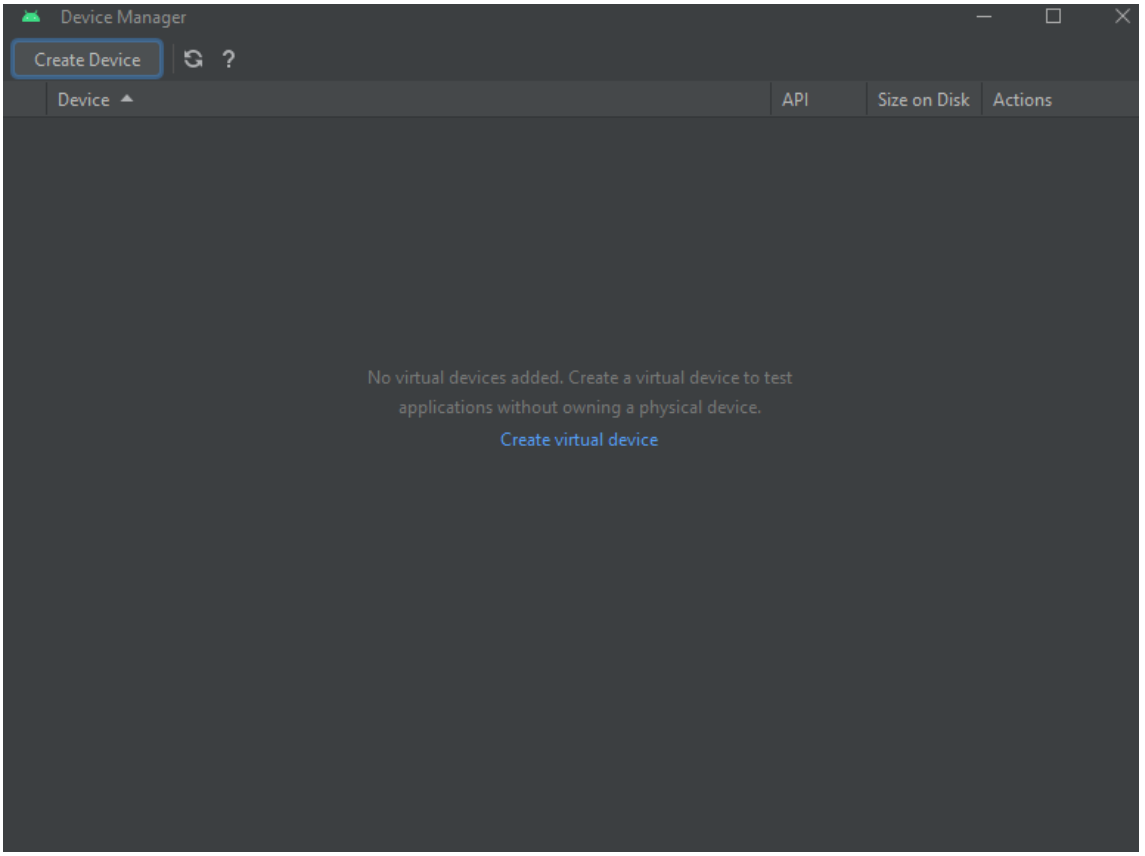


Figura 27 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Após isso, na janela seguinte, escolha um dispositivo de sua preferência, aqui, será utilizado o Pixel 7 Pro, e clique em Next

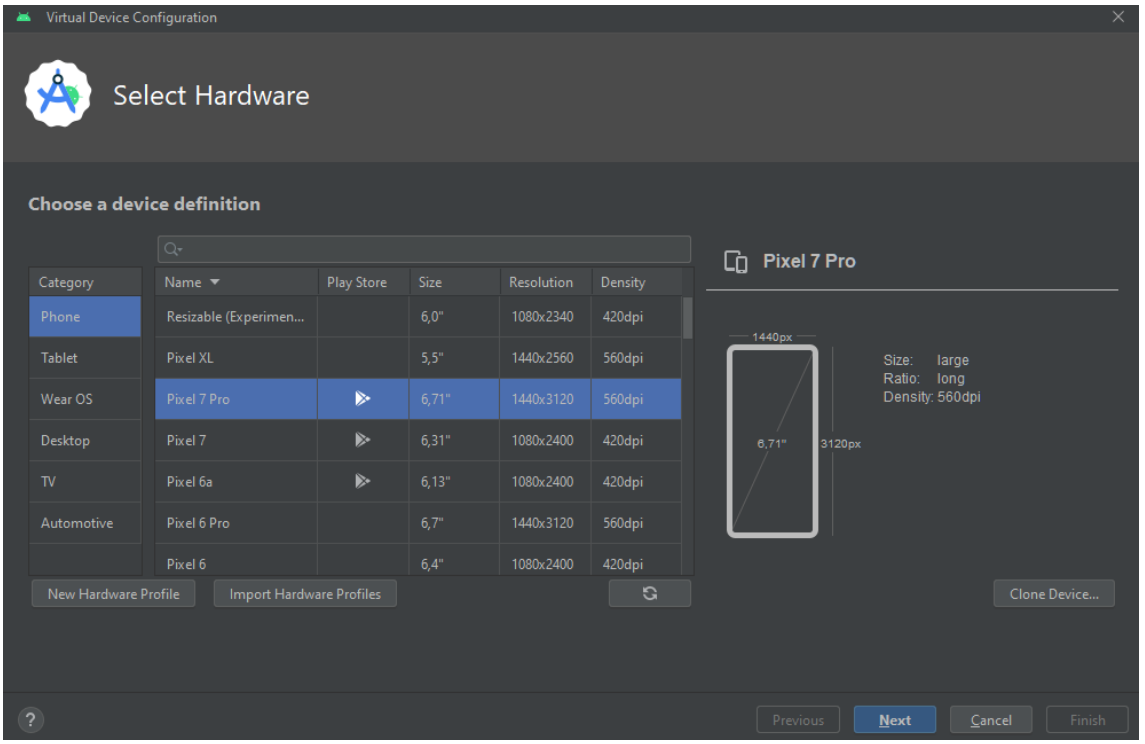


Figura 28 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Na janela seguinte, serão listados diversos sistemas operacionais recomendados, caso seja seu primeiro dispositivo, nenhum deles estará baixado, então, um deve ser escolhido e deve ser clicado no ícone de download. Aqui será utilizado o Android 13.0 Tiramisu para arquiteturas x86 e x64.

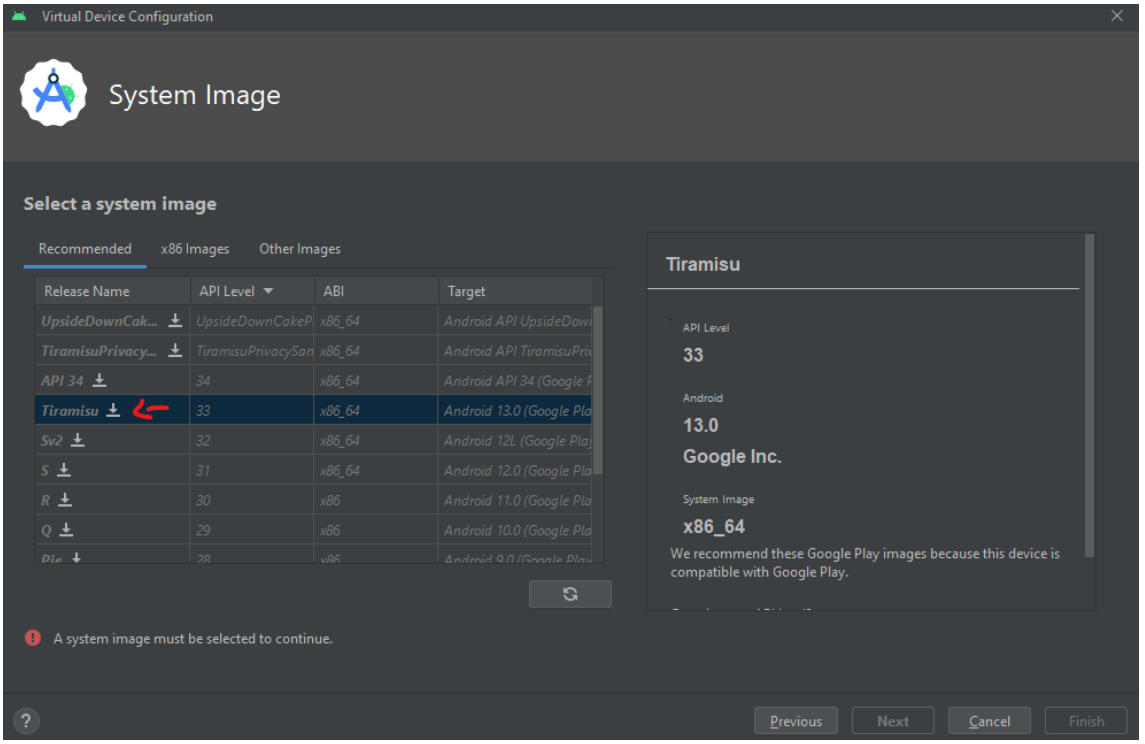


Figura 29 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Ao clicar no ícone de download do sistema desejado, ele já começará a ser baixado no pop-up que abrirá, quando a instalação finalizar, clique em Finish o pop-up fechará e então clique em Next

