UNIVERSIDADE DE SÃO CARLOS

INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

DEPARTAMENTO DE SISTEMAS DE INFORMAÇÃO

SCC0219 - Introduction to Web Development - 2ºSEM/2021

Grupo 4

Hugo de Azevedo Vitulli - 10295221
hugovitulli@usp.br
Matheus Vinicius - 10295217
mt_godoi@usp.br
Henrique Hiram Libutti Núñez - 11275300
henrique.nunez@usp.br

Prof. Dr. Dilvan de Abreu Moreira

1.Requirements

Além dos requerimentos previamente determinados pela proposta de trabalho, será necessário adicionar uma categorização dos calçados para que possamos implementar as funcionalidades relacionadas a filtragem por categoria.

2. Project Description

Neste projeto planejamos implementar funcionalidades de cadastro do usuário, login (um para usuário comum e outro para administrador, onde a conta de admin é gerada por nós e cedida ao administrador), verificação de detalhe do produto, adição/remoção de produto ao carrinho, conclusão de compra, com a conta de usuário ainda será possível verificar histórico de compra e editar informações da conta, para o administrador será herdado todas as funcionalidades do usuário e adicionaremos as funcionalidades de administrar produtos e administrar usuários.

Planejamos também adicionar filtros para melhor experiência na loja, podendo filtrar os sapatos por categorias como por exemplo: Esporte, Casual, Popular. Também pretendemos ter um filtro de ordenação, por preço (crescente/decrescente).

- **Mockup e Diagrama de Navegação:**
  Não foi desenvolvida uma tela de conclusão de compra no mockup, pois a API que utilizaremos com react no milestone - 2 já possui uma tela de compra através de um redirecionamento, portanto possui seu próprio design.
  Figma project - Outsider Footwear
  Figma presentation - Outsider Footwear

- **Informação salva do Banco:**
  Planejamos salvar as informações de cadastro do Usuário, sendo elas: email, senha, endereço, nome, telefone; e informações de produtos: Nome, descrição, preço, quantidade em estoque, categoria.
  Além disso, precisamos armazenar os produtos no carrinho de cada usuário, para manter o carrinho sempre atualizado.

3. Comments About the Code:

No milestone 1 escolhemos as páginas *homepage, signIn* e *signUp* para serem implementadas. Temos a página *headerTemplate.html* que utilizamos apenas para desenvolver o header, não há chamadas para o *headerTemplate.html* porém decidimos mantê-lo pois poderá ser reaproveitado como um componente em react no milestone 2.

Para desenvolver essa primeira Milestone, criamos a seguinte estrutura:

- /frontend/src/pages/ : contém todos os códigos html, onde procuramos modularizar ao máximo, separando as páginas html por funcionalidade assim como está no mockup.

- /frontend/src/styles/ : armazena todos os arquivos .css para serem utilizados no projeto. Os arquivos *global.css, header.css* e *footer.css* foram utilizados em todas as páginas, decidimos mantê-los modularizados para facilitar a futura implementação em react.

- /frontend/src/assets/ : contém as imagens e fontes utilizadas no site.

No milestone 2 começamos a criação do sistema, utilizando react como framework escolhido.

Para isso, reestruturamos o nosso projeto na seguinte estrutura:

Components:

- Aqui se encontram todas as renderizações que aparecerão em 2 páginas ou mais, por exemplo o header, assim basta chamar o Component no App (renderização geral) ou na página desejada.

Hooks:

- Nesta seção se encontram todas as regras de negócio que podemos efetuar no frontend, no nosso caso, criamos a useCart para lidar com interações entre compras de homepage, carrinho e gerar uma lista de histórico de compra, e criamos a useStock para gerenciamento de estoque. Aqui vale a ressalva que tivemos grandes dificuldades de conectar informações de estoque com homepage e carrinho, portanto para o useCart utilizamos as informações iniciais providenciadas pelo db.json(nosso database fake), já para o gerenciamento de produtos utilizamos o useStock a partir das mesmas informações iniciais, mas o efeito de um não se aplica ao outro.

Pages:

- Neste folder criamos todas as possíveis páginas do web app, sendo elas:

Homepage: tela inicial

ManageAccount: Gerenciamento da conta do usuário por parte do user, nesta página fizemos apenas a renderização.

