Plataformas de aplicações Web

Aula 13 - PetTopStore - PDV - Parte 2

Material Didático do Instituto Metrópole Digital - IMD

Termo de uso

Apresentação

Nesta aula vamos finalizar o sistema do PDV (Ponto de venda) da solução da loja Pet Top Store apresentando os recursos de adicionar cliente e realizar vendas.

Baixe os projetos

Na aula anterior fizemos o recurso de Login do PDV, mas disponibilizamos ao final o sistema completo. O segue o link caso precise baixar novamente: Clique aqui para download

Segue também o sistema administrativo/api completo: Clique aqui para download

Falta apresentar os recursos de realizar venda e adicionar cliente. Eles estão nos arquivos src/Sale.js e src/AddClient.js.

PDV - Realizar Vendas (src/Sale.js)

Antes de mais nada é importante lembrar que você pode executar o projeto a qualquer momento com o comando:

```
npm start
```

Pode ser usado o yarn start também, tanto faz.

O componente Sale, implementado em src/Sale.js é responsável por criar uma nova venda para um cliente na loja física. Nele você pode selecionar um cliente de uma lista, selecionar produtos e inserir em uma lista de itens da venda e finalizar a venda para o cliente.

Trata-se do componente mais complexo do PDV. Ele realiza várias operações e chamadas na API para cada uma de suas operações.

Como você deve lembrar, o componente principal (src/App.js) verifica se existe um employee logado e se existir ele retorna o Sale como componente que será exibido, ou seja, quando o usuário já está logado, o Sale é o que será exibido e visualmente ele é da como na imagem abaixo:

Vamos analisar cada parte.

Primeiro se tem um título com o nome PetTopStore - PDV

Depois existe o texto "Logado como "João Vendedor". Esse é o nome do employee logado que o App.js gerencia e que foi passado como prop (atributo) para o Sale, que mostra o seu nome.

O Botão "Sair do sistema", quando clicado, realiza uma operação super simples: Ele executa a função "props.setEmployee(null)". Essa função está implementada em App.js, e foi passada para o Sale por atributo (props). Efetivamente o que acontece é que o employee logado é setado para nulo e então o React reage a essa mudança e o App.js não exibe mais o componente Sale, e sim o Login.

Logo após temos uma seção chamada "Nova venda"

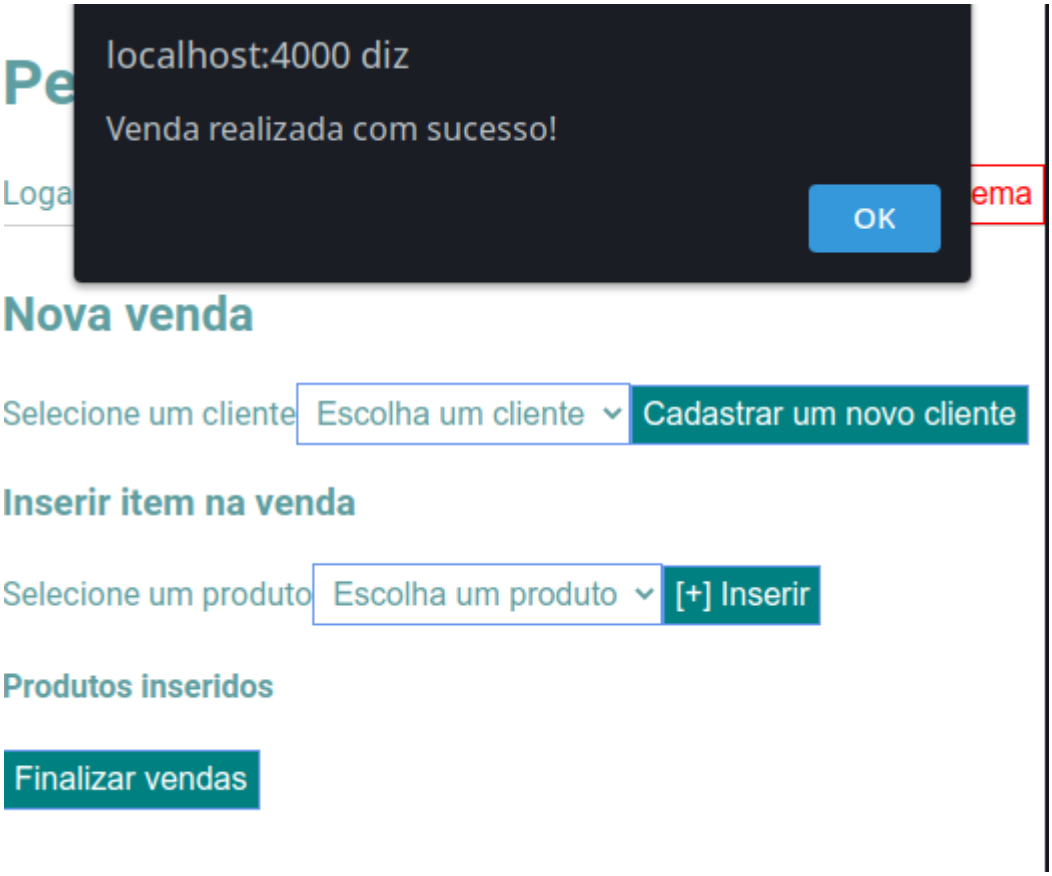
Essa seção tem inicialmente um Texto "Selecione um cliente", seguido de um SELECT com a lista de clientes cadastrados que veio da API. Esse cliente selecionado será o que será o cliente da venda que o funcionário está cadastrando.

