Setting up the development environment

Development OS: Windows

Target OS: Android

Installing dependencies

You will need Node, the React Native command line interface, Python2, a JDK, and Android Studio.

While you can use any editor of your choice to develop your app, you will need to install Android Studio in order to set up the necessary tooling to build your React Native app for Android.

Node, Python2, JDK

We recommend installing Node and Python2 via Chocolatey, a popular package manager for Windows.

React Native also requires Java SE Development Kit (JDK), as well as Python2. Both can be installed using Chocolatey.

Open an Administrator Command Prompt (right click Command Prompt and select "Run as Administrator"), then run the following command:

```
choco install -y nodejs.install python2 openjdk8
```

If you have already installed Node on your system, make sure it is Node 10 or newer. If you already have a JDK on your system, make sure it is version 8 or newer.

Android development environment

Setting up your development environment can be somewhat tedious if you're new to Android development. If you're already familiar with Android development, there are a few things you may need to configure. In either case, please make sure to carefully follow the next few steps.

1. Install Android Studio

Download and install Android Studio. While on Android Studio installation wizard, make sure the boxes next to all of the following items are checked:

`Android SDK``Android SDK Platform``Android Virtual Device` If you are not already using Hyper-V: `Performance (Intel ® HAXM)` (See here for AMD or Hyper-V)

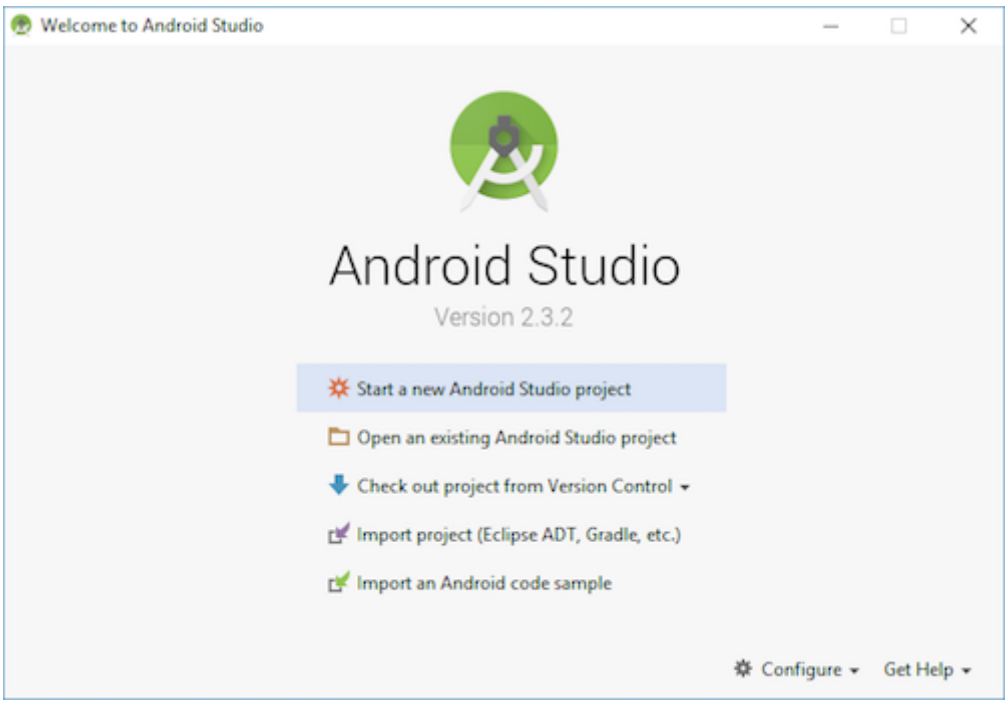
Then, click "Next" to install all of these components.

Once setup has finalized and you're presented with the Welcome screen, proceed to the next step.

2. Install the Android SDK

Android Studio installs the latest Android SDK by default. Building a React Native app with native code, however, requires the `Android 10 (Q)` SDK in particular. Additional Android SDKs can be installed through the SDK Manager in Android Studio.

To do that, open Android Studio, click on "Configure" button and select "SDK Manager".



