Sumário

Introdução

Componentes do React.js

- **App.js**
- **Home.js**
- **Cart.js**
- **Opencoffee.js**
- **AdminPage.js**
- **CartContext.js**
- **FilterContext.js**

Backend

- **imageapi.py**
- **images.py**
- **posvendas.py**
- **start.sh**
- **server.js**

Docker e Docker Compose

- **Dockerfile-python**
- **Dockerfile-node**
- **docker-compose.yml**

Configuração e Instalação

Conclusão

Introdução

Este documento fornece uma descrição detalhada dos componentes principais do nosso projeto de e-commerce, desenvolvido utilizando React.js, Node.js e Python. A seguir, descrevemos os componentes, suas funcionalidades e como interagem entre si para fornecer uma experiência de usuário rica e responsiva.

Header.js

```
import React, { useContext } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import logo from '../images/logo.webp';
import cartIcon from '../images/cart-icon.png';
import headlineImage from '../images/headline.jpeg';
import { FilterContext } from '../FilterContext';
import { CartContext } from '../CartContext';

const Header = () => {
 const { setFilter } = useContext(FilterContext);
 const navigate = useNavigate();
 const { cartItems } = useContext(CartContext);

 const handleFilterClick = (category) => {
  setFilter(category);
  navigate('/');
 };

 return (
  <div>
   <header className="relative text-white flex justify-center items-center bg-cover bg-center" style={{ backgroundImage: `url(${headlineImage})`, height: '280px' }}>
    <div className="absolute top-0 w-full h-20 bg-black bg-opacity-50"></div>
    <div className="absolute left-5 top-3">
     <img src={logo} alt="Indaia Eventos" className="w-36" />
    </div>
    <div className="font-bold absolute top-[60%] left-[50%] transform -translate-x-1/2 -translate-y-1/2 text-center">
     <h1 className="text-4xl">Área do Cliente</h1>
    </div>
    <div className="absolute right-5 top-5 flex flex-row items-center space-x-2">
     <button className="voltar-btn bg-white bg-opacity-20 text-white border border-white py-2 px-4 hover:bg-green-600 transition duration-300">
      Voltar à Área do Cliente
     </button>
     <button className="bg-white bg-opacity-20 text-white border border-white py-2 px-4 hover:bg-green-600 transition duration-300">
      Sair
     </button>
     <Link to="/cart" className="bg-white bg-opacity-20 text-white border border-white py-2 px-4 hover:bg-green-600 transition duration-300 flex items-center">
      <img src={cartIcon} alt="Carrinho de Compras" className="w-6 mr-2" />
```

```
        <span className="absolute top-0 right-0 bg-red-600 text-white rounded-full px-2 text-xs">
         {cartItems.length}
        </span>
        </Link>
      </div>
    </header>
    <section className="filter-title-container flex justify-center items-center py-4 bg-white shadow-md">
      <div className="filters-container flex flex-wrap justify-center space-x-2 sm:space-x-4">
       {['todos', 'cardápio', 'serviços', 'cerimonias', 'decoração', 'iluminação', 'contratados'].map(category => (
         <button
          key={category}
          className="filter-btn bg-green-500 text-white px-2 py-1 sm:px-4 sm:py-2 rounded-full my-1"
          onClick={() => handleFilterClick(category)}
         >
          {category.charAt(0).toUpperCase() + category.slice(1)}
         </button>
       ))}
      </div>
    </section>
   </div>
  );
};

export default Header;
```

Descrição:

O componente Header é responsável por renderizar o cabeçalho da página, incluindo o logo, o nome da área do cliente, os botões de navegação e o ícone do carrinho de compras. Ele utiliza o contexto FilterContext para definir filtros de categoria e CartContext para obter o número de itens no carrinho.

ProductDetails.js

```
import React, { useState, useEffect } from 'react';
import { useParams } from 'react-router-dom';
import axios from 'axios';

const ProductDetails = () => {
 const { productId } = useParams();
 const [ProductComponent, setProductComponent] = useState(null);
 const [productData, setProductData] = useState(null);
 const [error, setError] = useState(null);

 useEffect(() => {
  // Carregar o componente específico do produto
  import(`../pages/products/${productId}.js`)
   .then(module => setProductComponent(() => module.default))
   .catch(error => {
    console.error(`Erro ao carregar o componente para ${productId}:`, error);
    setError('Erro ao carregar o componente do produto.');
   });

  // Carregar os dados do produto
  axios.get(`https://loja.eventosindaia.com.br/api/products/${productId}`)
   .then(response => {
    if (response.data.active) {
     setProductData(response.data);
    } else {
     console.error('Produto inativo');
     setError('Produto inativo.');
    }
   })
   .catch(error => {
    console.error('Erro ao buscar detalhes do produto:', error);
    setError('Erro ao buscar detalhes do produto.');
   });
 }, [productId]);

 if (error) return <div>{error}</div>;
 if (!ProductComponent || !productData) return <div>Loading...</div>;

 return (
  <div className="p-5">
   <ProductComponent productData={productData} />
  </div>
 );
```

```
};
export default ProductDetails;
```

Descrição:

O componente ProductDetails é responsável por renderizar os detalhes de um produto específico. Ele carrega dinamicamente o componente e os dados do produto com base no productId presente na URL.

ProductItem.js

```
import React from 'react';
import { Link } from 'react-router-dom';

const ProductItem = ({ product }) => (
  <div className="product-item" data-category={product.category}>
    <img src={product.image} alt={product.name} />
    <h3>{product.name}</h3>
    <Link to={`/products/${product.id}`}>Veja detalhes</Link>
  </div>
);

export default ProductItem;
```

Descrição:

O componente ProductItem é responsável por renderizar as informações de um produto individual na lista de produtos. Ele exibe a imagem, o nome do produto e um link para a página de detalhes do produto.

ProductList.js

```
import React, { useContext, useState, useEffect } from 'react';
import axios from 'axios';
import ProductItem from './ProductItem';
import { FilterContext } from '../FilterContext';
import { useLocation } from 'react-router-dom';

const ProductList = () => {
 const { filter, contractedProducts, setContractedProducts } = useContext(FilterContext);
 const [products, setProducts] = useState([]);
 const location = useLocation();

 useEffect(() => {
  axios.get('https://loja.eventosindaia.com.br/api/products')
   .then(response => {
    setProducts(response.data.filter(product => product.active));
   })
   .catch(error => {
    console.error('Error fetching products:', error);
   });
 }, []);

 useEffect(() => {
  const query = new URLSearchParams(location.search);
  const cliente = query.get('cliente');
  if (cliente) {
   axios.get(`https://loja.eventosindaia.com.br/data`)
    .then(response => {
     const contracted = response.data.find(event => event['Cód'] === cliente);
     if (contracted) {
      const contractedItems = Object.keys(contracted).filter(key => contracted[key] === 1);
      setContractedProducts(contractedItems);
      console.log('Contratados:', contractedItems); // Log para verificação
     } else {
      console.log('Nenhum evento encontrado para o código:', cliente);
     }
    })
    .catch(error => {
     console.error('Error fetching contracted data:', error);
    });
  }
 }, [location.search, setContractedProducts]);
```

```
  const filteredProducts = filter === 'todos'
    ? products
    : filter === 'contratados'
      ? products.filter(product => contractedProducts.includes(product.name))
      : products.filter(product => product.category === filter);

  return (
    <section className="product-list grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-
4 p-4">
      {filteredProducts.map(product => (
        <ProductItem key={product.id} product={product} />
      ))}
    </section>
  );
};

export default ProductList;
```

Descrição:

O componente ProductList é responsável por renderizar a lista de produtos. Ele busca os produtos da API, aplica filtros com base na categoria selecionada e exibe os produtos filtrados.