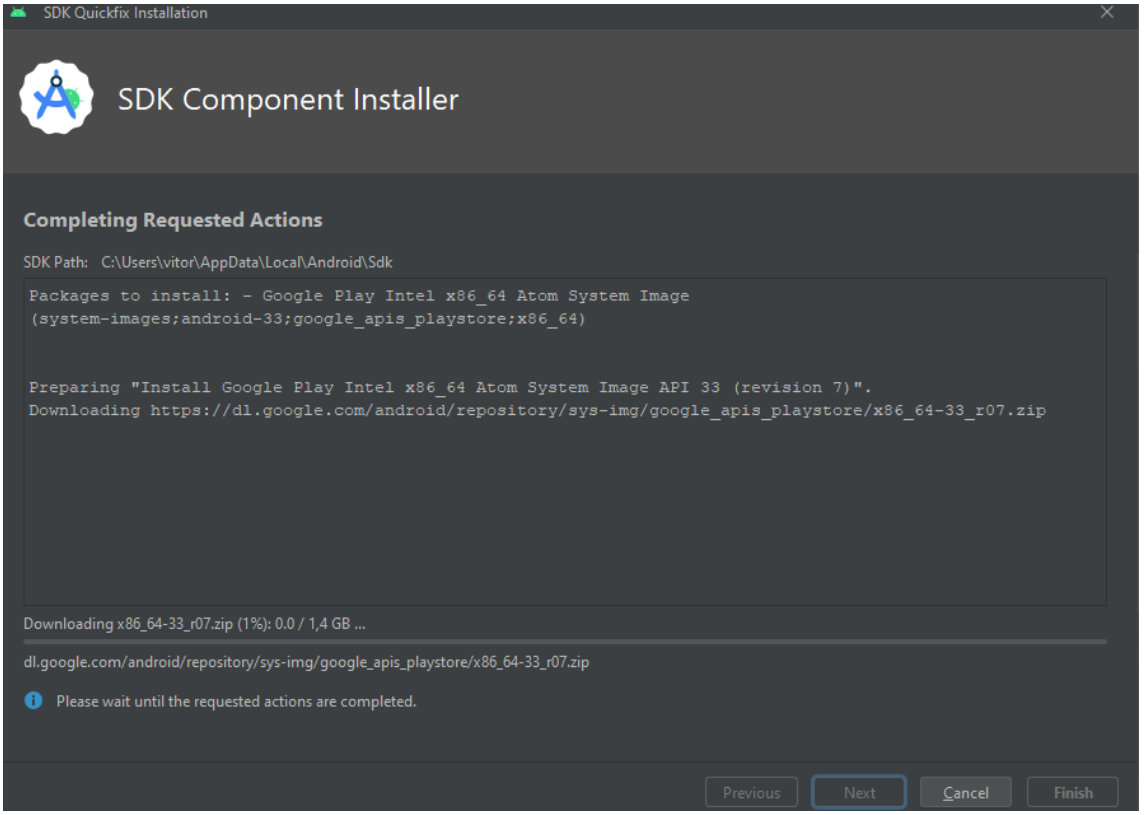


Figura 30 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Nessa tela, você consegue realizar mais alguns ajustes no seu dispositivo virtual, como quantidade de RAM que irá alocar, dispositivo de câmera que ele irá utilizar e outros. É recomendado que, inicialmente, utilize as opções padrão.

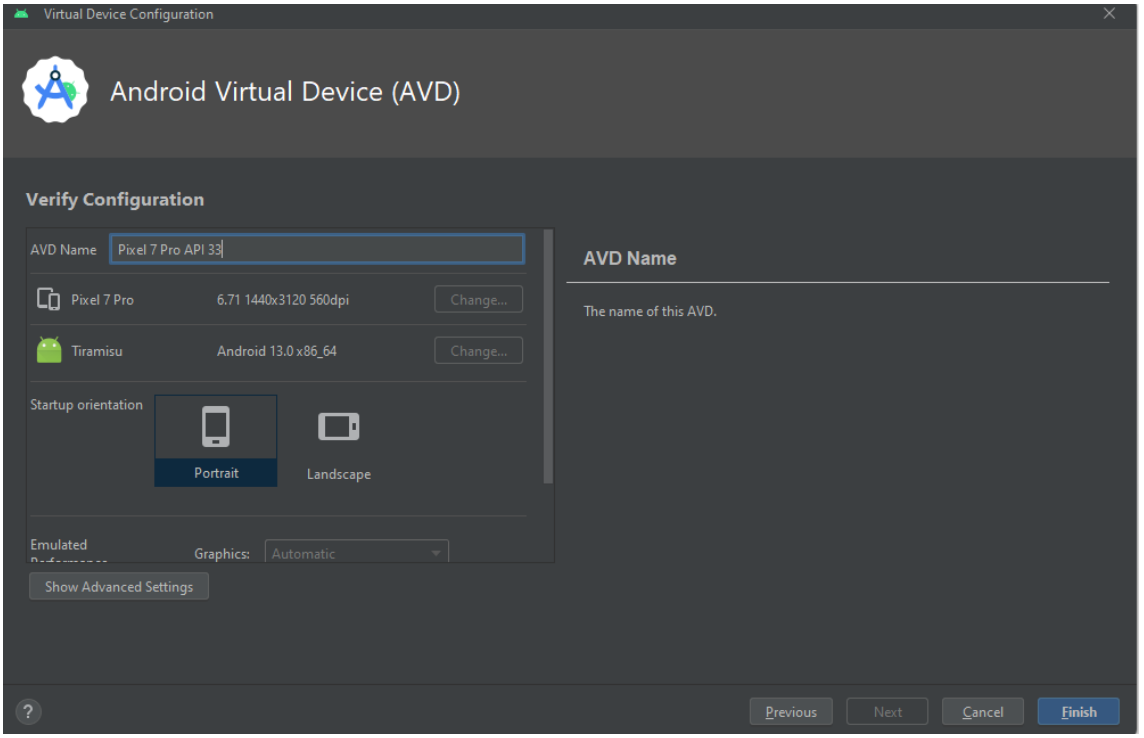


Figura 31 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Pronto, após isso, o dispositivo está pronto para uso, basta clicar no botão de Play e ele irá ser iniciado.

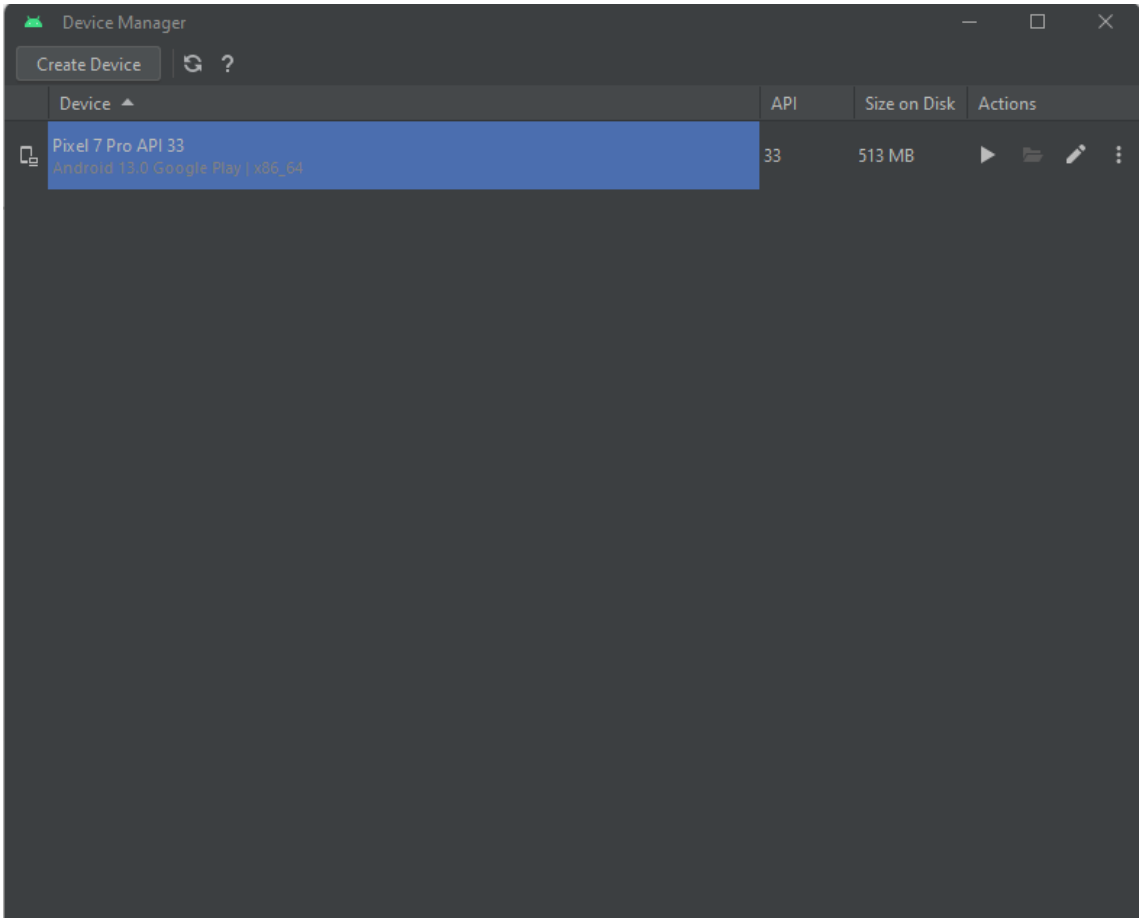


Figura 32 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Por hora, o emulador ainda não possui grande usabilidade, mas será utilizado nos próximos passos para rodar a primeira aplicação

