



FEJLESZTŐI DOKUMENTÁCIÓ

GYÜMIZÖLI WEBSHOP

BŐGÉR BENCE & MIHUCZA GERGŐ
BZSH KÜLKERESKEDELMI TECHNIKUM

Tartalomjegyzék

Célunk	4
Felhasznált technológiák.....	4
Készítéshez használt programok.....	4
Könyvtár szerkezet.....	4
Frontend projekt beüzemelése.....	5
Barion beállítása.....	5
Komponensek	7
aboutus	7
admin-orders.....	7
admin-products.....	8
admin-profile	9
admin-users	9
basket.....	10
errorpage	11
footer	11
fruits.....	11
home	12
login.....	13
navbar	14
payment	15
product-detail	16
profile.....	17
registration.....	18
sale	19
search.....	19
shipping-details.....	20
vegetables	21
Guards	22
admin.guard.ts	22
auth.guard.ts.....	22
Pipes.....	23
admin-order-search.ts	23
admin-product-search.ts.....	23
admin-user-search.ts	23
product-search.ts.....	24
Services	25
auth.service.ts.....	25
barion.service.ts.....	26

base.service.ts.....	27
basket.service.ts.....	28
search.service.ts.....	28
Backend Fejlesztői Dokumentáció	29
Adatbázis szerkezete és mezőnevek.....	29
users tábla.....	29
products tábla	31
orders tábla.....	33
Végpontok és azoknak a feladatai	35
Osztályok.....	39
AuthController	39
ProductController	40
OrderController.....	41
UserController.....	43
MailController	45
Tesztelés.....	47
Főoldal.....	47
Gyümölcsök.....	47
Zöldségek	48
Akciók.....	48
Rólunk	48
Termék adatlap	49
Kosár	50
Szállítási adatok.....	51
Sikeres rendelés	52
Keresés.....	52
Bejelentkezés	52
Regisztráció	53
Profil.....	54
Admin profil	55
Felhasználók kezelése	56
Termékek kezelése.....	57
Megrendelések kezelése.....	58
Navigációs sáv	59
Lábrész	59
Nyilvános Végpontok	60
Felhasználó bejelentkezés	60
Felhasználó regisztráció.....	60
Rendelés létrehozása	61

Termékek listázása	61
Egy termék megtekintése	62
E-mail küldés különböző eseményekhez	62
Hitelesítést Igénylő Végpontok	63
Kijelentkezés	63
Jelszó módosítása.....	63
Cím módosítása.....	64
Felhasználó saját rendeléseinek listázása	64
Rendelés frissítése	65
Rendelés törlése.....	65
Felhasználó rendeléseinek lekérdezése.....	66
Admin jogosultság beállítása	66
Új admin felhasználó hozzáadása	67
Felhasználó adatainak lekérdezése.....	67
Felhasználók listázása	68
Felhasználó adatainak frissítése	68
Felhasználó törlése	69
Új termék hozzáadása.....	69
Termék frissítése.....	70
Termék törlése	70
Fejlesztési lehetőségek	71
Összegzés	71

Frontend fejlesztői dokumentáció

Célunk

Egy kisvállalkozás számára egy gyümölcs zöldség webshop megalkotása. Azért terveztük ezt a webshopot, mert fontosnak tartjuk az egészséges táplálkozást és mindkettőnknek van otthon veteményes kertje és ebből jött az ötlet, hogy készítsük el ezt a webshopot mint lehetséges kiinduló alap egy vállalkozás számára.

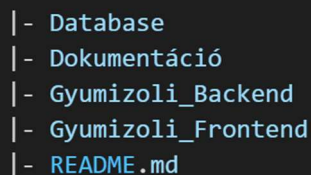
Felhasznált technológiák

- Angular 18.2.4
- ng-bootstrap 17.0.1
- Bootstrap 5.3.2
- bootstrap-icons 1.11.3
- Barion (kártyás fizetés)
- ngrok
- Laravel API (php keretrendszer)
- Mariadb serve(Adatbázis kiszolgáló)

Készítéshez használt programok

- Visual Studio Code 1.98.2
- Xampp
- Insomnia
- Google Chrome/Microsoft Edge → Konzol a hibák megjelenítéséről

Könyvtár szerkezet



```
| - Database  
| - Dokumentáció  
| - Gyumizoli_Backend  
| - Gyumizoli_Frontend  
| - README.md
```

Frontend projekt beüzemelése

Az GyumiZoli_Frontend című mappába belépve megnyitunk egy parancssort majd kiadjuk az:

- `npm i`

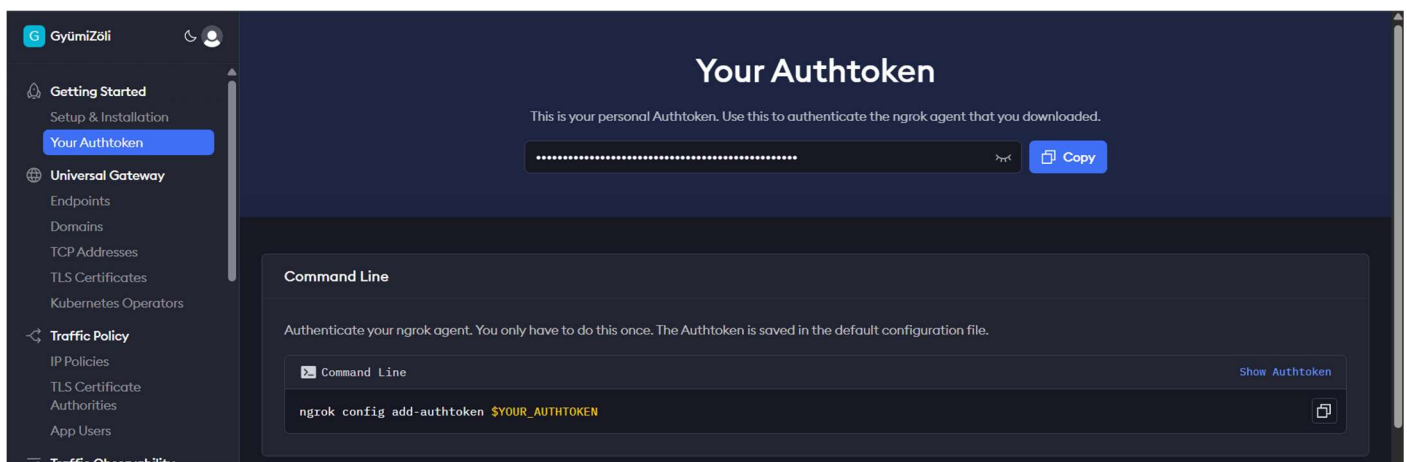
parancsot.

Barion beállítása

Az ngrok egy ingyenes publikus IP-címet biztosít ami szükséges a Barion beállításához. Az ngrok felületén létrehozunk egy felhasználói profilt. [\[Link\]](#) Miután létrehoztunk egy fiókot kiadjuk a konzolon a következő parancsot ami telepíti a gépre az ngrok csomagot:

- `npm i -g ngrok`

Miután feltelepítette belépünk a profilunkba és megkeressük az egyedi tokenünket.



Kimásoljuk a Command Line-ba írt szöveget majd a konzolunkba beillesztjük. A következő paranccsal pedig elindítjuk a csomagot:

- `ngrok http 4200`

Itt kapunk egy publikus URL-t ami a Barion beállításához szükséges. (Figyelem! Az IP cím mindig újragenerálódik, mert ingyenes felhasználói fiókunk van.)

[\[Itt\]](#) kell regisztrálni a Barion-ba. Miután sikeres a regisztráció be kell állítanunk a profilt máskülönben nem tudunk elfogadóhelyet beállítani. Sikeres profil beállítás után az elfogadóhelyekhez megyünk és beállítjuk a szükséges adatokat, a publikus URL-t kell megadni az elfogadóhely URL-jéhez.

barion

MAGYAR

Gyümi Zöli

gyumizoli10@gmail.com

Egyenlegek

4 139

Pénzközlés

Történet

Tárca

Feltöltés

Banki átutalás

Elfogadóhelyek

Fontos: Az elfogadóhelyeket megvizsgáljuk aszerint, hogy milyen kockázatot jelentenek számunkra (pénzügyi háttér, hírnév, termékek/szolgáltatások típusa). Előfordulhat, hogy egyedi feltételeket szabunk és olyan is lehetséges, hogy elutasítjuk, tehát egyáltalán nem szolgáljuk ki az adott tevékenységet, vagy céget. Az elfogadóhely jóváhagyásához implementálni kell a Barion fizetést, és elképzelhető olyan ritka eset, hogy csak ezt követően utasítjuk el a boltot. Mindent megteszünk, hogy ez ne így legyen, azonban, ha mégis megesik, a felmerülő költségeket nem áll módunkban megtéríteni. További információ [itt](#).

A technikai összekötés beállításához titkos azonosító kódra van szükség. Kérjük add meg az elfogadóhely nevét és mentsd el az űrlapot, ezáltal automatikusan létre jön a titkos azonosító (POS-key), amit megtekinthetsz az Elfogadóhely Részletekél. A jóváhagyást abban az esetben tudod kérni, ha már mindent kitöltöttél. Ezt követően a vizsgálat általában néhány napon belül megtörténik.

Elfogadóhely neve:

Itt jelenik meg a felhasználónál a Barion forrás felületén. Ha foglalt, választhatod.

Elfogadóhely URL-je:

Válaszd ki az URL-t, majd az alkalmazás esetén a letöltési URL-t (OS/Android), szimulációsprogram esetén a program URL-t. Csak https-sel működik, engedélyt kell hozzáadni az internethez.

https://example.com

Elfogadóhely leírása:

Röviden ne írd 120-200 karaktert, hogy milyen termékeket/szolgáltatásokat lehet megvásárolni az elfogadóhelyen. Itt az elfogadóhely jóváhagyásához szükséges.

Elfogadóhely logója:

A jóváhagyás az elfogadóhelyt saját logóval kell erősíteni. Ne használj írásmódot, csúszkát, szöveget.

Maximum feltöltési méret: 10 MB

Engedélyezett fájltípusok: .png, .gif, .jpg, .jpeg, .bmp

Fájl kiválasztása

Nincs fájl kiválasztva

Elfogadóhely kategóriája:

Tápellátás

Sikeres beállítás után így fog megjelenni:

Elfogadóhelyek

Ahhoz, hogy online elfogadóhelyeden Barionnal lehessen fizetni, létre kell hoznod egy új Barion elfogadóhelyet. Az integrációhoz szükséges leírást és számos egyéb információt a <https://docs.barion.com> oldalon találsz.

+ Elfogadóhely létrehozása

Elfogadóhely	Státusz	
<div> <div></div> <div>GyümiZöli</div> <div>https://4935-2a00-1110-20b-2ee5-c84e-76f7-a13f-793f...</div> </div>	<div> <div></div> <div>Nyitva</div> </div>	<div> <div>Barion Metrics</div> <div>Műveletek</div> </div>
<div> <div>1</div> <div>1 - 1. elfogadóhely, összesen: 1</div> </div>		

Lekérjük a Pos_keyt majd utána a projektünkbe létrehozunk egy mappát, amiben létrehozunk egy fájl a következő képpen:

- environments → environments.ts

A barion.service.ts komponensünkbe pedig átálítjuk az URL-t ami jelenleg megtalálható benne.

GyumiZoli_Frontend projektünket a következő paranccsal indítsuk el:

- ng serve --host 0.0.0.0 --disable-host-check

Bankkártya adatok:

- Successful 4444 8888 8888 5559
- ProblemWithCard 4444 8888 8888 4446
- LowFunds 4444 8888 8888 9999
- LostOrStolenCard 4444 8888 8888 1111
- Declined 4444 8888 8888 3331
- FraudulentTransaction 4444 8888 8888 6664
- CardSystemError 4444 8888 8888 7779
- ScaSoftDeclined 4444 8888 8888 0006

6

Komponensek

aboutus

Az aboutus komponens feladata, hogy információt nyújtson a webshopunkról a GyümiZöliről. Ez a komponens a Rólunk oldalt jeleníti meg.

admin-orders

Ezen a felületen tudja az admin a rendeléseket kezelni. Betöltéskor az összes rendelés adatot lekéri a BaseService-től. Keresni a rendelések között a SearchService és az egyedi pipe (adminOrderSearch) segítségével lehet. A rendelés szerkesztése modális ablakban lévő szerkesztőfelülettel lehetséges, státusz vagy szállítási idő megváltoztatásakor e-mailt küld a felhasználónak a rendelés változtatásról. Rendelés törlésénél megerősítést kér modális ablakban. A rendelt termékeket egy modális ablakban lehet megtekinteni. Értesítési üzeneteket jelenít meg minden fajta műveletekről.

Osztályváltozók:

- orderForm → Űrlap a rendelések szerkesztéséhez
- orders → A rendelések adatainak tárolására
- selectedOrder → A kiválasztott rendelés (szerkesztéshez vagy törléshez)
- selectedItems → A kiválasztott rendelés termékei
- word → Keresési kulcsszó
- toastMessage → Az értesítő üzenetek szövege
- toastType → Az értesítés típusa (success/danger/warning)
- isToastVisible → Az értesítés láthatóságát szabályozza
- columns → A rendelések adatmezőinek definíciója (mezőnév, típus, címke)
- orderItemsColumns → A rendelési tételek adatmezőinek definíciója

Metódusok:

- constructor() → Inicializálja a komponenst, lekéri a rendeléseket, beállítja a keresést
- ngOnInit() → Törli a keresési szót
- createForm() → Létrehozza a szerkesztő űrlapot a megadott oszlopdefiníciók alapján
- showToast() → Értesítési üzenet megjelenítése adott üzenettel és típussal
- chooseItems() → Kiválasztott rendelés termékeinek beállítása megjelenítéshez
- chooseEditOrder() → Rendelés kiválasztása szerkesztésre
- editOrder() → Rendelés adatainak frissítése és státusz vagy időpont változás esetén e-mail küldése
- chooseDeleteOrder() → Rendelés kiválasztása törlésre
- deleteOrder() → Kiválasztott rendelés törlése
- setSearch() → Keresési szó beállítása a keresőmezőből
- deleteSearch() → Keresés törlése

admin-products

Ezen a felületen tudja az admin a termékeket kezelni. Betöltéskor az összes termék adatot lekéri a BaseService-től. Keresni a termékek között a SearchService és az egyedi pipe (adminProductSearch) segítségével lehet. Új terméket egy modális ablakban megjelenő űrlappal lehetséges hozzáadni, képet is lehet hozzáadni, az adatokat egy FormData objektumként küldi tovább. A termékek szerkesztése modális ablakban lévő szerkesztőfelülettel lehetséges. Termék törlésénél megerősítést kér modális ablakban. Értesítési üzeneteket jelenít meg minden fajta műveletekről.