Logo depois temos um botão chamado "Cadastrar um novo cliente". Esse botão faz com que outro componente (AddClient) seja exibido com um simples formulário de cadastro de cliente. É passado para o AddClient uma função que ele pode chamar para dizer que um novo cliente foi cadastrada(clientCreated) e outra para cancelar a operação(cancelCreateClient). Quando um cliente é cadastrado com sucesso pelo AddClient, ele fica automaticamente como o cliente selecionado no select de cliente Em seguida temos uma seção chamada "Inserir item na venda". Basicamente existe

um SELECT com uma lista de produtos. O funcionário deve selecionar um produto e clicar em "[+] Inserir" O que faz com que esse produto entre na lista de produtos que fazem parte da venda atualmente sendo cadastrada. Vale lembrar que essa inserção de itens na venda ainda não foi persistida no banco de dados pois ela acontece somente no navegador, que vai exibindo a lista com os nomes dos produtos. Ao se inserir alguns produtos em "produtos inseridos" o formulário fica da seguinte forma:

Veja que no estado atual do componente Sale na imagem acima, existe um cliente selecionado chamado Amanda e 2 produtos na lista de produtos inseridos. A variável insertProductIDs é que guarda os IDs dos produtos que estão nessa listagem.

Quando se clica em "Finalizar venda" o cliente selecionado e os IDs dos produtos selecionados são enviados para a API para se criar uma nova venda (sale) e um logo em seguida é apresentada uma mensagem de sucesso e o formulário é limpo para a próxima venda:



Conteúdo comentado do src/Sale.js e src/Sale.css

```
import './Sale.css'; // Css de estilização
import { useState, useEffect } from 'react'; // React Rooks
import AddClient from './AddClient'; // importado o componente
AddClient que é responsável por se criar clientes

// Componente de realizar vendas (Sale)
function Sale(props) {
  // ** variáveis de estado **

  // lista de clientes para selecionar na venda
  const [clients, setClients] = useState([]);

  // lista de produtos disponíveis
  const [products, setProducts] = useState([]);

  // ID do cliente selecionado que será utilizado para cadastrar a
venda
  const [selectedClientID, setSelectedClientID] = useState('');
```

```
  // ID do produto atualmente selecionado que será adicionado ao
insertProductIDs
  const [selectedProductID, setSelectedProductID] = useState('');

  // Array com os IDs dos produtos que serão inseridos nos itens da
venda.
  const [insertProductIDs, setInsertProductIDs] = useState([]);

  // Variável boleana que determina se o formulário de criar novo
cliente deve ser exibido
  const [addClient, setAddClient] = useState(false);

  //função que "limpa" o formulário.
  function clearForm() {
    // limpa o clienteId selecionado
    setSelectedClientID('');
    // limpa o productId selecionado
    setSelectedProductID('');
    // limpa o array de productIDs (itens da venda)
    setInsertProductIDs([]);
  }

  // função que carrega da API a lista de clientes e guarda em
"clients"
  function loadClients() {
    // busca lista de clientes da api, passando o
"props.employee.token" como token JWT no cabeçalho da requisição
    return fetch('http://localhost:3000/api/clients', {
      method: 'GET',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${props.employee.token}`
      }
    }).then( (res) => {
      res.json().then(json => {
        // setar clientes para a lista retornada pela API
        setClients(json.clients);
      });
    });
  }

  // função que carrega a lista de produtos da API
  function loadProducts() {
    // realiza a busca de produtos da API, passando o
```

```
"props.employee.token" como token JWT no cabeçalho da requisição
    fetch('http://localhost:3000/api/products/search', {
      method: 'GET',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${props.employee.token}`
      }
    }).then(res => {
      if (res.ok) {
        res.json().then(json => {
          // setar a lista de produtos disponível para a lista
retornada pela API
          setProducts(json.products);
        })
      }
    })
  }

  // Função que é passada para o AddClient executar quando um
cliente é criado.
  // Ela deve ser passada par ao component AddClient e só é chamada
se o usuário não cancelar a operação e efetivamente criar o cliente
  function clientCreated(clientID) {
    // como um novo cliente foi adicionado, atualizar a lista de
clientes com loadClients()
    loadClients()
    .then(() => {
      // Quando o componente AddCliente executar essa função, ele
passará o ID do cliente que ele criou pela API no parâmetro.
      // Isso permite que possamos definir que o cliente selecionado
no SELECT será o que acabou de ser adicionado
      setSelectedClientID(clientID);
      // seta addCliente para falso, para que o componente AddClient
não seja mais exibido (já que seu trabalho foi finalizado)
      setAddClient(false);
    });
  }

  // função que o componente AddClient executar caso o usuário
cancele a operação
  // Somente seta addCliente para false, ou seja, a interfave vai
reagir e esconder, já que sua exibição depende desse boleano ser
'true'
  function cancelCreateClient() {
```

```
    setAddClient(false);
  }

  // Effect Hook que carreta os clientes e produtos assim que o
formulário iniciar (já que addClient mudará de valor) ou se ele
mudar de valor posteriomente.
  // Isso garante que o componente iniciará com os dados de clientes
e produtos carregados
  useEffect( () => {
    loadClients();

    loadProducts();
  }, [addClient]);

  if (addClient) {
    // caso a variável addClient seja true, exibir o componente de
adicionar cliente (AddClient)
    // É passado par ao componente AddCliente:
    // * employee logado
    // clientCreated que é a função que será chamada quando se criar
um cliente (essa função está implementada nesse componente (Sale.js)
mais acima)
    // cancelCreateClient que é a função que será chamada quando se
cancelar a criação de um cliente (essa função está implementada
nesse componente (Sale.js) mais acima)
    return <AddClient employee={props.employee}
clientCreated={clientCreated}
cancelCreateClient={cancelCreateClient} />
  }

  //Caso  addClient seja 'false' chega aqui, ou seja será retornado
o furmulário de adicionar nova venda.
  return (
    <div className="Sale">
      <h1>PetTopStore - PDV</h1>
      <div className="UserBox">
        <div>
          {/* Mostra o nome do funcionário logado (que foi passado
por atributo (props)) */}
          Logado como {props.employee.name}
        </div>

        {/* Botão de sair do sistema. Simplesmente chama
props.setEmployee(null), removendo a informação do usuário logado.
```

```
        essa função setEmployee foi passada para Sale por atributo
props pelo App.js
        */}
        <button
          className="logoutButton"
          onClick={e => props.setEmployee(null)}
        >
          Sair do sistema
        </button>
      </div>

      {/* Formulário de adicionar nova venda */}
      <h2>Nova venda</h2>
      <div>
        Selecione um cliente
        {/* Select com o valor associado a selectedClientId, ao
mudar troca o selectedClientId,

        os options são mapeados a partir da lista de clietnes
disponíveis carregada da API (clients)
        */}
        <select value={selectedClientID} onChange={(e) => {
            setSelectedClientID(e.target.value);
          }}>
          <option value="">Escolha um cliente</option>
          {clients.map((client) =>
            <option key={client.id}
value={client.id}>{client.name}</option>
          )}
        </select>

        {/* Se clicar em "Cadastraor novo cliente" é setada
addCliente para true, o que faz o componente correspondente aparecer
*/}
        <button
          className="success"
          onClick={() => {
            setAddClient(true);
          }}
        >
          Cadastrar um novo cliente
        </button>

        <h3>Inserir item na venda</h3>
```

```
        <div>
          Selecione um produto
          {/* select com os produtos da API. Ao se selecionar ele
muda selectedProductID */}
          <select value={selectedProductID} onChange={(e) => {
            setSelectedProductID(e.target.value);
          }}>
            <option value="">Escolha um produto</option>
            {/* Uma opção para cada produto disponível que veio da
API */}
            {products.map((product) =>
              <option key={product.id}
value={product.id}>{product.name}</option>
            )}
          </select>

          {/* Botão para se inserir o selectedProductID (associado
ao select acima) em productIDs que representa os itens da venda */}
          <button
            className="success"
            onClick={() => {
              if (selectedProductID) {
                setInsertProductIDs([
                  ...insertProductIDs,
                  selectedProductID
                ]);
                setSelectedProductID('');
              }
          }}>
            [+] Inserir
          </button>
          <br/>

          <h4>Produtos inseridos</h4>
          <ul>
            {/* Mostra reativamente os produtos escoilhidos, para
cada item em insertProductIDs, criar um LI com nome*/}
            {insertProductIDs.map((productID) => {
              const product = products.find(p => p.id ===
parseInt(productID));
              return <li key={product.id}>{product.name}</li>
            })}
          </ul>
```

```
          <button
            className="success"
            onClick={() => {
            // Tenta finalizar a venda (efetivar na API)

            // valida se existe cliente selecionado
            if (selectedClientID === '') {
              alert('Selecione um cliente');
              return;
            }

            // verificar se existe algum produto em adicionado em
insertProductIDs
            if (insertProductIDs.length === 0){
              alert('Adicione pelo menos um produto');
              return;
            }

            // cria nova venda na API passando o ID do cliente
selecionado (selectedClientID) e a lista de IDs de produtos
(insertProductIDs)
            fetch('http://localhost:3000/api/sales',{
              method: 'POST',
              headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${props.employee.token}`
              },
              body: JSON.stringify({
                client_id: selectedClientID,
                productIDs: insertProductIDs
              })
            }).then(res => {
              if (res.ok) {
                res.json().then(json => {
                  alert('venda ok!');
                })
              } else {
                alert('venda not ok');
              }
              clearForm();
            })
          }}>
            Finalizar vendas
          </button>
```

```
        </div>

      </div>
    </div>
  );
}

export default Sale;
```