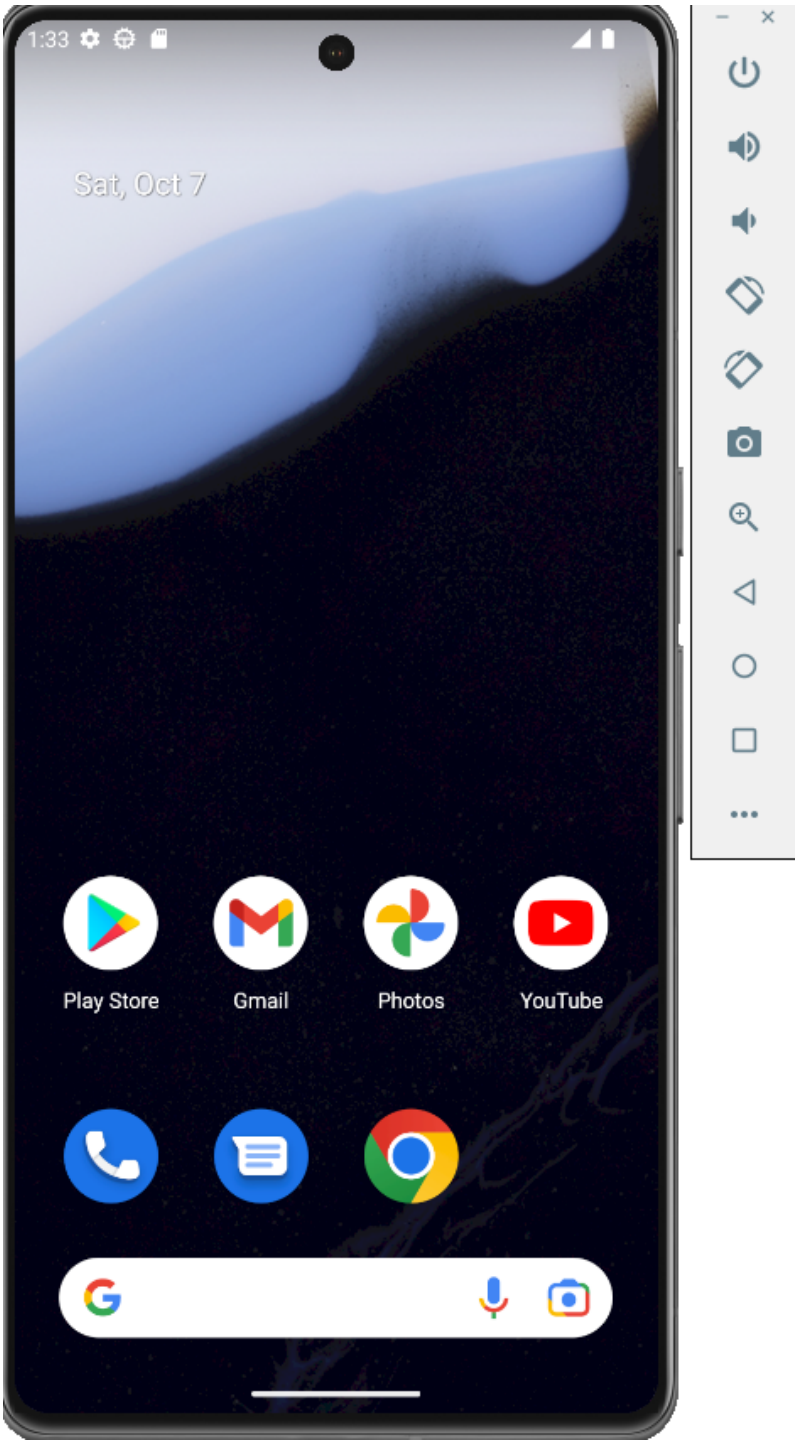


Figura 33 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

3.2 Linux

Para iniciar o desenvolvimento móvel no Linux, é necessário atender a alguns requisitos mínimos. Abaixo, explicaremos como instalar as dependências necessárias para executar o primeiro projeto em React Native utilizando o Expo.

É importante ressaltar que este guia pode variar de distribuição para distribuição. Durante a apostila, será utilizado Arch Linux e Ubuntu.

Fique ligado!
Atualmente, recomenda-se executar o Expo usando o comando **npx expo**, sem a necessidade de instalar um pacote adicional.

3.2.1 Git

Primeiramente, é essencial ter o Git instalado em sua máquina. Portanto, verifique se já está instalado, abrindo um terminal e digitando o comando abaixo e, caso contrário, instale-o.

```
# Caso o comando retorne alguma mensagem de erro é porque o git não está instalado
git --version
```

Caso não possua o git, ele pode ser instalado da seguinte maneira:

```
# Arch Linux
sudo pacman -S git

# Ubuntu
sudo apt install git

# Após a instalação, verifique-a:
git --version
```

3.2.2 Node

Para ter uma forma de instalação mais generalizada e com maior controle, é recomendado instalar o Node através do pacote NVM. É esse método que será utilizado.

```
#  Para começar, é preciso que tenha o cURL instalado, portanto para Arch Linux:
sudo pacman -S curl

# Ubuntu
sudo apt install curl

# Com o cURL instalado, é hora de baixar o script de instalação do NVM
curl https://raw.githubusercontent.com/creationix/nvm/master/install.sh | bash

# Após isso, basta atualizar o ambiente do bash com:
source ~/.bashrc

# Por fim, basta instalar o node na versão desejada, será utilizada a versão LTS (Long-Term Support)
nvm install --lts
```

Pronto! O ambiente está preparado para ser utilizado. Continue para as próximas páginas para obter mais informações sobre como executar o projeto, seja em um emulador ou no seu celular.

3.2.3 Emulador

Iniciando o desenvolvimento mobile, chega o momento de instalar o emulador do Android Studio para visualizar as aplicações no Linux.

Para realizar a instalação do emulador e do Android Studio no Linux, serão utilizados alguns comandos, eles podem divergir de acordo com sua distribuição, é válido ressaltar que no material será utilizado Arch Linux e Ubuntu.

Comece instalando os pacotes necessários:

```
# Arch Linux, para realizar a instalação no Arch é interessante que você possua algum gerenciador de pacotes AUR (pacotes da comunidade), no material será utilizado o YAY.
yay -S android-studio

# Ubuntu
sudo add-apt-repository ppa:maarten-fonville/android-studio
sudo apt-get update
sudo apt-get install android-studio
```

Ao iniciar o Android Studio pela primeira vez, será perguntado se deseja importar configurações anteriores, não é o caso, portanto, marque a opção “Do not import settings”.

Figura 34 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Depois, será perguntado se você deseja enviar dados de análise para o Google, para melhorar que eles possam evoluir o projeto, isso fica a seu critério, não irá afetar o uso geral.

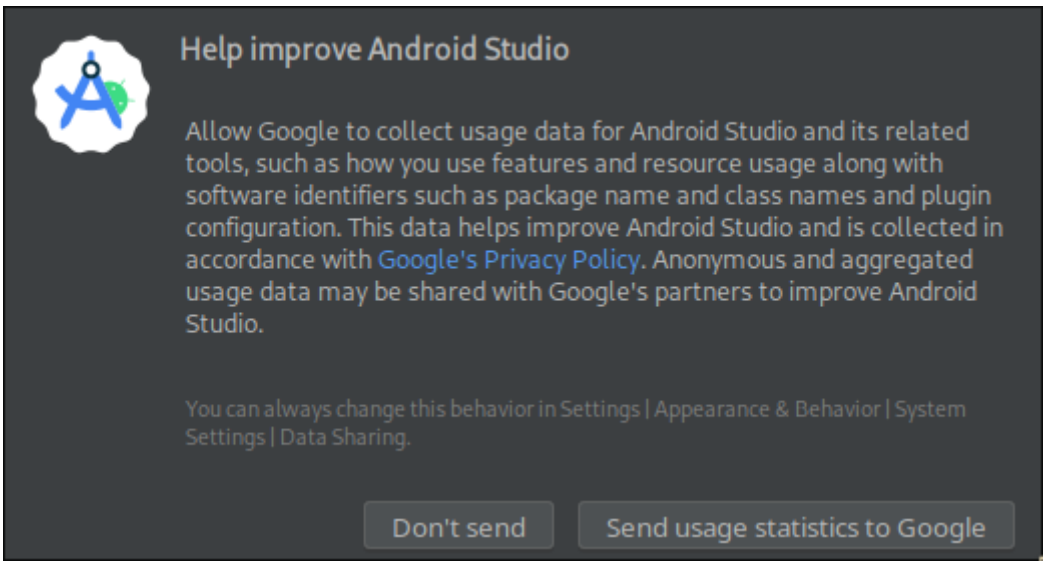


Figura 35 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Então, irá aparecer a tela de boas-vindas do instalador, basta seguir clicar em “Next”.

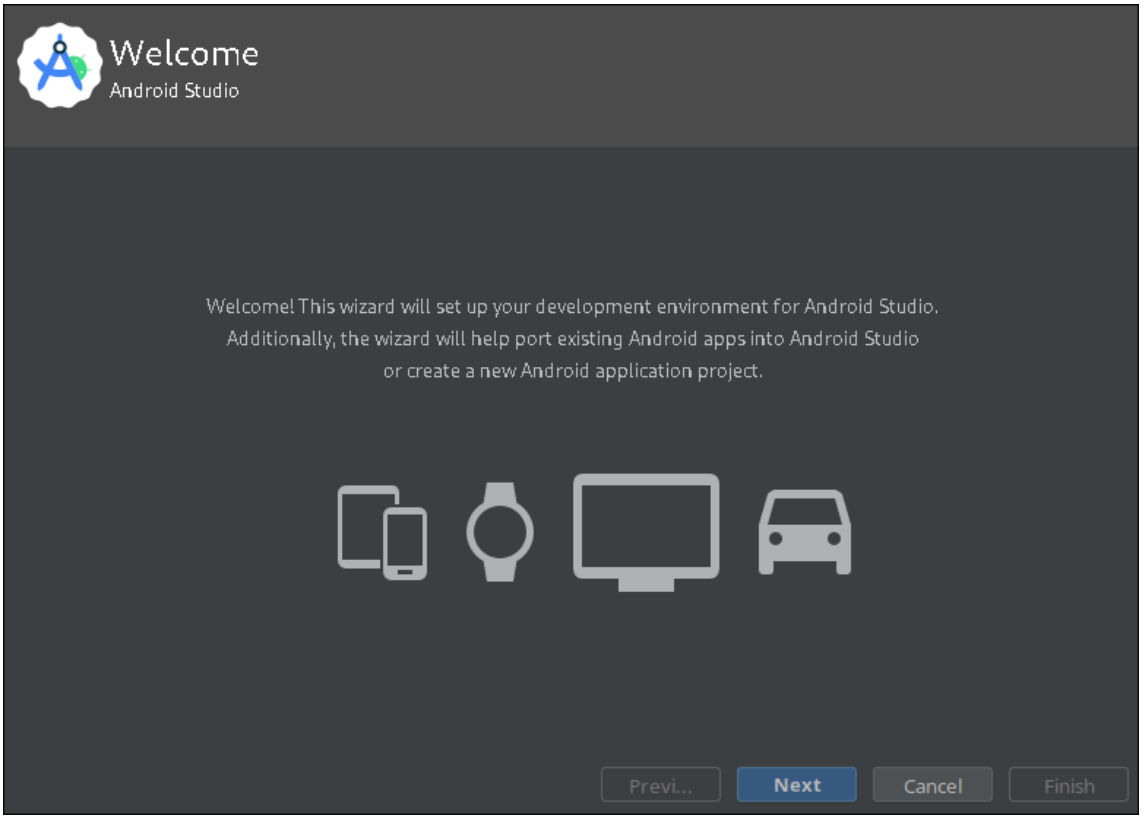


Figura 36 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

O instalador dá a opção de realizar a instalação de forma personalizada, escolhendo o que será baixado e configurações personalizadas, entretanto, utilize o “Standard”, ou seja, o padrão.