Select the "SDK Platforms" tab from within the SDK Manager, then check the box next to "Show Package Details" in the bottom right corner. Look for and expand the `Android 10 (Q)` entry, then make sure the following items are checked:

- `Android SDK Platform 29`
- `Intel x86 Atom_64 System Image` or `Google APIs Intel x86 Atom System Image`

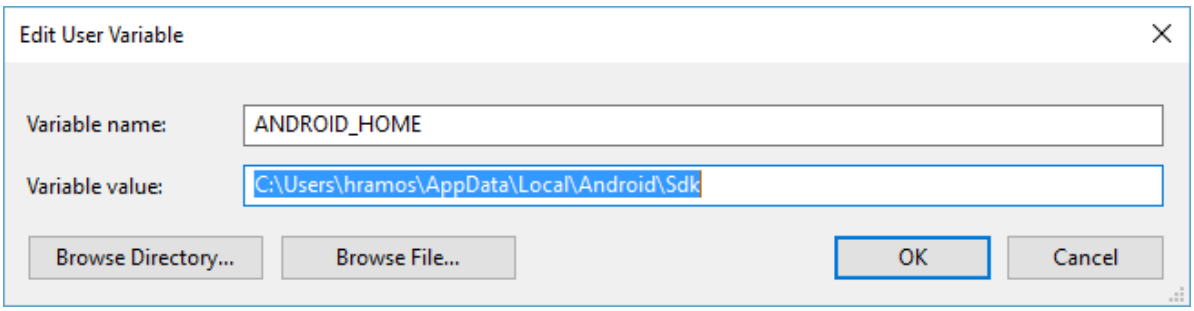
Next, select the "SDK Tools" tab and check the box next to "Show Package Details" here as well. Look for and expand the "Android SDK Build-Tools" entry, then make sure that `29.0.2` is selected.

Finally, click "Apply" to download and install the Android SDK and related build tools.

3. Configure the ANDROID_HOME environment variable

The React Native tools require some environment variables to be set up in order to build apps with native code.

1. Open the **Windows Control Panel.**
2. Click on **User Accounts,** then click **User Accounts** again
3. Click on **Change my environment variables**
4. Click on **New...** to create a new `ANDROID_HOME` user variable that points to the path to your Android SDK:



The SDK is installed, by default, at the following location:

```
%LOCALAPPDATA%\Android\Sdk
```

You can find the actual location of the SDK in the Android Studio "Settings" dialog, under **Appearance & Behavior** → **System Settings** → **Android SDK**.

Open a new Command Prompt window to ensure the new environment variable is loaded before proceeding to the next step.

1. Open powershell
2. Copy and paste **Get-ChildItem -Path Env:** into powershell
3. Verify `ANDROID_HOME` has been added

4. Add platform-tools to Path

1. Open the **Windows Control Panel.**
2. Click on **User Accounts,** then click **User Accounts** again
3. Click on **Change my environment variables**
4. Select the **Path** variable.
5. Click **Edit.**
6. Click **New** and add the path to platform-tools to the list.

The default location for this folder is:

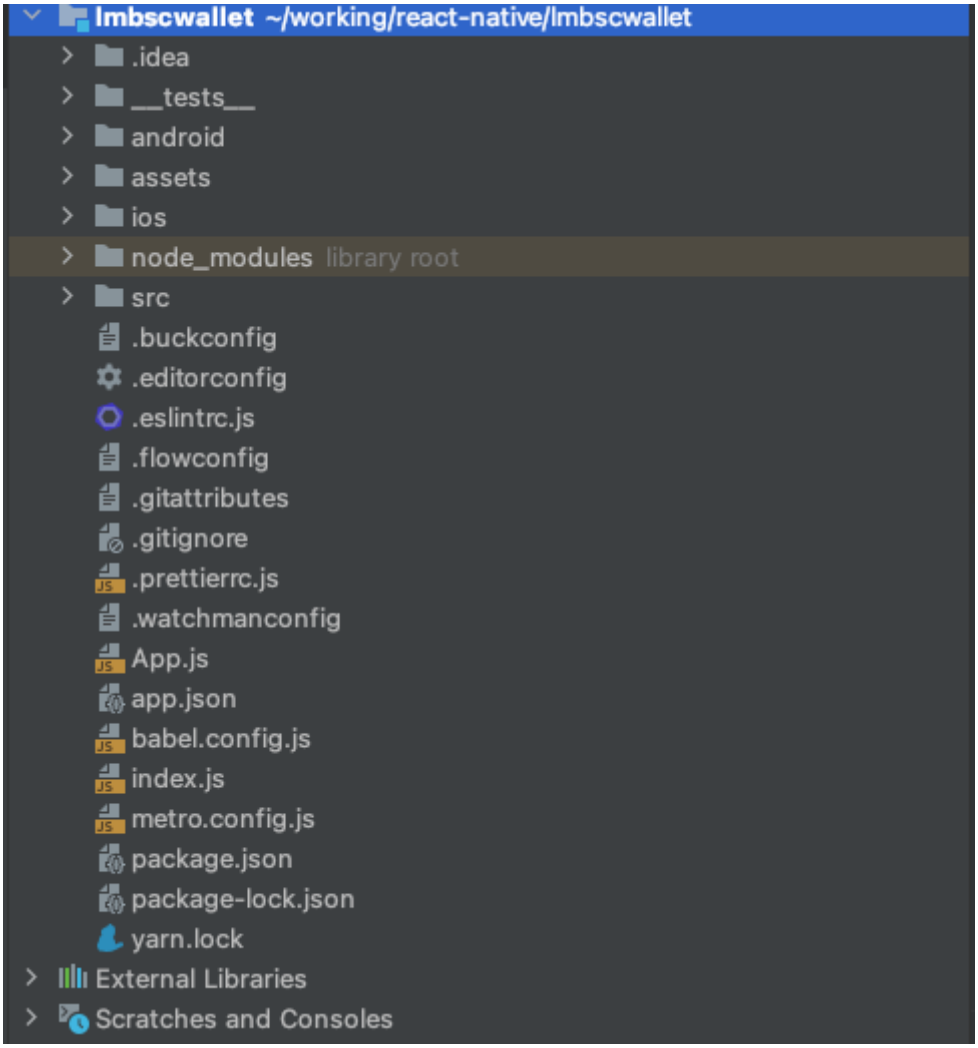
```
%LOCALAPPDATA%\Android\Sdk\platform-tools
```

React Native Command Line Interface

React Native has a built-in command line interface. Rather than install and manage a specific version of the CLI globally, we recommend you access the current version at runtime using `npx`, which ships with Node.js. With `npx react-native <command>`, the current stable version of the CLI will be downloaded and executed at the time the command is run.

Import BSCWallet project