ProductPage.js

```
import React, { useState, useEffect } from 'react';
import { useParams } from 'react-router-dom';
import axios from 'axios';
import '../styles.css';

const ProductPage = () => {
 const { productId } = useParams();
 const [product, setProduct] = useState(null);

 useEffect(() => {
  axios.get(` https://loja.eventosindaia.com.br/api/products/${productId}` )
   .then(response => setProduct(response.data))
   .catch(error => console.error('Erro ao buscar produto:', error));
 }, [productId]);

 if (!product) return <div>Loading...</div>;

 return (
  <div className="product-page">
   <h1>{product.name}</h1>
   <img src={product.image} alt={product.name} />
   <p>{product.description}</p>
   {/* Renderize outras informações do produto conforme necessário */}
  </div>
 );
};

export default ProductPage;
```

Descrição:

O componente ProductPage é responsável por renderizar a página de um produto específico, exibindo suas informações detalhadas.

App.js

```
import React from 'react';
import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';
import Header from './components/Header';
import ProductList from './components/ProductList';
import ProductDetails from './components/ProductDetails'; // Use ProductDetails se
você precisa de componentes dinâmicos
import Cart from './pages/cart';
import AdminPage from './pages/admin/AdminPage'; // Adicionado
import { CartProvider } from './CartContext';
import { FilterProvider } from './FilterContext';
import './styles.css';

const App = () => {
 return (
  <CartProvider>
   <FilterProvider>
    <Router>
     <Header />
     <Routes>
      <Route path="/" element={<ProductList />} />
      <Route path="/products/:productId" element={<ProductDetails />} /> {/* Use
ProductDetails */}
      <Route path="/cart" element={<Cart />} />
      <Route path="/admin" element={<AdminPage />} /> {/* Adicionado */}
     </Routes>
    </Router>
   </FilterProvider>
  </CartProvider>
 );
};

export default App;
```

Descrição:

O componente App é o componente principal do aplicativo. Ele define o provedor de contexto para carrinho de compras e filtro, configura as rotas de navegação e renderiza os componentes de cabeçalho e lista de produtos.

Home.js

```
import React, { useContext } from 'react';
import Header from '../components/Header';
import ProductList from '../components/ProductList';
import { FilterContext } from '../FilterContext';

const Home = () => {
 const { filter, setFilter } = useContext(FilterContext);

 return (
  <div>
   <Header setFilter={setFilter} />
   <main className="p-4">
    <section className="filter-title-container flex justify-between items-center">
     <div className="filters-container flex space-x-4">
      {['todos', 'cardapio', 'servicos', 'cerimonias', 'decoracao', 'iluminacao',
'contratados'].map(category => (
       <button
        key={category}
        className="filter-btn bg-green-500 text-white px-4 py-2 rounded"
        onClick={() => setFilter(category)}
       >
        {category.charAt(0).toUpperCase() + category.slice(1)}
       </button>
      ))}
     </div>
     <div className="product-title">
      <h2>Banco de Produtos</h2>
      <h3 id="product-category-title">{filter.charAt(0).toUpperCase() +
filter.slice(1)}</h3>
     </div>
    </section>
    <section className="product-list">
     <ProductList filter={filter} />
    </section>
   </main>
  </div>
 );
};

export default Home;
```

Descrição:

O componente Home renderiza a página inicial, que inclui o cabeçalho e a lista de produtos. Ele utiliza o contexto FilterContext para definir e aplicar filtros de categoria.

Cart.js

```
import React, { useContext } from 'react';
import { CartContext } from '../CartContext';
import './cart.css';

const Cart = () => {
 const { cartItems, clearCart, removeFromCart } = useContext(CartContext);

 const handleContactSupport = () => {
  const whatsappNumber = '+5547988544227';
  const cartItemsMessage = cartItems.map(item => item.name).join(', ');
  const message = `Olá, gostaria de falar sobre os seguintes itens no meu carrinho: ${cartItemsMessage}`;
  const whatsappLink = `https://wa.me/${whatsappNumber}?text=${encodeURIComponent(message)}`;
  window.location.href = whatsappLink;
 };

 return (
  <div>
   <main className="p-4">
    <section className="cart-details">
     <h2>Carrinho de Compras</h2>
     {cartItems.length === 0 ? (
      <p>Seu carrinho está vazio.</p>
     ) : (
      <ul className="cart-list">
       {cartItems.map((item, index) => (
        <li key={index} className="cart-item">
         <img src={item.image} alt={item.name} className="cart-item-image" />
         <div className="cart-item-details">
          {item.name}
          <button className="remove-item-btn" onClick={() => removeFromCart(item.cartItemId)}>Remover</button>
         </div>
        </li>
       ))}
      </ul>
     )}
     {cartItems.length > 0 && (
      <div className="cart-actions">
       <button className="clear-cart-btn" onClick={clearCart}>Esvaziar Carrinho</button>
```

```
          <button                                          className="contact-support-btn"
onClick={handleContactSupport}>Falar com Atendente</button>
         </div>
        )}
      </section>
    </main>
   </div>
  );
};

export default Cart;
```

Descrição:

O componente Cart renderiza a página do carrinho de compras, exibindo os itens adicionados e permitindo ao usuário remover itens, esvaziar o carrinho ou contatar o suporte via WhatsApp.