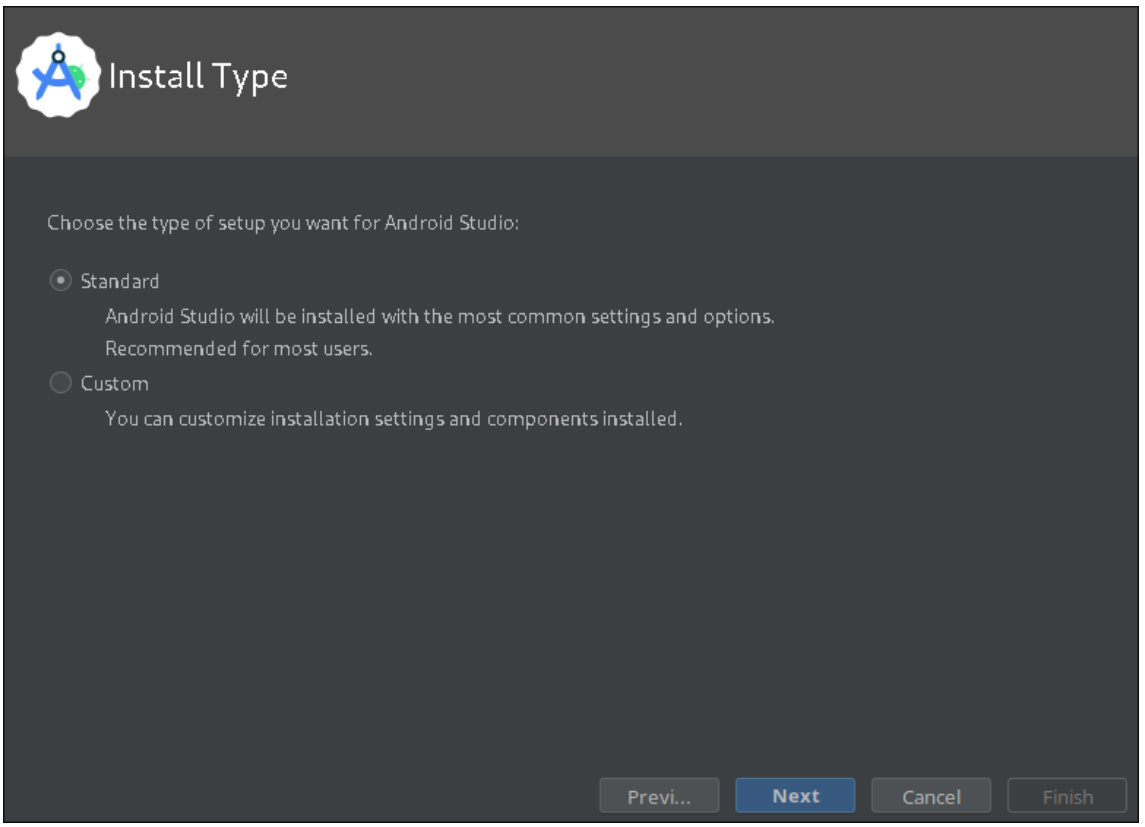


Figura 37 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Você pode escolher o tema que preferir abaixo.

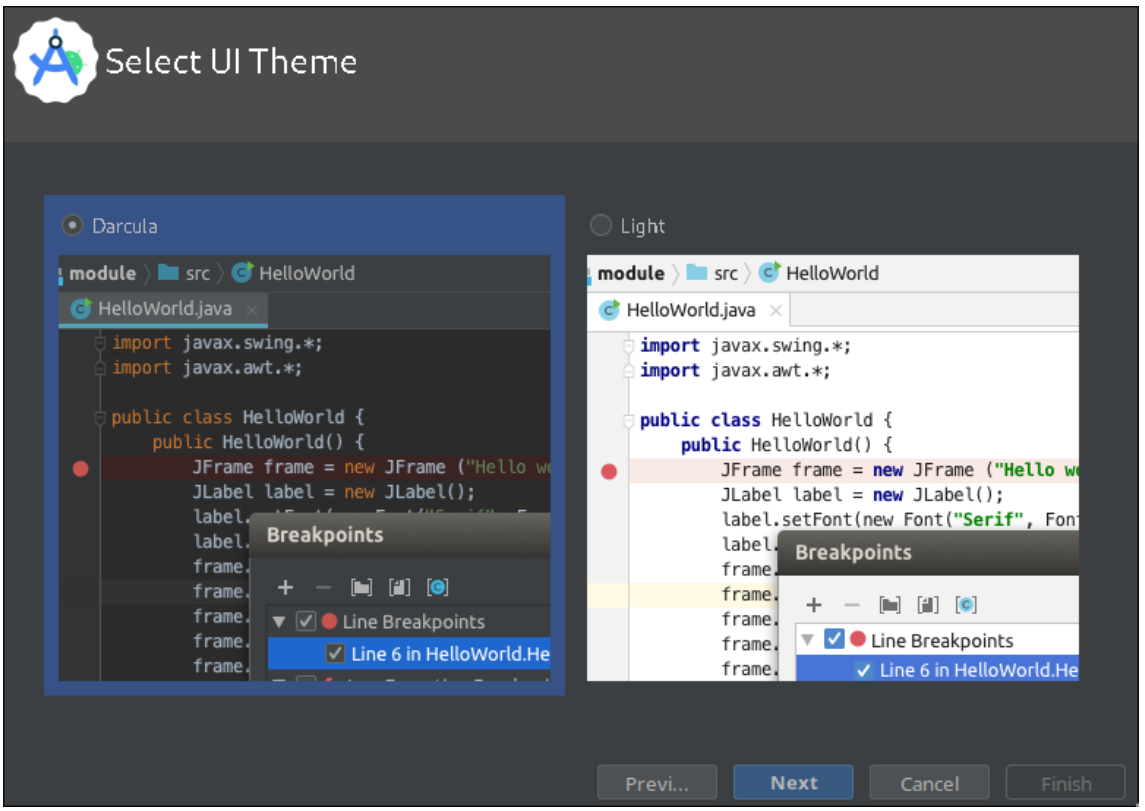


Figura 38 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Nessa tela ele irá mostrar um resumo de tudo que será instalado e configurado

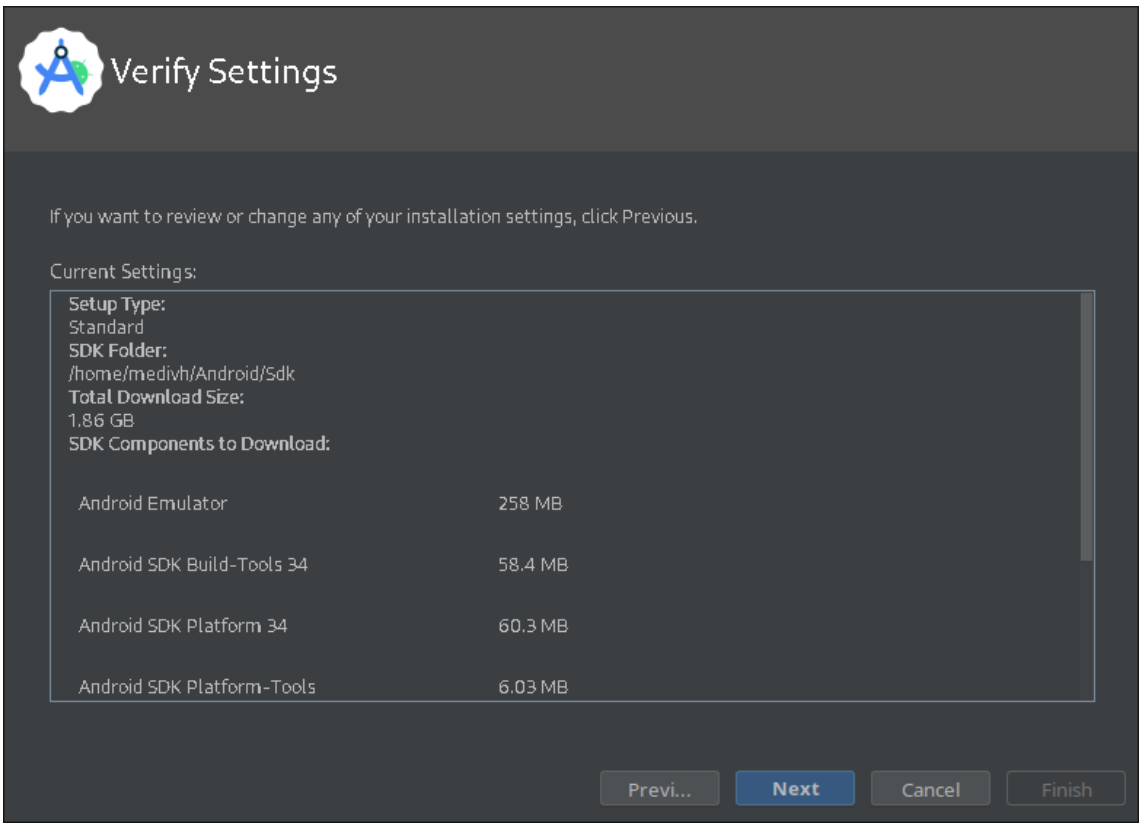


Figura 39 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Para prosseguir para a próxima tela, é necessário que você aceite os termos e regras de uso do Android SDK. Sendo assim, clique nos dois menus *dropdown* e depois em “Accept”.

