E-SHOP REDESIGN

Напреден Веб Дизајн

ИЗРАБОТИЛЕ:
ЈАНА ТРПКОВСКА 211070
СТЕФАНИЈА ФИЛИПАШИЌ 213215

5 СЕПТЕМВРИ 2024

Table of Contents

1. Вовед

1.1. Цел

Проектот за редизајн на eShop апликација беше иницран со цел да се подобрат функционалноста и корисничкото искуство на веќе постоечка платформа за е-трговија. Примарната цел беше да се модернизира апликацијата со инкорпорирање на поинтуитивен кориснички интерфејс, подобрување на backend процесите и воведување на напредни функции за подобро се исполнат потребите на корисниците.

Со овој проект се обидовме и да подобриме некои ограничувања на претходниот систем, вклучувајќи недостаток на сеопфатни опции за филтрирање и подобрување на одредени интеракции со продуктите. Со користење на современи технологии и најдобри практики, проектот има за цел да обезбеди одлично искуство за корисниците и да го олесни процесот на е-трговија.

1.2. Задачи

- Подобрено корисничко искуство – вклучува дизајнирање на интуитивен систем за навигација, имплементирање на принципи за респонзивен дизајн за да се обезбеди компатибилност на различни уреди и подобрување на визуелната привлечност на апликацијата за да се направи искуството попријатно и поефикасно.
- Напредно филтрирање и пребарување – имплементирање на филтри за категории и цени, како и филтер кои самите корисници може да го прилагодат за да им се овозможи лесно да пронајдат производи кои одговараат на нивните преференци.
- Робусен backend и API – развивање на доверлив API со поддршка за нови функции на frontend, вклучувајќи филтрирање на продукти, управување со кориснички профили, и функционалност за корисничка кошничка.
- Модерна архитектура на frontend – користење на React за да се изгради динамичен, интерактивен интерфејс.

2. Технологии

Проектот користи комбинација од современи технологии да ги постигне своите цели за подобрено корисничко искуство, напредна функционалност и ефикасни backend операции.

2.1. Frontend

- React – се користи како популарна JavaScript библиотека за градење на кориснички интерфејси. Архитектурата базирана на компоненти овозможува креирање на компоненти кои може да се реискористуваат, како и ефикасно управување со состојбата. Виртуелниот DOM на React овозможува ажурирањата на корисничкиот интерфејс да се вршат брзо и ефикасно.
- CSS и SCSS – се користат за стилизирање на апликацијата. SCSS е искористен поради неговите напредни функционалности како што се променливи, вгнездени правила и миксини, кои помагаат во одржувањето на модуларен и организиран лист со стилови. CSS се користи заедно со SCSS за да се обезбеди кохезивен и респонзивен дизајн, прилагодувајќи ја апликацијата за различни уреди и големини на екранот.
- React Router – се користи за навигација во апликацијата. Се олеснува создавањето на single page апликација со повеќе рути, овозможувајќи им на корисниците да се движат помеѓу различни прикази.

2.2. Backend

- Spring Boot – се користи како рамка која го поедноставува создавањето на stand-alone апликации.
- Java Persistence API (JPA) – се користи за управување со операциите на базата на податоци на поапстрактен начин.
- PostgreSQL – се користи како релациона база на податоци.
- Spring Security – се користи за автентикација и авторизација.

3. Имплементација на backend

3.1. Клучни функционалности

- Управување со корисници – вклучува функционалности за регистрација на корисник, најавување и управување со профилот. Се справува со автентикација на корисникот користејќи Spring Security, ги енкриптира лозинките за безбедност и управува со корисничките улоги за да го контролира пристапот до различни делови од апликацијата.

- Управување со продукти – поддржува сеопфатни функции за управување со продукти, вклучувајќи ја и можноста за додавање, ажурирање и бришење продукти, преглед на делати за секој продукт. Имплементацијата вклучува endpoints за преземање информации за продуктот и ажурирање детали за истиот.
- Филтрирање – за да се подобри корисничкото искуство овозможена е функционалност за филтрирање на производите. Има повеќе можности за филтрирање. Може да се филтрира според цена, според боја, или пак може корисникот сам да одбере спецификации за продуктот, како на пример дали станува за машка или женска облека, да одбере специфичен продукт како фустан или панталони и да даде спецификации да должината на продуктот, доколку станува збор за фустан или некоја блуза дали да е со кратки или долги ракави и сл.
- Кошничка – овозможена е и функционалност за корисничка кошничка во која корисникот може да додава продукти кои му се допаѓаат. Има можност да менува количина на самите продукти додека се тие во кошничка, како и големината на продуктите и има опција за да ја плати нарачката на продуктите кои се наоѓаат во кошничката и со тоа таа се празни.

3.2. Интеграција на податоци и import

За беспрекорна интеграција на постоечките продукти за кои е искористено web scraping, имплементиран е механизам за import на податоци од CSV датотеки во базата на податоци. Клучните аспекти на процесот на интеграција на податоци вклучуваат:

- CSV парсирање – развиен е CSV парсер за читање податоците од CSV-датотеки. Овој парсер ги конвертира податоците од CSV во Java објекти компатибилни со шемата на базата на податоци.
- Валидација на податоци – пред import, податоците се валидираат за да се обезбеди точност и конзистентност, како што е проверка на полиња што недостасуваат или неважечки вредности.

3.3. Бизнис логика и сервиси

Сервисите се дизајнирани да ја енкапсулираат основната бизнис логика на апликацијата. Оваа поделба на грижи гарантира дека деловните правила и операции се управуваат независно од нивоата на контролерот и пристапот до податоци. Клучните сервиси вклучуваат:

- Product Service – управува со операции поврзани со продуктите, како што се додавање, ажурирање и преземање производи. Вклучува бизнис правила за валидација на продукти и пресметки на цени.
- User Service – управува со операции поврзани со корисникот, вклучувајќи регистрација, најавување и управување со ингеренции и осигурување дека тие се безбедно складирани и управувани.
- Cart Service – управува со операциите на корисничка кошничка, вклучувајќи додавање ставки, пресметување на вкупните износи и примена на попусти.

3.4. Безбедност

Безбедноста е клучен аспект во имплементацијата на backend. За тоа е искористен Spring Security кој е конфигуриран да спроведува силни политики за автентикација и авторизација. Ова вклучува:

- Енкрипција на лозинка – корисничките лозинки се енкриптираат со користење на BCrypt хаширање за да се осигура дека се безбедно складирани.
- Контрола на пристап заснована на улоги – на различни кориснички улоги (Admin, User) им се доделуваат специфични дозволи за пристап или менување на одредени делови од апликацијата.
- Безбедни endpoints на API – крајните точки се заштитени за да се осигура дека само овластените корисници можат да вршат чувствителни операции.

4. Имплементација на frontend

Овој дел е дизајниран да обезбеди динамичен, лесен интерфејс кој користи модерни веб технологии. Ги детализира основните компоненти на имплементацијата на frontend, вклучувајќи архитектура, дизајн на компоненти, управување со состојби, рутирање, стилизирање и интеграција со backend.

4.1. Архитектура на frontend

Архитектурата на frontend е изградена со помош на React, популарна JavaScript библиотека за градење кориснички интерфејси. Клучните аспекти на архитектурата вклучуваат:

- Архитектура базирана на компоненти – апликацијата е структурирана во компоненти, од кои секоја е одговорна за одреден дел од интерфејсот. Компонентите се организирани хиерархиски, со родителски компоненти кои управуваат со состојбата и деца компоненти кои ракуваат со екранот и интеракциите со корисникот.
- React Hooks – hooks како useState, useEffect, useLayoutEffect се искористени за управување со состојбата и несаканите ефекти во функционалните компоненти. Тие обезбедуваат начин за справување со stateful логиката и животниот циклус без потреба од класни компоненти.
- Разделување на код (Code splitting) – се користи за да се оптимизираат перформансите и е имплементирано со помош на можностите за lazy loading на React. Овој пристап осигурува дека само потребниот код е вчитан кога е потребно намалувајќи ги почетните времиња на вчитување.

