Projeto Final: Aplicativo de Compras do Viveiro Paradise

Tempo estimado necessário: 110 minutos

Introdução

Neste projeto final, você criará um aplicativo de carrinho de compras para uma loja online de plantas que oferece uma variedade de plantas de interior.

As funcionalidades do carrinho de compras do *Viveiro Paradise* incluirão:

- Uma página inicial com um botão que leva à página de listagem de produtos
- Uma barra de navegação com links para as páginas inicial, de listagem de produtos e carrinho de compras
- Um cartão para cada planta que exibe as diferentes plantas junto com suas imagens, nome, descrição, custo e um botão de **Adicionar ao carrinho**.
- Um mínimo de duas seções descrevendo as plantas nessa seção. Por exemplo, “Plantas Aromáticas” e “Plantas Medicinais”.
- Uma página do carrinho que exibe os produtos no carrinho.
- O carrinho deve ter um cartão para cada tipo de planta no carrinho. Cada cartão deve ter a miniatura, o custo unitário, o custo total de todas as plantas desse tipo e botões para aumentar e diminuir a quantidade, além do botão **Excluir**.
- Botões de **Continuar Comprando** e **Finalizar Compra**

Você implementará o conhecimento e as habilidades que adquiriu ao trabalhar no projeto prático para lidar com funcionalidades dinâmicas, como mostrar a quantidade no carrinho no ícone da barra de navegação e atualizar o custo de todos os itens no carrinho quando o usuário altera o número de itens.

Envio do Projeto

Para enviar seu aplicativo para avaliação, você precisará implantá-lo. Você precisará fornecer a URL como parte do projeto de revisão por pares. Você pode usar o GitHub Pages para hospedar. Você pode encontrar as instruções no laboratório *Configurando o Ambiente do GitHub* anteriormente neste módulo.

Você pode optar por hospedá-lo em outro lugar e fornecer esse link em vez disso. Apenas certifique-se de que seus colegas possam acessar o aplicativo facilmente usando a URL que você fornecer.

Ao implantar, certifique-se de anotar a URL do aplicativo.

Objetivos de Aprendizagem

Após concluir este laboratório, você será capaz de:

- Componentes React: Criar componentes funcionais React usando composição e aninhamento de componentes.
- Gerenciamento de Estado com Hooks: Implementar Hooks do React, especificamente os hooks useState e useEffect. Você gerenciará o estado em nível de componente usando hooks para controlar a visibilidade dos elementos.
- Integração com Redux: Integrar Redux dentro de um aplicativo usando conceitos do Redux, como ações, redutores e a loja.
- Renderização de Dados Dinâmicos: Renderizar dinamicamente dados obtidos de um array de objetos na interface do usuário. Você irá mapear arrays para gerar listas de componentes.
- Manipulação de Eventos e Renderização Condicional: Manipular eventos do usuário, como seleção de botão e acionar ações correspondentes.

Pré-requisitos

- Conhecimento básico de GitHub e sua própria conta no GitHub
- Entendimento de componentes funcionais do React, props, hooks e React Redux Toolkit
- Navegador web com console (como Chrome DevTools ou Firefox Console)

Aviso importante sobre este ambiente de laboratório

Skills Network Cloud IDE (baseado em Theia e Docker) é um IDE (Ambiente de Desenvolvimento Integrado) de código aberto que fornece um ambiente para laboratórios práticos em cursos e projetos.

Por favor, esteja ciente de que as sessões para este ambiente de laboratório não são persistentes. Sempre que você se conectar a este laboratório, um novo ambiente será criado para você. Você perderá dados se sair do ambiente sem salvar no GitHub ou em outra fonte externa. Planeje concluir esses laboratórios em uma única sessão para evitar a perda de seus dados.

Configurando seu ambiente no GitHub

1. Faça um fork do repositório

Você precisa fazer um fork do repositório GitHub para o React da sua aplicação. O repositório GitHub com o código base para este projeto está aqui:

- https://github.com/ibm-developer-skills-network/e-plantShopping.git
- Após seguir o link acima, clique no botão de fork.

2. Este repositório contém o layout básico da aplicação React para este projeto.

3. Você usará este repositório forkado para enviar seu código mais recente e manter um registro do trabalho que você faz. Salve periodicamente todos os seus arquivos. Seus arquivos devem ser salvos para executar os comandos git.

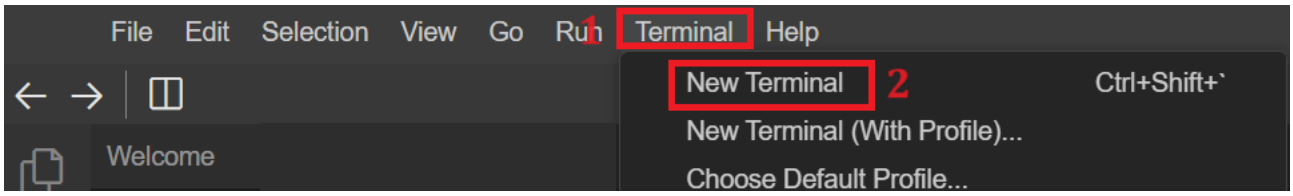
4. Execute os comandos `git add`, `git commit` e `git push` para atualizar as alterações da sua pasta de aplicação para o seu repositório GitHub para uma gestão adequada do código.

