**Cipő webáruház – Frontend Fejlesztői dokumentáció**

Tartalom

[App.jsx 1](#_Toc133788667)

[index.js 2](#_Toc133788668)

[responsive.js 3](#_Toc133788669)

[data.js 4](#_Toc133788670)

[Oldalak/Pages: 5](#_Toc133788671)

[Redux: 5](#_Toc133788672)

[Komponensek: 15](#_Toc133788673)

## App.jsx

A képen szöveg, computer, képernyőkép látható

Automatikusan generált leírás

Ez a kód az App komponenst definiálja, ami a fő alkalmazást reprezentálja. Az App komponens egy többszintű útvonaltervezést tartalmaz, amely az URL-eket a megfelelő komponensekkel rendeli össze a megjelenítéshez.

A Routes komponens a többszintű útvonaltervezés fő komponense, amely tartalmazza a Route komponenseket, amelyek az útvonalakat határozzák meg. Az exact attribútummal meghatározott útvonalak pontosan illeszkednek az URL-ekkel, és a path attribútum határozza meg az URL-eket, az element attribútum pedig a megfelelő komponenst adja meg az adott útvonalon való megjelenítéshez.

A kód tehát egy SPA (Single Page Application) alapján működő webalkalmazást hoz létre, amelynek navigációja és útvonaltervezése a Routes és a Route komponensek segítségével valósul meg.

## index.js

A képen szöveg látható

Automatikusan generált leírás

Ez a kód az alkalmazás belépési pontja, a ReactDOM.createRoot metódus segítségével létrehoz egy új gyökérelemet, amelynek az azonosítója "root". A gyökérelem tartalma a Provider komponens, amelynek az általános szerepe az alkalmazás állapotának kezelése, a Redux store-nak a komponensek közötti kommunikációhoz történő elérhetővé tétele. A store egy Redux által biztosított objektum, amely az alkalmazás állapotát tárolja.

A PersistGate komponens a React-Redux-Firebase által biztosított funkciók egyike, amely a Redux store persistálására használható. Ez azt jelenti, hogy az alkalmazás állapotát a PersistGate segítségével a helyi tárhelyen tároljuk, így az felhasználói munkamenetek vagy a böngésző újraindítása után is megmarad.

A BrowserRouter komponens az alkalmazás böngészőbéli útvonalait kezeli, és hozzárendeli azokhoz az egyes komponenseket.

Az App komponens az alkalmazás főkomponense, amely tartalmazza az összes egyes útvonalhoz rendelt komponenst. Az alkalmazás általános felépítése tehát egy SPA (Single Page Application), amelyben az egyes oldalak az útvonalaknak megfelelően jelennek meg és töltenek be.

A reportWebVitals függvény egy analitikai eszköz, amely a teljesítmény mérésére szolgál. A függvény meghívása lehetővé teszi a teljesítménymérési adatok rögzítését és küldését az analitikai szolgáltatók felé.

## responsive.js

A képen szöveg látható

Automatikusan generált leírás

Ez a kód a styled-components csomagból importálja a css függvényt. Ezután három különböző médiaképernyő méretre definiál stílusokat. A mobile a képernyők kisebb mint 599px szélességére alkalmazza a megadott stílusokat. A tablet a 600px-tól 1600px-ig terjedő méretű képernyőkre alkalmazza őket, míg a desktop a 1601px-nél szélesebb képernyőkre alkalmazza őket. Ezek a stílusok azt jelentik, hogy a props-ban megadott CSS szabályokat alkalmazzák az adott méretű képernyőkön.

## data.js

A képen szöveg látható

Automatikusan generált leírás

## Oldalak/Pages:

A képen szöveg látható

Automatikusan generált leírás

A fenti kód egy React komponenst definiál, amely a Home nevet viseli. A komponens egy egyszerű HTML struktúrát ír le, amely tartalmazza a weboldal felső navigációs sávját, egy slideshow-t, kategóriákat, valamint a legújabb és a legnépszerűbb termékeket, továbbá egy hírlevél feliratkozó formot és az oldal láblécét.

A komponens a Navbar, Slideshow, Categories, NewProducts, PopularProducts, Newsletter és Footer nevű komponenseket rendezi egymás után a DOM-ban. Ezeknek a komponenseknek az implementációja valószínűleg további React komponensekből és/vagy HTML elemekből áll.