Opencoffee.js

```
import React, { useState, useEffect, useContext } from 'react';
import axios from 'axios';
import { useNavigate } from 'react-router-dom';
import './products.css';
import MainImage from '../../images/Open Coffee/01.jpg';
import ImageGallery from 'react-image-gallery';
import 'react-image-gallery/styles/css/image-gallery.css';
import Modal from 'react-modal';
import { CartContext } from '../../CartContext';

Modal.setAppElement('#root');

const Opencoffee = () => {
  const { addToCart } = useContext(CartContext);
  const [images, setImages] = useState([]);
  const [galleryItems, setGalleryItems] = useState([]);
  const [isGalleryOpen, setIsGalleryOpen] = useState(false);
  const [isCartModalOpen, setIsCartModalOpen] = useState(false);
  const navigate = useNavigate();
  const productName = 'Open Coffee';

  useEffect(() => {

axios.get(` https://loja.eventosindaia.com.br/api/images/${encodeURIComponent(p
roductName)}` )
     .then(response => {
       const imageUrls = response.data.map(file => ({
         original: file,
         thumbnail: file
       }));

       setImages(imageUrls);
       setGalleryItems(imageUrls);
     })
     .catch(error => {
       console.error('Error fetching images:', error);
     });
  }, [productName]);

  const openGallery = (startIndex = 0) => {
    setGalleryItems(images);
    setIsGalleryOpen(true);
  };
```

```
 const closeGallery = () => {
  setIsGalleryOpen(false);
 };

 const handleAddToCart = () => {
  addToCart({ id: 'opencoffee', name: productName, image: MainImage });
  setIsCartModalOpen(true);
 };

 const closeCartModal = () => {
  setIsCartModalOpen(false);
 };

 const handleContinueShopping = () => {
  setIsCartModalOpen(false);
  navigate('/');
 };

 const handleCheckout = () => {
  setIsCartModalOpen(false);
  navigate('/cart');
 };

 return (
  <div>
   <main>
    <section className="product-details">
     <div className="product-image-container">
      <img id="product-image" src={MainImage} alt={productName} />
     </div>
     <div id="product-description">
      <p>Para os apaixonados por café, agora nossos eventos também contam com o serviço de Open Coffee! Nossa máquina de café é disponibilizada para a utilização durante o evento, onde você e todos os seus convidados poderão degustar de um delicioso cafezinho. Contamos com 7 opções de sabores, sendo eles, café suave, café forte, café c/ leite, cappuccino, mocaccino, chocolate, cappuccino e canela. Além dos itens incluídos, como copos descartáveis de isopor (120ml), adoçante, açúcar, mexedor, o open coffee também conta com a decoração da mesa!</p>
      <h4>Ambientes Disponíveis:</h4>
      <ul>
       <li>Castelo Blumenau (Blumenau)</li><li> Canto da Lagoa (Florianópolis)</li><li> Mirante da Lagoa (Florianópolis)</li><li> Espaço Joinville (Joinville)</li>
      </ul>
      <button className="add-to-cart-btn" onClick={handleAddToCart}>Adicionar produto ao carrinho</button>
```

```
        <h4>Veja mais fotos do produto</h4>
        <div className="product-gallery">
         {images.slice(0, 4).map((image, index) => (
          <div key={index} className="gallery-item" onClick={() =>
openGallery(index)}>
           <img src={image.original} alt={'Foto adicional ' + (index + 1)}
className="gallery-image" />
          </div>
         ))}
         {images.length > 4 && (
          <div className="gallery-item">
           <button className="view-more-btn" onClick={() => openGallery(4)}>Ver
mais</button>
          </div>
         )}
        </div>
       </div>
     </section>
    </main>

    <Modal
     isOpen={isGalleryOpen}
     onRequestClose={closeGallery}
     contentLabel="Galeria de Imagens"
     className="image-gallery-modal"
     overlayClassName="image-gallery-overlay"
    >
     <ImageGallery
      items={galleryItems}
      showThumbnails={true}
      showPlayButton={false}
      showFullscreenButton={false}
      startIndex={0}
     />
     <button className="close-gallery-btn" onClick={closeGallery}>Fechar</button>
    </Modal>

    <Modal
     isOpen={isCartModalOpen}
     onRequestClose={closeCartModal}
     contentLabel="Carrinho"
     className="cart-modal"
     overlayClassName="cart-overlay"
    >
     <div className="cart-modal-content">
      <h2>Produto adicionado ao carrinho</h2>
```

```
        <button className="close-cart-btn"
onClick={handleContinueShopping}>Continuar comprando</button>
        <button className="close-cart-btn" onClick={handleCheckout}>Finalizar
compra</button>
      </div>
    </Modal>
   </div>
  );
};

export default Opencoffee;
```

Descrição:

O componente Opencoffee renderiza os detalhes do produto "Open Coffee", incluindo a descrição, imagens, e permite adicionar o produto ao carrinho de compras. Ele também possui uma galeria de imagens e modais para exibir mais detalhes do produto e confirmar a adição ao carrinho.

AdminPage.js

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import Modal from 'react-modal';
import '../../App.css';
import './admin.css';
import { useNavigate } from 'react-router-dom';

Modal.setAppElement('#root');

const AdminPage = () => {
  const [product, setProduct] = useState({
    id: '',
    name: '',
    category: '',
    imageName: '',
    description: '',
    environments: [],
    active: true,
  });
  const [products, setProducts] = useState([]);
  const [modalIsOpen, setModalIsOpen] = useState(false);
  const [modalMessage, setModalMessage] = useState('');
  const [files, setFiles] = useState([]);
  const [isLoading, setIsLoading] = useState(false);
  const [selectAll, setSelectAll] = useState(false);
  const [isAuthenticated, setIsAuthenticated] = useState(false);
  const navigate = useNavigate();

  useEffect(() => {
    fetchProducts();
  }, []);

  const fetchProducts = () => {
    axios.get('https://loja.eventosindaia.com.br/api/products')
      .then(response => {
        setProducts(response.data);
      })
      .catch(error => {
        console.error('Erro ao buscar produtos:', error);
      });
  };

  const handleChange = (e) => {
```

```
  const { name, value, type, checked, files: selectedFiles } = e.target;
  if (name === 'environments' && type === 'checkbox') {
    setProduct((prevProduct) => ({
      ...prevProduct,
      environments: checked
        ? [...prevProduct.environments, value]
        : prevProduct.environments.filter(env => env !== value),
    }));
  } else if (name === 'files') {
    setFiles(selectedFiles);
  } else {
    let newValue = value;
    if (name === 'id') {
      newValue = newValue.toLowerCase().replace(/[^a-z0-9]/g, '');
    }
    setProduct((prevProduct) => ({
      ...prevProduct,
      [name]: newValue,
    }));
  }
};

const handleSelectAllChange = (e) => {
  const checked = e.target.checked;
  setSelectAll(checked);
  if (checked) {
    setProduct((prevProduct) => ({
      ...prevProduct,
      environments: [
        "Mezanino (Itapema)",
        "Espaço Panorâmico (Itapema)",
        "Salão de Eventos (Itapema)",
        "Lounge (Itapema)",
        "Canto da Lagoa (Florianópolis)",
        "Mirante da Lagoa (Florianópolis)",
        "Espaço Joinville (Joinville)",
        "Castelo Blumenau (Blumenau)"
      ],
    }));
  } else {
    setProduct((prevProduct) => ({
      ...prevProduct,
      environments: [],
    }));
  }
};
```

```
  const validateImageName = (name) => {
    const nameWithoutExtension = name.split('.').slice(0, -1).join('.');
    const invalidChars = /[^0-9]/;
    return !invalidChars.test(nameWithoutExtension);
  };

  const validateFileNames = (files) => {
    for (let i = 0; i < files.length; i++) {
      const fileNameWithoutExtension = files[i].name.split('.').slice(0, -1).join('.');
      const invalidChars = /[^0-9]/;
      if (invalidChars.test(fileNameWithoutExtension)) {
        return false;
      }
    }
    return true;
  };

  const formatForJson = (str) => {
    return str.replace(/\s/g, '%20');
  };

  const handleSubmit = (e) => {
    e.preventDefault();

    if (!validateImageName(product.imageName)) {
      setModalMessage('O nome da imagem principal deve conter apenas números
(excluindo a extensão).');
      setModalIsOpen(true);
      return;
    }

    if (!validateFileNames(files)) {
      setModalMessage('Os nomes dos arquivos de imagem devem conter apenas
números (excluindo a extensão).');
      setModalIsOpen(true);
      return;
    }

    setIsLoading(true);

    const formattedImageName = formatForJson(product.imageName);
    const formattedProductName = formatForJson(product.name);

    const newProduct = {
      ...product,
      imageName: formattedImageName,
```

```
    image:
` https://loja.eventosindaia.com.br/static/images/${formattedProductName}/${form
attedImageName}` ,
    environments: product.environments.join(', '),
   };

   if (files.length > 0) {
    const formData = new FormData();
    formData.append('folder', product.name);
    for (let i = 0; i < files.length; i++) {
     formData.append('files', files[i]);
    }
    axios.post('https://89.116.74.66:5001/upload-folder', formData)
     .then(uploadResponse => {
      axios.post('https://89.116.74.66:5000/api/products', newProduct)
       .then(response => {
        setModalMessage('Produto e arquivos cadastrados com sucesso!');
        setModalIsOpen(true);
        setProduct({ id: '', name: '', category: '', imageName: '', description: '',
environments: [], active: true });
        setFiles([]);
        fetchProducts();
       })
       .catch(error => {
        setModalMessage('Erro ao cadastrar produto: ' + error.message);
        setModalIsOpen(true);
        console.error('Erro ao cadastrar produto:', error);
       })
       .finally(() => setIsLoading(false));
     })
     .catch(uploadError => {
      setModalMessage('Erro ao enviar arquivos: ' + uploadError.message);
      setModalIsOpen(true);
      console.error('Erro ao enviar arquivos:', uploadError);
      setIsLoading(false);
     });
   } else {
    axios.post('https://89.116.74.66:5000/api/products', newProduct)
     .then(response => {
      setModalMessage('Produto cadastrado com sucesso!');
      setModalIsOpen(true);
      setProduct({ id: '', name: '', category: '', imageName: '', description: '',
environments: [], active: true });
      fetchProducts();
     })
     .catch(error => {
      setModalMessage('Erro ao cadastrar produto: ' + error.message);
```

```
        setModalIsOpen(true);
        console.error('Erro ao cadastrar produto:', error);
      })
      .finally(() => setIsLoading(false));
  }
 };

 const handleDelete = (id) => {
  axios.delete(`https://loja.eventosindaia.com.br/api/products/${id}`)
   .then(response => {
    alert('Produto deletado com sucesso!');
    fetchProducts();
   })
   .catch(error => {
    console.error('Erro ao deletar produto:', error);
   });
 };

 const handleInactivate = (id) => {
  const productToUpdate = products.find(product => product.id === id);
  const updatedProduct = { ...productToUpdate, active: !productToUpdate.active };

  axios.put(`https://loja.eventosindaia.com.br/api/products/${id}`,
updatedProduct)
   .then(response => {
    alert(`Produto ${updatedProduct.active ? 'ativado' : 'inativado'} com sucesso!`);
    fetchProducts();
   })
   .catch(error => {
    console.error('Erro ao inativar produto:', error);
   });
 };

 const handleViewDetails = (id) => {
  navigate(`/products/${id}`);
 };

 const handleLogin = () => {
  const username = prompt('Login:');
  const password = prompt('Senha:');

  const validUsers = [
   { username: 'gabriel.paduch', password: 'Indaia@2024' },
   { username: 'willian.indaia', password: 'Indaia@321' },
   { username: 'aline.indaia', password: 'Indaia@123' },
   { username: 'arles.robalo', password: 'Indaia@2024' }
  ];
```

```
   const isValidUser = validUsers.some(user => user.username === username &&
user.password === password);

   if (isValidUser) {
    localStorage.setItem('isAuthenticated', 'true');
    setIsAuthenticated(true);
   } else {
    alert('Login ou senha incorretos!');
   }
  };

  useEffect(() => {
   const authStatus = localStorage.getItem('isAuthenticated');
   if (authStatus === 'true') {
    setIsAuthenticated(true);
   } else {
    handleLogin();
   }
  }, []);

  if (!isAuthenticated) {
   return <div>Carregando...</div>;
  }

  return (
   <div className="admin-container">
    <h2>Cadastro de Produtos</h2>
    <form className="admin-form" onSubmit={handleSubmit}>
     <label>
      ID:
      <input type="text" name="id" value={product.id} onChange={handleChange}
required />
     </label>
     <label>
      Nome:
      <input type="text" name="name" value={product.name}
onChange={handleChange} required />
     </label>
     <label>
      Categoria:
      <select name="category" value={product.category} onChange={handleChange}
required>
       <option value="">Selecione uma categoria</option>
       <option value="cardápio">Cardápio</option>
       <option value="serviços">Serviços</option>
       <option value="cerimonias">Cerimônias</option>
```

```
        <option value="decoração">Decoração</option>
        <option value="iluminação">Iluminação</option>
        <option value="contratados">Contratados</option>
       </select>
      </label>
      <label>
       Nome da Imagem Principal (com extensão):
       <input type="text" name="imageName" value={product.imageName}
onChange={handleChange} required />
      </label>
      <label>
       Arquivos de Imagem:
       <input type="file" name="files" onChange={handleChange} multiple required />
      </label>
      <label>
       Descrição:
       <textarea name="description" value={product.description}
onChange={handleChange} required />
      </label>
      <label>
       Ambientes:
       <div className="checkbox-group">
        <label>
         <input
          type="checkbox"
          checked={selectAll}
          onChange={handleSelectAllChange}
         />
         Selecionar Todos
        </label>
        <label>
         <input type="checkbox" name="environments" value="Mezanino (Itapema)"
checked={product.environments.includes("Mezanino (Itapema)")}
onChange={handleChange} />
         Mezanino (Itapema)
        </label>
        <label>
         <input type="checkbox" name="environments" value="Espaço Panorâmico
(Itapema)" checked={product.environments.includes("Espaço Panorâmico
(Itapema)")} onChange={handleChange} />
         Espaço Panorâmico (Itapema)
        </label>
        <label>
         <input type="checkbox" name="environments" value="Salão de Eventos
(Itapema)" checked={product.environments.includes("Salão de Eventos (Itapema)")}
onChange={handleChange} />
         Salão de Eventos (Itapema)
```