4.2. Компоненти

4.2.1. App.js

Оваа компонента е централна компонента за рутирање, управување со состојби и вчитување податоци во апликацијата. Како класна компонента, таа координира различни операции поврзани со производи, категории, опции за бои и улоги на корисници. Исто така, интегрира повеќе клучни компоненти, како што се Product List, Product Details, Shopping Cart и страниците за автентикација (најава/регистрација), истовремено обезбедувајќи непречена навигација низ апликацијата.

4.2.1.1. Структура и состојба на компонентата

```
constructor(props) {
    super(props);
    this.state = {
        roles: [],
        products: [],
        productColorOptions: [],
        productImages: [],
        colors: [],
        selectedProduct: {},
        selectedProductColorOptions: [],
        selectedProductImages: [],
        selectedShoppingCart: {},
        categoriesWomen: [],
        categoriesMen: [],
        categoriesGirls: [],
```

```
      categoriesBoys: [],
      selectedPerson: {},
      selectedClothing: {},
      dataLoaded: false,
      username: null,
    }
}
```

- Управување со состојбата - компонентата App.js управува со сложена состојба која вклучува различни ентитети:
  - products – ја содржи листата на сите производи преземени од backend.
  - productColorOptions, productImages, colors – содржат дополнителни податоци за производите, вклучувајќи варијанти на бои за производите и слики.
  - selectedProduct, selectedShoppingCart – управува со тековно избраниот производ и деталите за количката.
  - categoriesWomen, categoriesMen, categoriesGirls, categoriesBoys – ги чува категориите на облека филтрирани по пол/возрасни групи.
  - roles, username – управува со информации поврзани со корисникот како што се достапните улоги и корисничкото име на тековниот корисник.
  - dataLoaded – се користи како flag за одредување кога потребните податоци се преземени од backend пред да се рендерира апликацијата.

4.2.1.2. Клучни методи

- Вчитување податоци – функцијата loadData() комбинира повеќе методи за преземање податоци, осигурувајќи дека производите, опциите за производи, сликите, категориите, боите, улогите и корисничкото име се асинхроно вчитани од backend. Ова гарантира дека состојбата на апликацијата е целосно пополнета кога компонентата е во mounted состојба.
- Операции со продукти
  - loadProducts(): Ги презема сите производи од backend.
  - getProduct(id): Добива еден производ според неговиот ID, заедно со придружните опции за боја и слики.
  - searchProducts(term): Пребарува производи по даден термин.
- Операции со филтри – достапни се неколку методи за филтрирање на листата на производи:

- filterProductsByPersonAndClothingCategory(person, clothing) – филтрира продукти врз основа на избраната личност (т.е маж, жена, девојче) и категорија на облека.
- filterPrice(min, max) – филтрира производи во одреден опсег на цени.
- filterColors(colors) – ги филтрира производите според нивната боја.
- filterCustom(person, clothing, length, sleeves, neckline, waist, fit) – овозможува поконкретно филтрирање врз основа на различни атрибути на производот.

- Операции со корисничка кошничка
  - getShoppingCart(username) – ги презема деталите за количката за наведениот корисник.
  - addToCart(productId, colorOptionId, quantity, size, navigate) – додава производ во кошничката на корисникот.
  - editProductInCart(id, productId, quantity, size, navigate) – ажурира продукт кој се наоѓа во кошничка.
  - removeProduct(id, navigate) – отстранува продукт од кошничка.
  - clearShoppingCart(username) – ја брише целата кошничка на корисникот.
- Управување со категорија
  - loadCategories() – ги вчитува сите категории на облека за мажи, жени, девојчиња од backend.

4.2.1.3. Рутирање

Компонентата користи react-router-dom за да дефинира повеќе рути, обезбедувајќи различни прикази за различни страници во апликацијата:

- /products – прикажува листа од сите производи.
- /products/:person – прикажува производи филтрирани по одредена категорија на лица (на пр. жени, мажи).
- /products/:person/:category – ги прикажува производите филтрирани и според категорија на лица и категорија на облека.
- /product/:id – ги прикажува деталите за избраниот производ.
- /shopping-cart/:username – ја прикажува кошничката на најавениот корисник.
- /login и /register – прикажуваат форми за најавување и регистрација на корисник.
- /: - почетна страница.