## Redux:

A képen szöveg látható

Automatikusan generált leírás

A fenti kód egy cartSlice nevű Redux slice-ot hoz létre. Ez az objektum tartalmazza a products, quantity és total állapotokat. Az addProduct és a removeProduct metódusok módosítják ezeket az állapotokat.

Az addProduct metódus egy új terméket ad a kosárhoz. Az állapot quantity értéke növekszik eggyel, a products tömb pedig az új terméket tartalmazó objektummal bővül, míg az action.payload által reprezentált termék ára hozzáadódik az állapot total értékéhez.

Az removeProduct metódus eltávolít egy terméket a kosárból. Először meghatározza a törlendő termék indexét a products tömbben. Ezután eltávolítja a terméket a tömbből, majd kiszámolja az eltávolított termék árát (removedProductPrice) és darabszámát (removedProductQuantity). Végül az állapot total és quantity értékei csökkentik az eltávolított termék árát és darabszámát.

Az utolsó három sor az addProduct és a removeProduct metódusokat exportálja, és a slice reducert is exportálja.

A képen szöveg látható

Automatikusan generált leírás

Ez a kód egy Redux store-ot és egy persistált store-ot hoz létre. A persistConfig objektum definiálja a persistált store-ot, amely a cart kulcsot használja és a storage-t használja a tároláshoz. Az így definiált persistedCartReducer az eredeti cartReducer-en alapul, és beállítja a persistálási konfigurációt a persistConfig szerint.

Az configureStore függvény létrehoz egy Redux store-t a persistált reducerekkel és a middleware konfigurációjával. A serializableCheck middleware beállításai kizárják azokat az akciókat, amelyek nem szerializálhatóak, például a FLUSH, REHYDRATE, PAUSE, PERSIST, PURGE és REGISTER. Az így létrehozott store és a persistálás funkciója elérhető az export parancsok segítségével, amelyek lehetővé teszik a store és a persistor importálását más modulokban.

A képen szöveg látható

Automatikusan generált leírás

Ez a kód egy kosár komponenst tartalmaz, ami megjeleníti a felhasználó kosarában lévő termékeket, azok adatait, árát és lehetővé teszi a szállítási opciók közötti választást. A kosár tartalmaz egy "Tovább vásárlás" gombot, amely visszairányít a termékek oldalra, valamint egy "Fizetés" gombot, amely megnyitja a fizetési felületet. A kosárban lévő termékeket lehet eltávolítani, és a szállítási opciókat lehet változtatni. A kód Redux store-t használ a kosár állapotának tárolására, és a persistStore segítségével biztosítja, hogy az állapot megmaradjon az oldal frissítése vagy újranyitása után.

A képen szöveg látható

Automatikusan generált leírás

Ez a kód egy React komponenst definiál "Info" néven, amely egy weboldal "Rólunk" oldalának tartalmát jeleníti meg. A komponensben található egy navigációs sáv, egy cím, szövegek, egy kép, egy térkép és egy lábléc. Az oldal bemutatja a Foot Frenzy nevű online cipőboltot, amely kiváló minőségű és divatos lábbeliket kínál a vásárlóknak. Az oldal továbbá említi, hogy a Foot Frenzy egy közösséget is jelent, amelyet a sneaker kultúra és érdeklődés hozott össze. A térkép mutatja az üzlet helyét, valamint a nyitvatartási időket is megadja. A lábléc tartalmazza a weboldal elérhetőségeit.

A képen szöveg látható

Automatikusan generált leírás

Ez egy React komponens, amely egy felhasználó bejelentkezési űrlapot jelenít meg a weboldalon. Először ellenőrzi, hogy van-e már eltárolt hozzáférési token a localStorage-ban, és ha van, akkor átirányítja a felhasználót a profil oldalra. Ha nincs, akkor megjeleníti a bejelentkezési űrlapot, és ha a felhasználó megadja az adatokat és elküldi a formot, akkor a komponens elküldi a bejelentkezési adatokat a szervernek, és ha a bejelentkezés sikeres volt, a komponens eltárolja a hozzáférési tokent a localStorage-ban, majd átirányítja a felhasználót a kezdőoldalra. Ha a bejelentkezés sikertelen volt, akkor a komponens megjeleníti az hibaüzenetet.

