

Documentação do App – Casa de Jogo

Como rodar no emulador

Para rodar no emulador é necessário ter instalado na máquina o **android studio**, e adicionados a variável de ambiente **PATH** os caminhos das ferramentas binárias do android studio como abaixo:

```
export ANDROID_SDK_HOME=/home/italo/Android/Sdk
export PATH=$PATH:$ANDROID_SDK_HOME/tools:$ANDROID_SDK_HOME/platform-tools
```

Claro, entenda que esta configuração está de acordo com o **linux ubuntu 22.0.4**. Se for fazer funcionar no **windows**, será necessária alterar para a configuração equivalente.

Feito isto, é necessário instalar o emulador. Na época que está sendo produzido este app, o **android** mais recente é o **14 API 34**. Logo, está sendo utilizado para testar um emulador correspondente a estas versões. O nome do **Android Virtual Device (AVD)** utilizado é “**Pixel_3a_API_34_extension_level_7_x86_64**”.

Tecnologias utilizadas além do android studio

- Node v20.11.0
- Npm 10.4.0
- React Native 0.73.4
- react-native-cli 12.3.2

O repositório

O código fonte produzido em React Native 0.73.4 está no github. Para clonar o projeto, faça o seguinte:

```
git clone https://www.github.com/italoherbert/cjapp
```

Instalando as dependências

Para instalar as dependências, entre no diretório cjapp correspondente ao projeto clonado e execute o seguinte comando:

```
npm install
```

Se ocorrer algum problema, execute:

```
rm -rf node_modules package-lock.json
npm install
```

Rodando o projeto

Para rodar execute o seguinte na raiz do projeto:

```
npx react-native start
```

Alterando o nome da aplicação

Acesse o arquivo `android/app/src/main/res/values/strings.xml` e altere o nome da aplicação nesse simples arquivo.

As libs do react-native utilizadas

- expo
- expo-sqlite
- @react-navigation/native – e dependências
- @react-navigation/native-stack
- @react-navigation/bottom-tabs
- @fontawesome – várias libs
- @react-native-picker/picker
- @react-native-community/datetimepicker
- react-native-bootsplash
- react-native-dialog
- react-native-snackbar

Foi utilizado o seguinte comando para criar o projeto:

`npx react-native init cjapps`

Após isto, instalar os [módulos do expo](#) com o seguinte comando:

`npx install-expo-modules@latest`

Feito isto, é necessário configurar o arquivo `android/app/build.gradle` como segue:

Localize o seguinte código:

```
// Added by install-expo-modules
entryFile = file(["node", "-e", "require('expo/scripts/resolveAppEntry')",
rootDir.getAbsoluteFile().getParentFile().getAbsolutePath(), "android", "absolute"].execute(null,
rootDir).text.trim())
cliFile = new File(["node", "--print", "require.resolve('@expo/cli)"].execute(null, rootDir).text.trim())
bundleCommand = "export:embed"
```

E comente estas linhas do arquivo.

Isto é necessário para o build de release funcionar corretamente. Caso contrário, é mostrado um erro referente ao hash dos arquivos de assets.

Para instalar o [expo-sqlite](#), utilize o seguinte comando:

`npx expo install expo-sqlite`

Para instalar o [@react-navigation](#), ver a seguinte página:

<https://reactnavigation.org/docs/getting-started>

Instalar `@react-navigation/native` e `@react-navigation/native-stack` com suas dependências

Para instalar `@react-natigation/bottom-tabs`, ver a seguinte página:

<https://reactnavigation.org/docs/bottom-tab-navigator/>

Para instalar `@fontawesome` e suas libs, ver a seguinte página:

<https://origin.fontawesome.com/docs/web/use-with/react-native>

Para instalar as outras libs, executar os seguintes comandos:

```
npm i @react-native-picker/picker
npm i @react-native-community/datetimepicker
npm i react-native-bootsplash
npm i react-native-dialog
npm i react-native-snackbar
npm i moment
```

Configurando o `@react-navigation` para evitar defeitos no aplicativo

Abra o arquivo: `android/app/src/main/java/com.cjapp/MainActivity.kt`, e altere como o seguinte:

```
class MainActivity: ReactActivity() {
    // ...
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(null)
    }
    // ...
}
```

Configurando o ícone do aplicativo

Foi utilizado o seguinte comando para gerar o ícone do aplicativo:

```
npx icon-set-creator create icon-and-splash.png
```