Osztályváltozók:

- @ViewChild("fileInput") fileInput!: ElementRef → Referencia a fájl feltöltő input elemre
- newProductForm → Űrlap új termék létrehozásához
- productForm → Űrlap meglévő termék szerkesztéséhez
- products → A termékek listája
- selectedProduct → A kiválasztott termék (szerkesztéshez vagy törléshez)
- selectedFile → A feltöltésre kiválasztott képfájl
- toastMessage → Az értesítő üzenetek szövege
- toastType → Az értesítés típusa (success/danger/warning)
- isToastVisible → Az értesítés láthatóságát szabályozza
- word → Keresési kulcsszó
- columns → A termék mezőinek definíciója, ami alapján az űrlapok dinamikusan generálódnak

Metódusok:

- constructor() → Inicializálja a komponenst, betölti a termékeket, beállítja a keresést
- ngOnInit() → Törli a keresési szót
- createNewForm() → Dinamikusan létrehozza az új termék űrlapot a columns alapján
- createForm() → Létrehozza a termékszerkesztő űrlapot, kiegészítve az image_url mezővel
- showToast() → Toast megjelenítése adott üzenettel és típussal
- removeImage() → Termék képének eltávolítása
- onFileSelect() → Kép feltöltés kezelése, fájltypus ellenőrzéssel (JPEG, JPG, PNG)
- confirmImageDelete() → Kép törlés megerősítése és végrehajtása
- addProduct() → Új termék hozzáadása FormData objektum segítségével
- chooseEditProduct() → Termék kiválasztása szerkesztésre és űrlap kitöltése
- editProduct() → Termékadatok frissítése, beleértve a képet is, ha van
- chooseDeleteProduct() → Termék kiválasztása törlésre
- deleteProduct() → Kiválasztott termék törlése
- setSearch() → Keresési szó beállítása a keresőmezőből
- deleteSearch() → Keresés törlése

admin-profile

Komponens betöltésekor automatikusan lekéri a bejelentkezett felhasználói adatokat és megjeleníti azokat. Három navigációs gomb helyezkedik el, rajta amik a különböző admin felületekhez irányítanak.

Osztályváltozók:

- userdata → Az admin felhasználó adatainak tárolására szolgál

Metódusok:

- constructor:
 - Inicializálja a komponenst
 - A BaseService getUserData() metódusát használva lekéri a bejelentkezett admin felhasználó adatait
 - Sikeres lekérdezés esetén a válasz adatait az userData objektumban tárolja
 - Hiba esetén konzolos hibaüzenetet jelenít meg

admin-users

Ezen a felületen tudja az admin a felhasználókat kezelni. Betöltéskor az összes felhasználói adatot lekéri a BaseService-től. Keresni a felhasználók között a SearchService és az egyedi pipe (adminUserSearch) segítségével lehet. Új felhasználót egy modális ablakban megjelenő mezőkkel lehetséges hozzáadni, sikeres hozzáadásról a felhasználónak elküld egy e-mailt, amiben egy beállított jelszót kap, amit első belépéskor célszerű megváltoztatni. A felhasználó szerkesztése és jogosultságának beállítása modális ablakban lévő szerkesztőfelülettel lehetséges. Felhasználó törlésénél megerősítést kér modális ablakban. Értesítési üzeneteket jelenít meg minden fajta műveletről.

Osztályváltozók:

- newUser → Új felhasználó adatainak ideiglenes tárolására szolgál
- users → A rendszerben található felhasználók listáját tárolja
- selectedUser → A műveletekhez kiválasztott felhasználó adatait tartalmazza
- toastMessage → Az értesítő üzenetek szövege
- toastType → Az értesítés típusa (success/danger/warning)
- isToastVisible → Az értesítés láthatóságát szabályozza
- word → A keresőmezőbe beírt keresési kulcsszó
- columns → A felhasználói adatok mezőinek definíciója, amely alapján az űrlapokat dinamikusan generálja

Metódusok:

- constructor() → Inicializálja a komponenst, betölti a felhasználókat, beállítja a keresést
- ngOnInit() → Törli a keresési kulcsszót
- showToast() → Értesítés megjelenítése adott üzenettel és típussal
- addUser() → Új felhasználó létrehozása a megadott adatokkal és értesítő e-mail küldése az új felhasználónak
- chooseEditUser() → Kiválasztott felhasználó adatainak betöltése szerkesztéshez
- editUser() → A kiválasztott felhasználó adatainak frissítése, Admin jogosultság beállítása
- chooseDeleteUser → Felhasználó kijelölése törlésre
- deleteUser() → Kiválasztott felhasználó törlése
- setSearch() → Keresési szó beállítása
- deleteSearch() → Keresés törlése (keresőmező ürítése)

basket

A webshop kosarát valósítja meg. Felhasználó által kiválasztott termékeket jeleníti meg. Termék mennyiséget módosítani és terméket törölni is lehet a kosárból. A végösszeg automatikusan kiszámolódik. Csak bejelentkezett felhasználók adhatnak le rendelést. Ha nincs bejelentkezett felhasználó akkor egy értesítést mutat, hogy nincs bejelentkezve.

Osztályváltozók:

- `basketItems` → A kosárban található termékek részletes adatait tároló tömb
- `totalPrice` → A kosár tartalmának végösszege
- `userData` → A bejelentkezett felhasználó adatai
- `selectedBasketItem` → Törlésre kijelölt kosárelem adatai
- `toastMessage` → Az értesítő üzenetek szövege
- `toastType` → Az értesítés típusa (success/danger/warning)
- `isToastVisible` → Az értesítés láthatóságát szabályozó változó

Metódusok:

- `constructor`:
 - Inicializálja a komponenst
 - Betölti a kosár tartalmát
 - Ellenőrzi a bejelentkezést és lekéri a felhasználó adatait, ha be van jelentkezve
- `loadBasket()`:
 - Lekéri a kosár elemeit a `BasketService`-ből
 - A termékadatokat feldolgozza megjelenítésre optimalizált formátumba
 - Minden terméknel kiszámítja az akciós vagy normál árat
 - Kiszámítja a termék összértékét (mennyiség x egységár)
 - Végül meghívja a `calculateTotalPrice()` metódust
- `calculateTotalPrice()`:
 - Összesíti a kosárban lévő termékek árát
 - Az értéket kerekíti egész számra
- `updateItemTotal()`:
 - Frissíti egy termék mennyiségét és árát
 - Ellenőrzi, hogy a kért mennyiség nem haladja-e meg a raktárkészletet
 - Újrászámítja a termék összértékét
 - Frissíti a kosár végösszegét
 - Frissíti a termék mennyiségét a `BasketService`-ben
- `chooseSelectedItem()` → Kiválasztja a terméket törléshez
- `removeItem()`:
 - Eltávolít egy terméket a kosárból a `BasketService` segítségével
 - Újratölti a kosár tartalmát
 - Törli a kiválasztott termék adatait
- `checkout()`:
 - Ellenőrzi, hogy a felhasználó be van-e jelentkezve
 - Ha igen, elmenti a kosár adatait a `localStorage`-ba
 - Átirányít a szállítási részletek oldalra
 - Ha nincs bejelentkezve felhasználó akkor értesítést jelenít meg a bejelentkezés szükségességéről
- `showToast()` → Értesítés megjelenítése adott szöveggel és típussal

errorpage

Felhasználó tájékoztatása, hogy az adott oldal nem található.

footer

Ez egy statikus komponens, amely a webshop minden oldalán megjelenik és alapvető információkat nyújt a felhasználó felé.

fruits

Ez a komponens az összes gyümölcsöt kilistázza az `*ngIf="product.category === 'Gyümölcs'"` feltétellel, akciót kiemeli, készlet információkat jelenít meg és navigációt biztosít a termék megtekintésére ha az adott termék készleten van.

Osztályváltozók:

- `products` → Az összes termék adatait tároló tömb, de a komponens csak a gyümölcsöket szűri és jeleníti meg.

Metódusok:

- `constructor()` → lekéri az összes terméket és belehelyezi a `products` tömbbe
- `viewProduct()`:
 - Egy adott gyümölcs részletes oldalára navigálást biztosít
 - Paraméterként megkapja a termék azonosítóját
 - A Router szolgáltatás `navigate()` metódusát használja az `"fruits/:id"` útvonalra való navigáláshoz

home

Ez a komponens a webshop főoldala. Üdvözlő üzenet fogadja a felhasználókat. Kategóriánként szűrt termék listákat görgethet a felhasználó, amiből választhat terméket és megtekintheti, ha van készleten. Készletinformáció és árak megjelenítése a termékekhez.

Osztályváltozók:

- `@ViewChild("scrollAreaFruits") scrollAreaFruits!`: `ElementRef` → Referencia a gyümölcs kategória görgethető konténeréhez
- `@ViewChild("scrollAreaVegetables") scrollAreaVegetables!`: `ElementRef` → Referencia a zöldség kategória görgethető konténeréhez
- `@ViewChild("scrollAreaSales") scrollAreaSales!`: `ElementRef` → Referencia az akciós termékek görgethető konténeréhez
- `products` → Az összes termék adatait tároló tömb
- `fruitsScrollable` → Jelzi, hogy a gyümölcs konténer tartalma túlnyúlik-e a látható területen
- `vegetablesScrollable` → Jelzi, hogy a zöldség konténer tartalma túlnyúlik-e a látható területen
- `salesScrollable` → Jelzi, hogy az akciós termékek konténerének tartalma túlnyúlik-e a látható területen

Metódusok:

- `constructor`:
 - Lekéri az összes terméket
 - az adatokat feldolgozza a megjelenítéshez használható formára
 - késleltetéssel ellenőrzi a görgetési állapotokat, hogy biztosan betöltsön a DOM
- `ngAfterViewInit`:
 - `ResizeObserver`-t állít be mindhárom görgethető konténerre
 - Figyeli a konténerek méretváltozását és frissíti a görgethetőségi állapotot
- `@HostListener("window:resize") onWindowResize()` → Ablak átméretezésekor ellenőrzi mindhárom konténer görgethetőségét
- `checkFruitsScrollOverflow()` → Ellenőrzi, hogy a konténer teljes szélessége nagyobb-e, mint a látható területe, és ennek megfelelően állítják be a `scrollable` változókat
- `checkVegetablesScrollOverflow()` → Ellenőrzi, hogy a konténer teljes szélessége nagyobb-e, mint a látható területe, és ennek megfelelően állítják be a `scrollable` változókat
- `checkSalesScrollOverflow()` → Ellenőrzi, hogy a konténer teljes szélessége nagyobb-e, mint a látható területe, és ennek megfelelően állítják be a `scrollable` változókat
- `scrollLeftFruits() & scrollRightFruits()` → Gyümölcs konténer 200 pixelnyi sima (animált) görgetést végez balra/jobbra
- `scrollLeftVegetables() & scrollRightVegetables()` → Zöldség konténer 200 pixelnyi sima (animált) görgetést végez balra/jobbra
- `scrollLeftSales() & scrollRightSales()` → Akciós termékek konténerének 200 pixelnyi sima (animált) görgetést végez balra/jobbra
- `viewProduct()`:
 - A termék részletes oldalára navigál a Router szolgáltatás segítségével
 - A magyar nyelvű kategórianevet ('Gyümölcs'/'Zöldség') átkonvertálja az angol URL-re ('fruits'/'vegetables')
 - Egyéb kategóriák esetén egyszerű normalizált formára alakítja

login

Ez a komponens a bejelentkezési felületet valósítja meg. Felhasználókat hitelesíti beléptetésük során. Jelszót megjelenítheti/elrejtetheti a felhasználó. Visszajelzéseket kap a felhasználó, ha nem sikerült bejelentkezni. Ha még nem regisztrált, akkor lehetősége van átnavigálni a regisztrációra.

Osztályváltozók:

- login → A bejelentkezési adatokat tartalmazó objektum
- isPasswordVisible → Jelzi, hogy a jelszó látható-e vagy el van rejtve
- toastMessage → Az értesítő üzenet szövege
- toastType → Az értesítés típusa (success/danger/warning)
- isToastVisible → Az értesítés láthatóságát szabályozza

Metódusok:

- constructor() → Inicializálja a komponenst
- signIn():
 - Elindítja a bejelentkezési folyamatot az AuthService segítségével
 - Sikeres bejelentkezés esetén a profil oldalra navigál
 - Sikertelen bejelentkezés esetén hibaüzenetet jelenít meg
 - Törli a bejelentkezési adatokat siker esetén
- togglePasswordVisibility():
 - A jelszó mezőben lévő tartalom láthatóságát kapcsolja be/ki
 - Az ikon is változik a láthatóságnak megfelelően
- showToast() → Értesítés megjelenítése adott üzenettel és típussal

navbar

Ez a komponens a webshop kulcsfontosságú komponense. Itt lehet navigálni a különböző oldalak között. Lehet kerestetni vele. Bejelentkezés/Regisztráció elérésében kulcsfontosságú szerepet tölt be. Felhasználói profil menüt mutatja. Kosár tartalmának jelzése. Admin és Felhasználói menü közötti automatikus váltás. Mobilnézetet folyamatosan nyomon követi.