Conteúdo do src/Sale.css (estilização):

```
.Sale
{
  display: flex;
  width: 100%;
  justify-content: center;
  flex-direction: column;
  align-items: stretch;
  color: cadetblue;
}

.UserBox {
  display: flex;
  border-bottom: 1px solid lightgray;
  margin-bottom: 10px;
  flex-direction: row;
  justify-content: space-between;
  align-items: center;
}

.logoutButton {
  padding: 5px;
  color: red;
  border-color: red;
}

input, select, option, button {
  color: cadetblue;
  background-color: white;
  border: 1px solid cornflowerblue;
  padding: 5px;
  font-size: 1em
}
```

```
input.success, select.success, option.success, button.success {
  color: white;
  background-color: teal;
  border: 1px solid cornflowerblue;
  padding: 5px;
  font-size: 1em
}
```

PDV - Adicionar Cliente (src/AddClient.js)

O componente AddClient adiciona um novo cliente usando a API e é exibido somente quando se escolhe essa opção no componente Sale.

Ele recebe por "props" o employee logado (necessário para se fazer requisições a api com o token JWT dele), recebe também a função clientCreated e cancelCreateClient que são ambas implementadas em Sale, mas são chamadas pelo componente AddClient, quando um cliente é criado com sucesso e quando o usuário cancela a criação do cliente.

