

# Deploy

## O que é?

Realizar o **deploy** significa gerar o instalador do aplicativo

**Android**



**.apk ou .aab**

**iOS**



**.ipa**

**Deploy**

**Por que isso é útil?**

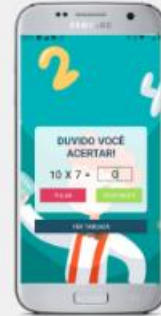
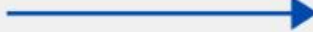
Para que nosso aplicativo fique disponível para download é necessário enviar os instaladores para as lojas **Google Play** e **App Store**.



Além de subir o instalador para a loja é possível instalar o aplicativo diretamente no celular (apenas no **Android**) através do arquivo **.apk**.



desafio-tabuada.apk



Celular android

# Deploy

## Como é feito?

## 1º Passo

Inserimos um ícone para o aplicativo



Ícone do aplicativo

## 2º Passo

Configuramos os dados do aplicativo.

```
1 {  
2   "expo": {  
3     "name": "Desafio Tabuada",  
4     "icon": "./assets/icone.png",  
5     "slug": "desafio-tabuada",  
6     "version": "1.0.0",  
7     "splash": {  
8       "image": "./assets/splash.png",  
9       "resizeMode": "contain",  
10      "backgroundColor": "#03989e"  
11    },  
12    "android": {  
13      "package": "com.devup.desafiotabuada",  
14      "versionCode": 1  
15    },  
16    "ios": {  
17      "bundleIdentifier": "com.devup.desafiotabuada",  
18      "buildNumber": "1.0.0"  
19    }  
20  }  
21 }
```

Arquivo app.json

É no passo 2 que vamos definir:

- O nome que vai aparecer no dispositivo do usuário.
- O nome do aplicativo no site do Expo.
- O local que salvamos o ícone do aplicativo.
- A versão do aplicativo.
- A imagem que será exibida enquanto o App estiver iniciando.
- As configurações para Android e iOS.

## 3º Passo

Executamos o comando para gerar o instalador.



```
expo build:android
```



```
expo build:ios
```

É no passo 3 que todo nosso código JS será resumido em um único arquivo de instalação



desafio-tabuada.ipa



desafio-tabuada.apk

ou



desafio-tabuada.aab

## 4º Passo

Subimos o instalador para a loja



desafio-tabuada.ipa



App Store (iOS)

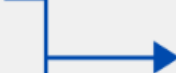


desafio-tabuada.apk

Ou



desafio-tabuada.aab



Google Play (Android)

Antes de gerar o instalador algumas configurações são necessárias para personalizar as informações sobre o aplicativo.

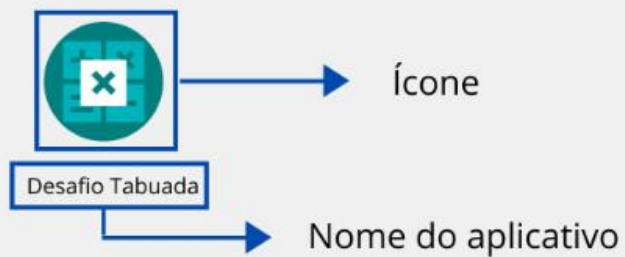
Quando falamos em informações do aplicativo estamos falando do arquivo **app.json**. Através dele codificamos as características exclusivas do nosso app.

Veja algumas das configurações que podemos fazer neste arquivo:

## O que personalizamos?



O ícone e o nome que aparecem para o usuário






## CODE BEHIND

Nome do aplicativo

```
1 {  
2   "expo": {  
3     "name": "Desafio Tabuada",  
4     "icon": "../assets/icone.png",  
5   },  
6 }
```

Local onde salvamos o ícone do aplicativo



O slug - que é o nome do projeto que será salvo no site do expo.




```
https://expo.io/dashboard/mvpmateus/desafio-tabuada
```

<https://expo.io/dashboard/> - **site do expo**  
mvpmateus - **nome do usuário**  
desafio-tabuada - **nome do projeto**

## Obs.:

O instalador do nosso aplicativo é gerado e ficará salvo no site do expo.

 <https://expo.io/dashboard/mvpmateus/desafio-tabuada>

O slug compõe o link que nos permite acessar o projeto e baixar o instalador.



## Controle de versão do aplicativo



Número da versão



## CODE BEHIND

```
1 {  
2   "expo": {  
3     "name": "Desafio Tabuada",  
4     "icon": "./assets/icone.png",  
5     "slug": "desafio-tabuada",  
6     "version": "1.0.0",  
  }
```



Versão do aplicativo

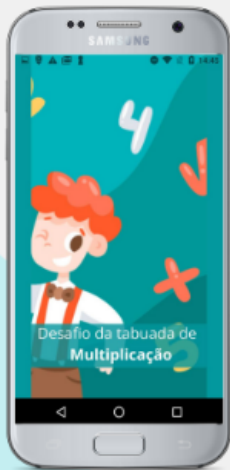


Uma imagem para exibir quando o usuário abrir o aplicativo



Imagem exibida

## CODE BEHIND



```
7 | "splash": {  
8 |   "image": "../assets/splash.png",  
9 |   "resizeMode": "contain",  
10 |  "backgroundColor": "#03989e"  
11 | },
```

- **image:** local da imagem que será exibida.
- **resizeMode:** Como a imagem vai se preencher na tela.
- **backgroundColor:** A cor que vai preencher a área que não for ocupada pela imagem

`resizeMode` possui os valores `cover` e `contain` (default). Essa propriedade é a mesma do componente `Image`

Configuração do Android

## Configuração do Android

Para gerarmos um aplicativo para Android precisamos definir duas configurações que são: o ID do aplicativo e o código da versão que podem ser vistos na **Figura 1**.

### Configuração do Android

```
12 |   "android": {  
13 |     "package": "com.devup.desafiotabuada",  
14 |     "versionCode": 1  
15 |   },
```

- **package**: ID do aplicativo na Google Play.
- **versionCode**: O código da versão do aplicativo Android.

Figura 1. Configuração necessária para gerar o aplicativo Android

## Configuração do iOS

Assim como no Android, não é diferente no iOS e por isso precisamos definir o ID e a versão do aplicativo, como mostra a **Figura 2**.

### Configuração do iOS

```
16 |   "ios": {  
17 |     "bundleIdentifier": "com.devup.desafiotabuada",  
18 |     "buildNumber": "1.0.0"  
19 |   }
```

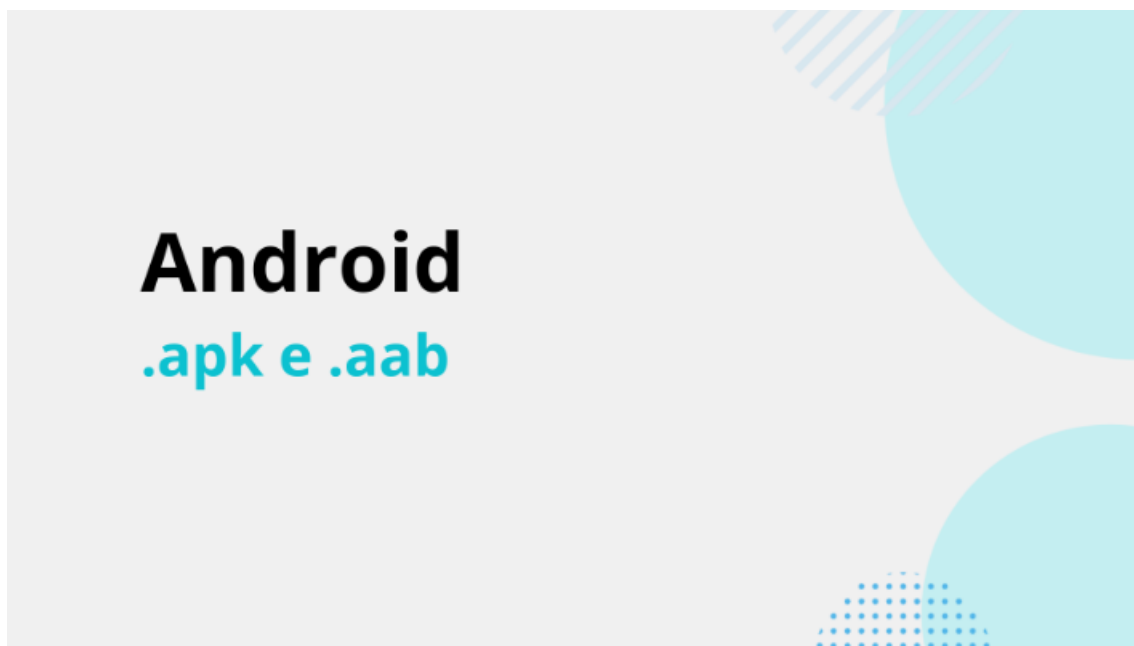
- **bundleIdentifier**: ID do aplicativo na App Store.
- **buildNumber**: O código da versão do aplicativo iOS.

O `package` (no caso do Android) e o `bundleIdentifier` (no caso do iOS) devem ser únicos e por isso não poderá ter outro aplicativo com este mesmo ID na loja

## Diferenças entre .apk e .aab do Android

Gerar um instalador é o processo necessário para que outros usuários instalem o aplicativo final. No Android podemos gerar dois arquivos: `.apk` e `.aab`.

Veja no slide abaixo a diferença entre eles:



Os dois arquivos podem ser enviados para a loja Google Play



O arquivo apk é o instalador que serve para qualquer dispositivo Android.





desafio-tabuada.apk

**Vantagem:** Não precisa enviar para a loja para que outros usuários possam instalá-lo. Basta abrir o arquivo diretamente no celular e o aplicativo será instalado.

**Desvantagem:** O tamanho do arquivo .apk e o aplicativo instalado é muito maior comparado ao .aab.

O .aab (Android App Bundle) não é o instalador mas um arquivo de publicação. Ele possui todo código necessário para gerar um .apk.



desafio-tabuada.aab



Google Play (Android)

Este arquivo só pode ser enviado para a loja - diferente do .apk que pode ser enviado diretamente para um dispositivo.



A partir do arquivo .aab a loja vai gerar um .apk contendo apenas as configurações necessárias para cada dispositivo.




desafio-tabuada.aab

**Vantagem:** O arquivo de instalação e o aplicativo instalado à partir do .aab será muito menor (cerca de 80%) comparado com o .apk.

**Desvantagem:** Por não ser de fato um instalador não é possível instalar em um dispositivo a não ser através da loja.

## Em resumo

 desafio-tabuada.apk



Pode ser instalado pela loja ou diretamente no celular, porém o tamanho do instalador é muito grande

 desafio-tabuada.aab



Só pode ser instalado pela loja porém o tamanho do instalador é muito menor e mais otimizado

# app.json

## Entendendo o código

```

{
  "expo": {
    "name": "Desafio Tabuada",
    "icon": "./assets/icone.png",
    "slug": "desafio-tabuada",
    "version": "1.0.0",
    "splash": {
      "image": "./assets/splash.png",
      "resizeMode": "contain",
      "backgroundColor": "#03989e"
    },
    "android": {
      "package": "com.devup.desafiotabuada",
      "versionCode": 1
    },
    "ios": {

```

Definimos o nome, o local do ícone, o nome amigável e a versão do aplicativo

```

1 {
2   "expo": {
3     "name": "Desafio Tabuada",
4     "icon": "./assets/icone.png",
5     "slug": "desafio-tabuada",
6     "version": "1.0.0",
7     "splash": {
8       "image": "./assets/splash.png",
9       "resizeMode": "contain",
10      "backgroundColor": "#03989e"
11    },
12    "android": {
13      "package": "com.devup.desafiotabuada",
14      "versionCode": 1
15    },
16    "ios": {

```

Definimos a imagem que vai aparecer quando o aplicativo for aberto.

```
1 {
2   "expo": {
3     "name": "Desafio Tabuada",
4     "icon": "./assets/icon.png",
5     "slug": "desafio-tabuada",
6     "version": "1.0.0",
7     "splash": {
8       "image": "./assets/splash.png",
9       "resizeMode": "contain",
10      "backgroundColor": "#03989e"
11    },
12    "android": {
13      "versionCode": 1
14    },
15  },

```

A imagem se ajustará uniformemente baseado nas suas dimensões

Definimos a cor que vai preencher o espaço que a imagem não ocupar

```
    "splash": {
      "image": "./assets/splash.png",
      "resizeMode": "contain",
      "backgroundColor": "#03989e"
    },
2    "android": {
3      "package": "com.devup.desafiotabuada",
4      "versionCode": 1
5    },
    "ios": {
      "bundleIdentifier": "com.devup.desafiotabuada",
      "buildNumber": "1.0.0"
    }
  },

```

Definimos o ID (package) e a versão do aplicativo Android

```
    "version": "1.0.0",
    "splash": {
      "image": "./assets/splash.png",
      "resizeMode": "contain",
      "backgroundColor": "#03989e"
    },
    "android": {
      "package": "com.devup.desafiotabuada",
      "versionCode": 1
    },
16    "ios": {
17      "bundleIdentifier": "com.devup.desafiotabuada",
18      "buildNumber": "1.0.0"
19    }
20  }
21 }

```

Definimos o ID (bundleIdentifier) e a versão do aplicativo iOS

O código do arquivo `app.json` segue no **Código 1**.

```
1  {
2    "expo": {
3      "name": "Desafio Tabuada",
4      "icon": "./assets/icone.png",
5      "slug": "desafio-tabuada",
6      "version": "1.0.0",
7      "splash": {
8        "image": "./assets/splash.png",
9        "resizeMode": "contain",
10       "backgroundColor": "#03989e"
11     },
12     "android": {
13       "package": "com.devup.desafiotabuada",
14       "versionCode": 1
15     },
16     "ios": {
17       "bundleIdentifier": "com.devup.desafiotabuada",
18       "buildNumber": "1.0.0"
19     }
20   }
21 }
```

**Código 1.** Arquivo `app.json`

## Gerando o instalador do Android

Chegamos ao ponto final da programação do nosso aplicativo que é gerar o instalador. Através dele vamos conseguir enviar nosso App para as lojas.

Antes de gerar o aplicativo você precisa se logar em sua conta no expo. Para se logar execute o **Código 1** no terminal e em seguida digite seu usuário e senha.

```
expo login
```

**Código 1.** Comando para se logar em uma conta expo

Em seguida precisamos instalar uma ferramenta responsável por gerar o instalador do Android. Para isso execute o **Código 2**.

```
npm install -g eas-cli
```

**Código 2.** Comando para se gerar um .apk ou .aab

Com a ferramenta `eas-cli` instalada o próximo passo é abrir o seu projeto no terminal - já logado na sua conta do expo, e executar o comando do **Código 3**.

```
eas build -p android
```

**Código 3.** Comando para se gerar um .apk ou .aab

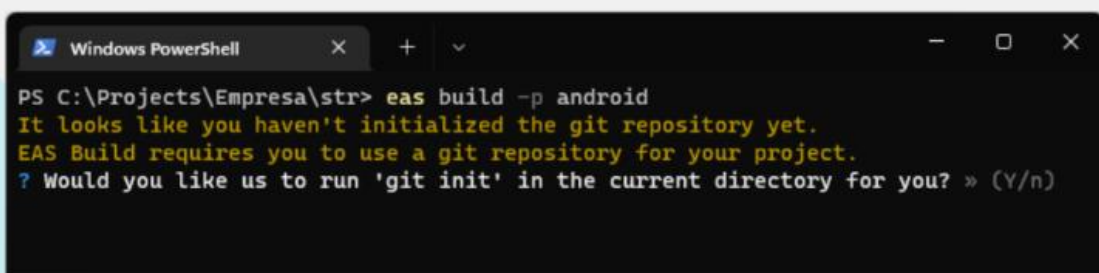
Entenda todo o processo no slide abaixo:



Abra o terminal na raiz do projeto e execute o seguinte comando:

```
c:\Desafio Tabuada> eas build -p android
```

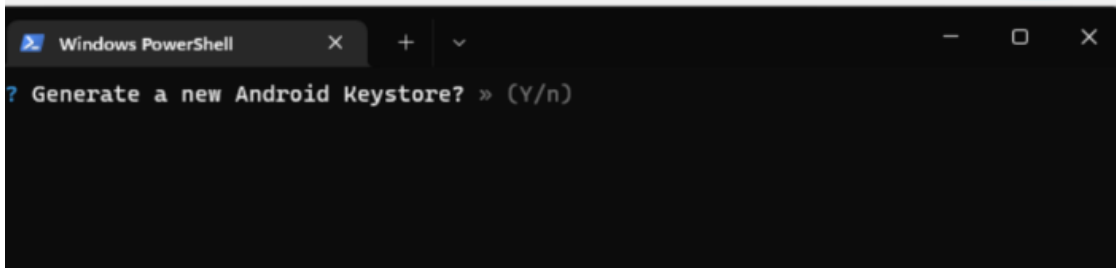
Ao executar o comando a seguinte mensagem vai aparecer na tela, basta teclar **Y**.



```
Windows PowerShell
PS C:\Projects\Empresa\str> eas build -p android
It looks like you haven't initialized the git repository yet.
EAS Build requires you to use a git repository for your project.
? Would you like us to run 'git init' in the current directory for you? » (Y/n)
```

```
PS C:\Projects\Empresa\str> eas build -p android
It looks like you haven't initialized the git repository yet.
EAS Build requires you to use a git repository for your project.
? Would you like us to run 'git init' in the current directory for you? » (Y/n)
? Would you like us to run 'git init' in the current directory for you? ... yes
We're going to make an initial commit for your repository.
Commit message: » Initial commit
```

Ao aparecer a pergunta abaixo, basta teclar **Y** para que o Expo gere pra você uma **keystore**.

A screenshot of a Windows PowerShell terminal window. The title bar shows 'Windows PowerShell' with standard window controls. The terminal text reads: '? Generate a new Android Keystore? » (Y/n)'.

```
Windows PowerShell
? Generate a new Android Keystore? » (Y/n)
```

A **keystore** é um 'registro' do seu aplicativo.

Quando terminar o processo um link vai aparecer no terminal.

You can monitor the build at

<https://expo.io/dashboard/mvpmateus/builds/e005536c-7fe4-42ab-8dc7-d4278819e7f3>

Waiting for build to complete.

You can press Ctrl+C to exit. It won't cancel the build, you'll be able to monitor it at the printed URL.

\ Build queued...

Isso significa que nosso .aab está sendo gerado no site do Expo

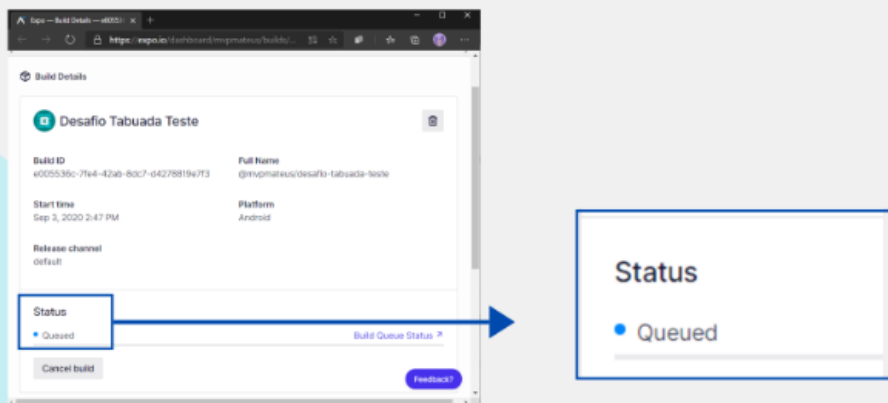


Podemos monitorar esse processo através do próprio terminal.