Osztályváltozók:

- `menuItems` → A normál felhasználói menü elemeit tartalmazza
- `adminMenuItems` → Az admin felhasználói menü elemeit tartalmazza
- `selectedLanguage` → Az aktuálisan kiválasztott nyelv kódja
- `languages` → A választható nyelvek listája
- `userData` → A bejelentkezett felhasználó adatai
- `totalItems` → A kosárban lévő tételek száma
- `basketSub` → Feliratkozás a kosár változásaira
- `userSub` → Feliratkozás a felhasználói adatok változásaira
- `isCollapsed` → A mobilnézet menüjének összecukott/kinyitott állapota
- `isMobile` → Jelzi, hogy mobilnézetet kell-e megjeleníteni
- `isAdminPage` → Jelzi, hogy admin oldalon vagyunk-e
- `@ViewChild("searchInput") searchInput!: ElementRef<HTMLInputElement>` → Referencia a keresőmező DOM elemére

Metódusok:

- `constructor():`
 - Inicializálja a komponens
 - Feliratkozik a bejelentkezett felhasználó adatainak változásaira
 - Feliratkozik a kosár tartalmának változására
- `ngOnInit():`
 - Ellenőrzi a mobilnézetet
 - Feliratkozik az útvonalváltozásokra, hogy követhesse, mikor vagyunk admin oldalon
- `@HostListener('window:resize') onResize()` → Ablakméret változásakor frissíti a mobilnézet állapotát
- `checkMobileView()` → Az ablak szélessége alapján meghatározza, hogy mobil- vagy asztali nézetet kell-e mutatni
- `onSearch():`
 - Keresés indítása a megadott kulcsszóval
 - A keresési eredmény oldalra navigál
 - Törli a keresőmezőt
- `logout():`
 - Kijelentkezteti a felhasználót az AuthService segítségével
 - Nullázza a felhasználói adatokat
 - Átirányít a kezdőlapra

payment

Ez a komponens a Barion fizetési folyamatának lezárása miatt lett létrehozva, hogy a felhasználó egy állapotot visszakapjon a fizetéséről. Ha sikeresen fizetett, akkor a rendelés leadásra kerül, a kosár tartalma törlődik. Sikertelen fizetés esetén a felhasználó újra megkísérelheti a fizetést.

Osztályváltozók:

- paymentId → A Barion fizetési szolgáltatástól kapott fizetési azonosító
- title → Az oldalon megjelenő címszöveg (sikeres/sikertelen fizetési üzenet)
- order → A rendelés adatai, amelyek ideiglenesen a localStorage-ban voltak tárolva
- orderNumber → A megrendelés azonosító száma, amely megjelenik a felhasználónak

Metódusok:

- constructor():
 - Inicializálja a komponens
 - Feliratkozik az URL lekérdezési paramétereire
 - Kinyeri a paymentId paramétert
 - Meghívja a checkPayment() metódust a kapott azonosítóval
- checkPayment():
 - A BarionService segítségével ellenőrzi a fizetés állapotát:
 - Sikeres fizetés esetén:
 - Beállítja a sikeres üzenetet
 - Megjeleníti a rendelés azonosítót
 - Kinyeri a várakozó rendelés adatait a localStorage-ból
 - Elküldi a rendelést a szervernek
 - Törli a kosár tartalmát és az ideiglenes adatokat
 - Sikertelen fizetés esetén:
 - Hibaüzenetet jelenít meg

product-detail

Ez a komponens a webshop termékadatlapját valósítja meg. Megjeleníti a termék fotóját, készletét, leírását, készletinformációját és normál vagy akciós árát. A felhasználó mennyiséget tud, választani, darabszám esetén a lépésköze az input mezőnek 1, súly esetén pedig 0.1. Az input mező semmi esetben nem lehet 0 erről a felhasználó értesítést is kap. Ezek alapján valós időben kiszámolódik a végösszeg. Miután belerakta a kosárba visszajelzést kap arról, hogy sikerült a művelet.

Osztályváltozók:

- oneproduct → Az aktuálisan megtekintett termék adatai
- id → A termék azonosítója, amit az URL paraméterekből olvas ki
- category → A termék kategóriája, szintén az URL-ből kiolvasva
- quantitySelected → A felhasználó által kiválasztott mennyiség
- totalPrice → A kiválasztott mennyiség összértéke
- toastMessage → Az értesítés szövege
- toastType → Az értesítés típusa (success/danger/warning)
- isToastVisible → Az értesítés láthatóságának szabályozása

Metódusok:

- constructor():
 - Inicializálja a komponens
 - Kinyeri a termék azonosítót és kategóriát az URL-ből
 - Lekéri a termék részletes adatait a BaseService-től
- updateTotalPrice():
 - Kiszámítja a kiválasztott mennyiség összértékét
 - Figyelembe veszi, hogy a termék akciós-e, és a megfelelő árat használja
 - Valós időben frissül, ahogy a felhasználó módosítja a mennyiséget
- addBasket():
 - Hozzáadja a terméket a kosárhoz a kiválasztott mennyiséggel
 - Ellenőrzi, hogy a mennyiség nagyobb-e nullánál
 - Ha nincs mennyiség kiválasztva, figyelmeztetést jelenít meg
 - A BasketService-t használja a művelet végrehajtásához
- showToast() → Értesítés megjelenítése adott üzenettel és típussal

profile

Ez a komponens a felhasználók adatait mutatja meg. A felhasználó az eddig megrendelt termékeit megtekintheti egy modális ablak táblázatában. Egy modális ablak űrlapjába új jelszót vagy címet adhat meg, de csak akkor, ha az új adatok megegyeznek.

Osztályváltozók:

- `userData` → A bejelentkezett felhasználó adatait tároló objektum
- `orders` → A felhasználó rendeléseinek listája
- `selectedOrderItems` → Az aktuálisan kiválasztott rendelés tételei
- `old_password` → A felhasználó jelenlegi jelszava
- `new_password` → A felhasználó új jelszava
- `confirmPassword` → Az új jelszó megerősítése
- `new_address` → A felhasználó új címe
- `confirmNewAddress` → Az új cím megerősítése
- `toastMessage` → Az értesítő üzenet szövege
- `toastType` → Az értesítés típusa (success/danger/warning)
- `isToastVisible` → Az értesítés láthatóságát szabályozó változó

Metódusok:

- `constructor()`:
 - Inicializálja a komponens
 - Lekéri a bejelentkezett felhasználó adatait
 - Sikeres/sikertelen adatlekérésről értesítést jelenít meg
- `getOrders()`:
 - Lekéri a felhasználó összes rendelését azonosító alapján
 - A "Rendelések" gombra kattintva aktiválódik
 - A rendelések adatait az `orders` tömbben tárolja
 - Sikeres/sikertelen lekérésről értesítést jelenít meg
- `chooseSelectedItem()`:
 - Kiválasztott rendelés tételeinek részletes megjelenítése
 - Feldolgozza a rendelés tételeit, biztosítva, hogy a JSON formátum megfelelően legyen értelmezve
 - Az eredményt a `selectedOrderItems` változóban tárolja
- `changePassword()`:
 - A jelszómódosítási folyamatot kezeli
 - Ellenőrzi az űrlap érvényességét
 - Meghívja a `BaseService changePassword()` metódusát
 - Sikeres módosítás esetén értesítő e-mailt küld és törli a form mezőket
 - Sikeres/sikertelen műveletről értesítést jelenít meg
- `changeAddress()`:
 - A címmódosítási folyamatot kezeli
 - Ellenőrzi az űrlap érvényességét
 - Meghívja a `BaseService changeAddress()` metódusát
 - Sikeres módosítás esetén értesítő e-mailt küld és törli a form mezőket
 - Sikeres/sikertelen műveletről értesítést jelenít meg
- `showToast()` → Értesítés megjelenítése adott üzenettel és típussal

registration

Ez a komponens a regisztrációs folyamatot biztosítja. Miután a felhasználó kitöltötte az űrlapot helyesen a rendszer aktiválja a regisztrációs gombot. Miután megnyomta a gombot a felhasználó regisztrációs folyamat elindul, és ha minden jól sikerült átirányítja a bejelentkezési felülethez.

Osztályváltozók:

confirmPassword → A jelszó megerősítésére szolgáló mező

isPasswordVisible → A jelszó láthatóságát szabályozza

isConfirmPasswordVisible → A jelszó-megerősítés láthatóságát szabályozza

toastMessage → Az értesítés szövege

toastType → Az értesítés típusa (success/danger/warning)

isToastVisible → Az értesítés láthatóságát szabályozza

registration → A regisztrációs adatokat tároló objektum

Metódusok:

- constructor() → Inicializálja a komponenst
- register():
 - A regisztrációs folyamatot kezeli
 - Az AuthService register() metódusát használja a felhasználó regisztrálásához
 - Sikeres regisztráció esetén:
 - Visszaigazoló e-mailt küld a BaseService segítségével
 - Törli az űrlapot
 - Átirányít a bejelentkezési oldalra
 - Sikertelen regisztráció esetén hibaüzenetet jelenít meg
- togglePasswordVisibility():
 - A jelszó láthatóságát kapcsolja be/ki
 - Az ikon is változik a láthatóságnak megfelelően
- toggleConfirmPasswordVisibility():
 - A jelszó-megerősítés láthatóságát kapcsolja be/ki
 - Az ikon is változik a láthatóságnak megfelelően
- showToast() → Értesítés megjelenítése adott üzenettel és típussal

sale

Ez a komponens az összes akciót kilistázza az `*ngIf="product.promotion === 1"` feltétellel, akciókat kiemeli, készlet információkat jelenít meg és navigációt biztosít a termék megtekintésére, ha az adott termék készleten van.

Osztályváltozók:

- `products` → Az összes termék adatait tároló tömb de a komponens csak a akciósokat szűri és jeleníti meg.

Metódusok:

- `constructor()` → lekéri az összes terméket és belehelyezi a `products` tömbbe
- `viewProduct()`:
 - Egy adott akciós termék részletes oldalára navigálást biztosít
 - Paraméterként megkapja a termék azonosítóját
 - A Router szolgáltatás `navigate()` metódusát használja az `"fruits/:id"` vagy `"vegetables/:id"` útvonalra való navigáláshoz

search

Ez a komponens a termékek közötti keresést teszi lehetővé a felhasználó számára. A navbar komponensben található keresőmező szövege átadódik a `SearchService`-nek és a `search` komponens erre feliratkozik. Ha érvényes szót talál elvégzi a szűrést a megadott paraméterekre amit a `productSearch` pipe-ban adtunk meg. Ha nincs találat a komponens jelzi a felhasználó felé, hogy „Nincs találat!”.

Osztályváltozók:

- `word` → A keresett szót/kifejezést tárolja
- `products` → Az összes termék adatait tartalmazó tömb

Metódusok:

- `constructor()` → Inicializálja a komponenst
- `ngOnInit()`:
 - Feliratkozik a `SearchService`-től érkező keresési kulcsszó változásaira
 - Ha van érvényes keresőszó, lekéri az összes terméket a `BaseService`-től
 - Üres keresőszó esetén üríti a termékek listáját
- `viewProduct()`:
 - A kiválasztott termék részletes oldalára navigál
 - A magyar nyelvű kategórianéveket angolra konvertálja (pl. "Gyümölcs" → "fruits")
 - Az egyéb kategórianéveknél eltávolítja az ékezeteket
 - A Router szolgáltatás segítségével navigál a megfelelő termékadatlap oldalra

shipping-details

Ez a komponens a szállítási és a fizetési adatokat gyűjti. A felhasználónak automatikusan betöltődik az adata az űrlapba, de ha szeretné megadhatja manuálisan is. Fizetési módot is tud választani. Kártyás fizetés esetén a Barion honlapjára irányít át a webshop. Készpénzes fizetés esetén a rendelés azonnal véglegesítődik, kosár tartalma törlődik és a megrendelő e-mailt kap a megadott e-mail címre.

Osztályváltozók:

- `basketData` → A kosár adatait tároló objektum, amelyet a `localStorage`-ból tölt be
- `order` → A végleges megrendelés adatait tartalmazó objektum
- `columns` → Az űrlap mezőinek konfigurációit tartalmazó tömb
- `toggleState` → A felhasználói adatok betöltésének állapotát jelző változó
- `toastMessage` → Az értesítés szövege
- `toastType` → Az értesítés típusa (`success/danger/warning`)
- `isToastVisible` → Az értesítés láthatóságát szabályozó változó
- `isModalVisible` → A sikeres megrendelés modális ablak láthatóságát szabályozó változó
- `countdown` → Visszaszámlálás a sikeres rendelés után

Metódusok:

- `constructor()`:
 - Inicializálja a komponens
 - Betölti a kosár adatait a `localStorage`-ból
- `ngOnInit()`:
 - A komponens inicializálásakor betölti a felhasználó adatait
 - Az adatokat beilleszti a megfelelő űrlapmezőkbe
 - Beállítja a `toggleState` értékét `true`-ra
- `toggleData()`:
 - Váltás a felhasználói adatok betöltése és törlése között
 - Ha az adatok be voltak töltve, törli azokat
 - Ha üres volt az űrlap, betölti a felhasználó adatait
- `allColumnsValid()`:
 - Ellenőrzi, hogy minden űrlapmező ki van-e töltve
 - `true` értéket ad vissza, ha minden mező ki van töltve
- `checkout()`:
 - A megrendelés folyamatát kezeli
 - Létrehozza a megrendelés objektumot az űrlap és a kosár adataiból
 - Beállítja a kézbesítési dátumot (3 nappal későbbre)
 - Elküldi a megrendelést a szervernek
 - Fizetési mód alapján:
 - Kártyás fizetésnél elindítja a Barion fizetési folyamatot
 - Készpénzes fizetésnél befejezi a rendelést és mutatja a sikeres modált
- `sendOrder()`:
 - Ellenőrzi, hogy minden mező ki van-e töltve
 - Ha igen, meghívja a `checkout()` metódust
 - Ha nem, hibaüzenetet jelenít meg
- `showSuccessModal()`:
 - Megjeleníti a sikeres megrendelés modális ablakot
 - Elindít egy 5 másodperces visszaszámlálást
 - A visszaszámlálás végén átirányít a kezdőlapra
 - Törli a kosár adatait a `localStorage`-ból és a szolgáltatásból
- `showToast()` → Értesítés megjelenítése adott üzenettel és típussal

vegetables

Ez a komponens az összes zöldséget kilistázza az `*ngIf="product.category === 'Zöldség'"` feltétellel, akciókat kiemeli, készlet információkat jelenít meg és navigációt biztosít a termék megtekintésére, ha az adott termék készleten van.

Osztályváltozók:

- `products` → Az összes termék adatait tároló tömb, de a komponens csak a zöldségeket szűri és jeleníti meg.