A képen szöveg látható

Automatikusan generált leírás

Ez egy React komponens, amely egy webshop termékeinek listáját jeleníti meg. A komponens a következőket tartalmazza:

* useLocation hook: Az URL útvonalának lekérdezésére használják. Az aktuális URL útvonalat meghatározza, majd eltárolja a location változóban.
* cat változó: A kategória azonosítóját tárolja az URL-ben. Az location.pathname segítségével kinyeri az útvonalból a kategória azonosítóját.
* filters állapot: Egy objektum, amely a különböző szűrők értékeit tartalmazza, amelyeket a felhasználó választ. Az állapotot az handleFilters függvény állítja be.
* sort állapot: Az aktuális rendezési módot tárolja, az alapértelmezett értéke "newest".
* handleFilters függvény: Egy eseménykezelő függvény, amely a felhasználó által választott szűrők értékeit állítja be az állapotban. Az állapotot egy objektum formájában tárolja.
* Products komponens: A termékek listáját jeleníti meg a megfelelő kategória és szűrők alapján.

A komponens tartalmaz egy FilterContainer div-et, amely két Filter div-et tartalmaz. Az első Filter div a kategória szűrőt tartalmazza, amely csak akkor jelenik meg, ha az URL-ben nincs kategória azonosító. A második Filter div az ár szerinti rendezést tartalmazza.

A képen szöveg látható

Automatikusan generált leírás

Ez a kód egy "Profile" nevű React komponenst definiál. A komponens a felhasználói fiók oldalát jeleníti meg.

A "user" változó értékét a "localStorage" segítségével kapja meg, ami a bejelentkezett felhasználó adatait tartalmazza. Ezután kinyeri a felhasználó felhasználónevét a "username" változóba, és ezt a felhasználónevet jeleníti meg az oldalon.

A képen szöveg látható

Automatikusan generált leírás

Ez a kód egy űrlapot tartalmazó React komponens, amely lehetővé teszi a felhasználók számára, hogy regisztráljanak az alkalmazásba. A regisztráció során a felhasználónév, az e-mail cím és a jelszó megadása kötelező, és az űrlap elküldése előtt minden mezőt validálnak.

Az űrlap elküldése után a kód egy POST kérést küld a "<http://localhost:3000/register>" címre a felhasználó által megadott adatokkal, és ha a regisztráció sikeres volt, akkor egy üzenetet jelenít meg a felhasználónak, majd 6 másodperc után átirányítja a felhasználót a bejelentkezési oldalra. Ha a regisztráció sikertelen volt, akkor az alkalmazás hibát jelent, és egy hibaüzenetet jelenít meg a felhasználónak.

A regisztrációs űrlap megjelenése a következőképpen épül fel: felhasználónév, e-mail cím és jelszó mezők, valamint egy adatvédelmi szabályzatra vonatkozó szöveges megállapodás, amelynek elfogadása szükséges a regisztrációhoz. Az űrlap tartalmaz egy "FIÓK LÉTREHOZÁSA" gombot, amely elküldi az űrlapot a szervernek. Ha valamilyen hiba történik, akkor az alkalmazás hibaüzenetet jelenít meg, ha a regisztráció sikeres volt, akkor pedig egy üzenetet jelenít meg a felhasználónak.

A képen szöveg látható

Automatikusan generált leírás

Ez egy React komponens, amely egyetlen terméket jelenít meg az alkalmazásban. A komponens a useParams hook segítségével megszerzi az URL-ben található termék azonosítóját, majd az azt azonosítóhoz tartozó adatokat az useState hook segítségével betölti. A useEffect hook segítségével az adatok betöltése a komponens mountolásakor történik meg. A komponens a felhasználó által kiválasztott termék méretét a useState hook segítségével állítja be. A felhasználó által történő kattintások kezelésére számos függvényt definiál, mint például a kosárba helyezéshez szükséges handleCartClick függvényt, a képek közötti váltást kezelő handlePreviewClick függvényt, és a kedvelt termékekhez adáshoz szükséges handleLikeClick függvényt. Ezenkívül a komponens megjeleníti a termék nevét, árát, leírását és egy felugró ablakot a kosárba helyezés sikeres befejezésének jelzésére. A felhasználó azonosítója a localStorage-ban tárolódik, és akkor jelenik meg, ha a felhasználó be van jelentkezve. A komponens stílusát a styled-components segítségével határozzuk meg.