```
        </label>
        <label>
          <input type="checkbox" name="environments" value="Lounge (Itapema)"
checked={product.environments.includes("Lounge (Itapema)")}
onChange={handleChange} />
          Lounge (Itapema)
        </label>
        <label>
          <input type="checkbox" name="environments" value="Canto da Lagoa
(Florianópolis)" checked={product.environments.includes("Canto da Lagoa
(Florianópolis)")} onChange={handleChange} />
          Canto da Lagoa (Florianópolis)
        </label>
        <label>
          <input type="checkbox" name="environments" value="Mirante da Lagoa
(Florianópolis)" checked={product.environments.includes("Mirante da Lagoa
(Florianópolis)")} onChange={handleChange} />
          Mirante da Lagoa (Florianópolis)
        </label>
        <label>
          <input type="checkbox" name="environments" value="Espaço Joinville
(Joinville)" checked={product.environments.includes("Espaço Joinville (Joinville)")}
onChange={handleChange} />
          Espaço Joinville (Joinville)
        </label>
        <label>
          <input type="checkbox" name="environments" value="Castelo Blumenau
(Blumenau)" checked={product.environments.includes("Castelo Blumenau
(Blumenau)")} onChange={handleChange} />
          Castelo Blumenau (Blumenau)
        </label>
      </div>
    </label>
    <button type="submit">Cadastrar Produto</button>
  </form>

  {isLoading && (
    <div className="admin-loading-overlay">
      <div className="admin-loading-spinner"></div>
      <p>Carregando...</p>
    </div>
  )}

  <h2>Produtos Cadastrados</h2>
  <ul className="admin-product-list">
    {products.map((product) => (
      <li key={product.id} className="admin-product-item">
```

```
        <img src={product.image} alt={product.name} className="admin-product-
image" />
        <div className="admin-product-details">
         <span><strong>Nome:</strong> {product.name}</span>
         <span><strong>Categoria:</strong> {product.category}</span>
         <span><strong>Status:</strong> {product.active ? 'Ativo' : 'Inativo'}</span>
        </div>
        <div className="admin-product-actions">
         <button className="admin-delete-btn" onClick={() =>
handleDelete(product.id)}>Excluir</button>
         <button className="admin-inactivate-btn" onClick={() =>
handleInactivate(product.id)}>
          {product.active ? 'Inativar' : 'Ativar'}
         </button>
         <button className="admin-view-btn" onClick={() =>
handleViewDetails(product.id)}>Ver Detalhes</button>
        </div>
       </li>
     ))}
    </ul>

    <Modal
     isOpen={modalIsOpen}
     onRequestClose={() => setModalIsOpen(false)}
     contentLabel="Retorno do Servidor"
     className="admin-modal"
     overlayClassName="admin-overlay"
    >
     <h2>Mensagem do Servidor</h2>
     <p>{modalMessage}</p>
     <button onClick={() => setModalIsOpen(false)}>Fechar</button>
    </Modal>
   </div>
  );
};

export default AdminPage;
```

Descrição:

O componente AdminPage permite a gestão de produtos, incluindo cadastro, edição, exclusão e ativação/inativação de produtos. Ele também inclui autenticação básica e envio de arquivos de imagem para o servidor.

CartContext.js

```
import React, { createContext, useState } from 'react';

export const CartContext = createContext();

export const CartProvider = ({ children }) => {
  const [cartItems, setCartItems] = useState([]);

  const addToCart = (product) => {
    const productWithId = { ...product, cartItemId: Date.now() };
    setCartItems((prevItems) => [...prevItems, productWithId]);
  };

  const removeFromCart = (cartItemId) => {
    setCartItems((prevItems) => prevItems.filter(item => item.cartItemId !==
cartItemId));
  };

  const clearCart = () => {
    setCartItems([]);
  };

  return (
    <CartContext.Provider value={{ cartItems, addToCart, removeFromCart, clearCart
}}>
      {children}
    </CartContext.Provider>
  );
};
```

Descrição:

O contexto CartContext fornece a funcionalidade para adicionar, remover e limpar itens do carrinho de compras. Ele é utilizado em vários componentes do aplicativo para gerenciar o estado do carrinho.

FilterContext.js

```
import React, { createContext, useState } from 'react';

export const FilterContext = createContext();

export const FilterProvider = ({ children }) => {
  const [filter, setFilter] = useState('todos');
  const [contractedProducts, setContractedProducts] = useState([]);

  return (
    <FilterContext.Provider value={{ filter, setFilter, contractedProducts,
setContractedProducts }}>
      {children}
    </FilterContext.Provider>
  );
};
```

Descrição:

O contexto FilterContext fornece a funcionalidade para definir e gerenciar filtros de produtos, bem como produtos contratados. Ele é utilizado para filtrar a lista de produtos exibidos no aplicativo.

Imageapi.py

```
import os
import json
from flask import Flask, jsonify, request, send_from_directory, url_for
from flask_cors import CORS

app = Flask(__name__)
CORS(app, resources={r"/*": {"origins": "*"}})  # Desativando a política de CORS

project_root = os.path.dirname(os.path.abspath(__file__))
products_path = os.path.join(project_root, '/var/www/areadocliente/my-
ecommerce-app/src/pages/products')
images_path = os.path.join(project_root, '/var/www/areadocliente/my-ecommerce-
app/src/images')

@app.route('/api/images/<product_name>', methods=['GET'])
def get_images(product_name):
    product_images_path = os.path.join(images_path, product_name)
    if not os.path.exists(product_images_path):
        return jsonify(error='Product not found'), 404

    image_files = [f for f in os.listdir(product_images_path) if f.lower().endswith(('.jpg',
'.jpeg', '.png', '.gif'))]
    # Criação das URLs das imagens
    image_urls = [url_for('static_files', filename=f'{product_name}/{image}',
_external=True) for image in image_files]
    return jsonify(image_urls)

@app.route('/api/products', methods=['GET'])
def get_products():
    with open(os.path.join(products_path, 'products.json')) as f:
        products = json.load(f)
    return jsonify(products)

@app.route('/api/products/<product_id>', methods=['GET'])
def get_product(product_id):
    with open(os.path.join(products_path, 'products.json')) as f:
        products = json.load(f)
    product = next((p for p in products if p['id'] == product_id and p.get('active', True)),
None)
    if not product:
        return jsonify(error='Product not found'), 404

    product_images_path = os.path.join(images_path, product['name'])
```

```
    if os.path.exists(product_images_path):
        image_files = [f for f in os.listdir(product_images_path) if f.lower().endswith(('.jpg', '.jpeg', '.png', '.gif'))]
        product['images'] = [url_for('static_files', filename=f'{product["name"]}/{image}', _external=True) for image in image_files]
    return jsonify(product)