Metódusok:

- `constructor()` → lekéri az összes terméket és belehelyezi a `products` tömbbe
- `viewProduct()`:
 - Egy adott zöldség részletes oldalára navigálást biztosít
 - Paraméterként megkapja a termék azonosítóját
 - A Router szolgáltatás `navigate()` metódusát használja a `"vegetables/:id"` útvonalra való navigáláshoz

Guards

admin.guard.ts

Feladata:

A weblap admin felületeket védi azáltal, hogy:

- Korlátozza a hozzáférést az admin részekhez
- Csak admin jogosultsággal rendelkezők érhetik el
- Megakadályozza a jogosultsággal nem rendelkező felhasználókat

Működése:

- Injektálja az AuthService-t
- Meghívja az auth.getIsAdmin metódust
- Visszaadott érték határozza meg, hogy hozzáférést kapjon-e az admin felülethez az adott felhasználó

auth.guard.ts

Feladata:

Biztosítsa, hogy csak a bejelentkezett felhasználók érjék el a profile.components-et.

Működése:

- Injektálja az AuthService-t
- filter → csak akkor enged tovább ha van bejelentkezett felhasználó
- map → a felhasználó objektumot true értéké alakítja

Pipes

admin-order-search.ts

Feladata:

Az admin felületen lévő rendeléseket keresi kulcsszó alapján. Egyszerűsítse az adat keresést.

Metódusok:

- transform:
 - orders → Szűrendő rendelések tömbje
 - word → A keresett szó
 - Szűri a rendeléseket és csak azokat adja vissza, amire keresett az admin

admin-product-search.ts

Feladata:

Az admin felületen lévő termékeket keresi kulcsszó alapján. Egyszerűsítse az adat keresést.

Metódusok:

- transform:
 - products → Szűrendő termékek tömbje
 - word → A keresett szó
 - Szűri a termékeket és csak azokat adja vissza, amire keresett az admin

admin-user-search.ts

Feladata:

Az admin felületen lévő felhasználókat keresi kulcsszó alapján. Egyszerűsítse az adat keresést.

Metódusok:

- transform:
 - users → Szűrendő termékek tömbje
 - word → A keresett szó
 - Szűri a felhasználókat és csak azokat adja vissza, amire keresett az admin

product-search.ts

Feladata:

A felhasználói felületen lévő termékeket keresi kulcsszó alapján. Egyszerűsítse az adat keresést.

Metódusok:

- transform:
 - products → Szűrendő termékek tömbje
 - word → A keresett szó
 - Szűri a termékeket és csak azokat adja vissza, amire keresett az admin

Services

auth.service.ts

Feladata:

- A felhasználói hitelesítést és a bejelentkezési állapotok kezelését végzi.
- Kezeli a felhasználók bejelentkezését, regisztrációját, admin jogosultságokat.
- Nyilvántartja a bejelentkezett felhasználó adatait, tokenjét.

Osztályváltozók:

- `loggedUserSubject` → bejelentkezett felhasználók adatait tárolja
- `adminSubject` → bejelentkezett felhasználó admin jogosultságát tárolja

Metódusok:

- `BaseService` → API hívásokért felel
- `checkAuthState()`:
 - ellenőrzi az autentikációs token
 - ha van token akkor frissíti a `loggedUserSubject`-et és az `adminSubject`-et
 - admin felhasználó esetén pedig betölti a felhasználókat és a rendeléseket
- `login()` → Bejelentkezteti a felhasználót, elmenti a tokenjét és frissíti az autentikációs állapotot
- `register()` → Új felhasználó regisztrálása
- `logout()`:
 - Kijelentkezteti a felhasználót
 - Törli a token
 - Nullázza a `loggedUserSubject`-et
 - Törli az admin jogosultságot az `adminSubject`-ből
- `getLoggedUser()` → Hozzáférést biztosít a bejelentkezett felhasználó adataihoz
- `getIsAdmin()` → Hozzáférést biztosít az admin jogosultsághoz

barion.service.ts

Feladata:

Barion fizetési rendszerrel való integrációt biztosítja, ami a következőket foglalja magába:

- Felhasználók számára az online fizetést a webáruházban
- Kezelje a fizetési kérelmek létrehozását
- Továbbítsa a fizetési adatokat a Barion API felé
- Lekérdezze a fizetés állapotát

Osztályváltozók:

Egyetlen egy osztályváltozó van a paymentRequest ami a Barion API-nak küldendő kérelem adatokat tárolja.

Metódusok:

HttpClient → http kliens befecskendezés az API kérések küldéséhez.

initializePaymentRequest:

- Feladata:
 - Fizetés kérelemi objektum létrehozása és inicializálása a Barion API számára.
- Paraméterei:
 - basketItems → Termékek listája
 - orderId → A rendelés egyedi azonosítója
- Működése:
 - Kiszámolja a teljes fizetendő összeget
 - Átalakítja a termékeket a Barion API által várt formátumra
 - Létrehozza az elküldésre szánt objektumot a megfelelő elemekkel

startPayment():

- Feladata:
 - Fizetési folyamat elindítása
- Működése:
 - http fejlécek beállítása
 - POST kérés küldése az inicializált paymentRequest objektummal
 - Visszaadja a válasz objektumát

getPaymentState():

- Feladata:
 - Fizetés állapotának lekérdezése
- Paramétere:
 - paymentId: A fizetés egyedi azonosítója
- Működése:
 - Összeállítja a lekérdezés URL-jét
 - Beállítja a http fejléceket
 - GET kérést küld a Barion API felé majd visszaadja a válasz objektumát

base.service.ts

Feladata:

Az adatkezelést és a backend kommunikációját kezeli ezen belül:

- CRUD műveletek
- E-mail küldés
- Authentikációval kapcsolatos API hívások kezelése

Osztályváltozók:

- apiUrl → backend API url címét tárolja
- userSubject → felhasználók adatait tárolja
- productsSubject → termékek adatait tárolja
- ordersSubject → rendelések adatait tárolja

Metódusok:

- getUsers() → Visszaadja a felhasználókat tartalmazó BehaviorSubject-et
- getProducts() → Visszaadja a termékeket tartalmazó BehaviorSubject-et
- getOrders() → Visszaadja a rendeléseket tartalmazó BehaviorSubject-et
- loadUsers() → Betölti a felhasználókat az API-ról
- loadProducts() → Betölti a termékeket az API-ról
- loadOrders() → Betölti és feldolgozza a rendeléseket az API-ról (JSON átalakítás)
- addUserAdmin() → Admin által új felhasználó létrehozása
- registerUser() → Új felhasználó regisztrálása
- loginUser() → Felhasználó bejelentkeztetése
- getUserData() → Bejelentkezett felhasználó adatainak lekérdezése
- logoutUser() → Felhasználó kijelentkeztetése
- updateUser() → Felhasználói adatok frissítése
- changePassword() → Jelszóváltoztatás
- changeAddress() → Cím változtatás
- setAdmin() → Admin jogosultság beállítása
- deleteUser() → Felhasználó törlése
- addProduct() → Új termék hozzáadása
- updateProduct() → Termék adatainak frissítése
- deleteProduct() → Termék törlése
- oneProduct() → Egy kiválasztott termék adatainak lekérése
- createOrder() → Új rendelés létrehozása
- updateOrder() → Rendelés adatainak frissítése
- deleteOrder() → Rendelés törlése
- getOrdersByUser() → Felhasználóhoz tartozó rendelések lekérése
- successRegistration() → Sikeres regisztráció e-mail értesítő
- successOrder() → Sikeres rendelés e-mail értesítő
- successChangeAddress() → Cím változtatás e-mail értesítő
- successChangePassword() → Jelszóváltoztatás e-mail értesítő
- sendOrderStatus() → Rendelés állapot e-mail értesítő
- sendAddUserMail() → Új felhasználó hozzáadása admin felületen e-mail értesítő

basket.service.ts

Feladata:

A webáruház kosarát kezeli. Ezen belül:

- Kosár állapotának nyilvántartása
- Hozzáadás/Frissítés/Törlés
- Kosár tartalmának megőrzése
- Lehetővé teszi a többi komponens számára az állapot követést

Osztályváltozó:

- basketItems → kosár tartalmát tárolja

Metódusok:

- addBasket():
 - Termék hozzáadása a megadott mennyiségben
 - Ha a termék már szerepel a kosárban akkor növeli annak a mennyiségét
 - Ha egy termék még nem szerepel a kosárban akkor új elemként adja hozzá
- removeBasket() → Termék eltávolítása a kosárból az azonosítója alapján
- updateBasket() → Termék mennyiség frissítése vagy ha 0 vagy kevesebb törlése
- deleteBasket() → Teljes kosár tartalmának törlése
- storeBasketItems() → Kosár elemeinek tárolása a localStorage-ban
- getStoreBasketItems() → Kosár elemeinek visszaolvasása a localStorage-ból

search.service.ts

Feladata:

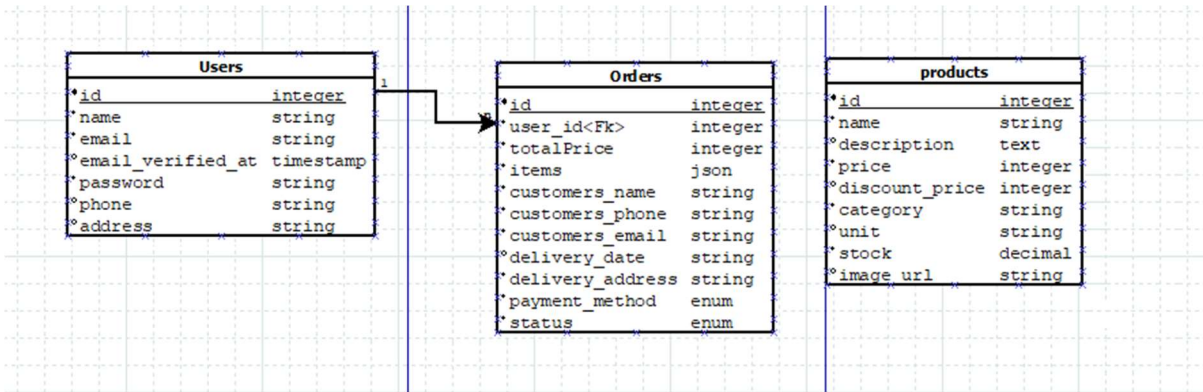
Keresési funkciókat lát el az oldalon. Tárolja és kezeli a felhasználó által beírt keresőszöveget. Lehetővé tegye a komponensek számára a keresési állapot elérését.

Osztályváltozók:

- getSearchingWord() → Visszaadja a keresőszöveget
- setSearchingWord() → Beállítja a keresendő szöveget/szót
- clearSearchingWord() → Törli a keresendő szöveget

Backend Fejlesztői Dokumentáció

Adatbázis szerkezete és mezőnevek



users tábla

id

Típus: automatikusan növekvő azonosító

Leírás: Minden felhasználó egyedi azonosítója.

name

Típus: string

Leírás: A felhasználó neve.

email

Típus: string (egyedi)

Leírás: A felhasználó email címe, amely egyedi érték.

email_verified_at

Típus: timestamp, nullable

Leírás: Az email cím hitelesítésének ideje, ha egyszer már ellenőrizték.

password

Típus: string

Leírás: A felhasználó jelszava (általában hashelt formában tárolva).

phone

Típus: string, nullable

Leírás: Opcionális telefonszám.

address

Típus: string, nullable

Leírás: A felhasználó lakcíme.

birth_date

Típus: string, nullable

Leírás: A felhasználó születési dátuma.

remember_token

Típus: string

Leírás: A felhasználó "emlékezzen rám" tokenje, hogy hosszabb ideig legyen bejelentkezve.

created_at / updated_at

Típus: timestamp

Leírás: A rekord létrehozásának és utolsó módosításának ideje.

products tábla

id

Típus: automatikusan növekvő azonosító

Leírás: A termék egyedi azonosítója.

name

Típus: string

Leírás: A termék neve.

description

Típus: text, nullable

Leírás: Részletes leírás a termékről.

price

Típus: integer

Leírás: A termék ára (egész szám, például forintban meghatározva).

promotion

Típus: boolean

Leírás: Jelezheti, hogy épp akciós-e a termék (true/false érték).

discount_price

Típus: integer, nullable

Leírás: Akciós termék elérhető ár, ha van kedvezmény.

category

Típus: string

Leírás: A termék kategóriáját jelölő mező (pl. elektronika, ruha, stb.).

unit

Típus: string, nullable

Leírás: A termék mértékegysége (pl. db, kg, liter), ha releváns.

stock

Típus: decimal(10,2)

Leírás: A készleten lévő mennyiség, pontosan meghatározva tizedes törtekkel.

image_url

Típus: string, nullable

Leírás: A termékről készült kép URL-je vagy elérési útja.

created_at / updated_at

Típus: timestamps

Leírás: A termék rekordjának létrehozási és utolsó módosítási ideje.

orders tábla

id

Típus: automatikusan növekvő azonosító

Leírás: A rendelés egyedi azonosítója.

user_id

Típus: foreignId

Leírás: A rendelést leadó felhasználó azonosítója, mely összekapcsolja a rendelést a users táblával.

totalPrice

Típus: integer

Leírás: A rendelés végösszege, azaz a megrendelt tételek összértéke.

items

Típus: json

Leírás: A rendeléshez tartozó tételek részletes leírása JSON formátumban (például termékazonosító, mennyiség, ár, stb.).

customers_name

Típus: string

Leírás: A megrendelő neve.

customers_phone

Típus: string

Leírás: A megrendelő telefonszáma, amely elérhetőséget biztosít a rendeléshez.

customers_email

Típus: string

Leírás: A megrendelő email címe a kommunikációhoz és értesítésekhez.

delivery_date

Típus: string, nullable

Leírás: A rendelés kiszállításának tervezett dátuma, mely opcionális.

delivery_address

Típus: string

Leírás: Az a cím, ahová a rendelés kiszállításra kerül.

payment_method

Típus: enum (értékek: 'card', 'cash')

Leírás: A fizetés módja, mely lehet bankkártyás ('card') vagy készpénzes ('cash').

status

Típus: enum (értékek: 'pending', 'processing', 'shipped', 'delivered', 'cancelled')

Leírás: A rendelés aktuális státusza, alapértelmezett értéke 'pending'.

pending: függőben lévő rendelés

processing: feldolgozás alatt

shipped: kiszállított rendelés

delivered: kézbesített rendelés

cancelled: törölt rendelés

created_at / updated_at

Típus: timestamps

Leírás: A rendelés létrehozásának és utolsó módosításának ideje.