4.2.1.4. Динамичност

- Филтрирање – корисниците можат да ги филтрираат производите според различни критериуми, вклучувајќи категорија на личност, опсег на цени, бои и приспособени атрибути на облека (на пр. должина на ракавите, деколте, величина).
- Корисничка кошничка – функционалноста на кошничката е целосно интегрирана, овозможувајќи им на корисниците да додаваат, уредуваат и отстрануваат производи во нивната кошничка.
- Состојба на вчитување – пред да се вчитаат податоците, се прикажува едноставна порака Loading..., па откако ќе се преземат сите потребни податоци, се прикажува целосната апликација.

4.2.1.5. Клучни компоненти

- Header – ја прикажува лентата за навигација, вклучувајќи функционалности специфични за корисникот како што се пребарување и филтрирање.
- Products – управува со приказот на листа од производи и се справува со интеракции како филтрирање и прегледување детали за производот.
- ProductDetails – ги прикажува деталите за одреден производ вклучувајќи опции за боја и слики.
- ShoppingCart – управува со кошничката на корисникот овозможувајќи управување со производите во истата.
- Login и Register – управува со автентикација на корисникот.
- Home и Footer – обезбедуваат дополнителна структура на секциите на почетната страница и footer на апликацијата.

4.2.2. Login.js, Register.js

- Login – оваа компонента обезбедува интерфејс каде што корисниците можат да ги внесат своите ингеренции за да добијат пристап до апликацијата. Содржи полиња корисничко име и лозинка на корисникот. По поднесувањето, компонентата ги испраќа ингеренциите до backend за верификација. Ако најавувањето е успешно, корисникот се автентицира и се креира JWT токен за одржување на состојбата низ различни делови од апликацијата.

  Компонентата содржи:

  - Механизам за справување со грешки при погрешно најавување.

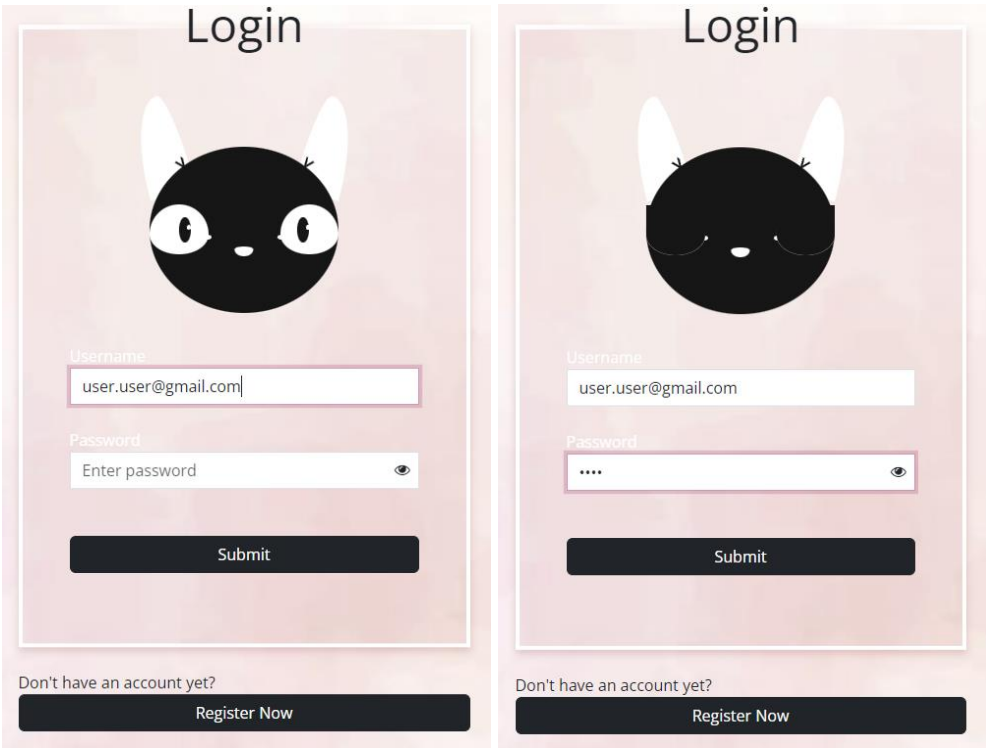
- Пристап до страната за регистрација, доколку корисникот нема постоечки кориснички профил.