@app.route('/api/products/<product_id>', methods=['PUT'])
def update_product(product_id):
    updated_product = request.json
    with open(os.path.join(products_path, 'products.json')) as f:
        products = json.load(f)

    product_index = next((index for (index, p) in enumerate(products) if p['id'] == product_id), None)
    if product_index is None:
        return jsonify(error='Product not found'), 404

    products[product_index] = updated_product

    with open(os.path.join(products_path, 'products.json'), 'w') as f:
        json.dump(products, f)

    return jsonify(updated_product)

@app.route('/api/products/<product_id>', methods=['DELETE'])
def delete_product(product_id):
    with open(os.path.join(products_path, 'products.json')) as f:
        products = json.load(f)

    product_index = next((index for (index, p) in enumerate(products) if p['id'] == product_id), None)
    if product_index is None:
        return jsonify(error='Product not found'), 404

    deleted_product = products.pop(product_index)

    with open(os.path.join(products_path, 'products.json'), 'w') as f:
        json.dump(products, f)

    return jsonify(deleted_product)

@app.route('/static/images/<path:filename>', methods=['GET'])
def static_files(filename):
    return send_from_directory(images_path, filename)

if __name__ == '__main__':
```

```
app.run(host='0.0.0.0', port=5003)
```

Descrição:

O script imageapi.py define uma API RESTful usando Flask para gerenciar produtos e imagens de produtos. As rotas implementadas incluem:

/api/images/<product_name>: Retorna URLs das imagens associadas a um produto específico.

/api/products: Retorna a lista de produtos.

/api/products/<product_id>: Retorna detalhes de um produto específico.

/api/products/<product_id> (PUT): Atualiza um produto existente.

/api/products/<product_id> (DELETE): Exclui um produto específico.

/static/images/<path:filename>: Serve arquivos de imagem estáticos.

Images.py

```
from flask import Flask, request, jsonify
from flask_cors import CORS
import os
from werkzeug.utils import secure_filename
import ssl

app = Flask(__name__)
CORS(app, resources={r"/*": {"origins": "*"}})  # Permite CORS de qualquer origem

UPLOAD_FOLDER = r'/var/www/areadocliente/my-ecommerce-app/src/images'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif'}

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

@app.route('/upload-folder', methods=['POST'])
def upload_folder():
    if 'files' not in request.files or 'folder' not in request.form:
        return jsonify({"error": "No files or folder part"}), 400

    folder_name = request.form['folder']
    folder_path = os.path.join(UPLOAD_FOLDER, folder_name)
    os.makedirs(folder_path, exist_ok=True)

    files = request.files.getlist('files')

    if not files:
        return jsonify({"error": "No selected files"}), 400

    for file in files:
        if not allowed_file(file.filename):
            return jsonify({"error": f"File type not allowed: {file.filename}"}), 400

        filename = secure_filename(file.filename)
        file_path = os.path.join(folder_path, filename)
        file.save(file_path)

    return jsonify({"message": "Files uploaded"}), 200

if __name__ == '__main__':
    context = ssl.SSLContext(ssl.PROTOCOL_TLS)
```

```
context.load_cert_chain(certfile='/etc/ssl/loja.eventosindaia.com.br/fullchain.pem',
keyfile='/etc/ssl/loja.eventosindaia.com.br/privkey.pem')
    app.run(host='0.0.0.0', port=5001, ssl_context=context, debug=True)
```

Descrição:

O script imageapi.py define uma API RESTful usando Flask para gerenciar produtos e imagens de produtos. As rotas implementadas incluem:

/api/images/<product_name>: Retorna URLs das imagens associadas a um produto específico.

/api/products: Retorna a lista de produtos.

/api/products/<product_id>: Retorna detalhes de um produto específico.

/api/products/<product_id> (PUT): Atualiza um produto existente.

/api/products/<product_id> (DELETE): Exclui um produto específico.

/static/images/<path:filename>: Serve arquivos de imagem estáticos.

Posvendas.py

```
import gspread
from oauth2client.service_account import ServiceAccountCredentials
from flask import Flask, jsonify
from flask_cors import CORS
from apscheduler.schedulers.background import BackgroundScheduler
import logging
import json
from datetime import datetime

# Configuração de logging
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s -
%(message)s')

app = Flask(__name__)
CORS(app)

# Configurações do Google Sheets
scope = ["https://spreadsheets.google.com/feeds",
'https://www.googleapis.com/auth/spreadsheets',
      "https://www.googleapis.com/auth/drive.file",
"https://www.googleapis.com/auth/drive"]

creds = ServiceAccountCredentials.from_json_keyfile_name('keys/backup-ti-
426913-2d9d57342f77.json', scope)
client = gspread.authorize(creds)

# Nome da planilha e da aba
spreadsheet_name = "PÓS-VENDAS (DISPONIBILIDADE VENDAS - ITAPEMA)"
sheet_name = "Pós-Vendas"

# Inicializa as variáveis globais
data = {}

def is_valid_date(date_string):
  try:
    datetime.strptime(date_string, '%d/%m/%Y')
    return True
  except ValueError:
    return False

def fetch_sheet_data():
  logging.debug("Iniciando a busca de dados da planilha.")
  global data
```

```
    try:
        sheet = client.open(spreadsheet_name).worksheet(sheet_name)
        logging.debug("Planilha e aba abertas com sucesso.")
        records = sheet.get_all_records()
        logging.debug(f"{len(records)} registros encontrados.")

        # Filtra os registros com base na data de realização
        filtered_records = []
        current_date = datetime.now()

        for record in records:
            hire_date_str = record.get('Data Real.', '')
            event_code = record.get('Cód', 'N/A')
            if hire_date_str and is_valid_date(hire_date_str):
                try:
                    hire_date = datetime.strptime(hire_date_str, '%d/%m/%Y')
                    if hire_date >= current_date:
                        filtered_records.append(record)
                except ValueError as e:
                    logging.error(f"Erro ao analisar a data de realização: {hire_date_str} - Código do Evento: {event_code}", exc_info=True)
            else:
                logging.warning(f"Data de realização está vazia ou inválida: {hire_date_str} - Código do Evento: {event_code}")

        data = filtered_records
        logging.debug("Dados atualizados com sucesso.")
    except Exception as e:
        logging.error("Erro ao buscar dados da planilha.", exc_info=True)

@app.route('/data', methods=['GET'])
def get_data():
    logging.debug("Recebida solicitação para /data.")
    return jsonify(data)

if __name__ == '__main__':
    logging.debug("Script iniciado.")
    fetch_sheet_data()  # Buscar dados inicialmente

    # Configuração do scheduler para atualizar os dados a cada 15 minutos
    scheduler = BackgroundScheduler()
    scheduler.add_job(fetch_sheet_data, 'interval', minutes=15)
    scheduler.start()
    logging.debug("Scheduler iniciado para atualizar os dados a cada 15 minutos.")