Végpontok és azoknak a feladatai

Íme egy összefoglaló a látott API végpontokról, azok feladatáról és a várt adatok típusáról (figyelem: az egyes végpontok pontos adatstruktúráját a vezérlők (Controllers) implementációja határozza meg, így itt csak általános leírás található):

- POST /login

- Feladat: Felhasználó bejelentkezésének az elindítása.
- Várt adat: Felhasználói bejelentkezési adatok (pl. email és jelszó), valamint válaszként esetleg egy hitelesítési token vagy felhasználói adatok.

- POST /register

- Feladat: Új felhasználó regisztrálása.
- Várt adat: Regisztrációhoz szükséges adatok (pl. név, email, jelszó, stb.), illetve válaszként a sikeres regisztrációról szóló visszajelzés, esetleg az új felhasználó adatai vagy token.

- POST /addorder

- Feladat: Új rendelés létrehozása.
- Várt adat: A rendelés létrehozásához szükséges paraméterek (pl. termékadatok, mennyiség, felhasználói adatok, stb.) és válaszként a megrendelt rendelés azonosítója vagy megerősítő üzenet.

- GET /products

- Feladat: Termékek lekérése.
- Várt adat: A rendszerben tárolt termékek listája (pl. termék neve, ára, leírás, stb.).

- GET /productshow

- Feladat: Egy adott termék részletes adatainak megjelenítése.
- Várt adat: A specifikus termék adatai (feltételezhetően kérdés paraméterként átadva azonosító formájában, bár részletek a vezérlő implementációjától függenek).

- POST /successorder

- Feladat: Rendelés sikerességének esetén megerősítő email küldése.
- Várt adat: A rendelés adatai az email tartalmához, valamint visszajelzés, hogy az email elküldése sikeres volt-e.

- POST /successregistration
 - Feladat: Regisztráció sikerességének visszaigazolása emailben.
 - Várt adat: Regisztráció adatai (pl. felhasználói információk) és visszajelzés az emailküldésről.

- POST /changepasswordmail
 - Feladat: Jelszóváltozás esetén értesítő vagy megerősítő email küldése.
 - Várt adat: A jelszóváltozással kapcsolatos információk, ezen keresztül a felhasználó értesítése.

- POST /changeaddressmail
 - Feladat: Címváltozás esetén értesítő email küldése.
 - Várt adat: A címváltozással kapcsolatos részletek, illetve visszajelzés az emailküldés állapotáról.

- POST /orderstatus
 - Feladat: Rendelés státuszának változását kommunikáló email küldése.
 - Várt adat: Friss rendelés státusz adatai, amiket az emailben közölnek a felhasználóval.

- POST /adduseremail
 - Feladat: Új felhasználó hozzáadása esetén értesítő email küldése.
 - Várt adat: Az új felhasználó adatai, illetve visszajelzés az emailküldés sikerességéről.

A következő végpontok kizárólag hitelesített (auth:sanctum) felhasználók számára érhetők el:

- POST /logout
 - Feladat: A felhasználó kijelentkeztetése.
 - Várt adat: Visszajelzés a sikeres kijelentkezésről.

- POST /change-password
 - Feladat: Jelszó módosítása a felhasználói fiókban.
 - Várt adat: Új jelszó (valószínűleg régi jelszóval együtt az azonosításhoz), illetve válaszként visszaigazolás a módosításról.

- POST /change-address

- Feladat: Cím módosítás a felhasználói fiókban.
- Várt adat: Új címadatok, és válaszként visszaigazolás az adatváltoztatásról.

- GET /orders

- Feladat: Rendelések lekérése (valószínűleg admin vagy ügyfélszolgálati szinten, de alkalmazási logikától függ).
- Várt adat: Rendelések listája.

- POST /updateorder

- Feladat: Rendelés adatainak módosítása.
- Várt adat: Módosítandó rendelés azonosítója és az új adatok, illetve válaszként a frissített rendelés adatai vagy megerősítő üzenet.

- DELETE /orderdestroy

- Feladat: Rendelés törlése.
- Várt adat: Az eltávolítandó rendelés azonosítója, illetve válaszként a törlés állapota (sikeres/sikertelen).

- GET /getcustomersorders

- Feladat: Az adott, bejelentkezett felhasználóhoz tartozó rendeléseinek lekérése.
- Várt adat: Az aktuális felhasználó rendeléseinek listája.

- PUT /users/set-admin

- Feladat: Egy felhasználói fiók jogosultsági szintjének adminisztrátorrá emelése.
- Várt adat: A módosítandó felhasználó azonosítója, illetve válaszként visszaigazolás vagy a módosított felhasználó adatai.

- POST /users/add

- Feladat: Új admin felhasználó létrehozása.
- Várt adat: Az új felhasználó adatai, illetve visszajelzés a sikeres létrehozásról.

- GET /getuser

- Feladat: Egy adott (bejelentkezett) felhasználó adatainak lekérése.
- Várt adat: A felhasználó részletes adatai.

- GET /users
 - Feladat: Felhasználók listájának lekérése (valószínűleg adminisztratív célokra).
 - Várt adat: A rendszerben található felhasználók listája.

- PUT /users/update
 - Feladat: Felhasználói adatok módosítása.
 - Várt adat: Módosítandó adatok, illetve visszajelzés a frissítésről.

- DELETE /users/delete
 - Feladat: Felhasználó törlése a rendszerből.
 - Várt adat: Törlendő felhasználó azonosítója, illetve válaszként a törlés eredménye.

- POST /addproduct
 - Feladat: Új termék hozzáadása a rendszerhez.
 - Várt adat: A termék részletes adatai (pl. név, ár, leírás, stb.) és válaszként a sikeres létrehozás visszaigazolása.

- POST /updateproduct
 - Feladat: Meglévő termék adatainak módosítása.
 - Várt adat: Módosítandó termék azonosítója, az új adatok, illetve a módosítás eredményének visszaigazolása.

- DELETE /productdestroy
 - Feladat: Termék törlése a rendszerből.
 - Várt adat: Törlendő termék azonosítója, illetve válaszként a törlés állapota.

Fontos, hogy a pontos adatformátumok (JSON szerkezet, mezőnevek) és a visszakapott értékek a kontrollerek implementációjától függenek, ezért ajánlatos megtekinteni az adott vezérlők kódját a részletes dokumentációért.

Osztályok

Az alábbiakban részletesen ismertetem a megadott osztályok metódusainak működését:

AuthController

- getUsers()

- Lekéri az összes felhasználót az adatbázisból a ``User::all()`` segítségével.
- A visszatérési érték JSON formátumban tartalmazza a felhasználók adatait.

- setAdmin(Request \$request)

- Ellenőrzi a jogosultságot a ``Gate::allows("admin")`` metódussal, így csak admin felhasználók továbbmehetnek.
- Ha nincs jogosultság, JSON hibát ad vissza.
- Megkeresi a kérésben kapott azonosítóval rendelkező felhasználót (``User::find($request["id"]``).
- Beállítja a felhasználó ``admin`` tulajdonságát a kérésből érkező értékre.
- Frissíti a felhasználó rekordját és JSON formátumban visszaküldi az aktualizált felhasználói adatokat.

- updateUser(Request \$request)

- Hasonló jogosultság ellenőrzést alkalmaz, mint a ``setAdmin()`` esetében.
- Megkeresi a módosítandó felhasználót azonosító alapján.
- Frissíti a következő mezőket:
 - ``name``
 - ``email``
 - ``birth_date``
 - ``address``
 - ``phone``
- Az adatok beállítása után elmenti a módosításokat, és visszaküldi a frissített felhasználó adatait JSON formában.

- `destroyUser(Request $request)`

- Jogosultsági ellenőrzést hajt végre az admin jogosultság miatt.
- Megkeresi a felhasználót az adott azonosító alapján.
- Törli a felhasználó rekordját, majd JSON formában visszaadja a törölt felhasználó adatait.

- `addUserAdmin(Request $request)`

- Csak admin jogosultság esetén érhető el.
- Létrehozza az új felhasználót a kapott adatokkal (név, email, jelszó – mely bcrypttel kerül hash-elésre –, telefonszám, cím, születési dátum, admin státusz).
- Visszaküldi a létrejött felhasználó adatait JSON válaszban.

ProductController

- `getProduct()`

- Lekéri az összes terméket a ``Product::all()`` segítségével.
- JSON formátumban visszaadja a termékek listáját.

- `addProduct(Request $request)`

- Először ellenőrzi, hogy az aktuális felhasználónak van-e admin jogosultsága.
- Új ``Product`` példányt hoz létre, majd a kérésből érkező adatok (név, ár, kategória, leírás, mértékegység, akciós státusz, készlet, kedvezményes ár) beállítása történik.
- Ellenőrzi, hogy érkezett-e kép (file) a ``image_url`` kulcs alatt. Ha igen, a képet feltölti a megadott tárolóba és beállítja a termék ``image_url`` mezőjét.
- Elmenti a termék rekordját az adatbázisba, majd JSON formátumban visszaküldi a létrehozott termék adatait.

- `showProduct(Request $request)`

- Megkeresi a terméket a kérésből érkező ``id`` alapján.
- Ha a termék nem található, visszaküld egy "Nem található a termék!" üzenetet.
- Ellenkező esetben visszaadja a termék adatait JSON formában.

- updateProduct(Request \$request)

- Először admin jogosultságot ellenőriz.
- Megkeresi a frissítendő terméket az `id` alapján (`findOrFail` használata miatt hiba esetén kivételt dob).
- Ha a kérés tartalmazza a `delete_image` jelet, és annak értéke true, a régi kép fájl törlésre kerül a storage-ból, majd a termék `image_url` mezője nullázódik.
- Ha új kép érkezik, törli a régi képet (ha van), majd feltölti az új képet és frissíti a `image_url` mezőt.
- A többi mezőt (név, ár, leírás, kategória, mértékegység, készlet, akciós státusz, kedvezményes ár) szintén frissíti a kérés alapján.
- Mentés után JSON üzenetet küld vissza, amely jelzi a sikeres frissítést.

- destroyProduct(Request \$request)

- Admin jogosultság ellenőrzést hajt végre.
- Megkeresi a törlendő terméket az `id` alapján.
- Ha nem található a termék, visszaad egy hibaüzenetet.
- Ha a termék rendelkezik képpel (`image_url`), a megfelelő fájl törlésre kerül a storage rendszerből.
- Törli a termék rekordját és JSON válaszban értesíti a sikeres törlésről.

OrderController

- createOrder(OrderRequest \$request)

- Az `OrderRequest` validálja a rendelés beküldött adatait.
- A kapott `items` tömbből ellenőrzi minden egyes tétel esetén, hogy a termék létezik-e és elegendő készlet van-e a rendeléshez. Ha bármelyik terméknel nincs elegendő készlet, hibaüzenetet küld vissza.
- Létrehozza az új rendelést a kapott adatokkal:
 - `user_id`
 - `items` (JSON formátumra alakítva)
 - `totalPrice`
 - vevői adatok: `customers_name`, `customers_phone`, `customers_email`
 - `delivery_address`
 - `payment_method`
 - `status`
 - `delivery_date`

- Miután a rendelés rögzítésre került, végigiterál a rendelési tételeken, és csökkenti az egyes termékek készletét a megadott `quantity` értékkel.
- JSON válaszban visszaküldi a sikeres rendelés üzenetét és az elkészült rendelés adatait (201-es státusz).

- getOrder()

- Összegyűjti az összes rendelést a `Order::all()` segítségével.
- Visszaadja a rendeléseket JSON formátumban.

- deleteOrder(Request \$request)

- Ellenőrzi az admin jogosultságot.
- Megkeresi a törölni kívánt rendelést az azonosító alapján.
- Ha a rendelés nem található, hibaüzenettel válaszol.
- Siker esetén törli a rendelést, és visszaad egy JSON üzenetet a törlésről.

- updateOrder(Request \$request)

- Admin jogosultság ellenőrzése történik.
- Megkeresi a frissítendő rendelést az `id` alapján.
- Frissíti a rendelés kulcsfontosságú mezőit:

- `totalPrice`
- `customers_name`
- `customers_phone`
- `delivery_address`
- `payment_method`
- `status`
- `delivery_date`

- Elmenti a módosításokat, majd JSON válaszban értesíti a sikeres frissítést és visszaadja az aktuális rendelési adatokat.

- getOrdersByUser(Request \$request)

- Lekérdezi azokat a rendeléseket, amelyeknél a `user_id` megegyezik a kérésben megadott `user_id` értékkel.
- Ha nincs találat, visszaad egy üzenetet, hogy nem található rendelés a megadott felhasználóhoz.
- Ellenkező esetben JSON formátumban visszaadja a felhasználó rendeléseit.

UserController

- register(Request \$request)

- Új felhasználó létrehozása a regisztráció során a megadott adatokkal:
 - `name`, `email`, `password` (bcrypt hash), `phone`, `address`, `birth_date`
- Az `admin` mező alapértelmezetten 0 (nem admin).
- A módszer egy sikeres regisztráció üzenetet küld vissza, a nevét is tartalmazva a válaszban.

- login(Request \$request)

- Ellenőrzi az email és jelszó párost az `Auth::attempt()` segítségével.
- Siker esetén:
 - Lekéri az aktuálisan bejelentkezett felhasználót (`Auth::user()`).
 - Létrehoz egy új személyes hozzáférési tokenet a Sanctum segítségével.
 - Visszaadja a felhasználó nevét és a létrehozott tokenet JSON formátumban.
- Sikertelen bejelentkezés esetén hibaüzenettel tér vissza (401-es státusz).

- logout()

- Lekéri az aktuálisan hitelesített felhasználó aktuális hozzáférési tokenjét, majd azt törli.
- JSON válaszban visszaadja a felhasználó nevét és a sikeres kijelentkezés üzenetét.

- getUser(Request \$request)

- A Bearer token alapján próbálja megkeresni a hozzáférési tokenet a `PersonalAccessToken::findToken()` módszerrel.
- Amennyiben a token hiányzik vagy érvénytelen, hibaüzenetet küld vissza.
- Ha érvényes, a tokenhez tartozó felhasználót (`tokenable`) adja vissza JSON formában.

- changePassword(Request \$request)

- Lekéri az aktuális felhasználót (`Auth::user()`).
- Ellenőrzi, hogy a kapott régi jelszó (`old_password`) megegyezik-e a tárolt hash jelszóval a `Hash::check()` módszerrel.
- Hiba esetén visszaadja a "Hibás jelszó" üzenetet (401-es).
- Ha illeszkedik, új jelszó hash-elés után elmentésre kerül, és igazoló üzenet kerül visszaküldésre.