## Komponensek:

A képen szöveg látható

Automatikusan generált leírás

Ez a "Categories" nevű komponens egy weboldalon megjelenít egy szöveget és kategóriákat, amelyek között a felhasználó választhat. A kategóriák listája az előre meghatározott "categories" tömbben található, amelyekhez a "CategoryItem" nevű alkomponens jeleníti meg a tartalmat. A komponens egy "Text" nevű elemet is tartalmaz, amely a felhasználót arra ösztönzi, hogy válasszon a kategóriák közül.

A képen szöveg látható

Automatikusan generált leírás

Ez a kódrészlet egy React komponens, amely megjeleníti az egyik kategória adatait a kategórialistában.

A CategoryItem komponens egy objektumot vár, amely az adott kategóriához tartozó információkat tartalmazza. Az objektum a következő tulajdonságokkal rendelkezik: "id", "cat", "img" és "title".

A komponens egy linket jelenít meg, amely az adott kategóriához tartozó termékek oldalára irányítja a felhasználót. A link megjelenítése egy div-containeren keresztül történik, amely magában foglalja a kategória címét és egy képet.

Az item objektum alapján megjelenítjük a kategória címét, és a hozzá tartozó képet. A Link komponensnek egy dinamikus útvonalat adunk meg a to kötőjel után, amely az adott kategória nevére vonatkozik, amelyet az "item.cat" segítségével kapunk meg.

Összességében, a CategoryItem komponens megjeleníti egy kategória adatait a kategórialistában, beleértve a címet és a képet, és lehetővé teszi a felhasználó számára, hogy az adott kategória termékeihez navigáljon.

A képen szöveg látható

Automatikusan generált leírás

Ez egy React komponens egy fizetési űrlaphoz. Az űrlap több input mezőt tartalmaz, hogy a felhasználó megadja a szállítási címét és fizetési adatait. Amikor a felhasználó elküldi az űrlapot, a komponens sorozat API kérést küld létrehozni egy rendelést, annak a hozzá tartozó tételeket és frissíti a készletet minden kiválasztott termék esetén.

A komponens több állapotváltozót is definiál, beleértve a showCheckout, addressObj, orderObj változókat, valamint az input mezők értékeit, mint például a streetAddress, city és postalCode. Az állapotot a useState hook használatával frissítik.

A komponens több funkciót is definiál, beleértve a handleCloseClick, handleCheckout, validatePostalCode, validateCardNumber, validateExpire és validateSecurity funkciókat. Ezek a funkciók az input alapján frissítik az állapotot és az input ellenőrzésére is használják, mielőtt az űrlapot elküldenék.

A handleCloseClick funkció beállítja a showCheckout értékét hamisra, majd meghívja a handleClose prop függvényt, amit a komponens szülője adott át neki.

A handleCheckout funkció akkor hívódik meg, amikor a felhasználó elküldi az űrlapot. Először az input ellenőrzése történik a validatePostalCode, validateCardNumber, validateExpire és validateSecurity funkciókkal. Ha ezek bármelyike hamis értéket ad vissza, akkor egy figyelmeztetést jelenít meg és a funkció visszatér anélkül, hogy bármilyen API kérést küldene.

Ha az input érvényes, akkor a funkció sorozat API kérést küld az adatbázisba cím, rendelés, rendelési tételek és a készlet frissítése érdekében. Ezek a kérések a fetch funkciót használják a backend API-hoz való kérésekhez. Ha bármelyik kérés nem sikerül, akkor a funkció hibaüzenetet ír a konzolra.

Végül, ha minden API kérés sikerrel jár, akkor a funkció figyelmeztetést jelenít meg a sikeres vásárlásról, törli a kosarat a helyi tárolóból és átirányítja a felhasználót a főoldalra 3 másodperces késleltetéssel.

A validatePostalCode, a validateCardNumber, a validateExpire és a validateSecurity függvények a felhasználó által megadott bevitel érvényesítésére szolgálnak. Minden függvény logikai értéket ad vissza, jelezve, hogy a bemenet érvényes-e, és frissít egy állapotváltozót, hogy hibaüzenetet jelenítsen meg, ha a bemenet érvénytelen. Ha a bemenet érvényes, a hibaüzenet üres karakterláncra van állítva.

