# Guia de Instalação da Biblioteca PasswordVerifier para React Native

## Introdução

Esse documento tem como objetivo ser um tutorial passo a passo para instalação da biblioteca PasswordVerifier para React Native.

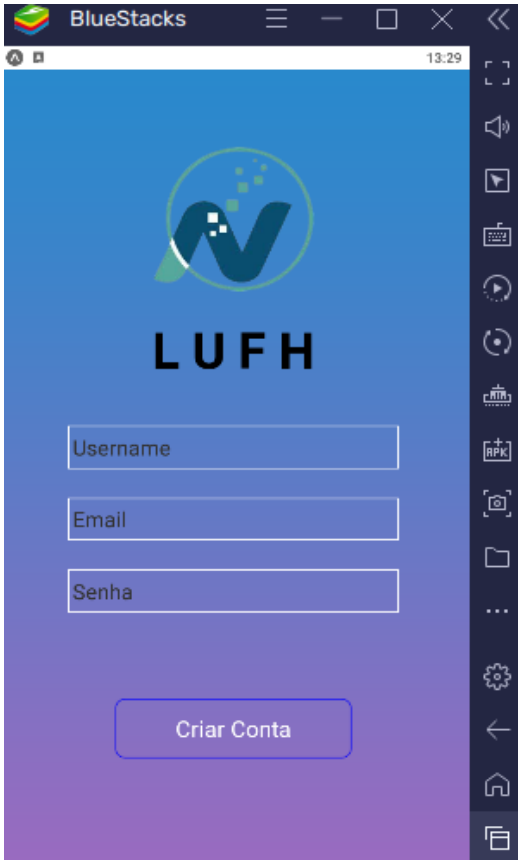
## Softwares necessários

- Node.js (v14.15.4) e NPM (6.14.10). Instalar Node/NPM
- React Native (versão recomendada: 6.0.0). Instalar React Native
- Editor de Texto para React Native (recomendado: Visual Studio Code).
- Celular Android (versão mínima: 5.0) ou emulador (sugestão: BlueStacks).
- Opcional: Expo (versão utilizada: 4.11.0). Instalar Expo.

## Instalação da Biblioteca PasswordVerifier

Para instalação e uso da biblioteca PasswordVerifier, será necessário ter um projeto do React Native criado e pronto para uso (veja como criar seu primeiro projeto aqui).

Para ilustrarmos, criamos uma simulação de tela de cadastro de usuário (criada inteira no VSCode e executada no emulador Bluestacks) para exemplificarmos os usos da biblioteca (código do aplicativo disponível aqui).




*Tela do Bluestacks*

Em seguida, instalamos a biblioteca Password Verifier (disponível aqui) utilizando o comando “npm install passwordverifier” no prompt de comando do VSCode.

```
PS C:\Users\luanf\OneDrive\Documentos\UEPB\LUFH\LUFH_App> npm install passwordverifier
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wante
d {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

+ passwordverifier@1.0.0
updated 1 package and audited 936 packages in 7.619s
```

*Instalação da biblioteca pelo VSCode*

Tendo a biblioteca instalada com sucesso nas dependências do seu app, ela já está pronta para ser importada e utilizada.

**Utilização da Biblioteca**

Para utilizarmos a biblioteca no nosso App, basta fazermos a importação da classe PasswordVerifier da biblioteca de mesmo nome.

```
5  import {PasswordVerifier} from 'PasswordVerifier';
```

Como os métodos da classe lidam com erros próprios, é interessante importarmos a classe de erros também, para lidarmos com eles apropriadamente.

```
import {PasswordVerifier, PasswordError} from 'PasswordVerifier';
```

Agora, podemos usar quaisquer métodos da classe (ver manual de uso completo da biblioteca aqui).

```
// Exemplo Verificação de Erros
try {
  // Utilização do método de verificar tamanho da senha digitada
  PasswordVerifier.assertSize("1234567",8);
} catch(erro){ // tratar o erro especifico de senhas
  if(erro instanceof PasswordError){
    alert(erro.message);
  } else { // Outro tipo de erro
    throw erro;
  }
}
```

*Exemplo de uso da biblioteca*