5. Como parte do projeto de revisão por pares, você precisará implantar seu aplicativo com o GitHub Pages para que seus colegas possam revisar a interface do usuário. Você pode revisar estas instruções para assistência com o GitHub e para as instruções de implantação da aplicação React.

Se você saiu e deseja retomar o trabalho no projeto, clone o repositório forkado onde você anteriormente enviou o código. Após clonar, você pode usar `git push origin` sem a necessidade de executar `git remote add...` para enviar suas alterações diretamente.

Crie seu projeto React

1. Se você não tiver um terminal aberto, selecione a aba “Terminal” no canto superior direito da janela e, em seguida, selecione “Novo Terminal”. A captura de tela abaixo mostra onde localizar essas opções na sua tela.

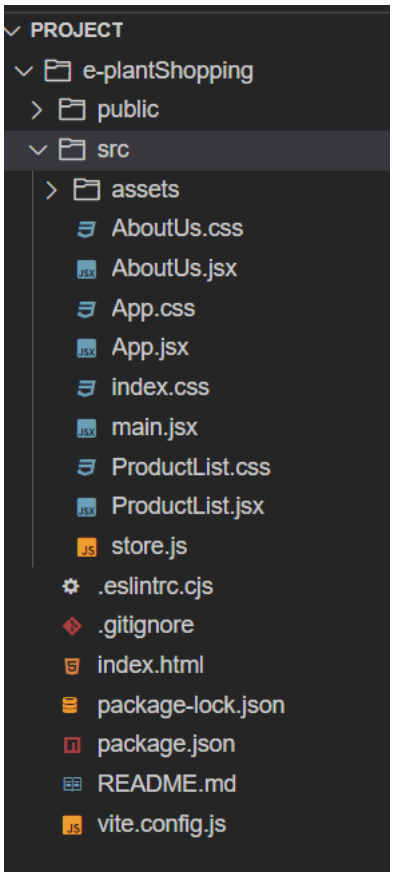


2. Agora clone o repositório forkado usando o comando fornecido, substituindo `<forked-repo-link>` pelo link do seu próprio repositório.

```
git clone <forked-repo-link>
```

3. Certifique-se de que o nome da sua aplicação corresponda ao seu projeto.

4. Após clonar o repositório, você verá a estrutura de pastas como na captura de tela abaixo.



5. Escreva o comando para entrar na pasta da aplicação no terminal. O comando definirá o caminho do seu terminal para executar a aplicação React na pasta `<forked-folder-name>`.

```
cd e-plantShopping
```

6. Para garantir que o código que você clonou está funcionando corretamente, você precisa seguir os passos dados:

   - Escreva o seguinte comando no terminal e pressione Enter para instalar todos os pacotes necessários para executar a aplicação.