ManageAccountAdmin: Gerenciamento da conta do usuário por parte do admin, nesta página fizemos apenas a rederização.

ManageProducts: Gerenciamento de produtos do estoque por parte do admin, nesta página fizemos interação de quantidade de produto em estoque utilizando localstorage, porém esta não afeta o db.json

ManageUsers: Gerenciamento de todos os usuários cadastrados no site por parte do admin, nesta página fizemos apenas a renderização.

ProductsInfo: Modal de informação do produto após clicar na foto de algum produto da homepage, esta página possui interação com o carrinho no fluxo de compra. Tentamos utilizar react-modal porém tivemos dificuldades em lidar com a ferramenta.

PurchaseHistory: Apresenta o histórico de compra do usuário. Esta página apresenta alguns problemas para executar os testes de interação será necessário se atentar aos casos em que o usuário executa compras distintas, uma vez que o usuário executa sua compra, os produtos permanecem no carrinho e é necessário limpar todos os produtos do carrinho(através da lixeira) para então selecionar novamente os produtos a partir da homepage ou productsInfo e então executar uma nova compra, caso contrário a quantidade de um produto pode herdar a quantidade do último produto comprado com o mesmo id. Outro fato é que para resetar o

histórico de compras é necessário deletar o localstorage "Group4:purchaseHistory" em Applications.

PurchaseHistoryAdmin: Mesma página de PurchaseHistory, mas feita para as compras do admin.

SingIn: Esta página está sem funcionalidade, uma vez que decidimos deixar a autenticação para ser realizada no milestone 3 através de um banco de dados.

SignUp: Esta página está sem funcionalidade, uma vez que decidimos deixar a autenticação para ser realizada no milestone 3 através de um banco de dados.

Styles:

- Possui o css global.

Services:

- Configuração do nosso servidor fake utilizando axios e json-server.

Util:

- Possui função de formatação de currency para facilitar a utilização desta formatação em preços do produto em todas as páginas que mostramos esses valores

App.tsx:

Renderização de todas as páginas

routes.tsx:

Seta o caminho de todas as páginas utilizando react-router-dom

4. Test Plan:

Planejamos utilizar insomnio ou postman para executar os testes de backend. Como utilizamos apenas um json para servir de backend, não fizemos os testes dos nossos bancos de dados, apenas da rota de produtos.

5. Test Results:

A princípio, planejamos utilizar uma tabela de entradas e saídas esperadas.

Os testes da API que utilizamos para servir os dados de produtos foram fornecidos como esperado.

6. Build Procedures:

Para fazer o build do projeto e executar, temos que seguir alguns passos. Primeiro, temos que instalar os pacotes do projeto:

yarn install

ou

npm install global

- Backend:

Para executar o nosso Backend, que se trata apenas de uma API para ler um json, basta executar:

yarn server

ou

npm run server

- Frontend:

Com o json server rodando, podemos executar o frontend e acessar o site,com o comando:

yarn start

ou

npm start

7. Problems:

Enfrentamos alguns problemas com nossas fontes de dados. Como optamos por utilizar apenas o json-server agindo como um mock do backend, tivemos dificuldade em renderizar propriamente todas as páginas com todos os dados que desejávamos, como por exemplo, a página de “manageUsers”, ou a função de edição do perfil do usuário. Por necessitar de um sistema mais robusto para tal, optamos por deixar essas funcionalidades para a próxima entrega, apesar das páginas existirem e mostrarem dados que foram escolhidos manualmente para as mesmas.

Também enfrentamos alguns problemas menores em usar React, já que não temos tanta experiência com o framework como um todo e com todas suas bibliotecas. Um exemplo disso foi o modal que aparece quando clicamos em um produto, não tínhamos conhecimento sobre a mesma anteriormente, então tivemos certos empecilhos para utilizar tal biblioteca.

Como optamos por deixar a autenticação do usuário para o milestone 3, optamos por deixar as 3 opções de caminhos no header para Sign In, Admin e User, embora a ideia do site seja apenas ter um caminho onde dependendo do usuário logado, gerará o caminho com suas devidas permissões.

8. Comments: