**Documentação de Código**

Este documento tem por objetivo explicitar o código do desafio front-end projeto Contajá. Sua funcionalidade é uma loja de quadrinhos, onde o usuário pode visualizar uma listagem com diversos exemplares, buscar exemplares através do título, ver detalhes sobre o quadrinho, optar por comprar ou adicionar ao carrinho e por fim, acessar seu carrinho de comprar e verificar os exemplares que selecionou para comprar.

O documento está escrito na ordem dos componentes mais simples para os mais complexos.

De forma geral, o funcionamento da aplicação se inicia logo que o usuário acessa a url. A primeira página a ser exibida é a página home, exibindo por *default* uma listagem de quadrinhos ordenados pelos mais recentes. O usuário pode alterar esta listagem acessando o campo busca presente na *Header*. Assim, o usuário insere um texto e será exibido uma lista de quadrinhos com títulos semelhantes ao informado.

Selecionando qualquer um dos quadrinhos da listagem, o usuário será redirecionado para a página de detalhes do quadrinho, onde poderá visualizar maiores informações.

Tanto na listagem quanto nos detalhes dos quadrinhos o usuário poderá acionar os botões de comprar e/ou adicionar ao carrinho. O botão comprar redirecionará automaticamente o usuário para o carrinho de compras, onde estará os exemplares adicionados ao carrinho, juntamente com o comprado. O botão de adicionar ao carrinho contabilizará o exemplar para a conferencia futura no carrinho de compras. O usuário poderá usar o botão do carrinho de compras presente na *Header* para ir para a página de Carrinho de compras.

**API**

Foi utilizada a biblioteca Axios para as requisições HTTP. Ficou previamente definido nos Services 03 chamadas do tipo GET. get() retorna utilizada pelo componente Home que retorna os dados dos elementos mais recentes. getSearch() retorna para o componente SearchResults dados dos elementos filtrados pela entrada fornecida pelo usuário. getId() retorna apenas um elemento, de acordo com o ID informado e é utilizado no componente details para carregamento sem uso do context.

**Routes**

É um componente criado para o gerenciamento de direcionamento. Estabelece a página home como padrão, com a URL “/”. Tem também adicionado ao seu escopo a página DetailsComic, SearchResults, CartRender e ErrorPage. Faz o redirecionamento de acordo com a url passada como parâmetro para useNavigate(). Url não definidas acionam o redirecionamento para a página ErrorPage.

**Contexts**

Contex é uma Hook do React que permite passar dados entre os nós, sem seguir a árvore do projeto. Após criado, o contexto pode ser chamado por qualquer nó que tanto pode usar quanto alterar os dados do contexto.

**Comic:** É o context que armazena a listagem de dados sobre os quadrinhos buscados na API. Seus dados são utilizados pelos componentes DetailsComic e SearchResults. O set acontece nos componentes Home e Search.

**CartList:** É o context que armazena os produtos quando o usuário clica em compra ou adicionar ao carrinho. É utilizado pelos componentes que precisam criar o componente carrinho de compras e alterado por componentes com botão comprar e adicionar ao carrinho.

**SelectedComic:** É o context utilizado pelo ComicElement e Details. Quando o usuário clicar sobre um produto para ver ser detalhes, os dados desse produtos serão alterados no context e utilizados pelo Details para montar o componente.

**Componentes**

**Footer:**Componente que fica na parte de baixo de todas as páginas com informação sobre o desenvolvedor do projeto.

**ShoppingCart:** O componente é um botão situado na Header. Tem o símbolo de um carrinho. Sua função é levar o usuário para a página de Carrinho de Compras sempre que for clicado. Ele também sinaliza quantos elemento tem no carrinho de compras.

**ShoppingButton:** É o botão que o usuário vai utilizar para efetuar a compra de algum produto. Quando disparado, ele salva os dados do produto no context CartList e redireciona para a página de checkout. Está presente tanto nos produtos listado na página Home e SearchResults, quanto na página de Details.

**CartButton:** É o botão que o usuário vai utilizar para adicionar produtos ao carrinho de compra. Quando disparado, ele salva os dados do produto no context CartList. Está presente tanto nos produtos listado na página Home e SearchResults, quanto na página de Details.

**Search:** Fica na Header e consiste em um campo de input e um botão de buscar. O usuário insere texto no input e ao clicar em buscar, dispara-se um evento que aciona a API getSearch(), buscando dados dos produtos com título semelhante ao texto inserido. Logo em seguida atualiza o context Comic e redireciona para a página SearchResults.

**CartElement**: O componente utiliza dados passados por parâmetro props para fazer sua montagem. Ele é utilizado para representar cada um dos elementos do carrinho de compras. Clicar sobre a imagem ou o título redirecionará o usuário para a página de detalhes.

**ComicElement:** Componente que utiliza dados passado como parâmetro props para fazer sua montagem. Ele é utilizado para representar cada um dos elementos das pagina Home e SearchResults. Clicar sobre a imagem ou o título redirecionará o usuário para a página de detalhes. Ele ainda possui os botões ShoppingButton e CartButton, que depara ações de compra e adicionar ao carrinho, respectivamente.

**RenderComics:** Este componente recebe uma listagem de produtos como parâmetro props e cria uma lista de ComicElement. Para tanto, à lista de produtos é aplicada a função map e cria-se para cada um dos elementos da lista, um componente ComicElement, sendo passado o elemento como parâmetro ao componente a ser criado.

**Header**: Componente presente no topo de todas as páginas, o Header é construído com uma logo Marvel, que serve de redirecionamento para a página Home, um componente Search e um componente ShoppingCart.

**Paginas**

**ErrorPage**: Pagina que contém mensagem de erro quando o usuário acessar url inválida. Tem na sua construção os componentes Header e Footer.

**CartRender**: Página acessa a lista de produtos adicionados ao carrinho através do context CartList. Na didMount do componente (useEffect), utilizando da lista acessada no context para contabilizar e agrupar os itens repetidos, alocando tais informações na lista comicsCart. Através desta nova lista, aplicando-se o map, cria-se uma lista de CartElement passando como parâmetro para sua montagem cada um dos elementos da comicList. A página ainda possui os componentes Header e Footer.

**DetailsComics**: Página criada toda vez que o usuário clica sobre um produto ou seu título, tanto nas páginas Home e SearchResults, quanto na página CartRender. Através do context SelectComic, acessa os dados do elemento selecionado pelo usuário para ver seus detalhes. Caso o context esteja sem elementos adicionados e o usuário tente acessar os detalhes diretamente através da url, na DidMount faz uma chamada HTTP getId(), passando como parâmetro o id presente na url. Tendo os dados, o componente é montado e exibido. Compõe a página ainda os componentes CartButton, ShoppingButton, Header e Footer.

**SearchResults**: Página que é acionada sempre que o usuário faz uma busca utilizando o componente Search. Após a busca os dados são armazenados no context Comics. Então a página busca esta lista de dados e os passa como parâmetros para o componente RenderComic. A página ainda possui os componentes Header e Footer.

**Home**: Página inicial do projeto. Na sua DidMount faz uma requisição HTTP get(). Os dados resposta da requisição são armazenados no context Comic e também utilizados como parâmetros para o componente RenderComic. A página ainda possui os componentes Header e Footer.