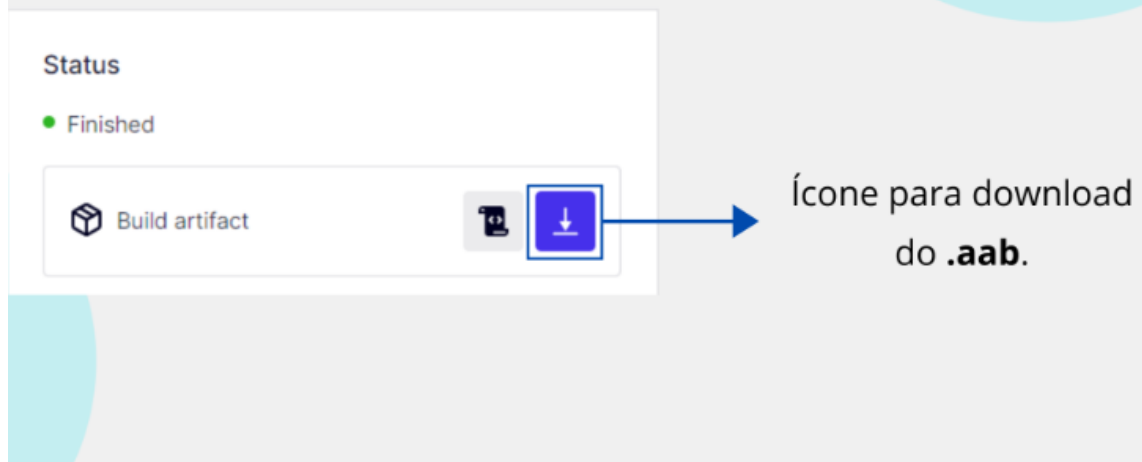
```
Waiting for build to complete.  
You can press Ctrl+C to exit. It won't cancel the build, you'll be able to monitor it at the printed URL.  
\\ Build queued...
```

Se preferir feche o terminal ou encerre o processo (CTRL + C) e acompanhe pelo link que foi gerado.

Link onde podemos ver o status do processo de gerar o arquivo.

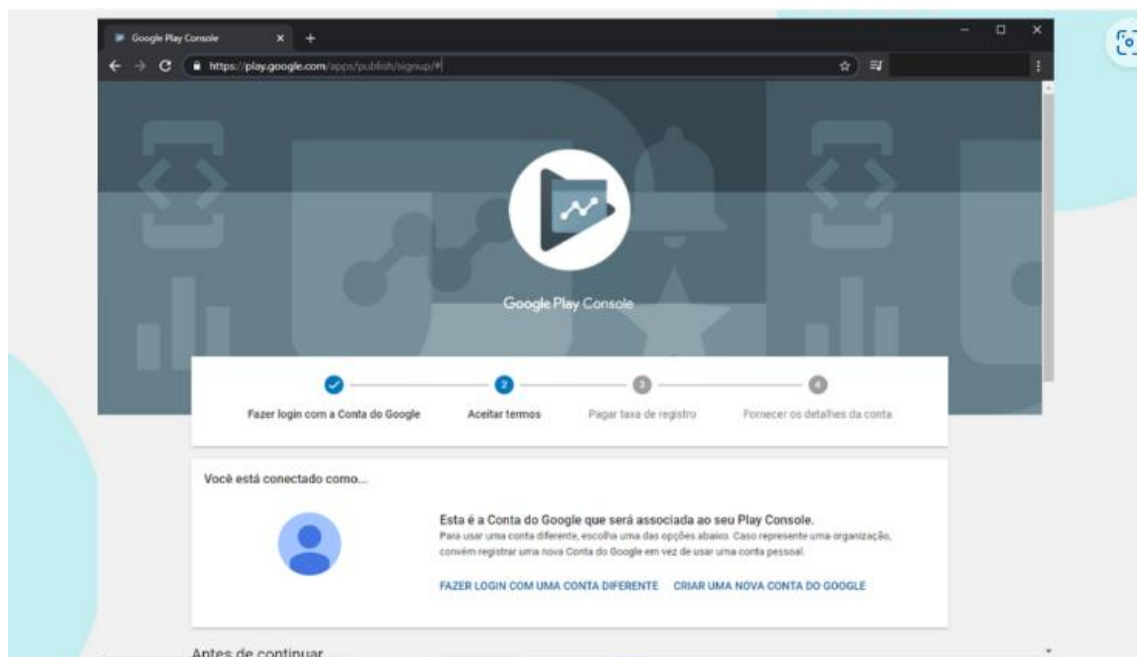


Ao finalizar podemos baixar o arquivo clicando no ícone.



O arquivo que você acabou de baixar (.aab) já pode ser enviado para a Google Play.

Para publicar Apps na Google Play é preciso [criar uma conta de desenvolvedor](#), que requer o pagamento de uma taxa única, como vemos na **Figura 1**.



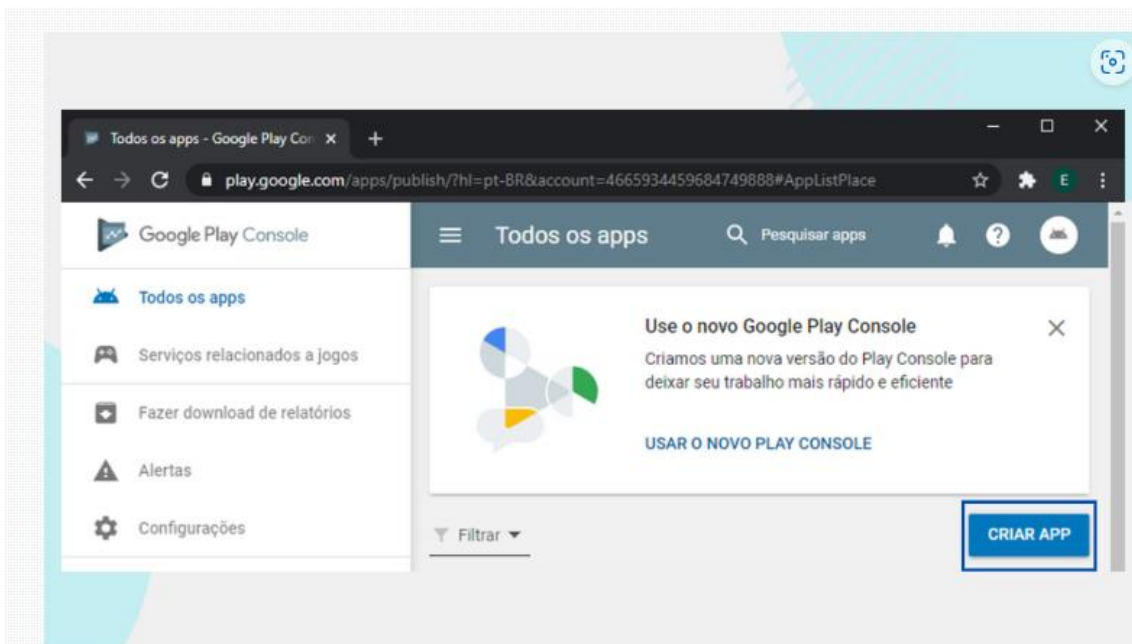


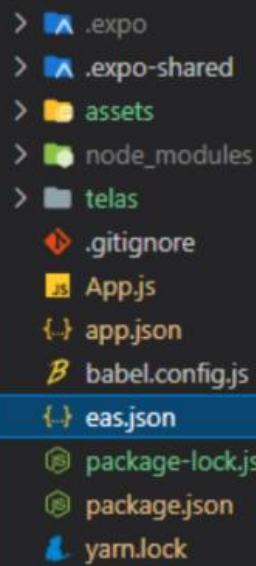
Figura 2. Painel de controle na loja Google Play

Caso você não queira criar uma conta na Google Play, o que requer pagamento de uma taxa, a alternativa é gerar o .apk e enviar diretamente para os dispositivos dos usuários.