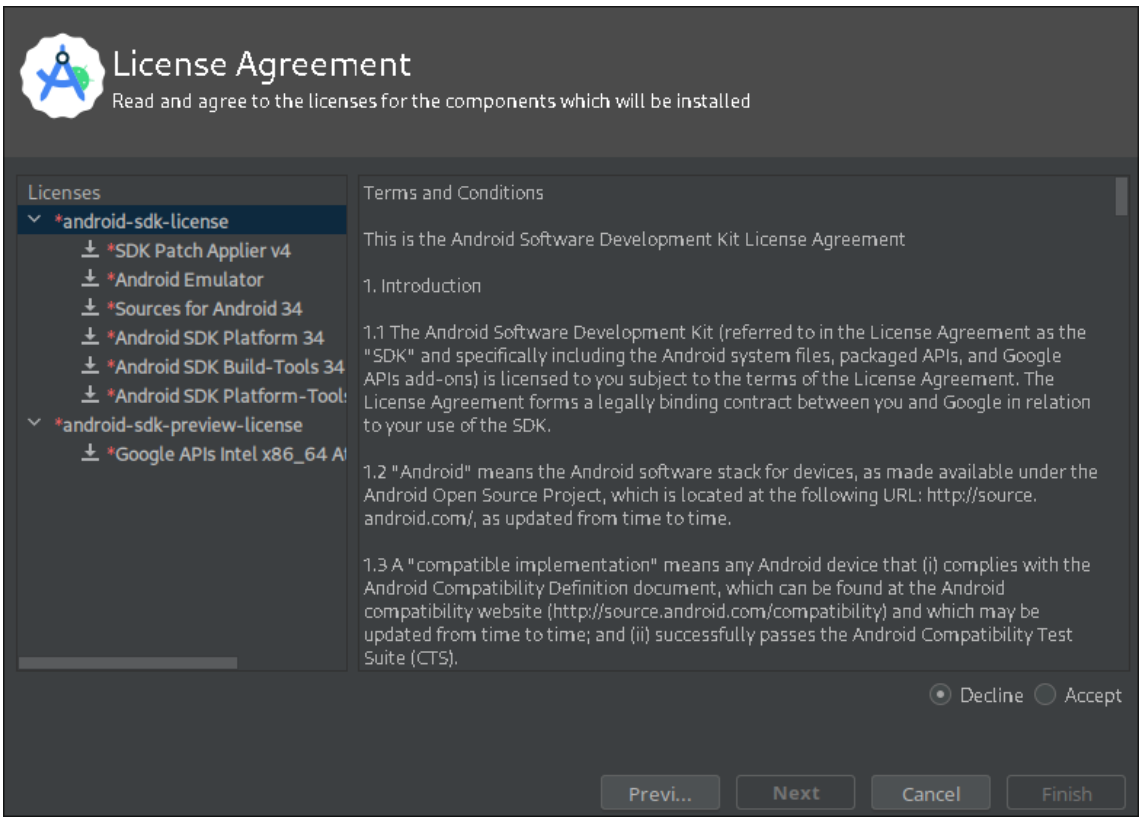


Figura 39 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Dependendo do seu sistema, essa tela pode aparecer e ela apenas informa que você pode rodar a aplicação no modo performance, deixando um link para ajudar a ativá-lo.

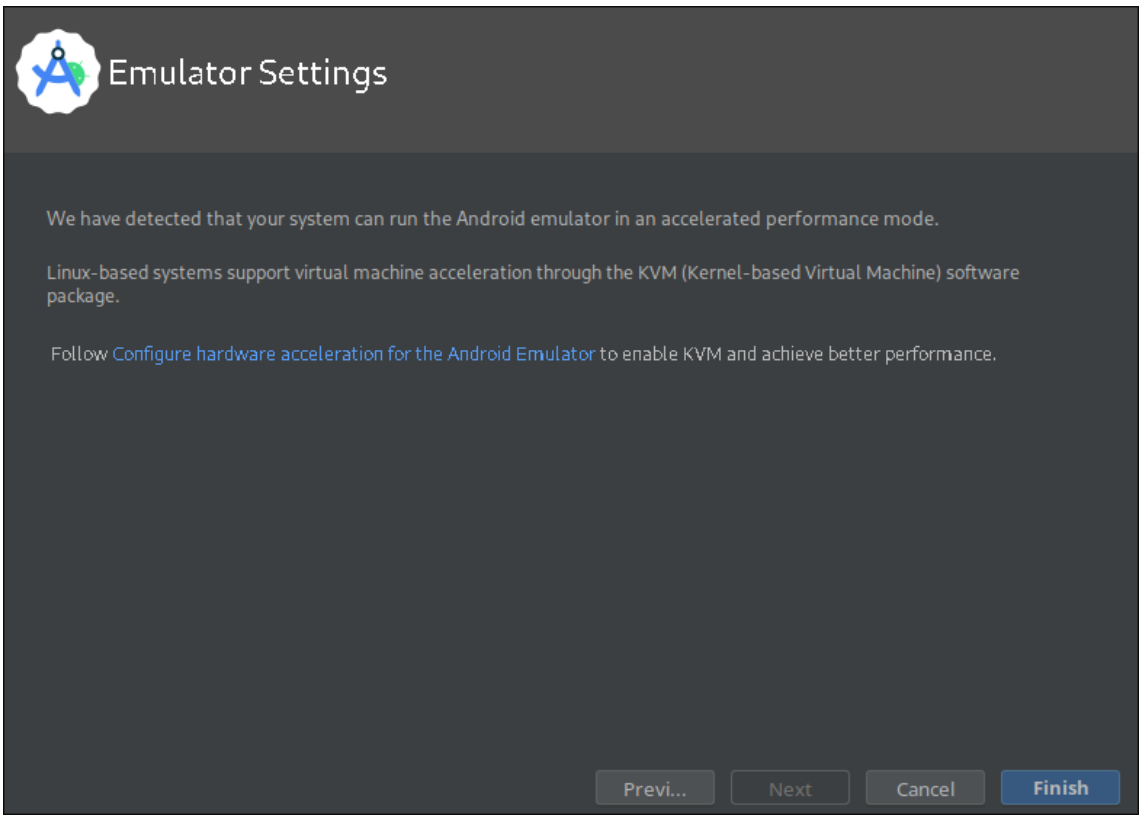


Figura 40 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Na tela inicial do Android Studio, clique em “More Actions” e selecione “Virtual Device Manager”

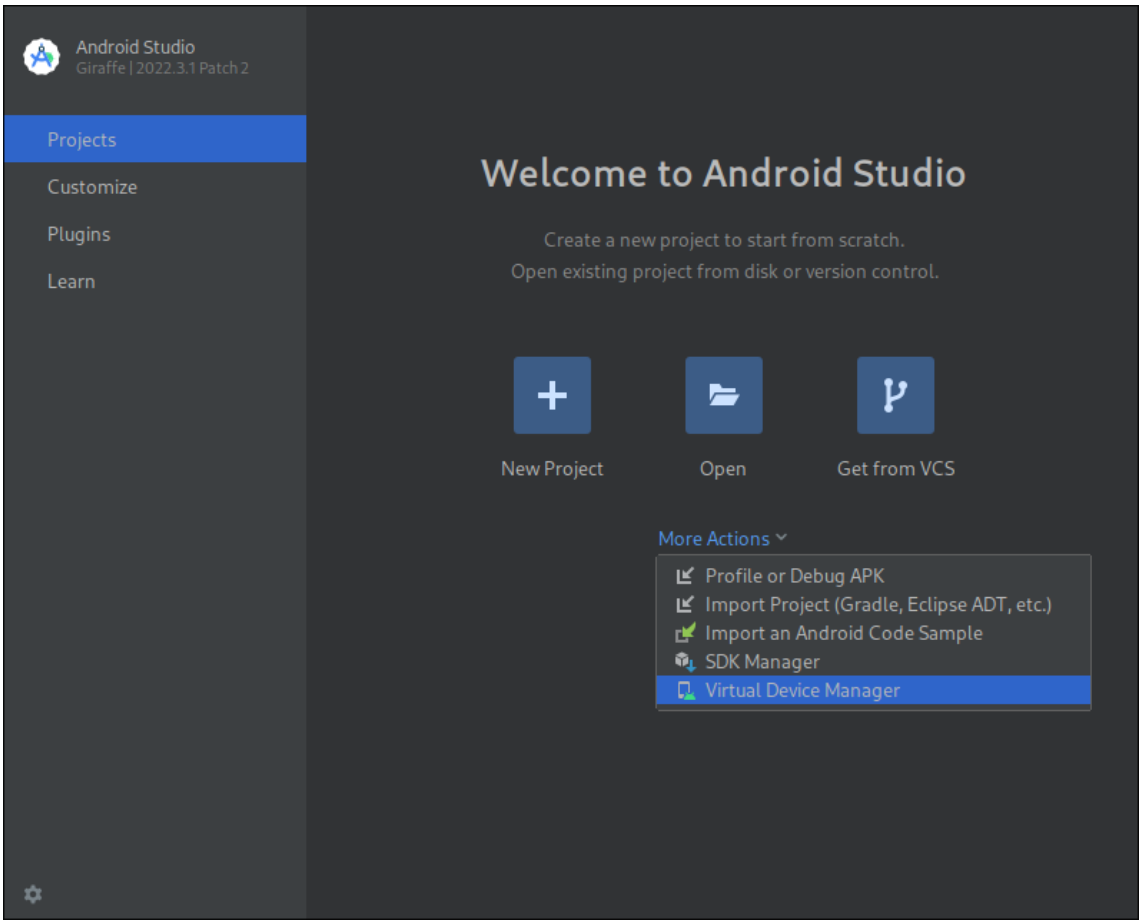


Figura 41 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Depois, clique em “Create device” e escolha um dispositivo, no material será utilizado o Pixel 7 Pro. Então, clique em “Next”.