Antes de ver o código vamos ver o visual do componente AddCliente quando se clica em "Cadastrar um novo cliente" em Sale:

Veja que é um componente simples, basicamente tem 3 inputs relacionados às variáveis de estado name, email e password. Quando se clica em "Salvar cliente" é feito o cadastro pelo acesso a API que retorna o ID do cliente cadastrado, e depois é executada a função props.clientCreated(json.clienteID), que foi passada pra ele pelo Sale. Essa função é chamada com o ID do cliente cadastrado como parâmetro para que o componente Sale (onde ele está implementado) consiga já automaticamente selecionar o novo cliente cadastrado no SELECT para se realizar uma nova venda.

O botão cancelar executa props.cancelCreateCliente() que somente some com o AddClient e volta para o Sale com o estado anterior mantido.

Conteúdo de src/AddClient.js:

Segue o conteúdo de src/AddClient.js

```
import { useState } from 'react';

// Componente que cria novos clientes
function AddClient(props) {

  // ** variáveis de estado **
  // nome do novo cliente
  const [name, setName] = useState('');
  // email do novo cliente
  const [email, setEmail] = useState('');
  // senha do novo cliente
  const [password, setPassword] = useState('');

  // função que salva novo cliente na API
  function salvarCliente() {
    // tenta salvar o cliente na API com o uso do fetch enviando o
nome, email e senha
    // é passado no cabeçalho o "props.employee.token" que é o token
JWT do employee (funcionário logado)
    fetch('http://localhost:3000/api/auth/client/registration', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${props.employee.token}`
      },
      body: JSON.stringify({ name, email, password })
    }).then(res => {
      if (res.ok) {
        alert('cliente salvo com sucesso');
        res.json().then(json => {
          // caso o cliente seja cadastrado na API com sucesso
          // é executada a função clientCreated passando por
parâmetro o ID do cliente cadastrado que retornou da API
          // essa função está dentro de props, ou seja, ela foi
passada para esse componente (AddCliente) mas está implementada em
Sale
          props.clientCreated(json.clienteID)
        });
      } else {
```

```
        alert('falha ao salvar cliente');
        // Ao falhar a criação do cliente informa ao Sale que o
processo está cancelado
        // e então esse componente irá sumir da tela
        props.cancelCreateClient();
      }
    })
  }

  // retorna um JSX com um formulário com o nome/email/senha do novo
cliente associado às variáveis relacionadas
  return (
    <div>
      <h2>Novo cliente</h2>
      <div>
        Nome: <input name="name" value={name} onInput={(e) => {
          setName(e.target.value);
        }}/><br/>
        E-Mail: <input name="email" value={email} onInput={(e) => {
          setEmail(e.target.value);
        }}/><br/>
        Senha: <input name="password" type="password"
value={password} onInput={(e) => {
          setPassword(e.target.value);
        }}/><br/>
        <button onClick={salvarCliente}>
          Salvar cliente
        </button>
        <button onClick={() => {
          // Se desistir do cadastro informar ao Sale que foi
cancelado chamando props.cancelCreateClient()
          // essa função foi passada como atributo da para o
AddCliente e está implementada em Sale
          props.cancelCreateClient()
        }}>
          Cancelar
        </button>
      </div>
    </div>
  );
}

export default AddClient;
```

Conclusão

Pronto. Criamos nosso Ponto de Venda (PDV) do projeto PetTopStore. O sistema foi criado como um SPA(Single Page Application) utilizando React. É importante ressaltar novamente que somente usamos recursos simples do React e que essas duas aulas não são um curso completo da tecnologia que hoje é uma das mais usadas no mundo. É possível fazer muito mais, não só com o React, mas como também com o próprio sistema do PDV que é passível de muitas melhorias que você pode fazer como exercício.

Duas sugestões somente para você iniciar: Implementar mais validações para campos que não podem ficar em branco, Melhorar as mensagens de erro e sucesso para que não sejam simples Alerts.

Espero que tenham notado o como é interessante separar sistemas grandes, como o de uma loja, em sistemas menores, porém integrados por uma camada de dados, como é nossa API. Seria possível criar o PDV em qualquer outra tecnologia/linguagem e mesmo assim a integração seria a mesma, assim como inclusive outras equipes poderiam manter o projeto da API e do PDV de forma independente, respeitando somente como a comunicação ocorre entre eles.

Nas próximas aulas já vamos para o último sistema, a Loja on-line que será implementado usando o Svelte, um outro Framework Front-end reativo e também integrado com nossa API.