### Gerando um arquivo .apk

Veja no flow abaixo como criar um .apk e instalar o aplicativo direto no seu celular:

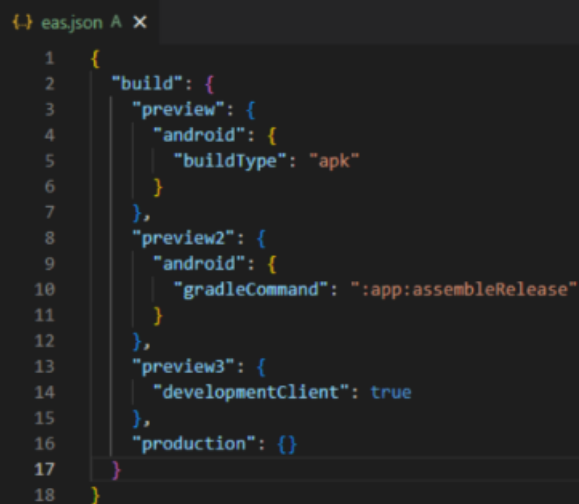
## Gerando o instalador .apk - Android



```
> .expo
> .expo-shared
> assets
> node_modules
> telas
.gitignore
App.js
app.json
babel.config.js
eas.json
package-lock.json
package.json
yarn.lock
```

Para gerar um .apk e instalar o aplicativo direto no dispositivo você precisa **criar o arquivo eas.json** na raiz do seu projeto caso ele ainda não tenha sido criado.

Seu código deve ser o seguinte:



```
{
  "build": {
    "preview": {
      "android": {
        "buildType": "apk"
      }
    },
    "preview2": {
      "android": {
        "gradleCommand": ":app:assembleRelease"
      }
    },
    "preview3": {
      "developmentClient": true
    },
    "production": {}
  }
}
```

Depois de salvar o arquivo, abra o terminal na raiz do projeto e execute o seguinte comando:

```
eas build -p android --profile preview
```

Mais uma vez um link vai aparecer no terminal.

```
You can monitor the build at
```

```
https://expo.io/dashboard/mvpmateus/builds/e005536c-7fe4-42ab-8dc7-d4278819e7f3
```

```
Waiting for build to complete.
```

```
You can press Ctrl+C to exit. It won't cancel the build, you'll be able to monitor it at the printed URL.
```

```
\ Build queued...█
```

Isso significa que nosso .apk está sendo gerado no site do Expo.

Ao finalizar podemos baixar o arquivo clicando no ícone.



Veja no **Código 4** o conteúdo do arquivo eas.json.

```
1  {
2    "build": {
3      "preview": {
4        "android": {
5          "buildType": "apk"
6        }
7      },
8      "preview2": {
9        "android": {
10         "gradleCommand": ":app:assembleRelease"
11       }
12     },
13     "preview3": {
14       "developmentClient": true
15     },
16     "production": {}
17   },
18   "cli": {
19     "version": ">= 0.53.1"
20   }
21 }
```

**Código 4.** Conteúdo do arquivo eas.json

## 8. Gerando o Instalador do iOS (.ipa)

Para gerar um arquivo .ipa - instalador do aplicativo para iOS, é necessário ter uma conta de desenvolvedor da Apple. A conta na Apple Store é paga anualmente.

Caso você tenha a conta de desenvolvedor então poderá seguir os próximos passos.

Faça o login em sua conta no expo, caso ainda não tenha feito, antes de executar o comando para gerar o arquivo .ipa. Feito isso execute o **Código 1** no terminal e insira o seu usuário e senha.

```
expo login
```

**Código 1.** Comando para se logar em uma conta expo

## Gerando o arquivo .ipa

Caso ainda não tenha instalado o `eas-cli` na aula anterior, abra o terminal e execute o comando do **Código 3**.

```
npm install -g eas-cli
```

**Código 3.** Comando para se gerar um .apk ou .aab

Com a ferramenta `eas-cli` instalada o próximo passo é abrir o seu projeto no terminal - já logado na sua conta do expo, e executar o comando do **Código 4**.

```
eas build -p ios
```

**Código 4.** Comando para se gerar um .ipa

Abra a raiz do projeto no terminal e execute o **Código 5**.

```
expo build:ios
```

**Código 5.** Executando código