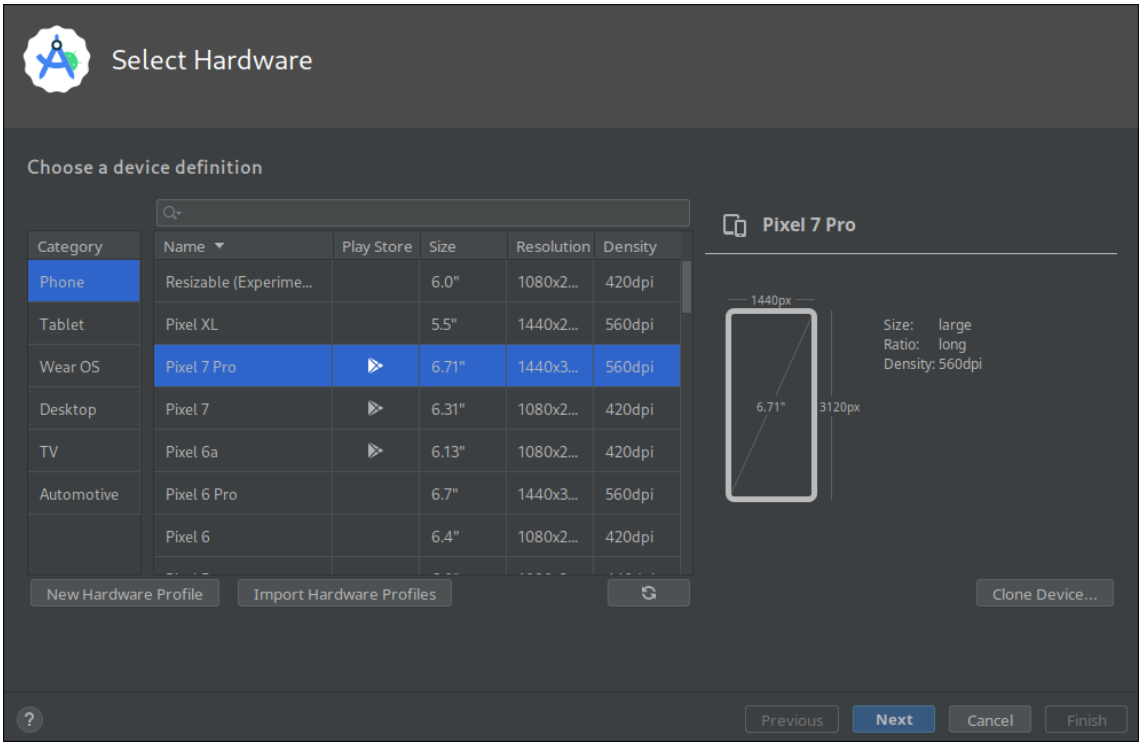


Figura 42 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Hora de escolher o sistema operacional. Na apostila será utilizado o API 34, basta clicar no ícone de download que ele já irá baixar automaticamente.

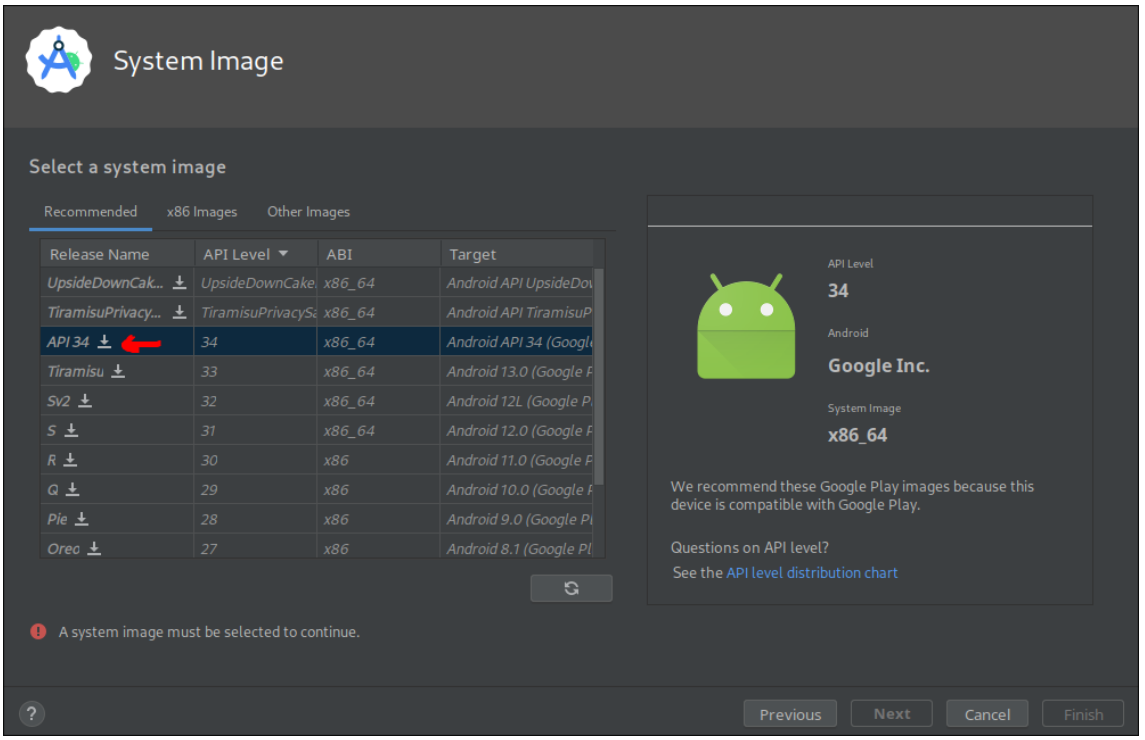


Figura 43 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Aqui você poderá realizar configurações extras do emulador, como quantidade de RAM dedicada, quais dispositivos a câmera irá utilizar e afins. Para o propósito atual, não é necessário realizá-las, basta clicar em “Finish”.

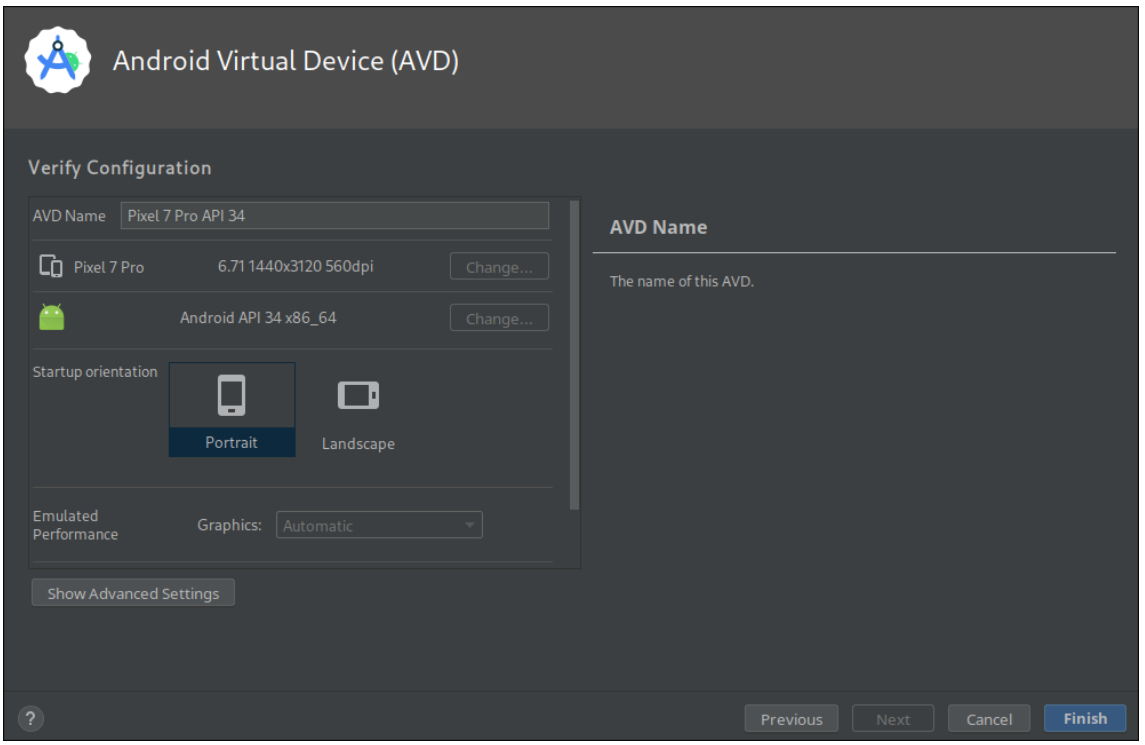


Figura 44 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Pronto, após isso, o dispositivo está pronto para uso, basta clicar no botão de Play e ele irá ser iniciado.