- Register – оваа компонента им овозможува на новите корисници да креираат кориснички профил со пополнување на форма за регистрација со детали како што се нивното корисничко име и лозинка. Откако ќе се поднесат, податоците од формата се валидираат, а информациите се испраќаат до backend за да се создаде нов кориснички запис.

  Компонентата содржи:

  - Валидација на лозинката.
  - Верификација на корисничко име за да се осигура дека корисникот ќе внесе валидно и единствено корисничко име.
  - Пренасочување на корисникот до страната за најава, по успешна регистрација.

Дополнително, во овие две компоненти уникатна и интерактивна карактеристика е употребата на динамичниот карактер, кој е стилизиран со помош на SCSS, за подобрување на корисничкото искуство за време на автентикацијата.



Клучни елементи при имплементација на карактерот:

- CSS анимации – карактерот ги менува изразите или позите врз основа на дејствата на корисникот. На пример, ако корисникот се фокусира на полето за внесување лозинка, карактерот ги „покрива очите“ за да покаже доверливост.
- Респонсивен дизајн – карактерот ги прилагодува позиционирањето и големината врз основа на димензиите на екранот, обезбедувајќи непречено искуство на различни уреди. SCSS помага во одржување на скалабилност и повторна употреба на стиловите.
- Стилизирање на карактерот – користејќи SCSS, карактерот е направен со сложени детали како што се шеми на бои, сенки и транзиции. Променливите и mixins се користат за да се осигура дека стиловите се конзистентни и може лесно да се променат доколку е потребно. На пример:
    - Mixins се користат за управување со различни состојби (среќни, неутрални, збунети) со транзиции.
    - Променливите ги контролираат боите, големините и димензиите за да го направат карактерот да може лесно да се приспособува.

4.2.3. Header.js

Оваа компонента игра голема улога во корисничкиот интерфејс на апликацијата и е одговорна за навигација, функционалност за пребарување и автентикација на корисникот. Динамички се прилагодува на различните кориснички интеракции и е многу значајна во подобрувањето на целокупното корисничко искуство.

4.2.3.1. Главни функционалности

- Респонзивна навигација - Заглавието содржи линкови за навигација до различни категории на производи (на пр. жени, мажи, девојки), кои им овозможуваат на корисниците да ги филтрираат производите врз основа на нивната избрана категорија. Овие категории се прикажани како копчиња и опаѓачки менија обезбедувајќи непречено искуство во прелистувањето.
    - Секоја категорија има свој сет на поткатегории кои динамично се прикажуваат во опаѓачко мени врз основа на податоците пренесени преку props.
    - Функцијата onNavigateToCategory се справува со логиката за навигација до соодветните страници со категорија и подкатегорија. Ја користи useNavigate hook од React Router за навигација на корисниците до посакуваните производи.

- Динамично прилагодување на padding на заглавието – useLayoutEffect hook обезбедува динамичко прилагодување на padding на заглавието врз основа на висината. Ова е особено корисно кога заглавието е фиксирано на врвот на страницата, бидејќи спречува содржината под него да биде поклопена.
- Функционалност за пребарување – компонентата вклучува лента за пребарување која им овозможува на корисниците да пребаруваат производи по име или клучен збор.
    - Функцијата handleSearchChange го ажурира терминот за пребарување додека корисникот пишува во полето за внесување, додека функцијата handleSearchSubmit се справува со поднесувањето на формата, пренесувајќи го терминот за пребарување во функцијата searchProducts и потоа навигира до страницата со производи за да се прикажат резултатите од пребарувањето.
- Автентикација на корисници и корисничка кошничка
    - Заглавието динамично ги прикажува опциите за автентикација врз основа на тоа дали корисникот е најавен или не, одредено од присуството на JSON Web Token (JWT) во localStorage.
    - Ако корисникот е најавен, компонентата го прикажува корисничкото име, врската до количката (со икона за количка) и копче за одјавување. Копчето за одјавување го брише JWT од локалното складирање и го пренасочува корисникот на страницата за најавување.
    - Ако корисникот не е автентициран, се прикажува копче за најава, кое го пренасочува на страницата за најавување кога ќе се кликне.

