**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

No front-end, ao qual está sendo utilizado o framework React com a linguagem Typescript, já foi implementada algumas telas como login, landing page, esqueceu a senha, cadastro do usuário.

Para estas foi realizada já a navegação entre telas e a validação de alguns formulários pelo front-end.

Dificuldade está sendo em deixar responsivo o site tanto para aparelhos móveis quanto desktop, alguns estilos acabam 'quebrando', mas estamos a caminho do refinamento, das telas concluídas, cerca de 60% delas já está responsiva para qualquer aparelho.

## Referências

Documentos utilizados como fonte de informações para desenvolvimento do projeto.

* Typescript - https://www.typescriptlang.org/ - acessado 15/12/2020 ;

# Organização e Estilo do Código

O projeto em seu frontend, segue o padrão de organização utilizado dentro da startup Rocketseat. Segue anexo de link sobre como é dispostos os itens para codificação, localização e entendimento de cada área: https://youtu.be/X2RKRKdqqwM ;

# Comentários

Trecho JSX

{/\*

<TagJsx />

\*/}

Trecho de funcionalidades TS

//Função anônima que renderiza botões defaults

const renderizaBotoes=()=>{

}

# Nomeação

Componentes React

Os componentes React são dispostos dentro de uma pasta, que terá o nome do componente e deverá iniciar com letra maiúscula que é o padrão do próprio React. Os arquivos em si, não possuem esta necessidade, podendo eles iniciar com letras minúsculas.

Exemplo:

NomeComponente

//Demais arquivos

index.tsx

# Declaração

1. Padrão de indentação de declaração de componentes:

const NomeComponente:React.FC = () =>{

return( );

}

# Reutilização

Esta seção fornece regras e diretrizes para facilitar a reutilização.

* O retorno dos dados devem ser sempre no formato de JSON
* Todo elemento passível de reuso, deve ser componentizado
* Nenhum elemento deve ficar fora de sua arquitetura pré definida.

# Tratamento de Erros e de Exceções

* Qualquer erro será tratado na sessão de services destinada a tratativas junto ao banco de dados/backend.

# teste unitário

Testes de responsividade do layout foram realizados até o momento, demais testes aguardam conexão com a outra camada do projeto.

# Diretrizes Gerais

Utilize sempre as diretrizes e recomendações de desenvolvimento de sistemas SPA's.