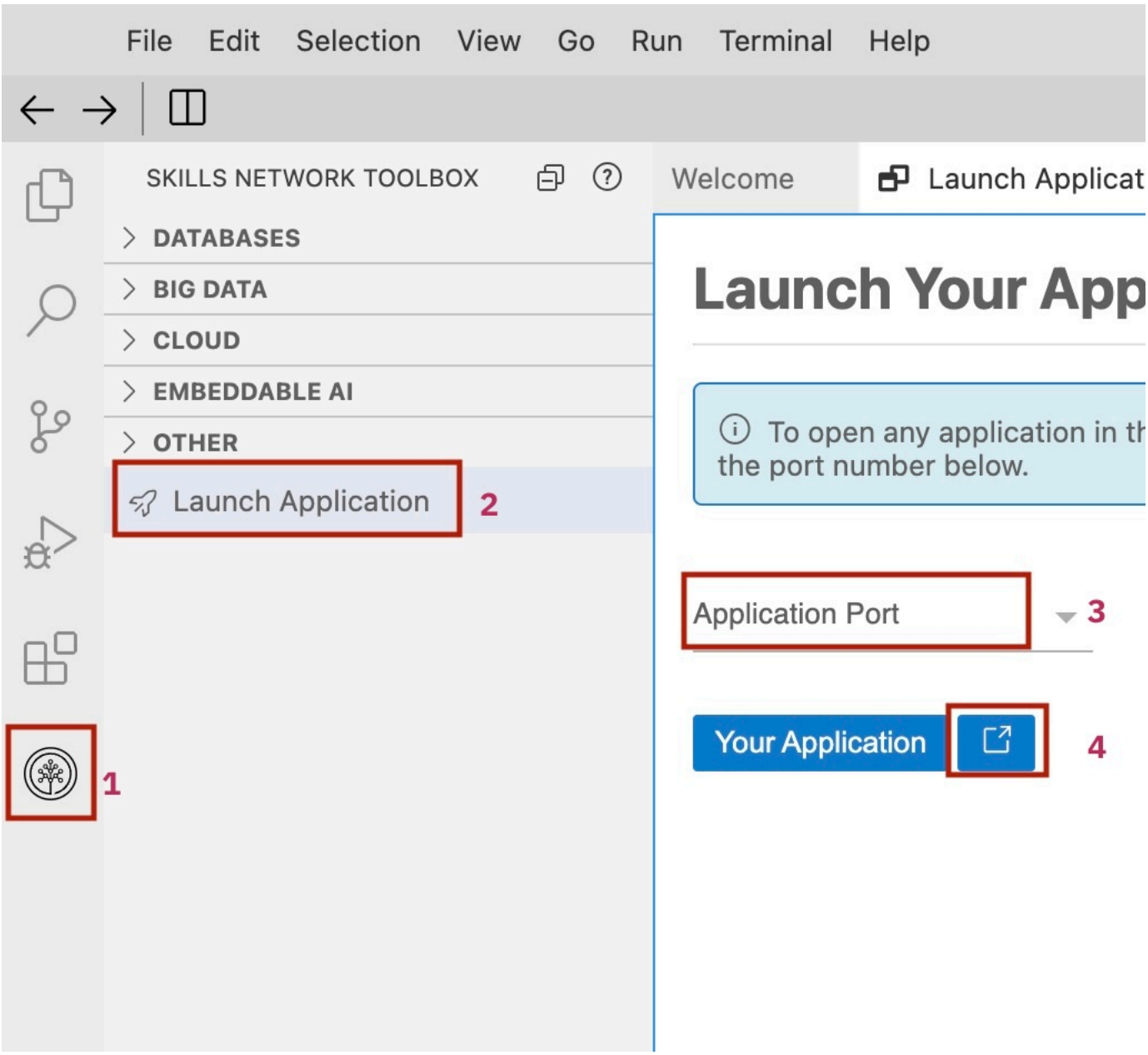
     ```
     npm install
     ```

   - Em seguida, execute o seguinte comando para rodar a aplicação, fornecendo o número da porta 4173.