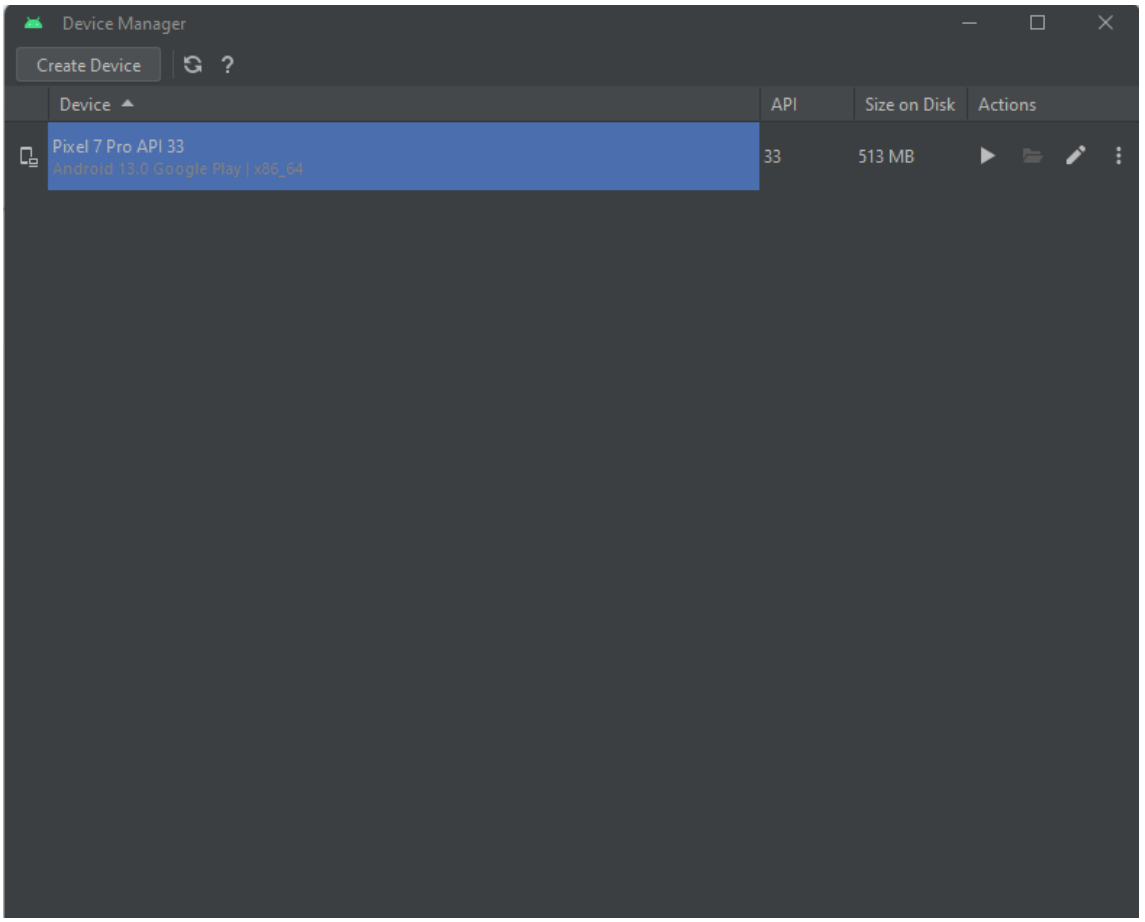


Figura 45 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

Por hora, o emulador ainda não possui grande usabilidade, mas será utilizado nos próximos passos para rodar a primeira aplicação

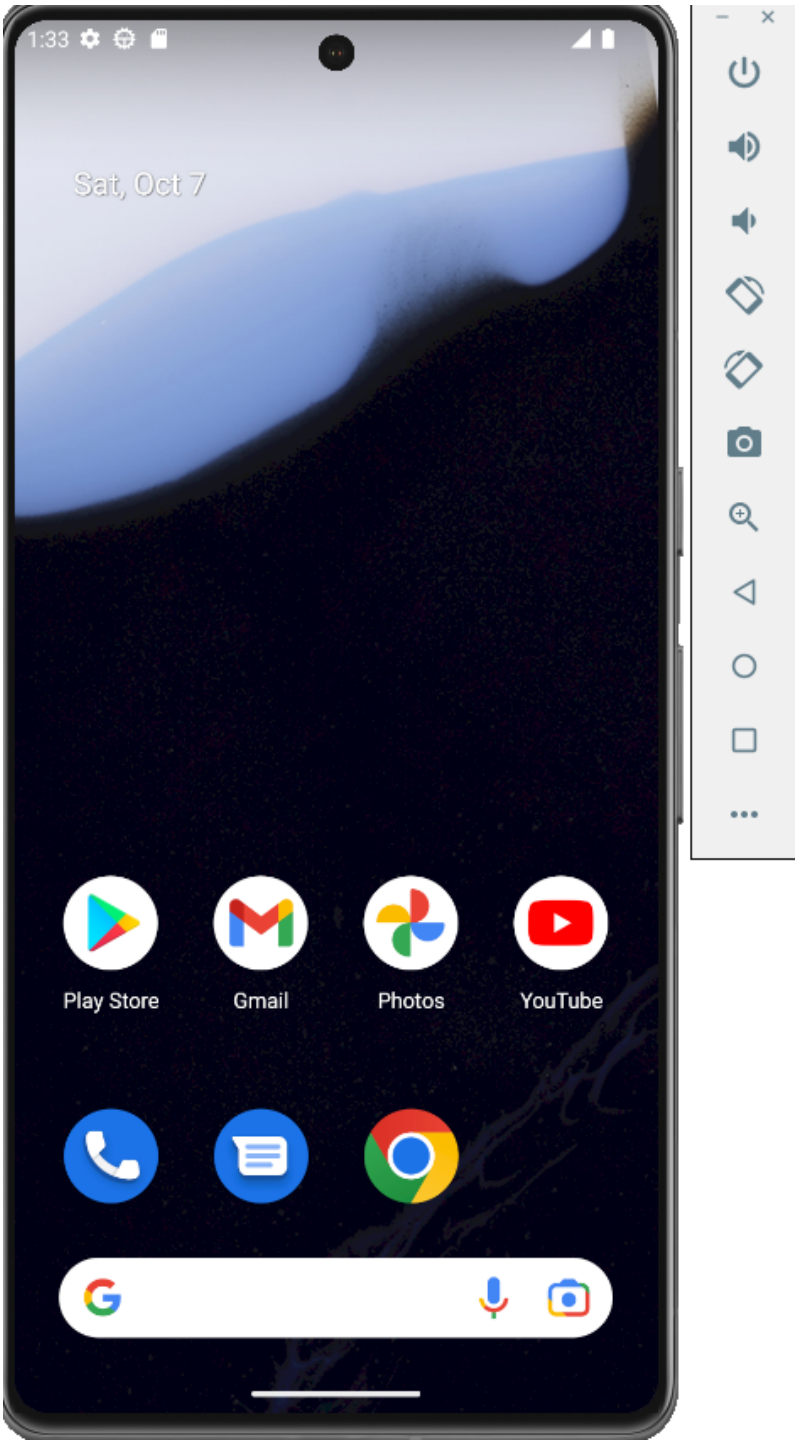


Figura 46 - Captura feita em 04/10/2023. Fonte: Reprodução do autor.

3.3 Build

Quando se trabalha com qualquer tipo de desenvolvimento, é comum existir diferentes ambientes. Quando a aplicação está nas mãos do desenvolvedor, apenas rodando ela localmente e sem executar o *build*, ela está em modo desenvolvimento. Ao utilizar o *build* a aplicação se torna uma simulação de como ela seria em modo produção, que é o que irá chegar para o usuário final. Portanto, o processo de *build* é muito importante para analisar a funcionalidade final da aplicação.

3.3.1 Como funciona?

Para realizar o procedimento utilizando Expo é bem simples. Basta executar um dos comandos abaixo.

```
# Este é o comando para buildar para Android
npx expo run:android

# Este é o comando para buildar para iOS
npx expo run:ios
```

4 Estilização

No contexto do React Native, o framework central deste curso, a estilização desempenha um papel crucial na criação de interfaces de usuário atraentes e funcionais. Existem três abordagens principais amplamente reconhecidas para estilizar componentes:

4.1 StyleSheet

Utilizando objetos que incorporam propriedades CSS in JS, a abordagem padrão do React Native permite a definição de estilos de forma programática. Isso oferece uma maneira eficiente e organizada de estruturar e aplicar estilos aos componentes, contribuindo para a consistência visual do aplicativo.

4.2 Styled Components

Bem parecida com a abordagem padrão, a biblioteca Styled Components é um tipo de estilização que utiliza propriedades CSS convencionais, tornando possível a inclusão de condições diretamente nos estilos. Isso proporciona uma maior flexibilidade na criação e aplicação de estilos.

4.3 Bibliotecas de componentes

Algumas bibliotecas oferecem a conveniência de estilizar componentes com base em propriedades passadas diretamente para eles, ou através de componentes feitos previamente. Embora isso simplifique a aplicação de estilos, é importante notar que essa abordagem pode limitar a personalização, tornando-a mais restrita em comparação com outras opções.

4.4 Densidade de Pixels

A densidade de pixels (PPI - pixels per inch, ou pixels por polegada) refere-se ao número total de pixels contidos dentro de uma determinada área física da tela. No âmbito do hardware, cada pixel representa um ponto de luz na tela. Por outro lado, no contexto do software, a densidade de pixels é dinâmica e ajusta-se de acordo com a quantidade de pixels na tela

A importância da densidade de pixels reside na capacidade de proporcionar detalhes visuais. Em dispositivos como smartphones, onde os usuários estão mais próximos da tela, uma maior densidade de pixels é necessária para garantir uma experiência nítida e detalhada. Porém, em dispositivos maiores, como TVs, a densidade de pixels pode ser menor sem comprometer a qualidade visual.

Ao desenvolver aplicações móveis, a unidade independente de densidade é crucial. No Android, utiliza-se o DPI (dots per inch), enquanto no iOS, emprega-se o conceito de pontos. Ao especificar o tamanho de um elemento como 24, estamos indicando que, independentemente da densidade de pixels, esse elemento manterá proporções consistentes em todos os dispositivos, proporcionando uma experiência visual uniforme. Essa abordagem facilita a criação de interfaces adaptáveis e consistentes em uma variedade de dispositivos móveis.

Saiba mais:
- “CSS in JS”, ou CSS em JavaScript, é um termo utilizado para denominar objetos de estilização que possuem sintaxe diferente do CSS convencional, nesse caso utilizando atributos JavaScript.

5 Hooks e Context

Nesta seção, serão abordados dois pontos importantes: hooks e context. O primeiro auxiliará na execução e uso de determinadas funcionalidades, enquanto o segundo permitirá o compartilhamento de dados entre os componentes da aplicação.

5.1 O que são hooks?

Segundo a documentação do React (2023), hooks permitem que você use diferentes funcionalidades do React em seus componentes. Você pode usar os Hooks integrados ou combiná-los para criar os seus próprios.

Isto é, hooks são poderosas ferramentas que já são integradas ao React, algumas delas são:

5.1.1 useState

Permite adicionar o ótimo controle de estados do React a componentes funcionais, proporcionando uma alternativa aos estados de componentes de classe.

5.1.2 useEffect

Utilizado para realizar efeitos colaterais em componentes funcionais, como requisições a APIs, manipulação de DOM ou subscrição a eventos.

5.1.3 useContext

Facilita o acesso a contextos no React Native, simplificando o compartilhamento de dados entre componentes sem a necessidade de propagação manual de *props*.

5.1.4 useCallback e useMemo

São usados para otimizar a performance, memorizando funções e valores calculados, respectivamente, para evitar cálculos ou renderizações desnecessárias.