4.2.4. ProductList.js

Оваа компонента е клучен дел од апликацијата што се справува со динамичниот приказ и филтрирање на податоците за продуктите. Клучните фукнционалности на оваа компонента вклучуваат обезбедување на кориснички интерфејс за листање производи во различни layouts, примена на различни филтри (цена, боја, customize) и управување со состојби на интеракција, како што се ефектите на менување на филтри.

4.2.4.1. Функционалности

- Динамичен side bar за филтрите - Страничната лента содржи филтри за цена, боја и прилагодување на продукти по избор, при што компонентите како PriceFilter, ColorFilter и CustomizeFilter се користат за логика на филтрирање.

Секој дел од филтерот може да се отвара и затвара, што му овозможува на корисникот да го прошири или да го собере врз основа на неговата желба.

Компонентата ја зачувува проширената/склопена состојба на овие делови во состојбата на проширени секции. Со кликнување на заглавието на секцијата се менува неговата видливост преку функцијата toggleExpand.

- Clear all filters – оваа опција им овозможува на корисниците да ги ресетираат сите филтри. Функцијата активира дејства за ресетирање на сите компоненти на филтерот со повикување на нивните методи за ресетирање и повикување props.clearFilters за ресетирање на листата со продукти.
- Промена на Layout на продуктите – корисниците можат да приспособат како се прикажуваат производите користејќи различни режими на приказ (една колона, три колони или четири колони). Компонентата нуди три различни опции за преглед користејќи ја функцијата handleViewChange, која соодветно го прилагодува распоредот на колоните на картичките на производите и нивните димензии.

  Прилагодувањето на приказот се визуелизира со додавање/отстранување на CSS класи на картичките на производите за да се измени нивната ширина и распоред во реално време, што озовможува респонзивност и подобрување на корисничкото искуство.
- Ефекти при hover врз продуктите – ефектот ја менува сликата на производот кога корисникот лебди над некој производ. Ова е овозможено со користење на hoveredProductId, кој го следи моментално лебдениот производ. Ако е достапна алтернативна слика за некој производ, таа се прикажува за време на лебдењето. Управувано од функциите handleMouseEnter и handleMouseLeave, кои ја ажурираат состојбата на hoveredProductId.
- Филтрирање според URL – компонентата ги слуша промените во URL (користејќи useParams за извлекување username и категорија од патеката) и соодветно ја прилагодува листата на продукти. При промена на URL, го активира props.onFilter да ги филтрира прикажаните производи врз основа на параметрите во URL.

  Ова е особено корисно за да се осигура дека прикажаните производи се совпаѓаат со избраната категорија или корисник при промени во URL.
- Приказ на продукти – продуктите се прикажуваат како картички, и ги содржат името на производот, цената и сликата. Секоја картичка динамички се префрла помеѓу главната слика на производот и алтернативна слика при hover (ако е

достапна). Ако производот има попуст, се прикажуваат и целосната цена и цената на попустот; во спротивно се прикажува само целосната цена.

- CSS, Bootstrap и респонзивен дизајн – компонентата користи различни CSS и Bootstrap класи за управување со распоредот на производи. Приказот на продуктите е респонзивен, овозможувајќи различни конфигурации на колони во зависност од опцијата за преглед избрана од корисникот.
- Детали за продукти – секоја картичка за продукт е обвиткана во Link компонента, која овозможува да се навигира до приказот на детали за секој продукт, во кој може да се прочитаат детали за материјалите од кои е составен продуктот, инструкции за перење, како и да се одбере боја, величина и да се стави тој продукт во корисничката кошничка.

4.2.5. ColorFilter.js

Оваа компонента е одговорна да им дозволи на корисниците да ги филтрираат производите по боја, користејќи checkbox полиња за да изберат една или повеќе бои и се користи во ProductList.js. Изградена е со помош на ForwardRef на React и useImperativeHandle за изложување на одредени функции на родителските компоненти, овозможувајќи да се контролира однадвор.