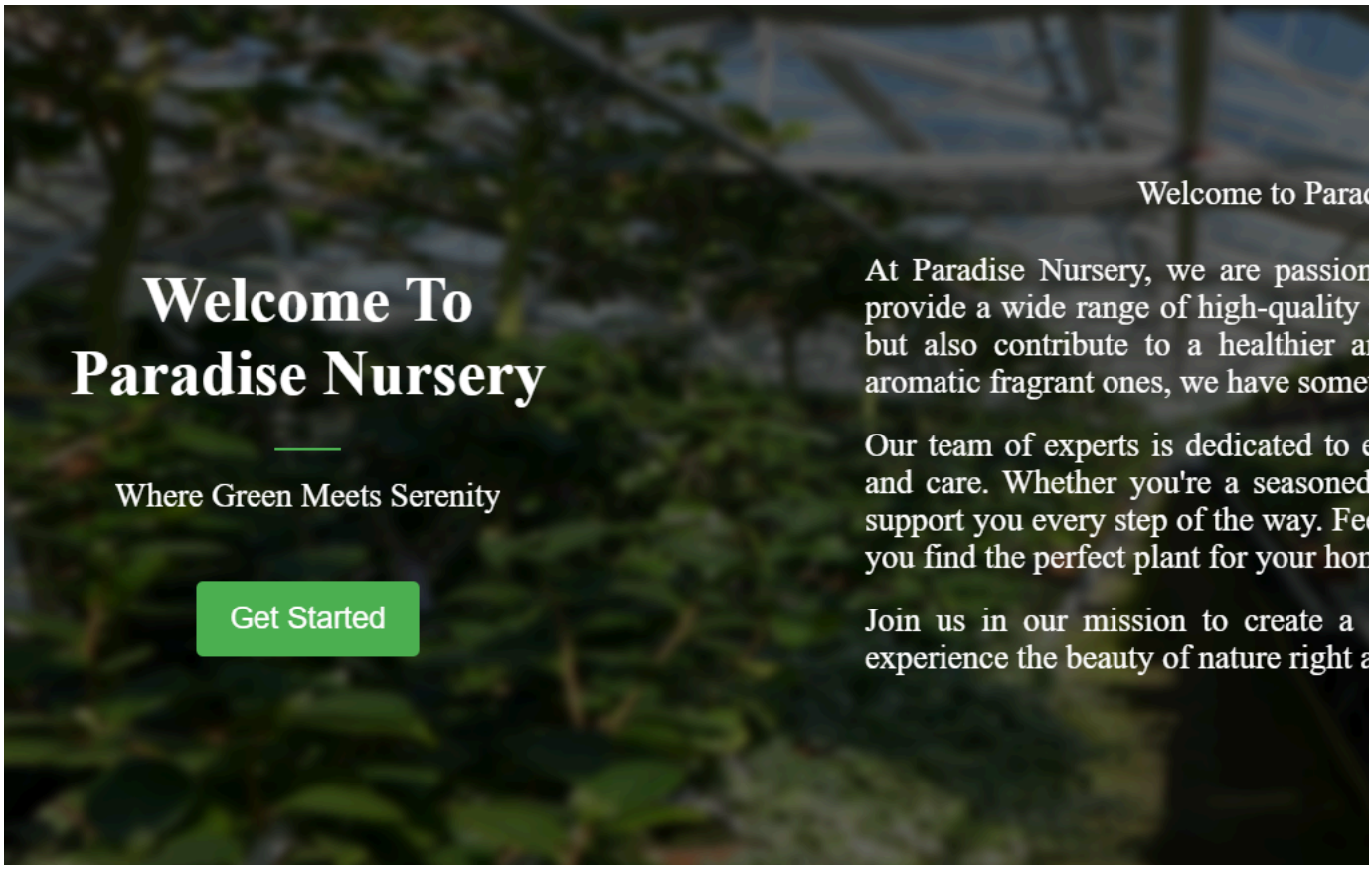
     ```
     npm run preview
     ```

7. Para visualizar sua aplicação React, clique no ícone Skills Network no painel esquerdo (consulte o número 1). Essa ação abrirá a **Caixa de Ferramentas do Skills Network**. Em seguida, clique em **Lançar Aplicação** (consulte o número 2). Insira o número da porta **4173** em **Porta da Aplicação** (consulte o número

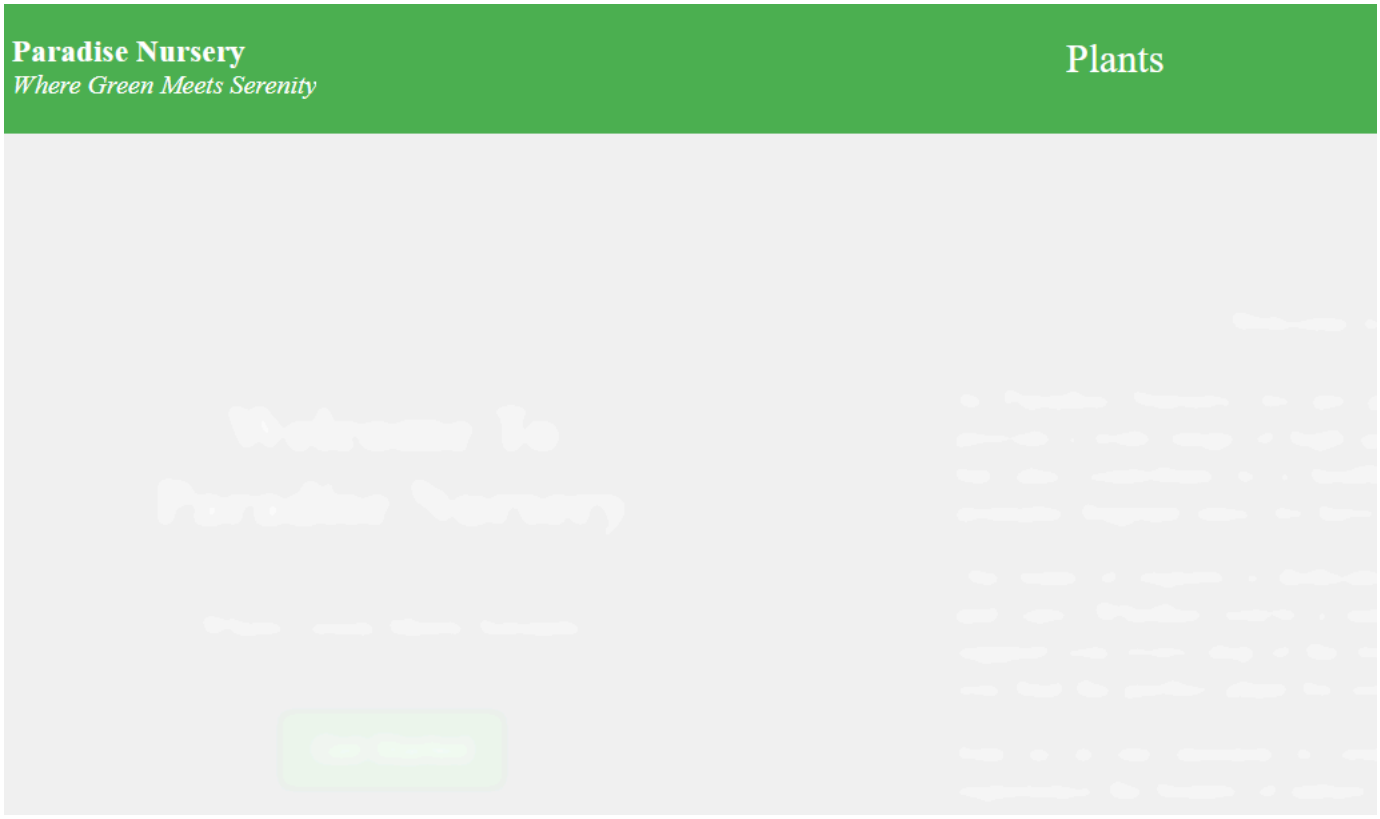
3) e clique  .