5.2 O que é um Context?

Trata-se de um recurso que permite compartilhar dados, como estados ou configurações, entre componentes na árvore de componentes, sem a necessidade de passar esses dados manualmente por meio de props. É particularmente útil quando há muitos componentes aninhados que precisam acessar determinados valores, evitando a propagação excessiva de props. O Context cria uma espécie de "canal" global onde valores específicos podem ser fornecidos e consumidos por componentes que estão em diferentes níveis da hierarquia de componentes.

Fique ligado!
- *Props* no React Native são usadas para transmitir dados de pais para filhos, possibilitando a reutilização de código e facilitando a comunicação entre componentes.

6 Navegação

A navegação é um aspecto crucial no desenvolvimento de aplicativos móveis, e no React Native, ela se refere à transição entre diferentes telas ou componentes para criar uma experiência de usuário fluida. A navegação eficiente é vital para garantir a usabilidade e a lógica de fluxo em um aplicativo. No framework, há várias bibliotecas e abordagens para lidar com a navegação, sendo duas das mais comuns: React Navigation e React Native Navigation (Wix). Ambas as bibliotecas possuem documentação abrangente e ativa comunidade de desenvolvedores, tornando o processo de aprendizado e implementação mais acessível. A escolha entre elas dependerá das necessidades específicas do projeto e das preferências do desenvolvedor. Independente da escolha, compreender os princípios da navegação é essencial para criar aplicativos React Native coesos e intuitivos.

6.1 React Navigation

O React Navigation é uma biblioteca popular que oferece uma solução flexível e personalizável para a navegação em React Native. Ele suporta diferentes tipos de navegação, como navegação em pilha, guias e gavetas, adaptando-se a diversos cenários de aplicativos. A configuração básica envolve a criação de um navegador (Navigator) e a definição de rotas que mapeiam para diferentes componentes e telas.

6.2 React Native Navigation (Wix)

Esta é outra opção popular que oferece um desempenho notável ao delegar a navegação para as bibliotecas nativas do iOS e Android. O React Native Navigation, desenvolvido pela Wix, é especialmente indicado para aplicativos que buscam otimização de desempenho e experiência nativa.Esta é outra opção popular que oferece um desempenho notável ao delegar a navegação para as bibliotecas nativas do iOS e Android. O React Native Navigation, desenvolvido pela Wix, é especialmente indicado para aplicativos que buscam otimização de desempenho e experiência nativa.

7 Armazenamento local

O armazenamento local desempenha um papel crucial no desenvolvimento de aplicativos, proporcionando uma maneira eficiente de preservar dados no dispositivo do usuário. Ao invés de depender exclusivamente de servidores remotos ou armazenamento volátil, como é o caso do useState, que, ao reiniciar a aplicação, reinicia todos os estados, o armazenamento local oferece benefícios importantes para a experiência do usuário.

Quando se trata de armazenamento local em React Native, uma ferramenta amplamente empregada é o AsyncStorage. Essa biblioteca desempenha um papel crucial ao oferecer uma maneira assíncrona e eficiente de preservar dados no dispositivo do usuário.

7.1 Benefícios do Async Storage

Ao adotar o Async Storage em aplicativos React Native, os desenvolvedores colhem diversos benefícios essenciais para a eficiência, desempenho e experiência do usuário.

7.1.1 Persistência de Dados

Ao contrário do armazenamento volátil, como o useState, o Async Storage preserva dados entre sessões, garantindo consistência mesmo após a reinicialização do aplicativo.

7.1.2 Assincronia

Operando de forma assíncrona, o Async Storage não bloqueia a execução do código, permitindo a execução eficiente de tarefas sem impactar o desempenho do aplicativo.

7.1.3 Facilidade de Uso

Sua interface simples torna fácil para os desenvolvedores salvar e recuperar dados, utilizando pares chave-valor de maneira intuitiva.

7.1.4 Adaptação a Fluxos de Trabalho Assíncronos

Ideal para situações em que o armazenamento ou recuperação de dados precisa ocorrer de forma não bloqueante, como em operações de rede ou interações do usuário.

7.1.5 Armazenamento de Dados Sensíveis

Adequado para salvar dados sensíveis, como tokens de autenticação, de forma segura no dispositivo do usuário.

7.1.6 Suporte a Dados Complexos

Capaz de armazenar não apenas strings simples, mas também objetos JavaScript, fornecendo versatilidade para diferentes tipos de dados.

Conclusão

Esta apostila abrange uma extensa gama de tópicos, proporcionando uma visão abrangente do desenvolvimento de aplicações móveis usando React Native. Desde os conceitos iniciais de desenvolvimento móvel até tópicos como Hooks, Context, Navegação e Armazenamento local. Dessa forma, cada leitor(a) é guiado(a) através de uma jornada educativa que visa não apenas apresentar informações, mas também cultivar uma compreensão e prática.

A organização em seções distintas facilita a absorção progressiva de conhecimento, permitindo que os(as) leitores(as) se aprofundem gradualmente em conceitos mais complexos. A seção inicial fornece uma sólida introdução aos fundamentos do desenvolvimento de aplicações, enquanto cada seção subsequente expande e aprofunda áreas específicas do Desenvolvimento móvel.

A inclusão de tópicos como Estilização, Hooks, Context, Navegação e Armazenamento local reflete a difusão de conceitos básicos para o desenvolvimento de aplicativos, abrangendo desde a interface do usuário até aspectos importantes de gerenciamento de estado e persistência de dados.

Assim, o conteúdo do curso Introdução ao Desenvolvimento para Dispositivos Móveis familiariza o leitor com o mundo tecnológico, em um campo tecnológico que está sempre evoluindo, crescendo exponencialmente e apresentando constantemente novas soluções e desafios.

Obrigado por fazer parte do curso e ter realizado a leitura desta apostila. Espero que o interesse por Desenvolvimento Mobile apresentado neste curso esteja aguçado, para que você pratique e conheça ainda mais as maravilhas do Desenvolvimento para Dispositivos Móveis.

Referências

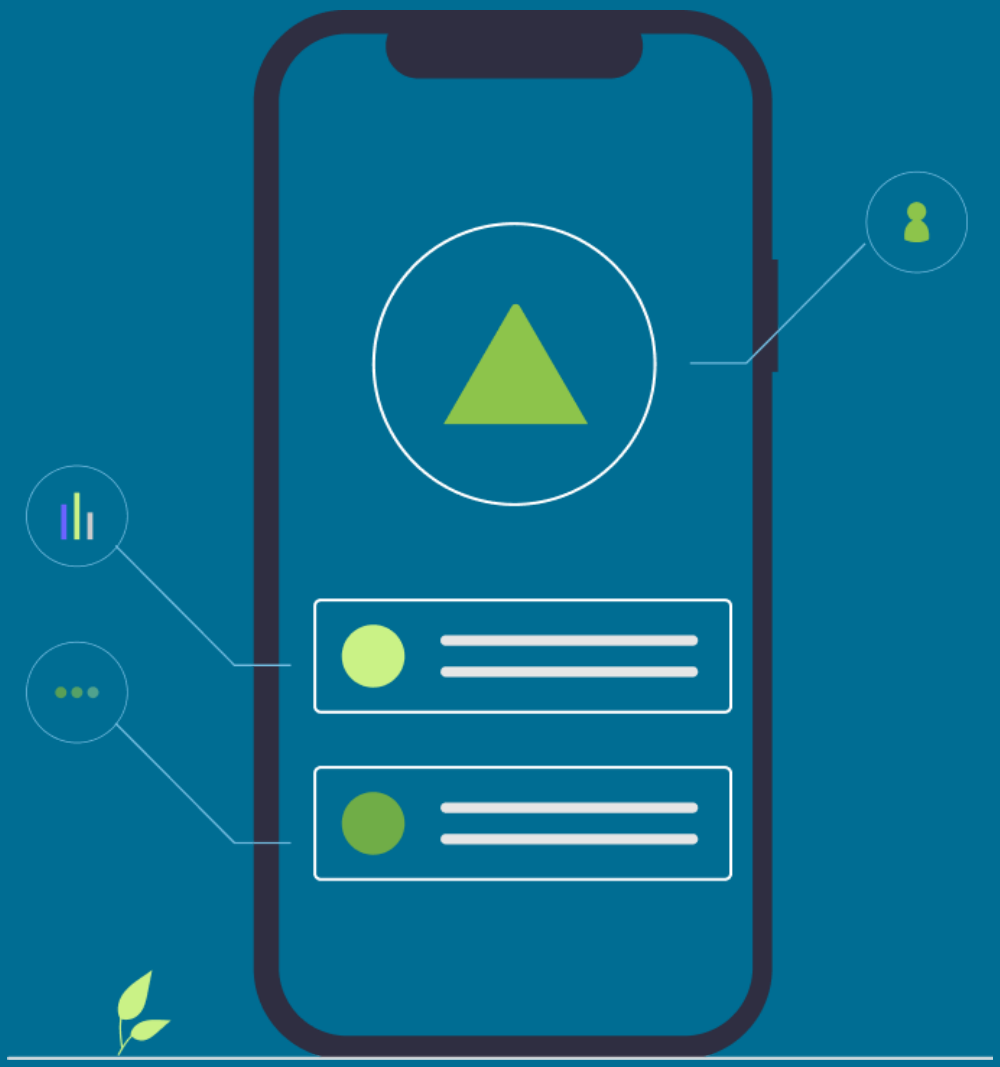
ANDROID DEVELOPERS. Android Studio. Disponível em: https://developer.android.com/studio. Acesso em dezembro de 2023.

EXPO. Documentação do Expo. Disponível em: https://docs.expo.dev. Acesso em dezembro de 2023.

GIT. Documentação do Git. Disponível em: https://git-scm.com/doc. Acesso em dezembro de 2023.

NODE.JS. Página de download do Node.js. Disponível em: https://nodejs.org/en/download/current. Acesso em dezembro de 2023.

REACT NATIVE. Documentação oficial. Disponível em: https://reactnative.dev. Acesso em dezembro de 2023.



BOM CURSO!

cepedi