**Guia de Implementação**

SUMÁRIO

[1. Introdução 2](#_Toc485801465)

[1.1. Referências 2](#_Toc485801466)

[2. Organização e Estilo do Código 2](#_Toc485801467)

[3. Comentários 2](#_Toc485801468)

[4. Nomeação 2](#_Toc485801469)

[5. Declaração 3](#_Toc485801470)

[6. Reutilização 3](#_Toc485801471)

[7. Tratamento de Erros e de Exceções 3](#_Toc485801472)

[8. teste unitário 3](#_Toc485801473)

[9. Diretrizes Gerais 4](#_Toc485801474)

# Introdução

Este guia apresenta os padrões de codificação do Apontador de presença para eventos do IFSP – Campus Salto, desenvolvido para a disciplina de Desenvolvimemento para Web II no curso de Análise e Desenvolvimento de Sistemas do IFSP - Campus Salto.

O Apontador de presença foi desenvolvido utilizando o framework Javascript React Native, em sua interação front-end e na linguagem de programação php para a maioria de suas interações back-end, para tanto este documento estará dividido em duas partes de padronização, sendo elas, a utilizada para criação do aplicativo e sua interface e a de modelagem de negócios.

## Referências

* JavaScript Guia de estilo e convenções de codificação. Disponível em: <http://www.w3bai.com/pt/js/js\_conventions.html>
* Guias de leitura React Navigation. Disponível em: < https://reactnavigation.org/pt-BR/>
* Guias React Native. Disponível em: < https://facebook.github.io/react-native/docs/getting-started.html>

# Organização e Estilo do Código

* Arquitetura de pastas a ser mantida dentro da pasta src:
  + actions (Redux)
  + components (Componentes customizados para a aplicação, não os por padrão fornecidos pelo react)
  + reducers (Redux)
  + screens (telas da aplicação)
    - images (imagens utilizadas pelo sistema)
    - styles (estilos gerais do aplicativo, configuráveis)
  + services (endpoint de api's)
  + store (Redux)
* Os caminhos/ rotas das telas deve estar no arquivo index.js mais externo, na pasta src
* Tamanho de classes php: não mais que 1000 linhas
* Tamanho de métodos: não mais que 20 linhas
* Declaração de variáveis sempre no início do bloco que irá utiliza-la, mesmo que o java script permita o “hoisting”
* Importação de bibliotecas sempre no topo do código
* Padrão de nomeação de métodos: camelCase
* Padrão de nomeação de variáveis: camelCase

# Comentários

Classes

/\*\*

\* <Descrição da funcionalidade da Classe>

\* @author: <Nome autor>

\*/

Métodos

// <Descrição das funcionalidades do método>

# Nomeação

Arquivos Javascript

* nomearquivo.js

Exemplo:

Arquivo de tela de leitura de qrcode

qrcode.js

Variáveis

* nomeVariavel

Scripts SQL

* NOMESCRIPT.sql

# Declaração

1. Padrão de indentação de declaração de variáveis const:

const nomeVariavel = atribuicao (

{

[...],

[...],

});

1. Padrão de indentação de declaração de classes:

Class NomeClasse {

}

# Reutilização

* Criar arquivos distintos para cada nova tela do sistema
* Manter sempre um arquivo index.js como responsável pela execução da aplicação
* Manter atualizado arquivo de pacotes utilizados pela aplicação

# Tratamento de Erros e de Exceções

* Criar uma casse de tratamento de exceções na camada de apresentação.
* Garantir que erros e exceções não corrompam o estado atual da aplicação.

# teste unitário

* Criar plano de teste sempre se baseando no fluxo principal e alternativo do sistema e em suas exceções.

# Diretrizes Gerais

Não se aplica.