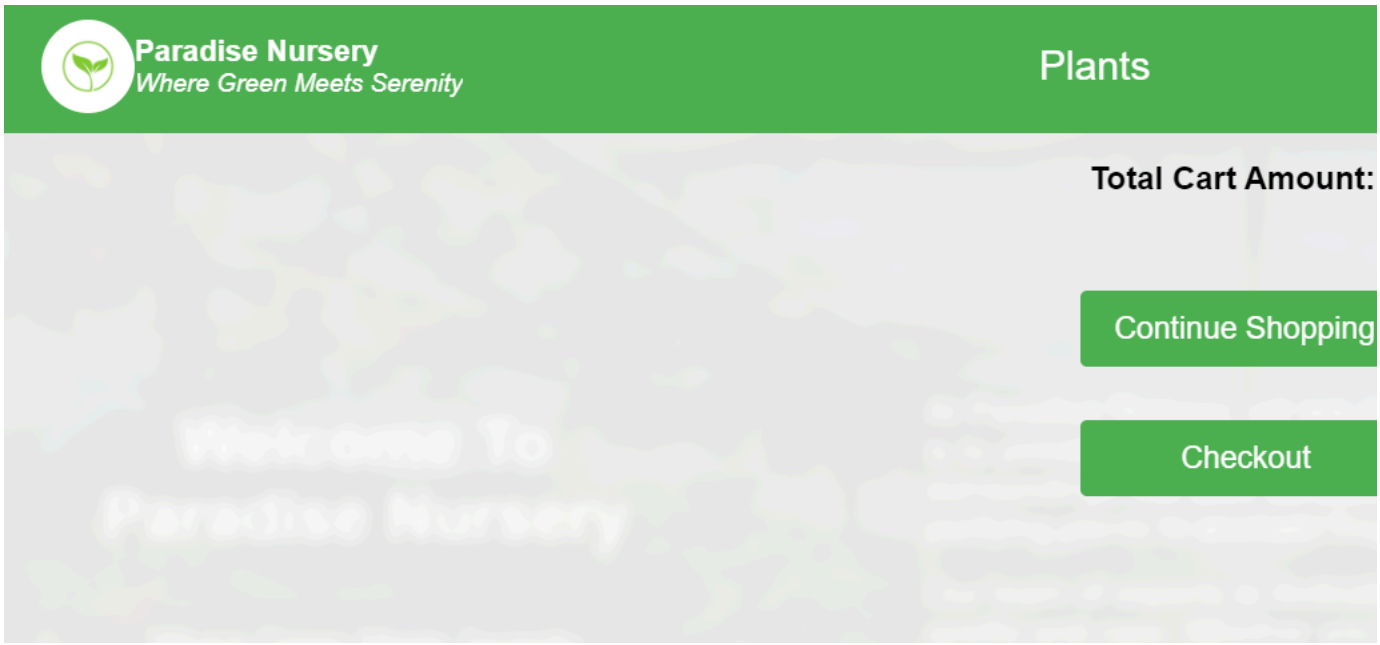
8. A saída será de acordo com a captura de tela dada, com a imagem de fundo.



9. Agora clique no botão `Get Started` e então você verá o layout dado de acordo com a captura de tela, que inclui uma barra de navegação com cor de fundo verde.



10. A barra de navegação contém três links:
    - `Paradise Nursey` - Isso levará você de volta à página inicial da aplicação.
    - `Plants` - Isso o navegará para a página onde as informações relacionadas à página estarão visíveis.
    - `ícone do carrinho` - Isso o navegará para a seção de itens do carrinho.
11. Quando você clicar no `ícone do carrinho`, a saída será visível conforme a captura de tela dada.



Complete as tarefas necessárias explicadas nas páginas que seguem. Seus colegas o avaliarão usando uma rubrica fornecida com base nessas tarefas.

Tarefa 1: Layout do componente ProductList

A página do produto permitirá que seus usuários comprem as diferentes plantas que você vende. Cada planta será exibida em seu próprio "cartão" com os dados relacionados armazenados no objeto planta. Você armazenará os objetos planta em um array. Siga estas etapas para o array e os objetos planta.

1. Exibir o Array de Plantas

- Navegue até o componente `ProductList.jsx` e você verá um array chamado `plantsArray` com os detalhes das plantas.
- Cada objeto planta contém as categorias, nome das propriedades, URL da imagem, descrição e custo.

2. Exibir os Detalhes da Planta dentro da tag `div` com o nome da classe **product-grid**.

- Utilize métodos de array para mapear sobre o array de plantas.

    Dica: use o método `map()` para iterar o array.

- Renderize os detalhes de cada planta na página, incluindo nome, imagem, descrição e custo.

3. Exibir um botão **Adicionar ao Carrinho** para cada planta.

▶ Solução para exibir plantas e botão de adicionar ao carrinho

- Inclua o código acima dentro da classe chamada **product-grid**.

4. Crie uma variável chamada `addedToCart` para gerenciamento de estado usando o **useState** hook para rastrear quais produtos foram adicionados ao carrinho.

▶ Solução de exemplo para o hook useState

5. Funcionalidade de Adicionar ao Carrinho

- Crie a função `handleAddToCart` para implementar a funcionalidade de adicionar uma planta ao carrinho quando o usuário selecionar o botão **Adicionar ao Carrinho**. Esta função deve receber um parâmetro que contém as informações da planta selecionada. Essas informações devem ser despachadas para o addItem dentro do componente de função `CartSlice`.

- Além disso, reflita que o produto foi adicionado ao carrinho. Atualize o estado `setAddedToCart` definindo o nome do produto como chave e seu valor como verdadeiro.