Configurando o splash screen

Utilize o seguinte comando para instalar na pasta drawable a imagem de splash screen:

```
npx react-native generate-bootsplash icon-and-splash.png \  
  --platforms=android,ios,web \  
  --background=F5FCFF \  
  --logo-width=100 \  
  --assets-output=assets \  
  --flavor=main \  
  --html=index.html
```

Agora, é necessário configurar a [splashscreen](https://github.com/zoontek/react-native-bootsplash) conforme a documentação encontrada em:

<https://github.com/zoontek/react-native-bootsplash>

Gerando o APK de debug e release

Utilize o arquivo [./build-and-install.sh](#) para gerar o **apk** conforme os seguintes comandos:

Para debug:

```
./build-and-install.sh debug r
```

Para release:

```
./build-and-install.sh release r
```

Atenção: A opção “r” significa “[reinstall](#)”, para reinstalar o aplicativo em seu celular. Para utilizar esta opção, certifique-se de que o telefone está conectado via cabo ao computador e permita pelo celular o acesso via computador.

Se quiser apenas fazer **build** sem instalar no celular, basta não colocar a opção “r” no final conforme o seguinte:

```
./build-and-install.sh debug  
./build-and-install.sh release
```

Obs: continua na página seguinte....

Configurações adicionais para versão RELEASE

Para a versão release é necessário o arquivo de assinatura com extensão .keystore. Para gerá-lo utilize o arquivo [./gen-keystore.sh](#) com o seguinte comando:

[./gen-keystore.sh](#)

isto gerará o arquivo de nome [android/app/release.keystore](#) que deve ser configurado no arquivo [android/app/build.gradle](#) conforme a seguir:

Localize o seguinte:

```
android {  
    ...  
    signingConfigs {  
        ...  
        debug {  
            ...  
        }  
        // adicione aqui o seguinte:  
        release {  
            storeFile file('release.keystore')  
            storePassword '221088'  
            keyAlias 'release'  
            keyPassword '221088'  
        }  
        // Substituindo os dados pelos conforme a criação do arquivo “release.keystore”  
    }  
  
    buildTypes {  
        debug { ... }  
        release {  
            debuggable true // necessário para acessar os dados do aplicativo via:  
                           // adb shell run-as  
  
            signingConfig signingConfigs.release  
            ...  
        }  
    }  
    ...  
}
```

Um tutorial explicando como gerar a chave .keystore e configurar o arquivo acima pode ser encontrado em:

<https://reactnative.dev/docs/signed-apk-android>

Problema com o METRO-CONFIG

Pode ser necessário desfazer uma alteração feita automaticamente com a instalação dos [módulos expo](#). Então faça o seguinte:

No arquivo [metro.config.js](#), comente as linhas:

```
//const { getDefaultConfig } = require('expo/metro-config');  
//const { mergeConfig } = require('@react-native/metro-config');
```

E, abaixo ou acima das linhas comentadas, adicione a linha:

```
const {getDefaultConfig, mergeConfig} = require('@react-native/metro-config');
```

Tornando o aplicativo de release em produção DEBUGÁVEL

Para fazer isso, basta localizar no arquivo [android/app/build.gradle](#) o seguinte: **android > buildTypes > release**, e adicionar a opção [debuggable](#) com valor [false](#) ou [true](#).

Se o valor for false, então não será possível acessar os dados da aplicação como, por exemplo, o banco de dados em formato sqlite.

Se o valor de [debuggable](#) for [true](#), então será possível fazer o seguinte:

```
adb shell  
$ run-as com.cjapp cp /data/user/0/com.cjapp/files/SQLite/cjapp.db /sdcard/Android/obb/cjapp.db
```

O banco de dados SQLite

Como visto acima, o banco de dados tem nome cjapp.db e está localizado em: [/data/user/0/com.cjapp/files/SQLite/cjapp.db](#) ou [/data/data/com.cjapp/files/SQLite/cjapp.db](#)

Para copiar o banco de dados [do celular para o computador](#) no diretório atual, faça:

```
adb shell  
$ run-as com.cjapp cp /data/user/0/com.cjapp/files/SQLite/cjapp.db /sdcard/Android/obb/cjapp.db  
  
adb pull /sdcard/Android/obb/cjapp.db .
```

Para copiar o banco de dados da pasta local [do computador para o celular](#), faça:

```
adb push cjapp.db /sdcard/Android/obb/cjapp.db  
  
adb shell  
$ run-as com.cjapp cp /sdcard/Android/obb/cjapp.db /data/user/0/com.cjapp/files/SQLite/cjapp.db
```