    # Executa o servidor Flask com HTTP
    app.run(host='0.0.0.0', port=5002)
```

```
    logging.debug("Servidor Flask iniciado na porta 5002 com HTTP.")
```

Descrição:

O script posvendas.py gerencia dados de uma planilha do Google Sheets relacionada a vendas pós-evento. Utiliza Flask para fornecer uma API que retorna esses dados, e o APScheduler para atualizar os dados a cada 15 minutos. As funcionalidades principais incluem:

/data: Retorna os dados filtrados da planilha do Google Sheets.

Função de agendamento que atualiza os dados a cada 15 minutos.

start.sh

```
#!/bin/bash
python imageapi.py &
python posvendas.py &
python images.py
```

Descrição:

O script start.sh é um script de inicialização que executa múltiplos scripts Python em segundo plano. Ele inicia os scripts imageapi.py, posvendas.py e images.py.

Server.js

```
const express = require('express');
const fs = require('fs');
const path = require('path');
const morgan = require('morgan');
const cors = require('cors');
const https = require('https');
const { exec } = require('child_process');

const app = express();
const PORT = process.env.PORT || 5000;
const projectRoot = '/var/www/areadocliente/my-ecommerce-app';

// Caminhos dos certificados
const privateKey = fs.readFileSync('/etc/ssl/loja.eventosindaia.com.br/privkey.pem',
'utf8');
const certificate =
fs.readFileSync('/etc/ssl/loja.eventosindaia.com.br/fullchain.pem', 'utf8');
const credentials = { key: privateKey, cert: certificate };

app.use(morgan('combined'));
app.use(cors());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

app.use(express.static('public'));
app.use('/src/images', express.static(path.join(__dirname, '../src/images')));

app.post('/api/products', (req, res) => {
 const newProduct = req.body;
 const productsPath = path.join(__dirname, '../src/pages/products/products.json');
 console.log(` Adicionando novo produto: ${JSON.stringify(newProduct)}` );

 fs.readFile(productsPath, 'utf8', (err, data) => {
  if (err) {
   console.error(` Erro ao ler o arquivo de produtos: ${err.message}` );
   return res.status(500).json({ error: 'Failed to read products file' });
  }
  const products = JSON.parse(data);
  products.push(newProduct);
  fs.writeFile(productsPath, JSON.stringify(products, null, 2), (err) => {
   if (err) {
    console.error(` Erro ao escrever no arquivo de produtos: ${err.message}` );
    return res.status(500).json({ error: 'Failed to write products file' });
```

```
    }

    // Criar o novo arquivo de página de produto e executar o build
    const componentName = newProduct.id.charAt(0).toUpperCase() +
newProduct.id.slice(1);
    const productPageContent = `
import React, { useState, useEffect, useContext } from 'react';
import axios from 'axios';
import { useNavigate } from 'react-router-dom';
import './products.css'; // Importando o novo CSS
import MainImage from
'../../images/${newProduct.name}/${newProduct.imageName}';
import ImageGallery from 'react-image-gallery';
import 'react-image-gallery/styles/css/image-gallery.css';
import Modal from 'react-modal';
import { CartContext } from '../../CartContext'; // Importando o contexto do carrinho

Modal.setAppElement('#root');

const ${componentName} = () => {
  const { addToCart } = useContext(CartContext); // Usando o contexto do carrinho
  const [images, setImages] = useState([]);
  const [galleryItems, setGalleryItems] = useState([]);
  const [isGalleryOpen, setIsGalleryOpen] = useState(false);
  const [isCartModalOpen, setIsCartModalOpen] = useState(false);
  const navigate = useNavigate();
  const productName = '${newProduct.name}';

  useEffect(() => {

axios.get(\`https://loja.eventosindaia.com.br/api/images/\${encodeURIComponent(
productName)}\`)
    .then(response => {
      const imageUrls = response.data.map(file => ({
        original: file,
        thumbnail: file
      }));

      console.log('Imagens carregadas:', imageUrls); // Adicione este log para verificar
as imagens
      setImages(imageUrls);
      setGalleryItems(imageUrls);
    })
    .catch(error => {
      console.error('Error fetching images:', error);
    });
  }, [productName]);
```

```
 const openGallery = (startIndex = 0) => {
  setGalleryItems(images);
  setIsGalleryOpen(true);
 };

 const closeGallery = () => {
  setIsGalleryOpen(false);
 };

 const handleAddToCart = () => {
  addToCart({ id: '${newProduct.id}', name: productName, image: MainImage });
  setIsCartModalOpen(true);
 };

 const closeCartModal = () => {
  setIsCartModalOpen(false);
 };

 const handleContinueShopping = () => {
  setIsCartModalOpen(false);
  navigate('/');
 };

 const handleCheckout = () => {
  setIsCartModalOpen(false);
  navigate('/cart');
 };

 return (
  <div>
   <main>
    <section className="product-details">
     <div className="product-image-container">
      <img id="product-image" src={MainImage} alt={productName} />
     </div>
     <div id="product-description">
      <p>${newProduct.description}</p>
      <h4>Ambientes Disponíveis:</h4>
      <ul>
       ${newProduct.environments.split(',').map(env => `<li>${env}</li>`).join('')}
      </ul>
      <button className="add-to-cart-btn" onClick={handleAddToCart}>Adicionar
produto ao carrinho</button>
      <h4>Veja mais fotos do produto</h4>
      <div className="product-gallery">
       {images.slice(0, 4).map((image, index) => (
```

```
          <div key={index} className="gallery-item" onClick={() =>
openGallery(index)}>
            <img src={image.original} alt={'Foto adicional ' + (index + 1)}
className="gallery-image" />
          </div>
         ))}
         {images.length > 4 && (
          <div className="gallery-item">
           <button className="view-more-btn" onClick={() => openGallery(4)}>Ver
mais</button>
          </div>
         )}
        </div>
       </div>
      </section>
    </main>

    <Modal
     isOpen={isGalleryOpen}
     onRequestClose={closeGallery}
     contentLabel="Galeria de Imagens"
     className="image-gallery-modal"
     overlayClassName="image-gallery-overlay"
    >
     <ImageGallery
      items={galleryItems}
      showThumbnails={true}
      showPlayButton={false}
      showFullscreenButton={false}
      startIndex={0}
     />
     <button className="close-gallery-btn" onClick={closeGallery}>Fechar</button>
    </Modal>

    <Modal
     isOpen={isCartModalOpen}
     onRequestClose={closeCartModal}
     contentLabel="Carrinho"
     className="cart-modal"
     overlayClassName="cart-overlay"
    >
     <div className="cart-modal-content">
      <h2>Produto adicionado ao carrinho</h2>
      <button className="close-cart-btn"
onClick={handleContinueShopping}>Continuar comprando</button>
      <button className="close-cart-btn" onClick={handleCheckout}>Finalizar
compra</button>
```

```
      </div>
    </Modal>
   </div>
  );
};

export default ${componentName};
`;

    const productPagePath = path.join(__dirname,
`../src/pages/products/${newProduct.id}.js`);
    fs.writeFile(productPagePath, productPageContent, (err) => {
     if (err) {
      console.error(`Erro ao criar o arquivo de página do produto: ${err.message}`);
      return res.status(500).json({ error: 'Failed to create product page' });
     }

     // Executar o build
     exec('npm run build', { cwd: projectRoot }, (err, stdout, stderr) => {
      if (err) {
       console.error(`Erro ao executar build: ${stderr}`);
       return res.status(500).json({ error: 'Erro ao executar build' });
      }
      console.log(`Resultado do build: ${stdout}`);
      res.status(201).json(newProduct);
     });
    });
   });
  });
});

const httpsServer = https.createServer(credentials, app);

httpsServer.listen(PORT, () => {
 console.log(`Server is running on port ${PORT}`);
});
```

Descrição:

O script server.js é um servidor Node.js que utiliza o framework Express para gerenciar solicitações HTTP e HTTPS. As funcionalidades principais incluem:

Configuração de CORS: Permite solicitações de qualquer origem.

Servir arquivos estáticos: Serve arquivos estáticos a partir dos diretórios public e src/images.

Gerenciamento de Produtos:

POST /api/products: Adiciona um novo produto ao arquivo products.json e cria uma nova página de produto em React. Após adicionar o produto, o script executa um comando de build (npm run build) para compilar o projeto.

Configuração HTTPS: Utiliza certificados SSL para criar um servidor HTTPS seguro.

Dockerfile-python

```
# Usar a imagem oficial do Python
FROM python:3.9-slim

# Definir o diretório de trabalho no contêiner
WORKDIR /var/www/areadocliente/my-ecommerce-app/backend/api

# Copiar os arquivos de requisitos
COPY backend/api/requirements.txt .

# Instalar as dependências do Python
RUN pip install --no-cache-dir -r requirements.txt

# Copiar o código da aplicação para o diretório de trabalho
COPY backend/api /var/www/areadocliente/my-ecommerce-app/backend/api

# Copiar os certificados SSL para o contêiner
COPY backend/api/loja.eventosindaia.com.br /etc/ssl/loja.eventosindaia.com.br

# Dar permissão de execução para o script de inicialização
RUN chmod +x /var/www/areadocliente/my-ecommerce-app/backend/api/start.sh

# Comando para rodar o script de inicialização
CMD ["/var/www/areadocliente/my-ecommerce-app/backend/api/start.sh"]
```

Descrição:

O arquivo Dockerfile-python é utilizado para criar uma imagem Docker personalizada para o backend em Python. Ele define as instruções necessárias para configurar o ambiente de execução, incluindo a instalação de dependências e a configuração dos certificados SSL.

Dockerfile-node

```
# Usar a imagem oficial do Node.js
FROM node:14

# Definir o diretório de trabalho no contêiner
WORKDIR /var/www/areadocliente/my-ecommerce-app/backend

# Copiar os arquivos package.json e package-lock.json
COPY backend/package*.json ./

# Instalar as dependências do Node.js
RUN npm install

# Copiar o código da aplicação para o diretório de trabalho
COPY backend /var/www/areadocliente/my-ecommerce-app/backend

# Copiar os certificados SSL para o contêiner
COPY backend/api/loja.eventosindaia.com.br /etc/ssl/loja.eventosindaia.com.br

# Comando para rodar a aplicação Node.js
CMD ["node", "server.js"]
```

Descrição:

O arquivo Dockerfile-node é utilizado para criar uma imagem Docker personalizada para o backend em Node.js. Ele define as instruções necessárias para configurar o ambiente de execução, incluindo a instalação de dependências e a configuração dos certificados SSL.

docker-compose.yml

```
version: '3.8'

services:
 python-backend:
  build:
   context: .
   dockerfile: backend/api/Dockerfile-python
  volumes:
   - ./backend/api:/var/www/areadocliente/my-ecommerce-app/backend/api
   - ./src:/var/www/areadocliente/my-ecommerce-app/src
  environment:
   - PYTHONUNBUFFERED=1
  ports:
   - "5001:5001"
   - "5002:5002"
   - "5003:5003"

 node-backend:
  build:
   context: .
   dockerfile: backend/Dockerfile-node
  volumes:
   - ./backend:/var/www/areadocliente/my-ecommerce-app/backend
   - ./src:/var/www/areadocliente/my-ecommerce-app/src
  ports:
   - "3000:3000"
  environment:
   - NODE_ENV=development

 frontend:
  image: nginx:alpine
  volumes:
   - ./build:/usr/share/nginx/html
  ports:
   - "8080:80"
  depends_on:
   - node-backend
```

Descrição:

O arquivo docker-compose.yml é utilizado para definir e gerenciar os serviços Docker para o projeto. Ele especifica como os contêineres devem ser construídos e configurados, bem como as dependências entre eles.

Configuração e Instalação

Para configurar e instalar o projeto utilizando o repositório do GitHub e Docker, siga os passos abaixo:

Clonar o repositório:

```
git clone https://github.com/GabePaduch/https-loja.eventosindaia.com.br-.git
cd https-loja.eventosindaia.com.br-
```

Instalar dependências do Frontend:

```
cd frontend
npm install
cd ..
```

Instalar dependências do Backend:

```
cd backend
npm install
cd ..
```

Construir e iniciar os contêineres Docker:

Certifique-se de estar no diretório raiz do projeto (https-loja.eventosindaia.com.br-), onde o arquivo docker-compose.yml está localizado.

```
docker-compose up --build
```

Este comando irá construir as imagens Docker definidas nos Dockerfiles e iniciar os contêineres conforme especificado no docker-compose.yml.

Conclusão

Esta documentação fornece uma visão abrangente dos componentes principais e da estrutura do nosso projeto de e-commerce. O projeto é desenvolvido utilizando uma combinação de tecnologias, incluindo React.js para o frontend, Node.js e Python para o backend, e Docker para facilitar a implantação e o gerenciamento dos ambientes de desenvolvimento e produção.

Os componentes React.js são projetados para serem reutilizáveis e modulares, facilitando a manutenção e a escalabilidade do projeto. O backend fornece APIs robustas para gerenciar produtos e imagens, utilizando tanto Node.js quanto Python para diferentes funcionalidades.

A configuração Docker garante um ambiente de desenvolvimento consistente e simplifica a implantação da aplicação. Com Docker, é possível replicar facilmente o ambiente de produção localmente, garantindo que a aplicação funcione de forma previsível em qualquer ambiente.

Por fim, seguindo os passos de configuração e instalação fornecidos, você estará pronto para executar e desenvolver o projeto localmente, bem como preparar a aplicação para produção.