4.2.5.1. Клучни функционалности

- Менаџирање на состојба – компонентата одржува состојба checkboxStates за да следи кои бои се избрани. Состојбата е мапа каде што клучевите се имињата на боите, а вредностите се boolean вредности кои покажуваат дали полето за избор за таа боја е означено или не.
- Ракување со промени – кога ќе се кликне полето за избор на боја, се активира функцијата handleChange. Таа го ажурира полето во checkboxStates за соодветната боја со префрлање на нејзината булова вредност во true (или во false доколку се ресетира) со што симулира штиклирање (ресетирање) на одредена боја.
- Рендерирање на опциите за бои – компонентата динамички генерира checkbox полиња врз основа на низата props.colors, која ги содржи достапните бои за филтрирање. Секое поле за избор на боја е стилизирано според вредноста на бојата (на пр. бојата на позадината).

4.2.6. CustomizeFilter.js

Оваа компонента им овозможува на корисниците да ги приспособат опциите за облека врз основа на избрани атрибути како должина, ракави, деколте, струк и сл.

- Форми за проспособување – формата започнува со тоа што им овозможува на корисниците да изберат личност (жени, мажи, девојчиња) и тип на облека (фустан, блуза, панталони). На вториот чекор, корисниците можат да изберат специфични атрибути (должина, ракави, деколте, итн.), а овие атрибути се користат за филтрирање на производите.

  Визуелното претставување на избраните атрибути динамички се ажурира со слики, кои се прикажани врз основната слика (на пр. генеричка силуета за жени, мажи или девојки), поставувајќи ја секоја опција, додека корисниците избираат атрибути.

4.2.6.1. Клучни карактеристики

- Динамично прикажување на атрибути – во зависност од избраната личност и типот на облека, соодветните атрибути (должина, ракави, итн.) се динамично прикажани и можат да се изберат.
- Визуелна повратна информација – обезбеден е преглед во живо на избраните опции, помагајќи им на корисниците да ги видат промените за како изгледа продуктот кој тие го избираат пред филтрирање.
- Модуларни конфигурации – компонентата користи конфигурациски објект за да ги дефинира можните полиња и слики за секоја комбинација од типот на личност и облека.

4.2.7. PriceFilter.js

Оваа компонента им овозможува на корисниците да ги филтрираат продуктите кои спаѓаат во одреден опсег според нивната цена. Компонентата користи Material-UI, популарна библиотека во React, за да обезбеди лизгач за избор на опсег на цени.

4.2.7.1. Material-UI

- Box Component – ова е компонента од библиотеката Material-UI која се користи за креирање на layout компоненти. Служи како wrapper и врз неа може да се применат различни својства за стилизирање и распоред.
- Slider Component – исто така компонента од библиотеката Material-UI која обезбедува контрола на лизгач за избор на опсег на вредности.

Употребата на овие компоненти го поедноставува процесот на развој и обезбедува конзистентност со системот за дизајн на Material-UI.

4.2.7.2. Функционалност

- valuetext функција – ја конвертира вредноста на лизгачот во текстуален формат за прикажување.
- useImperativeHandle – го изложува методот на ресетирање на родителските компоненти преку refs, што го ресетира лизгачот на неговиот стандарден опсег [0, 100].
- handleChange функција – ја ажурира состојбата кога се менува вредноста на лизгачот.
- onFormSubmit функција – ја активира onFilterPrice функцијата која ги филтрира продуктите во дадениот опсег.

4.2.8. ShoppingCart.js

Оваа компонента управува и ја прикажува кошничката на корисникот, овозможувајќи интеракции како што се ажурирање на детали за производот, отстранување ставки и испразнување на целата кошничка.

4.2.8.1. Hooks

- React Hooks – useEffect, useState и useRef се користат за управување со несакани ефекти, состојби и референци.
- React Router Hooks – useParams и useNavigate се користат за рутирање и извлекување на параметрите на рутата.

4.2.8.2. Состојба и ефекти

- Состојба:
  - formData – чува податоци од формулари за секоја ставка од кошничката, вклучувајќи ID на производ, количина и големина.
- Ефекти:
  - Преземање податоци за кошничката – useEffect со getShoppingCart ги презема деталите за кошничката кога ќе се промени параметарот на корисничкото име.
  - Иницијализирање на податоците за формата – друг useEffect го иницијализира formData од shoppingCart.products секогаш кога се менуваат податоците за количката.