▶ Solução de exemplo para adicionar ao carrinho

Nota: Certifique-se de que você `importou` o `addItem` reducer de `CartSlice.jsx`

6. A função `handleAddToCart()` carregará os detalhes daquela planta que o usuário deseja adicionar ao carrinho. E os detalhes da planta para o carrinho em um nível global usando `CartSlice.jsx`.

7. Certifique-se de salvar essas alterações enviando seu código para o seu repositório do GitHub.

Tarefa 2: Gerenciamento de estado usando Redux

1. Você tem o layout básico no arquivo `CartSlice.jsx`.

2. Defina Funções Redutoras

- Agora implemente a propriedade redutora do slice para adicionar, remover e atualizar o número de itens no carrinho.

- Essas funções redutoras serão chamadas quando o usuário quiser adicionar ou remover a quantidade de plantas dentro do componente `cartItems`.

- A redutora `addItem()` adiciona um novo item de planta ao array `items` que você inicializou na etapa anterior.

- A função `addItem()` deve ser chamada quando o usuário selecionar um **Adicionar ao carrinho** na página de listagem de plantas. Subsequentemente, a `handleAddToCart()` é chamada, que tem o tipo de planta como parâmetro.
- A função `handleAddToCart()` então despacha os detalhes da planta para a função redutora `addItem()` em `CartSlice.jsx`.

▶ solução de exemplo da redutora addItem()

- Agora você precisa completar o código para os reducers `removeItem()` e `updateQuantity()`.
- `removeItem()`: Este reducer remove um item do carrinho com base em seu nome e é chamado quando o usuário deseja remover produtos do carrinho.

▶ solução de exemplo do reducer removeItem()

- `updateQuantity()`: Para criar esta função, comece extraindo o nome e a quantidade do item do `action.payload`. Em seguida, procure o item no array `state.items` que corresponda ao nome extraído. Se o item for encontrado, atualize sua quantidade para a nova quantidade fornecida no payload. Isso garante que a quantidade do item seja atualizada corretamente com base na ação.

▶ solução de exemplo do reducer updateQuantity()

3. Manipular Ações

- Exporte os criadores de ações, `addItem()` para usar em `ProductList.jsx`, `removeItem()`, e `updateQuantity()`, para usar em `CartItem.jsx`.
- Também exporte o redutor como o padrão para usar em `store.js`.

4. Certifique-se de salvar essas alterações enviando seu código para o seu repositório do GitHub.

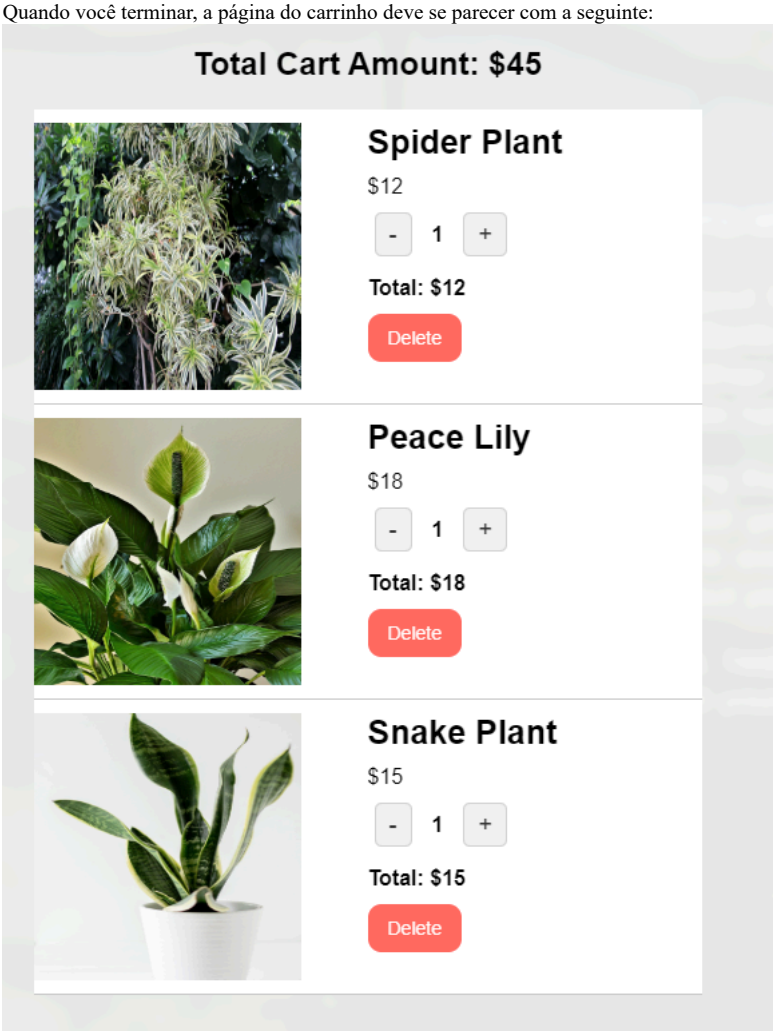
Tarefa 3: Componente CartItems

Em seguida, você completará o desenvolvimento do componente `CartItem.jsx`, que exibe os itens contidos no carrinho de compras. Este componente possui várias funcionalidades que você encontra em um carrinho de compras típico:

- Calcular o total de todos os itens no carrinho.
- Calcular o subtotal para cada tipo de planta no carrinho.
- Continuar comprando
- Incrementar e decrementar a quantidade de cada tipo de planta no carrinho
- Remover (excluir) um tipo de planta do carrinho completamente.