- changeAddress(Request \$request)

- Az aktuális felhasználót lekéri és ellenőrzi a bejelentkezést.
- Ellenőrzi, hogy az új cím (`new_address`) nem üres.
- Frissíti a felhasználó `address` mezőjét, elmenti a változást, és sikeres üzenettel tér vissza.

MailController

- sendOrderConfirmationMail(Request \$request)

- Lekéri a rendeléshez tartozó adatokat a kérésből, például:
 - `customers_email`, `customers_name`, `customers_phone`, `delivery_address`, `payment_method`
 - `delivery_date`, `totalPrice` és az `items` (JSON vagy dekódolt tömb).
- A `payment_method` értékét ember számára olvasható formára alakítja ("Kártyával" vagy "Készpénz").
- Összeállít egy tartalom tömböt, amely a rendelés visszaigazolásához szükséges adatokat tartalmazza.
- A Laravel Mail osztály segítségével elküldi az OrderConfirmationMail e-mailt a vásárló e-mail címére.

- sendRegistrationSuccessMail(Request \$request)

- Lekéri a felhasználó nevét és e-mail címét.
- Összeállít egy tartalom tömböt az üzenethez (cím: "Üdvözljük", valamint a felhasználó neve).
- Az e-mail küldés a RegistrationSuccessMail osztállyal történik a kapott e-mail címre.

- sendChangePasswordMail(Request \$request)

- A kérésből kinyeri a felhasználó nevét és e-mail címét.
- Összeállít egy tartalom tömböt a "Jelszó megváltoztatva" üzenettel.
- E-mail küldés történik a ChangePasswordMail osztály használatával.

- sendChangeAddressMail(Request \$request)

- Lekéri a felhasználó nevét, e-mail címét és új címét.
- Tartalomként összeállítja az "Cím megváltoztatva" üzenetet.
- Az e-mailt a ChangeAddressMail osztály segítségével küldi el a felhasználó e-mail címére.

- sendOrderStatusMail(Request \$request)

- Lekéri a rendelés állapotára vonatkozó adatokat: a vásárló e-mail címét, nevét, az új státuszt, kiszállítási dátumot és rendelési azonosítót.
- A `match` szerkezettel emberi nyelvű státuszt rendel az egyes enum értékekhez (pl. `pending` → "Függőben", stb.).
- Összeállít egy tartalom tömböt, majd az OrderStatusMail osztály segítségével elküldi az e-mailt a vásárlónak.

- sendAddUserMail(Request \$request)

- Lekéri a felhasználó nevét, e-mail címét és jelszavát a kérésből.
- Összeállítja az üzenetet, amely tartalmazza a "Felhasználó hozzáadva" címet, a felhasználó nevét, a jelszót és egy utasítást, hogy a felhasználó azonnal változtassa meg a jelszavát.
- Az e-mailt az AddUserMail osztály segítségével küldi el a megadott e-mail címre.

Ez a részletes ismertető lefedi az összes főbb metódust az osztályokban, beleértve a jogosultsági ellenőrzéseket, adatfeldolgozást, adatbázis műveleteket és az e-mail küldési folyamatokat.

Tesztelés

Tesztelve:

- Windows 11 operációs rendszeren
- Google Chrome böngészőben (Verzió: 135.0.7049.42)

Főoldal

A frontend sikeresen lekérte a backendtől az adatokat.

	Bemenet	Kimenet	Eredmény
Megnézem gomb	Kattintás	Átírányít az adott termék oldalára és betölti az adatokat.	Sikeres
Akcio felirat	Adat, amiben szerepel, hogy akcióban van	Kiírja az akció feliratot	Sikeres
Nincs készleten!	Adat, amiben a raktáron lévő mennyiség 0	Kiírja, hogy nincs készleten. <i>Megnézem</i> gomb tiltása	Sikeres
Lista görgetés	Kattintás vagy húzás	Lista mozgása	Sikeres
Reszponzivitás	Mobilnézet	Az oldalon lévő összes elem alkalmazkodik a képernyőhöz	Sikeres

Gyümölcsök

A frontend sikeresen lekérte a backendtől az adatokat.

	Bemenet	Kimenet	Eredmény
Megnézem gomb	Kattintás	Átírányít az adott termék oldalára és betölti az adatokat.	Sikeres
Akcio felirat	Adat, amiben szerepel, hogy akcióban van	Kiírja az akció feliratot	Sikeres
Nincs készleten!	Adat, amiben a raktáron lévő mennyiség 0	Kiírja, hogy nincs készleten. <i>Megnézem</i> gomb tiltása	Sikeres
Reszponzivitás	Mobilnézet	Az oldalon lévő összes elem alkalmazkodik a képernyőhöz	Sikeres

Zöldségek

A frontend sikeresen lekérte a backendtől az adatokat.

	Bemenet	Kimenet	Eredmény
Megnézem gomb	Kattintás	Átírányít az adott termék oldalára és betölti az adatokat.	Sikeres
Akció felirat	Adat, amiben szerepel, hogy akcióban van	Kiírja az akció feliratot	Sikeres
Nincs készleten!	Adat, amiben a raktáron lévő mennyiség 0	Kiírja, hogy nincs készleten. <i>Megnézem</i> gomb tiltása	Sikeres
Reszponzivitás	Mobilnézet	Az oldalon lévő összes elem alkalmazkodik a képernyőhöz	Sikeres

Akciók

A frontend sikeresen lekérte a backendtől az adatokat.

	Bemenet	Kimenet	Eredmény
Megnézem gomb	Kattintás	Átírányít az adott termék oldalára és betölti az adatokat.	Sikeres
Akció felirat	Adat, amiben szerepel, hogy akcióban van	Kiírja az akció feliratot	Sikeres
Nincs készleten!	Adat, amiben a raktáron lévő mennyiség 0	Kiírja, hogy nincs készleten. <i>Megnézem</i> gomb tiltása	Sikeres
Reszponzivitás	Mobilnézet	Az oldalon lévő összes elem alkalmazkodik a képernyőhöz	Sikeres

Rólunk

Sikeresen betöltődtek az adatok.

	Bemenet	Kimenet	Eredmény
<i>Böngésszen termékeink között gomb</i>	Kattintás	Főoldara navigálás	Sikeres
Reszponzivitás	Mobilnézet	Az oldalon lévő összes elem alkalmazkodik a képernyőhöz	Sikeres

Termék adatlap

A frontend sikeresen lekérte a backendtől az adatokat.

	Bemenet	Kimenet	Eredmény
Mennyiség választás	Termék mennyiség választása	Termék mennyiség kiírása	Sikeres
Összesen	Mennyiség és az ár/akciós ár szorzata	Végeredmény	Sikeres
<i>Kosárba</i> gomb	Kattintás	Modális ablak megjelenése. Sikeresen hozzáadva a kosárhoz	Sikeres
Sikeres kosárba helyezés modális ablak	-	Két gomb választás	Sikeres
<i>Folytatom a vásárlást</i> gomb	Kattintás	Modális ablak bezárása	Sikeres
<i>Kosár</i> gomb	Kattintás	Navigálás a kosárhoz	Sikeres
Reszponzivitás	Mobilnézet	Az oldalon lévő összes elem alkalmazkodik a képernyőhöz	Sikeres

Kosár

A frontend sikeresen lekérte a backendtől az adatokat.

	Bemenet	Kimenet	Eredmény
Mennyiség választás	Termék mennyiség választása	Termék mennyiség kiírása	Sikeres
Összeg	Mennyiség és az ár/akciós ár szorzata	Végeredmény	Sikeres
<i>Törlés</i> gomb	Kattintás	Modális ablak megjelenése	Sikeres
Modális ablak <i>Törlés</i> gomb	Kattintás	Termék törlése a kosárból	Sikeres
<i>Mégse</i> gomb	Kattintás	Modális ablak bezárása	Sikeres
Összesen	Termék összegek összeadása	Végeredmény kerekítve	Sikeres
<i>Tovább a megrendeléshez</i> gomb	Kattintás	Ha a felhasználó be van jelentkezve, akkor tovább visz a szállítási adatokhoz. Ha nincs bejelentkezve, akkor egy hiba üzenetet dob fel.	Sikeres
Reszponzivitás	Mobilnézet	Az oldalon lévő összes elem alkalmazkodik a képernyőhöz	Sikeres

Szállítási adatok

A frontend sikeresen lekérte a backendtől az adatokat.

	Bemenet	Kimenet	Eredmény
Adatok betöltve/törölve kapcsoló	Kattintás	Felhasználó adatainak betöltése vagy törlése a szövegbeviteli mezőkből	Sikeres
Fizetés kártyával	Kiválasztás	Kiválasztva	Sikeres
Fizetés boltban	Kiválasztás	Kiválasztva	Sikeres
Megrendelem gomb	Kattintás	<p>Üres valamelyik mező akkor üzenetet ad.</p> <p>-</p> <p>Kártyás: Betölti a Barionba a szükséges adatokat, amik a fizetéshez kellenek. Átnavigál a Barion fizetéshez.</p> <p>-</p> <p>Bolt: Sikeres rendelés. 5 másodperc múlva visszairányít a Kezdőlapra.</p>	Sikeres
Reszponzivitás	Mobilnézet	Az oldalon lévő összes elem alkalmazkodik a képernyőhöz	Sikeres

Sikeres rendelés

A frontend sikeresen lekérte a backendtől az adatokat.

	Bemenet	Kimenet	Eredmény
Rendelés azonosító	Adat	Rendelési azonosító. Sikeres megrendelés	Sikeres

Keresés

A frontend sikeresen lekérte a backendtől az adatokat.

	Bemenet	Kimenet	Eredmény
<i>Megnézem</i> gomb	Kattintás	Átirányít az adott termék oldalára és betölti az adatokat.	Sikeres
Akció felirat	Adat, amiben szerepel, hogy akcióban van	Kiírja az akció feliratot	Sikeres
Nincs készleten!	Adat, amiben a raktáron lévő mennyiség 0	Kiírja, hogy nincs készleten. <i>Megnézem</i> gomb tiltása	Sikeres
Keresés eredmény	Szó, amit a navigációs sáv keresőjébe írtunk	Keresési eredmények	Sikeres
Reszponzivitás	Mobilnézet	Az oldalon lévő összes elem alkalmazkodik a képernyőhöz	Sikeres

Bejelentkezés

Sikeresen betöltődtek az adatok.

	Bemenet	Kimenet	Eredmény
Jelszó láthatósága	Kattintás	Megjelenik/eltűnik a jelszó	Sikeres
Helyes adatok bevitele + <i>Bejelentkezés</i> gomb	Adatok	Bejelentkezés. Profil oldalra navigálás	Sikeres
Helytelen adatok bevitele + <i>Bejelentkezés</i> gomb	Adatok	Értesítés a sikertelen bejelentkezésről	Sikeres
<i>Regisztráció</i> gomb	Kattintás	Navigálás a Regisztráció oldalra	Sikeres
Reszponzivitás	Mobilnézet	Az oldalon lévő összes elem alkalmazkodik a képernyőhöz	Sikeres

Regisztráció

Sikeresen betöltődtek az adatok.

	Bemenet	Kimenet	Eredmény
Jelszó láthatósága	Kattintás	Megjelenik/eltűnik a jelszó	Sikeres
Helyes adatok bevitele + <i>Regisztráció</i> gomb	Adatok	<i>Regisztrációs</i> gomb engedélyezése. Bejelentkezés oldalra navigálás	Sikeres
<i>Bejelentkezés</i> gomb	Kattintás	Navigálás a Bejelentkezés oldalra	Sikeres
Reszponzivitás	Mobilnézet	Az oldalon lévő összes elem alkalmazkodik a képernyőhöz	Sikeres

Profil

A frontend sikeresen lekérte a backendtől az adatokat.

	Bemenet	Kimenet	Eredmény
Felhasználói adatok	-	Betöltődnek a felhasználói adatok	Sikeres
<i>Rendelések</i> gomb	Kattintás	Megnyílik egy modális ablak amiben a felhasználói rendelései szerepelnek. Üzenet a sikeres rendelés betöltésről	Sikeres
<i>Termékek</i> gomb	Kattintás	Az adott rendelés termékei megjelennek	Sikeres
<i>Vissza</i> gomb a megrendelt termékeknél	Kattintás	Visszairányít a megrendelésekhez	Sikeres
<i>Jelszó módosítása</i> gomb	Kattintás	Megjelenik a modális ablak ahol jelszót lehet változtatni	Sikeres
<i>Módosítás</i> gomb	Kattintás	Ha minden adat ki van töltve és az új jelszók egyeznek, akkor jelszómódosítás és a módosításról egy e-mail küldés. Értesítés megjelenítése a sikerességről.	Sikeres
<i>Cím módosítása</i> gomb	Kattintás	Megjelenik a modális ablak ahol címet lehet változtatni	Sikeres
<i>Módosítás</i> gomb	Kattintás	Ha a két e-mail megegyezik akkor cím módosítás. E-mail küldés a változtatásról. Értesítés megjelenítése a sikerességről.	Sikeres
Reszponzivitás	Mobilnézet	Az oldalon lévő összes elem alkalmazkodik a képernyőhöz	Sikeres

Admin profil

A frontend sikeresen lekérte a backendtől az adatokat.

	Bemenet	Kimenet	Eredmény
Felhasználói adatok	-	Betöltődnek a felhasználói adatok	Sikeres
<i>Felhasználók módosítása gomb</i>	Kattintás	Navigáció a Felhasználók kezelése oldalra	Sikeres
<i>Termékek módosítása gomb</i>	Kattintás	Navigáció a Termékek kezelése oldalra	Sikeres
<i>Rendelések módosítása gomb</i>	Kattintás	Navigáció a Rendelések kezelése oldalra	Sikeres
Reszponzivitás	Mobilnézet	Az oldalon lévő összes elem alkalmazkodik a képernyőhöz	Sikeres

Felhasználók kezelése

A frontend sikeresen lekérte a backendtől az adatokat.

