


Document React Native **pool inspector app**





Summary

1. Accessibility/Improvements
 - 1.1 Accessibility
 - 1.2 Responsive
2. Security
 - 2.1 Initial strategy
 - 2.2 Sensitive Information
 - 2.3 Serverless function
 - 2.4 Security layers
 - 2.5 Observability/to monitor
3. Good habits

1 Accessibility and improvements

It is a viable idea to use the new 0.68 architecture of React Native and seek to develop a more interactive interface.

[Announcing React Native 0.68](#)

1.1 Accessibility

Use Tooltip to give feedback on '<TextInput/>' when clicked.

```
keyExtractor={this.keyExtractor}
showsVerticalScrollIndicator={false}
/>
</View>
</SafeAreaView>
</Modal>
<View style={{marginBottom: 15}}>
  <View style={containerViewStyle}>
    <TouchableOpacity
      onPress={toggleModal}
      style={{flexDirection: 'row', marginRight: '2%'}}>
      <Text>{flagValue}</Text>
      /* <Text style={styles.dialCode}></Text> */
    </TouchableOpacity>
    <TextInput
      placeholder={placeholder}
      editable={editEvent}
      style={inputStyles}
      placeholderTextColor="#000000"
      value={this.state.value}
      onChangeText={this.handleChange}
      onBlur={this.handleBlur}
      {...restProps}
    />
  </View>
  {touched && error && <Text style={styles.errorText}>{error}</Text>}
</View>
</View>
```

[react-native-walkthrough-tooltip - npm](#)

1.2 Responsive

Use Flex Wrap to declare screen limits.

[Layout with Flexbox · React Native](#)

To limit the font of the app, use the necessary Tags, the `allowFontScaling` or `maxFontSizeMultiplier` attributes can also be used, See the examples below:



```
import { Text } from 'react-native';

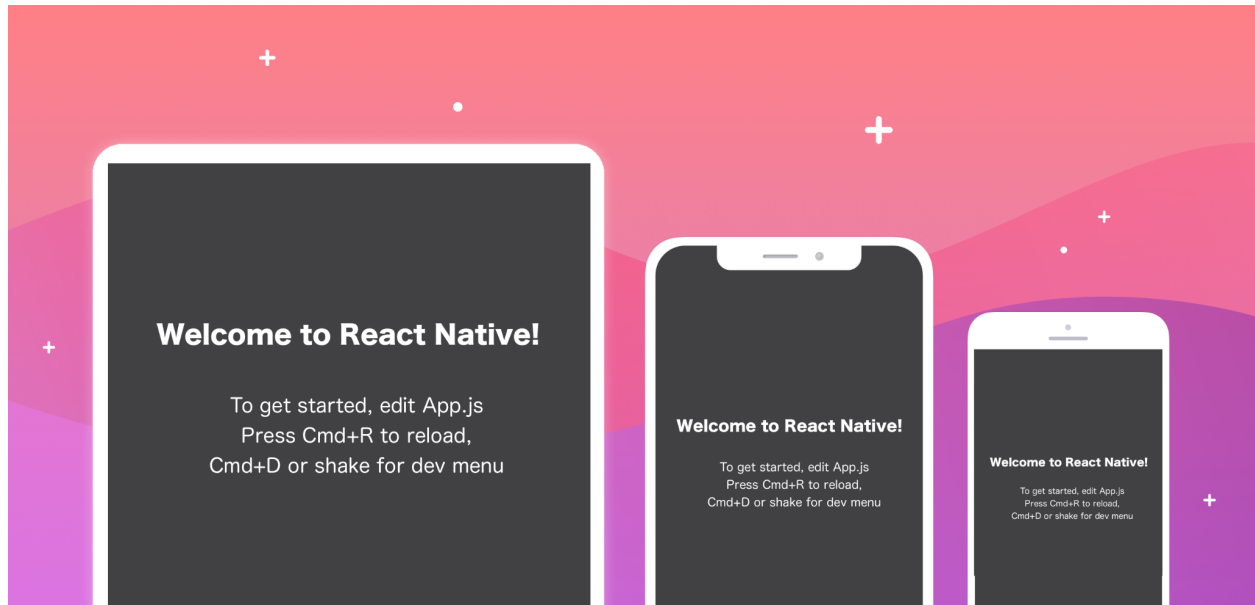
const TextWrapper = (props) => {
  return <Text {...props} />
}

TextWrapper.defaultProps = {
  maxFontSizeMultiplier: 1,
};

export default TextWrapper;
```

[Text · React Native](#)


You can also use font libraries to work with relative sizes:



[react-native-responsive-fontsize - npm](https://www.npmjs.com/package/react-native-responsive-fontsize)

Use the status bar in the project:





It is also a good idea to use the library [GitHub - IjzerenHein/react-native-shared-element](#).
It would be interesting to use the animatable([react-native-animatable - npm](#)) In the
project.

2 Security

Details problems brought about by the existence of the superuser.
Part of the topics here simulates the app already in production.

2.1 Initial strategy

Add a check to the code. You can use the 'Jail Monkey' library. Below is an example of a code that badges the application to prevent intrusions.



```
// Importe o BackHandler do React Native
import {AppRegistry, BackHandler} from 'react-native';
import App from './App';
import {name as appName} from './app.json';

// importação do JailMonkey
import JailMonkey from 'jail-monkey'

// Aqui você pode simplesmente crashar o aplicativo quando o dispositivo do usuário tiver Root ou JailBroken ativo.
if (JailMonkey.isJailBroken()) {
  BackHandler.exitApp();
}

AppRegistry.registerComponent(appName, () => App);
```

[jail-monkey - npm](#)

2.3 Sensitive Information

Initially, it is recommended to use Hashes to encrypt sensitive information. Below is an example:

Plain text encryption

```
import CryptoJS from "react-native-crypto-js";

// Encrypt
let ciphertext = CryptoJS.AES.encrypt('my message', 'secret key 123').toString();

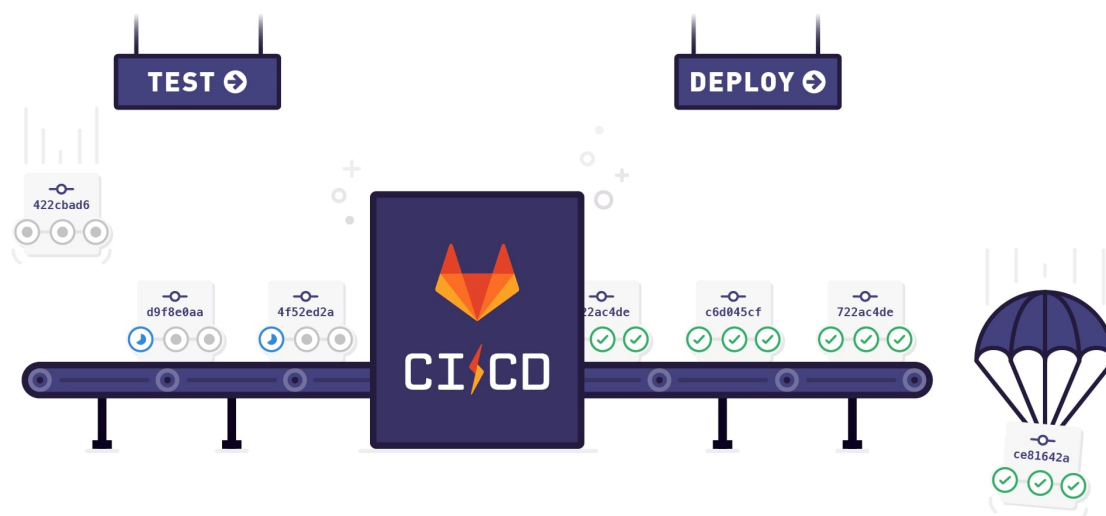
// Decrypt
let bytes = CryptoJS.AES.decrypt(ciphertext, 'secret key 123');
let originalText = bytes.toString(CryptoJS.enc.Utf8);

console.log(originalText); // 'my message'
```

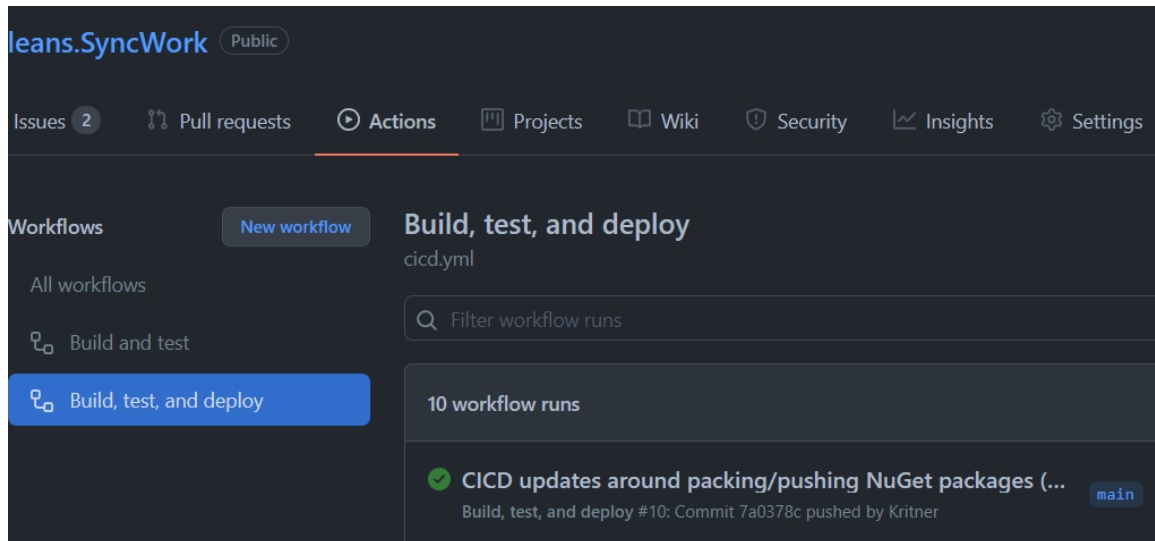
It is also recommended to use Hash SHA256.

[react-native-crypto-js - npm](#)

Generate the dotenv(.env) file to store sensitive information, but do not store API keys. Attention: Do not leave the dotenv file, keystore and p12 certificate in repositories. And let the CI/CD process fill in the files.



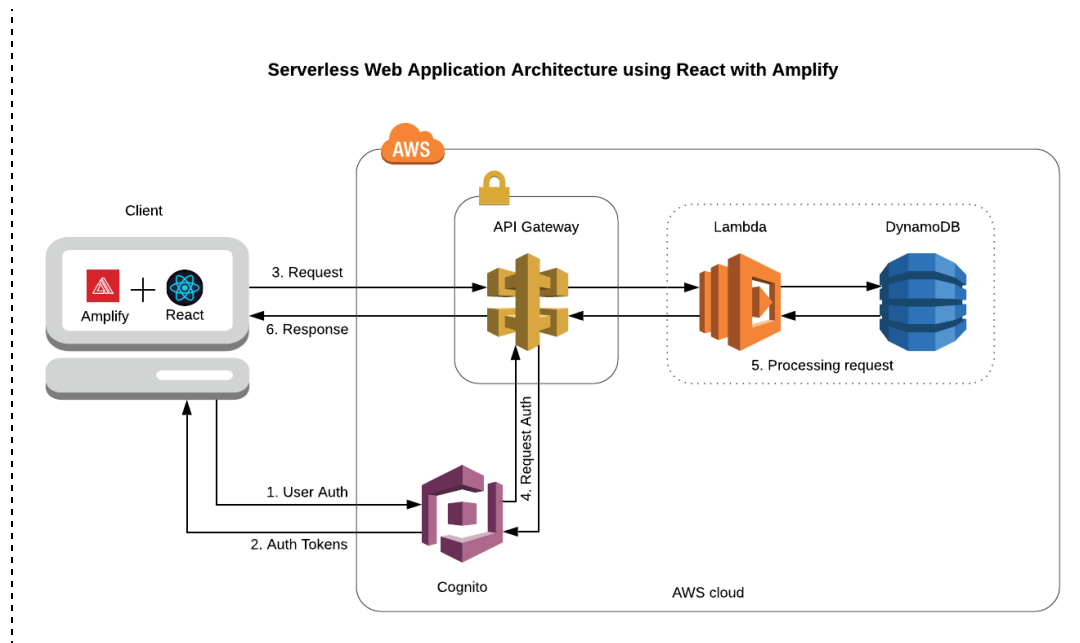
You can still use secrets actions to generate variables where the value cannot be changed or seen.



[secrethub/actions: Load secrets into GitHub Actions](#)

2.4 Serverless function

Use Serverless function to store keys and protect them, such as the GoogleMaps api keys used in the project.



[Serverless Functions – Vercel Docs](#)

Another tip that if using the proxy is to use the library:


<https://www.npmjs.com/package/http-proxy-middleware>.

2.5 Security layers

When the APP is played to some store the Bundle file (add-on file) is generated. The purpose here is to generate layers and avoid information exposed in this file and the APP can still be accessed by an apk and the information can be leaked.

To place these layers you can use the library [obfuscator-io-metro-plugin - npm](#). It puts extra layers in the Bundle, making attacks difficult.

Do not use all seen in the document as it can generate performance loss in the APP.



The library will generate the `metro.config.js` file in the project root.
You can also use the analysis tool [Diffchecker](#).

2.5 Observability/to monitor

To know about application failures for production use these tools:

[Crashlytics | React Native Firebase](#)

[React Native | Sentry Documentation](#)

[React Native | New Relic Instant Observability](#)

You can also use [GitHub - dotlib/tech-day-observability: Repositório com foco em observabilidade de aplicações, utilizando Elasticsearch, Kibana, Strapi e React.js para o DotLib Tech Day](#) to create performance metrics and user experience alerts.

Also use SSL Pinning([react-native-ssl-pinning - npm](#)) which is a technique to avoid attacks.

3 Good habits

Prevent the app from crashing:

```
iconContainer: { position: 'absolute', top: Platform.OS === 'ios' ? 10 : 20, right: 20, },|
```

Declaring global variables and defaults:

```
export const SET_USER_DATA = 'SET_USER_DATA';

export const EDIT_USER_PROFILE_START = 'EDIT_USER_PROFILE_START';
export const EDIT_USER_PROFILE_SUCCESS = 'EDIT_USER_PROFILE_SUCCESS';
export const EDIT_USER_PROFILE_FAILURE = 'EDIT_USER_PROFILE_FAILURE';
export const CHANGE_PASSWORD_START = 'CHANGE_PASSWORD_START';
export const CHANGE_PASSWORD_SUCCESS = 'CHANGE_PASSWORD_SUCCESS';
export const CHANGE_PASSWORD_FAILURE = 'CHANGE_PASSWORD_FAILURE';

export const GET_ORG_DETAILS_START = 'GET_ORG_DETAILS_START';
export const GET_ORG_DETAILS_SUCCESS = 'GET_ORG_DETAILS_SUCCESS';
export const GET_ORG_DETAILS_FAILURE = 'GET_ORG_DETAILS_FAILURE';

export const EDIT_ORG_DETAILS_START = 'EDIT_ORG_DETAILS_START';
export const EDIT_ORG_DETAILS_SUCCESS = 'EDIT_ORG_DETAILS_SUCCESS';
export const EDIT_ORG_DETAILS_FAILURE = 'EDIT_ORG_DETAILS_FAILURE';

export const GET_ORG_ACCOUNT_DETAILS_START = 'GET_ORG_ACCOUNT_DETAILS_START';
export const GET_ORG_ACCOUNT_DETAILS_SUCCESS =
  'GET_ORG_ACCOUNT_DETAILS_SUCCESS';
export const GET_ORG_ACCOUNT_DETAILS_FAILURE =
  'GET_ORG_ACCOUNT_DETAILS_FAILURE';

export const EDIT_ORG_ACCOUNT_DETAILS_START = 'EDIT_ORG_ACCOUNT_DETAILS_START';
export const EDIT_ORG_ACCOUNT_DETAILS_SUCCESS =
  'EDIT_ORG_ACCOUNT_DETAILS_SUCCESS';
export const EDIT_ORG_ACCOUNT_DETAILS_FAILURE =
  'EDIT_ORG_ACCOUNT_DETAILS_FAILURE';
```

```
    } = this.props;
    return (
      <View style={viewStyles}>
        <View style={borderStyle}>
          {countryCode !== undefined && countryCode === true &&
            <Text style={countryCodeStyle}>
              +230{' '}
            </Text>
          }
          {icon && icon === true && (
            <Image source={src} style={styles.icon} resizeMode='contain' />
          )}
          <TextInput
            placeholder={placeholder}
            style={inputStyles}
            placeholderTextColor={placeholderTextColor}
            //onBlur={onBlur}
            value={this.state.value}
            onChangeText={this.handleChange}
            onBlur={this.handleBlur}
            {...restProps}
          />
          {forgot && forgot === true && (
            <Text
              onPress={onForgotPress}
              style={styles.forgotText}
            >Forgot?</Text>
          )}
        </View>
      </View>
    );
  }
}
```

Responsiveness with Direction:

```
    </Text>
    <View style={{flexDirection: 'row'}}>
      <Image
        source={require('../assets/calendarRoundGret.png')}
        style={detailStyles.Icons}
        resizeMode="contain"
      />
      <Text style={detailStyles.detailText}>
        {getSingleInspectionData.pool_construction_date}
      </Text>
    </View>
```