Você irá despachar os detalhes de incremento, decremento e atualização de quantidade de um arquivo Redux.

Quando você terminar, a página do carrinho deve se parecer com a seguinte:



1. Custo de todos os itens no carrinho

- Na função `calculateTotalAmount()`, você precisa de uma função para calcular o custo de todos os itens no carrinho. Existem várias maneiras de calcular isso.

▶ Descrição do algoritmo de exemplo para calcular o custo total

2. Continuar comprando

- Os usuários devem ser capazes de retornar à página de listagem de plantas para continuar comprando enquanto estão na página do carrinho de compras. Portanto, na função `handleContinueShopping()`, chame a função passada do componente pai.

3. Finalizar compra

- Neste projeto, você não é obrigado a fornecer a função `handleCheckoutShopping()`, mas pode querer fazê-lo para exploração e prática adicionais. Por enquanto, adicione o seguinte código para alertar o usuário de que essa função será adicionada em uma data posterior.

```
const handleCheckoutShopping = (e) => {
  alert('Functionality to be added for future reference');
};
```

4. Incrementar e decrementar

- Para as funções `handleIncrement()` e `handleDecrement()`, você precisa despachar o redutor `updateQuantity()` no arquivo `CartSlice.jsx`. No argumento da função, adicione um ao valor de `item.quantity` ou subtraia um, respectivamente.
- Além disso, para o `handleDecrement()`, você precisará de um if-else para lidar com o caso em que o número de itens é decrementado para 0. Nesse caso, você precisará despachar o método removeItem().

5. Remover planta do carrinho

- Para a função `handleRemove()`, você precisa despachar o método `removeItem()`.

6. Subtotal do item

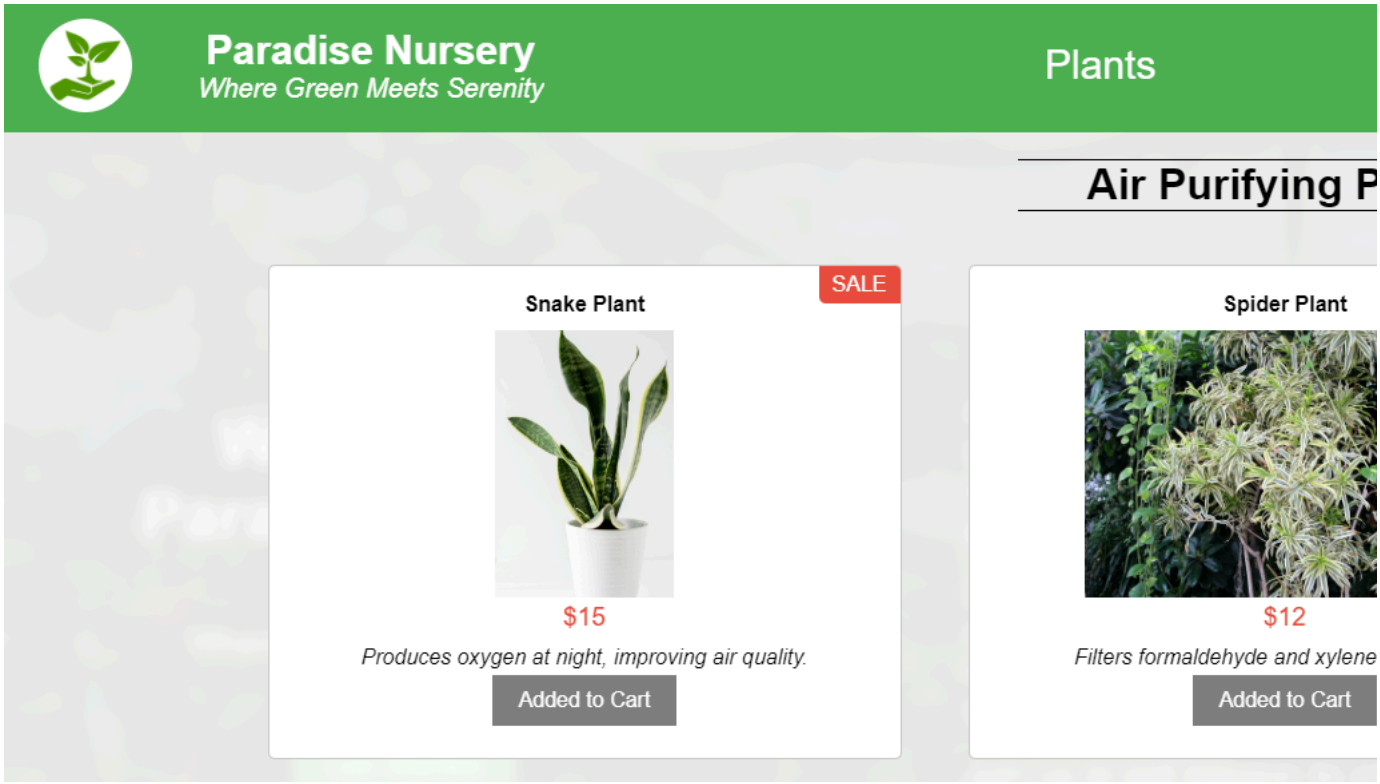
- Calcule o custo total do número de plantas de um tipo específico com a função `calculateTotalCost()` multiplicando o custo unitário de uma planta pelo número desse tipo de planta no carrinho.