A képen szöveg látható

Automatikusan generált leírás

Ez egy DropdownMenu nevű React komponens, amely egy menüt jelenít meg. A menü gombra kattintva megjelenik a menü, amelyben néhány menüpont található. Ha a felhasználó valamelyik menüpontot kiválasztja, akkor az átirányítja a felhasználót a megfelelő oldalra.

A komponens az useState hookot használja a anchorEl állapot változó tárolására, amely meghatározza a menü megjelenítésének helyét. Az állapotváltozó null értékkel kezdődik.

A handleClick függvényt akkor hívja meg a menü gombra kattintás esetén, amely beállítja az anchorEl állapotváltozót a kattintás helyére, így megjeleníti a menüt.

A handleClose függvény akkor hívódik meg, amikor a felhasználó kiválaszt valamilyen menüpontot, vagy a menü bezárása gombra kattint. Ez a függvény beállítja az anchorEl állapotváltozót null értékre, így eltűnik a menü.

A képen szöveg látható

Automatikusan generált leírás

A LikeButton komponens egy gombot jelenít meg, amely lehetővé teszi a felhasználók számára, hogy kedveljék vagy kedveljék az adott terméket. A gombra kattintva a komponens küld egy API kérést a kedvelés státuszának lekérdezésére, majd a gombot a megfelelő állapotban állítja be.

Ha a gombra kattintva az állapota már kedvelt, akkor a komponens egy másik API kérést küld a kedvelés eltávolítására. Ha az állapota még nem kedvelt, akkor a komponens egy API kérést küld a kedvelés hozzáadására.

Az API kérések fetch segítségével történnek, és a userId és productId prop-ok alapján történnek. A komponens egy useEffect hookot használ, amely lefut, amikor a userId vagy a productId változik.

Ha a felhasználó nincs bejelentkezve, a gomb letiltva van, és egy Tooltip segítségével információ jelenik meg arról, hogy előbb be kell jelentkeznie.

Összességében a LikeButton komponens segítségével a felhasználók lehetőséget kapnak arra, hogy kedveljék vagy kedveljék a termékeket, amelyeket megtekintenek.

A képen szöveg látható

Automatikusan generált leírás

Ez egy React komponens, amely a felhasználó kedvelt elemek listáját jeleníti meg.

A komponens useEffect hookot használ adatai lekéréséhez. Amikor a komponens létrejön vagy a userId változik, az useEffect hook végrehajtja a fetchData függvényt, amely aszinkron módon adatokat kér le a szerverről. Amíg a kérés folyamatban van, a komponens a loading állapotot mutatja. Ha az adatok sikeresen letöltődtek, azokat a data állapotban tárolja, és azokat megjeleníti az oldalon. Ha hiba történik a lekérés során, a komponens az error állapotot mutatja.

Az oldalra az adatok sikeres letöltése után egy listaelem kerül, amely a kedvelt elemeket tartalmazza. A kedvelt elemeket a data állapotban tárolt adatokból jeleníti meg a map függvény segítségével.

A képen szöveg látható

Automatikusan generált leírás

A fenti kódrészlet egy LikedItem komponenst definiál, amely egy kedvelt tételt jelenít meg a felhasználó kedvenc termékeinek listájában. A komponens egy objektumot kap item néven, amely az adott kedvelt tételhez kapcsolódó adatokat tartalmazza.

A useNavigate hook segítségével a komponens navigálhat a megadott útvonalra, ha a felhasználó rákattint a kedvelt tételre. Az item objektum productId tulajdonságának értékét felhasználva az útvonal dinamikusan generálódik.

Az Image komponens megjeleníti a kedvelt tétel első képét, a Title komponens a tétel nevét jeleníti meg. Az Info komponensnek nincs megadva tartalma a kódrészletben, de vélhetően kiegészítő információkat jelenítene meg a kedvelt tételről. A Container komponens a kedvelt tétel megjelenítésére szolgáló tartály, amelyre rá lehet kattintani a navigáláshoz.

A képen szöveg látható

Automatikusan generált leírás

A függvény egy kijelentkezési gombot hoz létre, amely segítségével a felhasználó kijelentkezhet az alkalmazásból. Az onClick eseménykezelő segítségével meghívódik a handleLogout() függvény, amely eltávolítja a felhasználó autentikációs tokenjét és a felhasználói adatokat a helyi tárolóból (local storage), majd újratölti az aktuális oldalt, hogy kikényszerítse a kijelentkezést. A gomb szövege "KIJELENTKEZÉS".

A képen szöveg látható

Automatikusan generált leírás

Ez a kód egy Google térképet jelenít meg egy konténerben. A térkép középpontja a defaultCenter koordinátáin található. Az api key-t a LoadScript komponens által biztosított googleMapsApiKey property-ben kell megadni. A térkép egyetlen markert tartalmaz, amelynek a pozíciója megegyezik a térkép középpontjával. A mapStyles objektum meghatározza a térkép stílusát, és a height és width tulajdonságok beállítják a konténer méretét. A komponens a react-google-maps csomagot használja, amely az npm csomagkezelővel telepíthető.

A képen szöveg látható

Automatikusan generált leírás

A fenti kódrészlet egy Navbar komponenst definiál, ami a weboldal navigációs sávját reprezentálja. A komponens a következő részekből áll:

* const quantity = useSelector((state) => state.cart.quantity);: A Redux store-ból lekéri a kosárban lévő termékek számát.
* const isDesktop = useMediaQuery("(min-width:1024px)");: A React Responsive Library segítségével lekérdezi, hogy az eszköz, amelyen a weboldal megnyitásra kerül, nagyobb-e, mint 1024 pixel széles. Ha igen, akkor desktop eszközről van szó.
* A <Wrapper>, <Left>, <Center>, és <Right> komponensekkel megadja a navigációs sáv alapstruktúráját.
* A <Language> komponens megjeleníti a weboldal nyelvét.
* A <Logo> komponens megjeleníti a FootFrenzy logót.
* Az <Linkek> komponensben találhatók a weboldal főbb linkei (FŐOLDAL, SNEAKEREK, MEGJELENÉSEK, INFÓ), amelyek a <NavbarLink> komponens segítségével lettek implementálva. A <DropdownMenu> komponens a mobil nézetben megjelenő hamburger menü.
* Az <IconLink> komponens a megadott linkekre visz el a weboldalon belül.
* Az <AccountCircle> ikon a bejelentkezett felhasználó avatarját jelképezi.
* A <StyledBadge> komponens az ikon mellé jelzi a kosárban lévő termékek számát.
* A <Username> komponens a bejelentkezett felhasználó nevét jelképezi.

A képen szöveg látható

Automatikusan generált leírás

A NewProducts komponens egy listát jelenít meg az új cipő termékekről. A komponens a következő állapotokat használja:

* data: A termékek tömbjét tartalmazza.
* loading: Az állapot jelzi, hogy a termékek adatok betöltése folyamatban van vagy sem.
* error: Az állapot jelzi, hogy a termékek adatok betöltése során hiba történt-e vagy sem.

Az useEffect hookot használja, hogy aszinkron módon lekérje a termékek adatait egy API-n keresztül. Amennyiben a lekérdezés során hiba történik, az error állapotba kerül az üzenet, amit majd kiír a felhasználónak. Ha sikeres a lekérdezés, akkor az adatokat rendezés után az data állapotba menti.

Végül a komponens visszatérési értéke a termékek listáját jeleníti meg egy konténerben. Amennyiben az adatok még betöltés alatt állnak, egy betöltési szöveg jelenik meg, és amennyiben az adatok lekérdezésekor hiba történt, akkor az error állapotban tárolt hibaüzenet jelenik meg.

A képen szöveg látható

Automatikusan generált leírás

Ez a kód egy funkciót definiál, amely a népszerű termékeket jeleníti meg. A funkció egy komponens, amely az oldal egy részének megjelenítéséért felelős, és a népszerű termékeket a localhost:3000/shoes/popular URL-ről tölti be az oldal betöltésekor.

Az useState hookot használja a data, loading és error állapotok kezeléséhez. Az useEffect hookot használja adatai betöltéséhez, és az adatokat a data állapotba menti, miután letöltötte őket a szerverről. A loading állapot azt jelzi, hogy az adatok betöltése még nem fejeződött be, és amíg igaz, egy töltő animációt jelenít meg. Az error állapot akkor jelenik meg, ha hiba történt az adatok betöltése során.