I'm going to use IntelliJ IDEA as the IDE. You can choose which IDE as your preferable IDE as well



Change the Project Package

Note: This step is really important. So pay your attention please

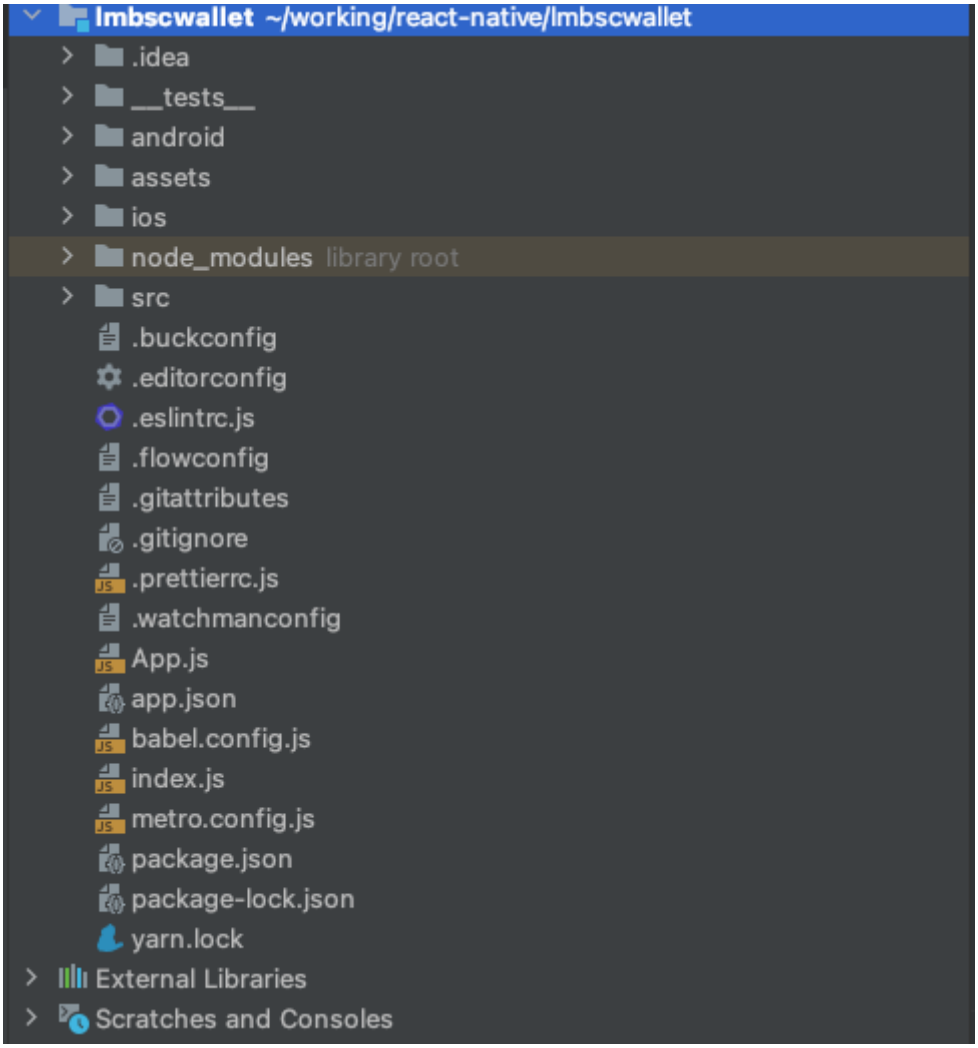
1. Install the **react-native-rename** lib
   - Open the CMD with Administrative Privileges
   - `npm install react-native-rename -g`
2. Your root project folder. Run the following command line: **com.lm.yourpackage** (you should use **your own package**)
   - `npx react-native-rename "New BSCWallet App" -b com.lm.yourpackage`