7. Manipulação de Eventos

- Implemente manipuladores de eventos para incrementar e decrementar a quantidade do item no carrinho.
- Quando o usuário altera o número de um tipo de planta no carrinho, os seguintes dados precisam ser atualizados: o ícone do carrinho, o número daquele tipo de planta, o subtotal e o custo total.
- Implemente um manipulador de eventos para remover o item do carrinho.

8. Contador de Quantidade Total

- Mantenha uma variável dedicada a contar o número total de itens adicionados ao carrinho.
- Atualize essa variável à medida que o usuário adiciona ou remove plantas do carrinho.
- Exiba a quantidade total no ícone do carrinho na barra de navegação.



9. Certifique-se de salvar essas alterações enviando seu código para o seu repositório do GitHub.

Tarefa 4: Integrar a funcionalidade Redux em seus componentes

1. Componente ProductList
    - Inicialize o estado dos itens do carrinho no armazenamento Redux.

- Despache a ação `addItem` para adicionar itens ao carrinho.
- Recupere a quantidade de todos os itens no carrinho do armazenamento Redux.

2. Componente CartItem

   - Despache a ação `updateQuantity` para atualizar a quantidade do item do carrinho.
   - Despache a ação `addItem` para adicionar o item do carrinho.
   - Despache a ação `removeItem` para remover o item do carrinho.

Nota: Certifique-se de salvar essas alterações enviando seu código para seu repositório GitHub

Tarefa 5: Importar detalhes para store.js

1. Importando Funções e Arquivos Necessários:

   - A função `configureStore()` do pacote `@reduxjs/toolkit` é importada para configurar o store do Redux.
   - O `cartReducer` do arquivo `CartSlice.jsx` que é importado, gerencia a fatia de estado relacionada ao carrinho de compras.

```
import { configureStore } from '@reduxjs/toolkit';
import cartReducer from './CartSlice';
```

2. Configurando a Loja:

   - A função `configureStore()` é usada para configurar a loja Redux.
   - Dentro do objeto de configuração passado para `configureStore()`, a chave `reducer` especifica as funções redutoras. Neste caso, o `cartReducer` é atribuído para gerenciar o trecho `cart` do estado.

```
const store = configureStore({
    reducer: {
        cart: cartReducer,
    },
});
```

3. Exportando a Loja:

   - A loja Redux configurada é exportada usando `export default store;`, para que possa ser utilizada em toda a aplicação para gerenciar o estado.

```
export default store;
```

Nota:
- Este código está pré-configurado no repositório e pronto para uso.
- Certifique-se de salvar essas alterações enviando seu código para o seu repositório do GitHub

Tarefa 6: Configurar o armazenamento global

1. Navegue até o arquivo `main.jsx` na pasta `src`.

2. O componente `Provider` da biblioteca `react-redux` já está importado. Este componente permite que todos os componentes da aplicação acessem o armazenamento Redux.

```
import { Provider } from 'react-redux';
```

3. A loja Redux é importada do arquivo `store.js`. Esta loja mantém o estado da aplicação, utilizando o redutor definido no arquivo `CartSlice.jsx`.

```
import store from './store.js';
```

4. O componente `App` é envolvido pelo componente `Provider`, com a loja Redux passada como uma prop. Isso permite que todos os componentes do aplicativo acessem e interajam com o estado global gerenciado pelo Redux.

```
<Provider store={store}>
  <App />
</Provider>
```

Nota:
- Este código está pré-configurado no repositório e pronto para uso.
- Certifique-se de salvar essas alterações enviando seu código para o seu repositório do GitHub

Conclusão do Projeto

Parabéns!

Você completou este projeto! Ótimo trabalho.

Se ainda não o fez, implante sua aplicação. Você pode implantá-la usando o GitHub Pages ou seu próprio site de hospedagem. Anote a URL da aplicação, você precisará disso para enviar seu projeto para revisão por pares.

As instruções para implantar sua aplicação usando o GitHub Pages estão encontradas no último passo deste laboratório, *Configurando o Ambiente do GitHub*, anteriormente neste curso.

A implantação de sua aplicação React é importante, pois seu colega irá revisar seu trabalho com base nisso.

Resumo

- Você criou componentes funcionais React, os compôs e aninhou.
- Você implementou React Hooks, especificamente os hooks `useState` e `useEffect` para gerenciar estados em nível de componente.

- Você integrou o Redux em sua aplicação React e aplicou conceitos do Redux como ações, reducers e a store.
- Você renderizou dinamicamente dados buscados de um array de objetos na interface do usuário. Você mapeou arrays para gerar listas de componentes.
- Você lidou com eventos de usuário, como seleção de botões, e acionou ações correspondentes para adicionar e remover itens no carrinho.

Autor

Richa Arora