	Bemenet	Kimenet	Eredmény
<i>Új felhasználó felvétele</i> gomb	Kattintás	Modális ablak megnyitása ahol a felhasználót hozzálehet adni	Sikeres
Adatok kitöltése + <i>Felvétel</i> gomb	Adatok	Adatok sikeresen hozzáadva az adatbázishoz. E-mail küldése az adott felhasználónak. Értesítés a sikerességről	Sikeres
Felhasználók keresése	Adat	Keresési eredmény	Sikeres
X gomb	Kattintás	Keresési mező tartalmának törlése	Sikeres
<i>Szerkesztés</i> gomb	Kattintás	Modális ablak megnyitása. Az adott felhasználó adatainak betöltése	Sikeres
Módosított adatok + <i>Szerkesztés</i> gomb	Adat	Adatbázisba mentés. Értesítés a sikerességről	Sikeres
<i>Törlés</i> gomb	Kattintás	Modális ablak megnyitása. Az adott felhasználó adatainak betöltése	Sikeres
<i>Törlés</i> gomb	Kattintás	Felhasználó törlése. Értesítés sikerességről	Sikeres
Reszponzivitás	Mobilnézet	Az oldalon lévő összes elem alkalmazkodik a képernyőhöz	Sikeres

Termékek kezelése

A frontend sikeresen lekérte a backendtől az adatokat.

	Bemenet	Kimenet	Eredmény
<i>Új termék felvétele</i> gomb	Kattintás	Modális ablak megnyitása ahol a termékek hozzálehet adni	Sikeres
Kép feltöltése	Kép fájl (JPEG, JPG és PNG)	Sikeres hozzáadás	Sikeres
Adatok kitöltése + <i>Felvétel</i> gomb	Adatok	Adatok sikeresen hozzáadva az adatbázishoz. Értesítés sikerességről	Sikeres
Termékek keresése	Adat	Keresési eredmény	Sikeres
X gomb	Kattintás	Keresési mező tartalmának törlése	Sikeres
<i>Szerkesztés</i> gomb	Kattintás	Modális ablak megnyitása. Az adott termék adatainak betöltése	Sikeres
<i>Kép törlése</i> gomb	Kattintás	Fotó ideiglenes törlése	Sikeres
Módosított adatok + <i>Szerkesztés</i> gomb	Adat	Adatbázisba mentés. Értesítés a sikerességről	Sikeres
<i>Törlés</i> gomb	Kattintás	Modális ablak megnyitása. Az adott termék adatainak betöltése	Sikeres
<i>Törlés</i> gomb	Kattintás	Termék törlése. Értesítés sikerességről	Sikeres
Reszponzivitás	Mobilnézet	Az oldalon lévő összes elem alkalmazkodik a képernyőhöz	Sikeres

Megrendelések kezelése

A frontend sikeresen lekérte a backendtől az adatokat.

	Bemenet	Kimenet	Eredmény
Rendelések keresése	Adat	Keresési eredmény	Sikeres
X gomb	Kattintás	Keresési mező tartalmának törlése	Sikeres
<i>Termékek</i> gomb	Kattintás	Modális ablak megnyitása. Az adott megrendelés termékeinek betöltése	Sikeres
<i>Szerkesztés</i> gomb	Kattintás	Modális ablak megnyitása. Az adott megrendelés adatainak betöltése	Sikeres
Módosított adatok + <i>Szerkesztés</i> gomb	Adat	Adatbázisba mentés. Értesítés a sikerességről. E-mail küldése a módosított adatokkal a megrendelőnek	Sikeres
<i>Törlés</i> gomb	Kattintás	Modális ablak megnyitása. Az adott megrendelés adatainak betöltése	Sikeres
<i>Törlés</i> gomb	Kattintás	Megrendelés törlése. Értesítés sikerességről	Sikeres
Reszponzivitás	Mobilnézet	Az oldalon lévő összes elem alkalmazkodik a képernyőhöz	Sikeres

Navigációs sáv

A frontend sikeresen lekérte a backendtől az adatokat.

	Bemenet	Kimenet	Eredmény
Felhasználói navigációs elemek (Kezdőlap, Gyümölcsök, Zöldségek, Akciók, Rólunk, Bejelentkezés, Regisztráció, Kosár)	Kattintás	Navigálás az adott oldalra	Sikeres
Admin navigációs elemek (Felhasználók, Termékek, Rendelések)	Kattintás	Navigálás az adott oldalra	Sikeres
Profil lekérdezése	-	Név megjelenítése a navigációs sávban	Sikeres
Profil elemek	Kattintás	Navigálás az adott oldalra	Sikeres
Keresés sáv + Keresés ikon	Keresési szó + Kattintás	Navigálás a kereső oldalra	Sikeres
Kijelentkezés	Kattintás	Felhasználó kijelentkeztetése	Sikeres
Kosár ikon	Termékszám	Termékszám megjelenítése	Sikeres
Reszponzivitás	Mobilnézet	Az oldalon lévő összes elem alkalmazkodik a képernyőhöz	Sikeres

Lábrész

Sikeresen betöltődtek az adatok.

	Bemenet	Kimenet	Eredmény
<i>Katt ide!</i> gomb	Kattintás	Navigáció a Rólunk oldalra	Sikeres
E-mail cím	Kattintás	Alapértelmezett e-mail szolgáltatás megnyitása, hogy e-mailet lehessen küldeni	Sikeres

Nyilvános Végpontok

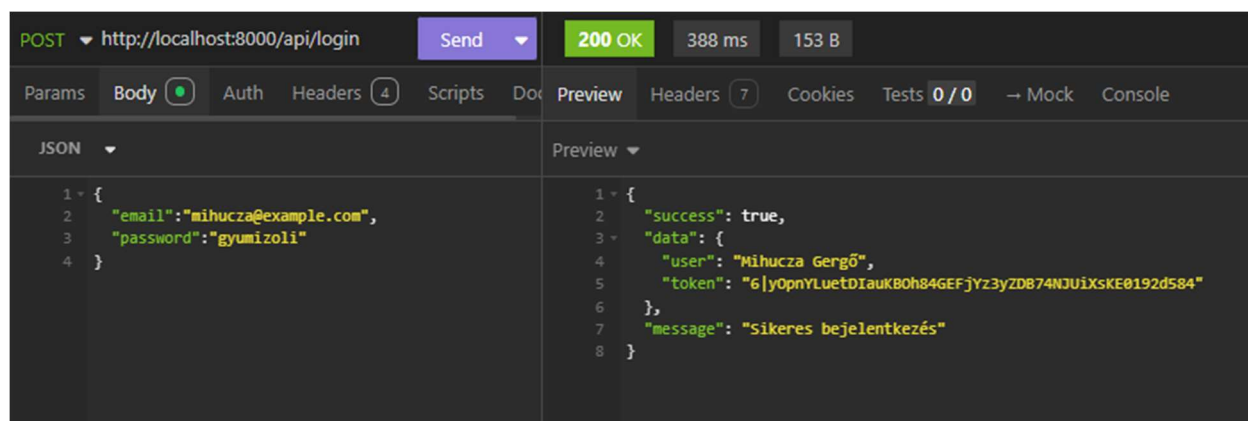
Ezek a végpontok hitelesítés nélkül érhetőek el. Tesztelve Windows 10 operációs rendszeren és Insomnia programmal való tesztelés történt.

Felhasználó bejelentkezés

URL: POST /login

Leírás: Felhasználó bejelentkezése.

Body: { "email": "string", "password": "string" }

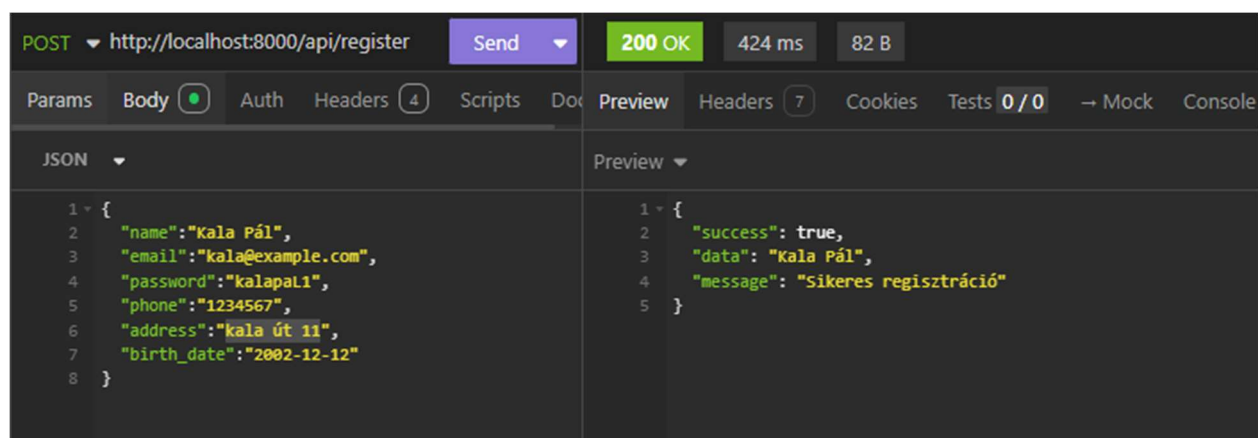


Felhasználó regisztráció

URL: POST /register

Leírás: Új felhasználó regisztrációja.

Body: { "name": "string", "email": "string", "password": "string" }



Rendelés létrehozása

URL: POST /addorder

Leírás: Új rendelés létrehozása.

Body: Rendelés adatai.

POST http://localhost:8000/api/addorder Send 201 Created 177 ms 511 B

Params Body Auth Headers 4 Scripts Docs Preview Headers 7 Cookies Tests 0/0 Mock Console

JSON

```
1 {
2   "user_id": 3,
3   "items": [
4     {
5       "product": {
6         "id": 1
7       },
8       "quantity": 2
9     },
10    {
11      "product": {
12        "id": 2
13      },
14      "quantity": 1
15    }
16  ],
17  "totalPrice": 15000,
18  "customers_name": "Teszt Elek",
19  "customers_phone": "+36123456789",
20  "customers_email": "teszt.elek@example.com",
21  "delivery_address": "1234 Budapest, Teszt utca 1.",
22  "payment_method": "card",
23  "status": "pending",
24  "delivery_date": "2025-04-10"
25 }
```

Preview

```
1 {
2   "message": "Megrendelés sikeresen létrehozva!",
3   "order": {
4     "user_id": 3,
5     "items": "[{"product":{"id":1,"quantity":2},"product":{"id":2,"quantity":1}]",
6     "totalPrice": 15000,
7     "customers_name": "Teszt Elek",
8     "customers_phone": "+36123456789",
9     "customers_email": "teszt.elek@example.com",
10    "delivery_address": "1234 Budapest, Teszt utca 1.",
11    "payment_method": "card",
12    "status": "pending",
13    "delivery_date": "2025-04-10",
14    "updated_at": "2025-04-07T18:17:59.000000Z",
15    "created_at": "2025-04-07T18:17:59.000000Z",
16    "id": 3
17  }
18 }
```

Termékek listázása


URL: GET /products

Leírás: Az összes termék lekérdezése.

GET http://localhost:8000/api/products Send 200 OK 175 ms 5.7 KB

Params Body Auth Headers 3 Scripts Docs Preview Headers 7 Cookies Tests 0/0 Mock Console

No Body



Enter a URL and send to get a response

Select a body type from above to send data in the body of a

Preview

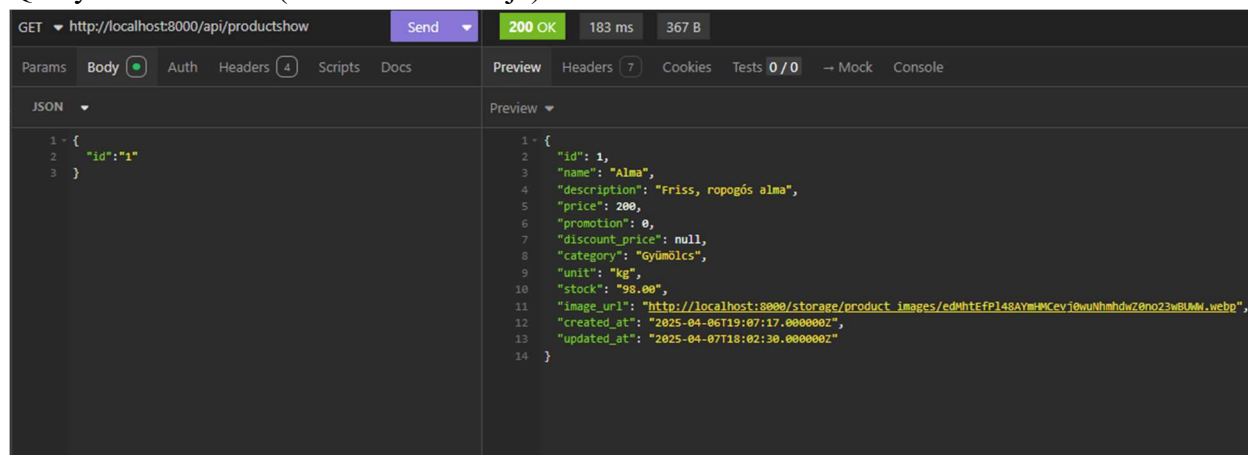
```
1 [
2   {
3     "id": 1,
4     "name": "Alma",
5     "description": "Friss, ropogós alma",
6     "price": 200,
7     "promotion": 0,
8     "discount_price": null,
9     "category": "Gyümölcs",
10    "unit": "kg",
11    "stock": "98.00",
12    "image_url": "http://localhost:8000/storage/product_images/edMhtFP148AYmWCEvj0uUhmhdwz0no23wBUm.webp",
13    "created_at": "2025-04-06T19:07:17.000000Z",
14    "updated_at": "2025-04-07T18:02:30.000000Z"
15  },
16  {
17    "id": 2,
18    "name": "körte",
19    "description": "Zamatos és ízletes körte",
20    "price": 250,
21    "promotion": 1,
22    "discount_price": 200,
23    "category": "Gyümölcs",
24    "unit": "kg",
25    "stock": "79.00",
26    "image_url": null,
27    "created_at": "2025-04-06T19:07:17.000000Z",
28    "updated_at": "2025-04-07T18:02:30.000000Z"
29  }
30 ]
```

Egy termék megtekintése

URL: GET /productshow

Leírás: Egy adott termék adatainak lekérdezése.

Query Paraméter: id (termék azonosítója)



E-mail küldés különböző eseményekhez

URL-ek:

POST /successorder

POST /successregistration

POST /changepasswordmail

POST /changeaddressmail

POST /orderstatus

POST /adduseremail

Leírás: E-mail küldése különböző eseményekhez (pl. rendelés visszaigazolás, regisztráció sikeressége stb.).