A komponens visszatérési értéke egy <div>, amely tartalmazza a betöltési állapotot, és ha az adatok betöltődtek, akkor a népszerű termékeket jeleníti meg. A termékeket egy Container komponensbe rendezi, és minden termékhez egy Product komponenst rendel, amely az adatokat és a termék megjelenítéséért felelős. A betöltött termékek közül csak a legnépszerűbbek első 8 darabját jeleníti meg a komponens.

A képen szöveg látható

Automatikusan generált leírás

Ez egy Popup nevű komponens, amely egy üzenetet jelenít meg egy felugró ablakban. A komponens props-ként kap egy message string-et, amely az üzenetet tartalmazza, valamint egy handleClose függvényt, amelyet akkor hív meg, amikor az üzenet megjelenése véget ér.

A useEffect hook segítségével beállítunk egy időzítőt, amely 3 másodperc múlva meghívja a handleClose függvényt. Ezt a hookot azért használjuk, hogy biztosítsuk, hogy az üzenet csak rövid ideig jelenik meg a felhasználó számára.

A komponens egy div-et (PopupContainer) tartalmaz, amely egy p tag-ben jeleníti meg az üzenetet. Az üzenet tartalmát a message prop-ból származó string adja.

A képen szöveg látható

Automatikusan generált leírás

A fenti kód egy Product komponenst definiál, amely egy terméket jelenít meg az oldalon. A komponens egy termék objektumot kap prop-ként, amely tartalmazza a termék adatait.

A useNavigate hookot használja a react-router-dom modulból a termék részleteinek oldalára való navigáláshoz. Az onClick eseményre egy handleImageClick függvény van hozzárendelve, amely meghívja a navigate függvényt a megfelelő URL-el.

Az Image komponens egy képet jelenít meg a termékről, amelynek forrása a item objektum imageUrl1 tulajdonsága.

Az Info, Title, és Price komponensek a termék nevét, árát és egyéb információit jelenítik meg a termék alatt.

A képen szöveg látható

Automatikusan generált leírás

Ez a kódrészlet egy olyan React komponenst definiál, amely két useEffect hookot használ. Az egyik useEffect hook a komponens renderelésekor lefut, és aszinkron módon adatokat kér a szerverről, majd beállítja a state változókat (filteredData, data, loading, error) a kapott adatoknak megfelelően. Ha az adatkérés során hiba történik, akkor az error állapot beállítódik a hibaüzenetre. A másik useEffect hook a szűrőfeltételek változása esetén frissíti a megjelenített termékek listáját a szűrőfeltételeknek megfelelően. Ezután a termékek sorrendjét a sort változó értékének megfelelően módosítja, majd a termékek listáját megjeleníti a képernyőn. Ha nincs megfelelő termék a szűrőfeltételeknek megfelelően, akkor egy üzenet jelenik meg a felhasználónak. A komponens props-okat kap a kategória, a szűrőfeltételek és a sorrend beállítására.

A képen szöveg látható

Automatikusan generált leírás

Ez egy képcsúszkát megvalósító komponens, amely egy képernyőn három képet jelenít meg, amelyek között balra és jobbra lehet navigálni. A komponens a következőket tartalmazza:

* State-et használ a slideIndex számának tárolására, amely meghatározza, hogy melyik slide jelenik meg.
* handleClick függvény, amely beállítja a slideIndex-et a megfelelő értékre attól függően, hogy melyik nyílra kattintottak.
* Egy Arrow komponens bal és jobb nyilakkal, amelyek a handleClick függvényt hívják meg a megfelelő irányba történő navigáláshoz.
* Egy Wrapper komponens, amely tartalmazza a sliderItems tömb elemeit, amelyek mindegyike egy Slide komponensbe van beágyazva. A Wrapper beállítja az aktuális slideIndex-et a megfelelő slide megjelenítéséhez.
* A Slide komponens egy kép hátteret és információkat jelenít meg, például címet és leírást.
* A Slideshow komponens összesen három Slide komponenst jelenít meg, amelyek között balra és jobbra navigálhatunk a nyilakra kattintva.