Install dependencies

Execute the following command to install the dependencies needed

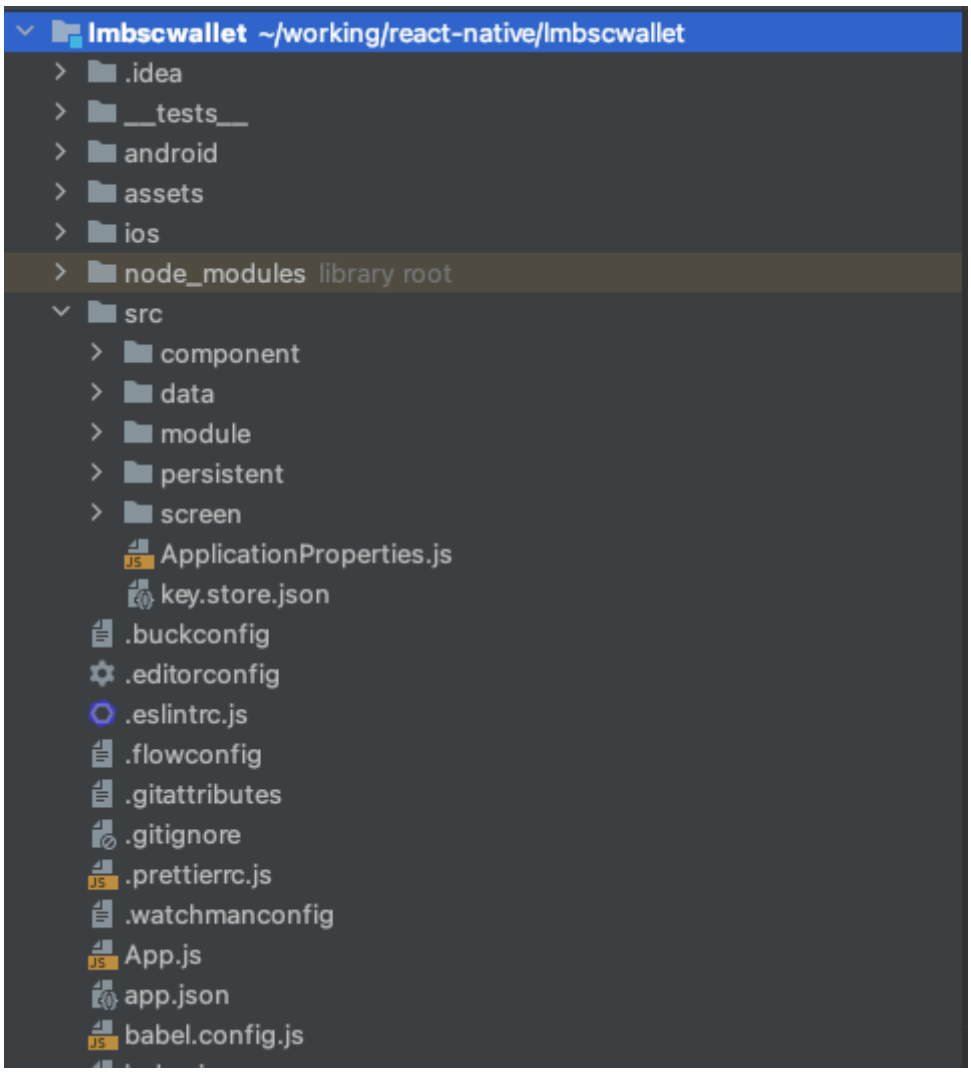
```
npm install
```

Once It has installed, You will see the `node_modules` folder in your project as below. It has all the dependencies needed for our project.



Edit BSCWallet Project

To customize our project as your needed. You will have to take a look at these folders below



- Firstly, please take a look at the screen folder where you can create a new screen or modify something as your request.
- Secondly, the persistence folder is where you can interact with the API server, redux layer, etc ...

Run BSCWallet Project

Then run the following command to start our application

```
$ cd android
$ gradlew clean
$ cd ..
$ npx react-native run-android
```

After a while, Our application will run on your connected android device.

Publishing to Google Play Store

Generating an upload key

You can generate a private signing key using `keytool`. On Windows `keytool` must be run from `C:\Program Files\Java\jdkx.x.x_x\bin`.

```
$ keytool -genkeypair -v -keystore wpay-key.keystore -alias wpay_realease_key -
keyalg RSA -keysize 2048 -validity 10000
```

his command prompts you for passwords for the keystore and key and for the Distinguished Name fields for your key. It then generates the keystore as a file called `wpay-key.keystore`.

The keystore contains a single key, valid for 10000 days. The alias is a name that you will use later when signing your app, so remember to take note of the alias.

Setting up Gradle variables

1. Place the `wpay-key.keystore` file under the `android/app` directory in your project folder.
2. Edit the file `~/.gradle/gradle.properties` or `android/gradle.properties`, and add the following (replace `*****` with the correct keystore password, alias and key password)

```
MYAPP_UPLOAD_STORE_FILE=wpay-key.keystore
MYAPP_UPLOAD_KEY_ALIAS=wpay_realease_key
MYAPP_UPLOAD_STORE_PASSWORD=*****
MYAPP_UPLOAD_KEY_PASSWORD=***
```

Adding signing config to your app's Gradle config

The last configuration step that needs to be done is to setup release builds to be signed using upload key. Edit the file `android/app/build.gradle` in your project folder, and add the signing config

```
...
android {
    ...
    defaultConfig { ... }
    signingConfigs {
        release {
            if (project.hasProperty('MYAPP_UPLOAD_STORE_FILE')) {
                storeFile file(MYAPP_UPLOAD_STORE_FILE)
                storePassword MYAPP_UPLOAD_STORE_PASSWORD
                keyAlias MYAPP_UPLOAD_KEY_ALIAS
                keyPassword MYAPP_UPLOAD_KEY_PASSWORD
            }
        }
    }
    buildTypes {
        release {
            ...
            signingConfig signingConfigs.release
        }
    }
}
...
```

Generating the release APK

Run the following in a terminal:

```
$ cd android
$ ./gradlew bundleRelease
```

The generated AAB can be found under `android/app/build/outputs/bundle/release/app.aab`, and is ready to be uploaded to Google Play.

Testing the release build of your app

Before uploading the release build to the Play Store, make sure you test it thoroughly. First uninstall any previous version of the app you already have installed. Install it on the device using the following command in the project root:

```
npx react-native run-android --variant=release
```

Support:

We will be happy to answer all the app related questions/issues.

If you need support, please send us an email using the contact form on the user page. We usually respond to support requests within 24 hours so please feel free to contact us with problems of any kind or even simple questions, We don't mind responding.

- Send us a ticket via Envato
- Email Support: lm.envato1820@gmail.com](mailto:lm.envato1820@gmail.com)
- Skype Support: lm.envato1820@gmail.com]([lm.envato1820@gmail.com)