4.2.8.3. Пресметки

- Вкупна цена – ја пресметува вкупната цена на сите артикли во количката, земајќи ја предвид цената на попустот или целосната цена.
- Број на артикли – ги сумира количините на сите производи во количката.

4.2.9. EShopRepository.js

Оваа компонента делува како централизиран комуникациски слој помеѓу frontend и backend, користејќи Axios за правење барања за HTTP. Таа ја поедноставува комуникацијата помеѓу компонентите на frontend и API на backend, овозможувајќи преземање, ажурирање и управување со продукти преку добро дефинирани методи. Секој метод директно одговара на крајната точка на API, што го олеснува извршувањето на задачите како што се автентикација на корисникот, филтрирање на производи и управување со кошничка. Оваа компонента го одржува frontend делот чист и модуларен, бидејќи не треба директно да се справува со детали за HTTP на ниско ниво.

4.2.9.1. Цел

Компонентата е дизајнирана да ги поедностави интеракциите со backend со апстрахирање на деталите за HTTP барањата. Секоја функција во EShopService одговара на крајната точка на API, што го олеснува управувањето со корисничките дејства и операциите поврзани со производот.

4.2.9.2. Axios

Инстанцата на axios е употребена преку import од сопствена конфигурација, која вклучува основни поставки за URL-адреса, заглавија и други конфигурации потребни за правење барања за HTTP.

```
import axios from "axios";

const instance = axios.create({
    baseURL: 'http://localhost:8080/api',
    headers: {
        'Access-Control-Allow-Origin' : '*',
        'Authorization': `${localStorage.getItem("JWT")}`
    }
})

instance.interceptors.request.use(
    config => {
        const token = localStorage.getItem("JWT");
        if (token) config.headers.Authorization = `Bearer ${token}`;
        return config;
```

```
  },
  error => {
    if (error.response.status === 403) {
      console.log("interceptor error 403");
      localStorage.removeItem("JWT");
      window.location.href = '/login';
    }
    return Promise.reject(error);
  }
);
export default instance;
```

4.2.9.3. Корисничка автентикација и управување со улоги

- login – испраќа кориснички ингеренции (корисничко име, лозинка) до крајната точка /login за автентикација.
- register - Регистрира нов корисник со испраќање на неговите податоци до крајната точка /user/register.
- getUserRole и getUserUsername – ја презема улогата и корисничкото име на најавениот корисник, соодветно.

4.2.9.4. Управување со производи и кошничка

- getAllProducts, getProduct – ги презема сите производи или детали за одреден производ од backend.
- getShoppingCart, addProductToShoppingCart, editProductInShoppingCart, removeProductFromShoppingCart – управува со операции поврзани со корисничка кошничка.

4.2.9.5. Филтрирање и пребарување

- Филтрирање – filterProductsByPersonCategory, filterProductsByPrice, filterProductsByColor, filterProductsByCustom.
- Пребарување – searchProducts овозможува пребарување на производи со помош на низа за пребарување.

5. Заклучок

Овој проект вклучува редизајн на софистицирана апликација за е-трговија со функции за управување со производи, опции за филтрирање и справување со интеракциите со корисниците. Клучните компоненти вклучуваат:

- Управување со производи – имплементирано со React, овозможувајќи им на корисниците да ги филтрираат производите по категорија и цена. Компонентите CategoryFilter и PriceFilter им овозможуваат на корисниците интуитивни опции за филтрирање, користејќи Material-UI за модерен изглед и чувство.
- Корисничка кошничка – овозможува сеопфатно искуство со кошничката каде што корисниците можат да прегледуваат, уредуваат и отстрануваат производи. Исто така, ја пресметува вкупната цена и бројот на артикли, интегрирајќи го управувањето со состојба и React Router за беспрекорна навигација и интеракција со корисниците.
- Интеграција на Backend – API-то на backend поддржува операции со производи и кошнички, вклучувајќи филтрирање и ажурирање детали за кошничката. Frontend делот е во интеракција со овие API за да обезбеди ажурирања во реално време и респонзивен дизајн.

Генерално, проектот ја комбинира моќната архитектура заснована на компоненти на React со Material-UI за конзистентност на дизајнот и корисничко искуство, обезбедувајќи функционална и привлечна платформа за е-трговија.