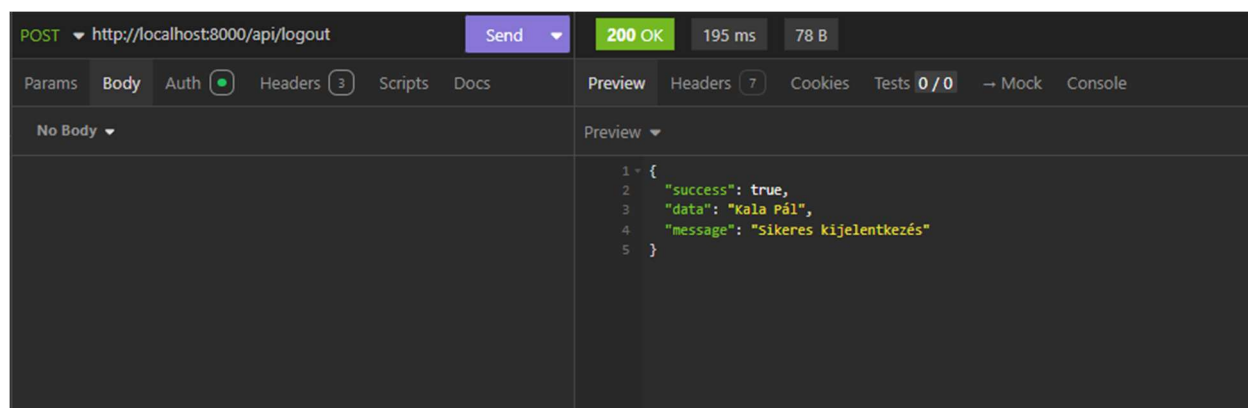
Hitelesítést Igénylő Végpontok

Ezek a végpontok csak hitelesített felhasználók számára érhetők el (auth:sanctum middleware).

Kijelentkezés

URL: POST /logout

Leírás: Felhasználó kijelentkezése.

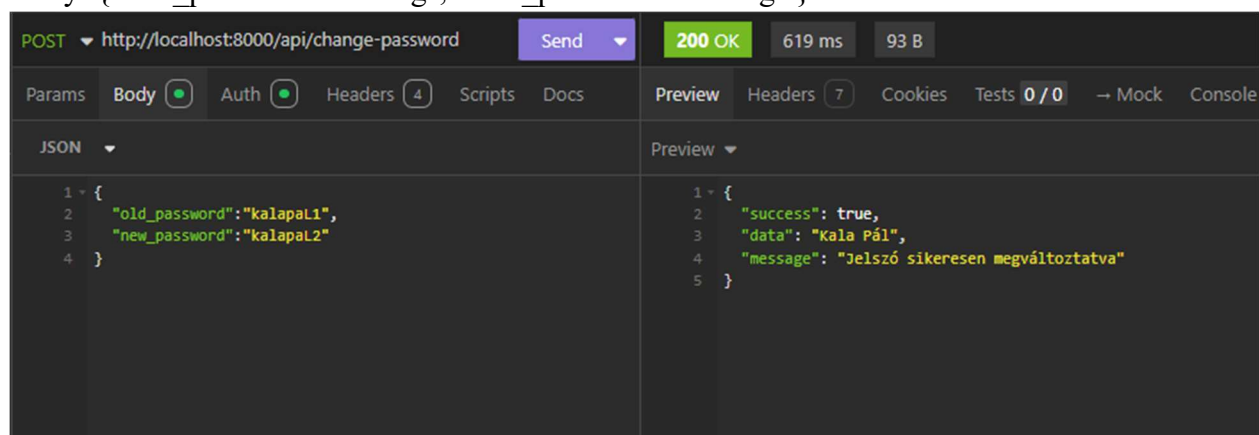


Jelszó módosítása

URL: POST /change-password

Leírás: Felhasználó jelszavának módosítása.

Body: { "old_password": "string", "new_password": "string" }



Cím módosítása

URL: POST /change-address

Leírás: Felhasználó címének módosítása.

Body: { "address": "string" }

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:8000/api/change-address
- Status:** 200 OK
- Time:** 167 ms
- Size:** 90 B
- Body (JSON):**

```
1 {
2   "new_address": "kala út 12"
3 }
```
- Preview:**

```
1 {
2   "success": true,
3   "data": "Kala Pál",
4   "message": "Cím sikeresen megváltoztatva"
5 }
```

Felhasználó saját rendeléseinek listázása

URL: GET /orders

Leírás: Az összes rendelés lekérdezése.

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:8000/api/orders
- Status:** 200 OK
- Time:** 163 ms
- Size:** 447 B
- Body:** No Body
- Preview:**

```
1 [
2   {
3     "id": 1,
4     "user_id": 1,
5     "totalPrice": 15000,
6     "items": "[{"product":{"id":1,"quantity":2},"product":{"id":2,"quantity":1}}",
7     "customers_name": "Teszt Elek",
8     "customers_phone": "+36123456789",
9     "customers_email": "teszt.elek@example.com",
10    "delivery_date": "2025-04-10",
11    "delivery_address": "1234 Budapest, Teszt utca 1.",
12    "payment_method": "card",
13    "status": "pending",
14    "created_at": "2025-04-07T18:02:30.000000Z",
15    "updated_at": "2025-04-07T18:02:30.000000Z"
16  }
17 ]
```

Rendelés frissítése

URL: POST /updateorder

Leírás: Egy rendelés adatainak frissítése.

Body: Rendelés adatai.

The screenshot shows a REST client interface with a POST request to `http://localhost:8000/api/updateorder`. The request body is a JSON object containing order details. The response is a 200 OK status with a 162 ms response time and 520 B body. The response body is a JSON object containing a success message and the updated order details.

```
POST http://localhost:8000/api/updateorder 200 OK 162 ms 520 B
```

Params Body Auth Headers 4 Scripts Docs Preview Headers 7 Cookies Tests 0/0 → Mock Console

JSON

```
1 {
2   "id": 1,
3   "totalPrice": 20000,
4   "customers_name": "Teszt Elek",
5   "customers_phone": "+36123456789",
6   "delivery_address": "1234 Budapest, Frissített utca
7   2.",
8   "payment_method": "card",
9   "status": "shipped",
10  "delivery_date": "2025-04-15"
11 }
```

Preview

```
1 {
2   "message": "Megrendelés sikeresen frissítve!",
3   "order": {
4     "id": 1,
5     "user_id": 1,
6     "totalPrice": 20000,
7     "items": "[{"product":{"id":1,"quantity":2},"product":{"id":2,"quantity":1}]",
8     "customers_name": "Teszt Elek",
9     "customers_phone": "+36123456789",
10    "customers_email": "teszt.elek@example.com",
11    "delivery_date": "2025-04-15",
12    "delivery_address": "1234 Budapest, Frissített utca 2.",
13    "payment_method": "card",
14    "status": "shipped",
15    "created_at": "2025-04-07T18:02:30.000000Z",
16    "updated_at": "2025-04-07T18:14:11.000000Z"
17  }
18 }
```

Rendelés törlése

URL: DELETE /orderdestroy

Leírás: Egy rendelés törlése.

Body: { "id": "integer" }

The screenshot shows a REST client interface with a DELETE request to `http://localhost:8000/api/orderdestroy`. The request body is a JSON object containing the order ID. The response is a 200 OK status with a 195 ms response time and 37 B body. The response body is a string message indicating the order was successfully deleted.

```
DELETE http://localhost:8000/api/orderdestroy 200 OK 195 ms 37 B
```

Params Body Auth Headers 4 Scripts Docs Preview Headers 7 Cookies Tests

JSON

```
1 {
2   "id": "1"
3 }
```

Preview

```
1 "Megrendelés törölve!"
```

Felhasználó rendeléseinek lekérdezése

URL: GET /getcustomersorders

Leírás: Egy adott felhasználó rendeléseinek lekérdezése.

Query Paraméter: user_id (felhasználó azonosítója)

GET http://localhost:8000/api/getcustomersorders Send 200 OK 162 ms 447 B

Params Body Auth Headers (4) Scripts Docs Preview Headers (7) Cookies Tests 0/0 → Mock Console

JSON

```
1 {
2   "user_id": "3"
3 }
```

Preview

```
1 [
2   {
3     "id": 3,
4     "user_id": 3,
5     "totalPrice": 15000,
6     "items": "[{"product":{"id":1,"quantity":2},"product":{"id":2,"quantity":1}}]",
7     "customers_name": "Teszt Elek",
8     "customers_phone": "+36123456789",
9     "customers_email": "teszt.elek@example.com",
10    "delivery_date": "2025-04-10",
11    "delivery_address": "1234 Budapest, Teszt utca 1.",
12    "payment_method": "card",
13    "status": "pending",
14    "created_at": "2025-04-07T18:17:59.000000Z",
15    "updated_at": "2025-04-07T18:17:59.000000Z"
16  }
17 ]
```

Admin jogosultság beállítása

URL: PUT /users/set-admin

Leírás: Egy felhasználó admin jogosultságának beállítása.

Body: { "id": "integer", "admin": "boolean" }

PUT http://localhost:8000/api/users/set-admin Send 200 OK 153 ms 289 B

Params Body Auth Headers (4) Scripts Docs Preview Headers (7) Cookies Tests 0/0 → Mock Console

JSON

```
1 {
2   "id": "3",
3   "admin": 1
4 }
5 }
```

Preview

```
1 {
2   "id": 3,
3   "name": "Kala Pál",
4   "email": "kala@example.com",
5   "email_verified_at": null,
6   "phone": "1234567",
7   "address": "kala út 12",
8   "birth_date": "2002-12-12",
9   "admin": 1,
10  "login_counter": 0,
11  "banning_time": null,
12  "created_at": "2025-04-07T18:06:07.000000Z",
13  "updated_at": "2025-04-07T18:21:21.000000Z"
14 }
```

Új admin felhasználó hozzáadása

URL: POST /users/add

Leírás: Új admin felhasználó létrehozása.

Body: Felhasználó adatai.

POST http://localhost:8000/api/users/add

Send 200 OK 412 ms 240 B

Params Body Auth Headers 4 Scripts Docs

JSON

```
1 {
2   "name": "Nagy Kata",
3   "email": "kata@example.com",
4   "password": "password123",
5   "phone": "+36123456789",
6   "address": "1234 Budapest, Admin utca 1.",
7   "birth_date": "1990-01-01",
8   "admin": 0
9 }
```

Preview

```
1 {
2   "name": "Nagy Kata",
3   "email": "kata@example.com",
4   "phone": "+36123456789",
5   "address": "1234 Budapest, Admin utca 1.",
6   "birth_date": "1990-01-01",
7   "admin": 0,
8   "updated_at": "2025-04-07T18:25:08.000000Z",
9   "created_at": "2025-04-07T18:25:08.000000Z",
10  "id": 4
11 }
```

Felhasználó adatainak lekérdezése

URL: GET /getuser

Leírás: Egy adott felhasználó adatainak lekérdezése.

GET http://localhost:8000/api/getuser

Send 200 OK 193 ms 377 B

Params Body Auth Headers 3 Scripts Docs

Bearer Token

ENABLED ☒

TOKEN

PREFIX

Preview

```
1 {
2   "success": true,
3   "data": {
4     "id": 1,
5     "name": "Mihucz Gergő",
6     "email": "mihucz@example.com",
7     "email_verified_at": null,
8     "phone": "1234567890",
9     "address": "Révay utca 1, Budapest",
10    "birth_date": "2004-11-08",
11    "admin": 1,
12    "login_counter": 0,
13    "banning_time": null,
14    "created_at": "2025-04-06T19:07:17.000000Z",
15    "updated_at": "2025-04-06T17:23:54.000000Z"
16  },
17   "message": "Felhasználó adatai"
18 }
```

Felhasználók listázása

URL: GET /users

Leírás: Az összes felhasználó adatainak lekérdezése.

GET http://localhost:8000/api/users

Send

200 OK

157 ms

1222 B

Params

Body

Auth

Headers 3

Scripts

Docs

Bearer Token

Preview

Headers 7

Cookies

Tests 0 / 0

→ Mock

Console

ENABLED ☒

TOKEN

.....

PREFIX

Preview

```
1 [
2 {
3   "id": 1,
4   "name": "Mihucz Gergő",
5   "email": "mihucz@example.com",
6   "email_verified_at": null,
7   "phone": "1234567890",
8   "address": "Révay utca 1, Budapest",
9   "birth_date": "2004-11-08",
10  "admin": 1,
11  "login_counter": 0,
12  "banning_time": null,
13  "created_at": "2025-04-06T19:07:17.000000Z",
14  "updated_at": "2025-04-06T17:23:54.000000Z"
15 },
16 {
17   "id": 2,
18   "name": "Bölgér Bence",
19   "email": "boger@example.com",
20   "email_verified_at": null,
21   "phone": "0987654321",
22   "address": "Révay utca 2, Debrecen",
23   "birth_date": "2002-11-02",
24   "admin": 1,
25   "login_counter": 0,
26   "banning_time": null,
27   "created_at": "2025-04-06T19:07:17.000000Z",
28   "updated_at": "2025-04-06T19:07:17.000000Z"
29 },
30 ]
```

Felhasználó adatainak frissítése

URL: PUT /users/update

Leírás: Egy felhasználó adatainak frissítése.

Body: Felhasználó adatai.

PUT http://localhost:8000/api/users/update

Send

200 OK

188 ms

308 B

Params

Body

Auth

Headers 4

Scripts

Docs

JSON

Preview

Headers 7

Cookies

Tests 0 / 0

→ Mock

1 {

2 "id": "4",

3 "name": "Nagy Sára",

4 "email": "kata@example.com",

5 "password": "password123",

6 "phone": "+36123456789",

7 "address": "1234 Budapest, Admin utca 1.",

8 "birth_date": "1990-01-01",

9 "admin": 0

10 }

Preview

```
1 {
2   "id": 4,
3   "name": "Nagy Sára",
4   "email": "kata@example.com",
5   "email_verified_at": null,
6   "phone": "+36123456789",
7   "address": "1234 Budapest, Admin utca 1.",
8   "birth_date": "1990-01-01",
9   "admin": 0,
10  "login_counter": 0,
11  "banning_time": null,
12  "created_at": "2025-04-07T18:25:08.000000Z",
13  "updated_at": "2025-04-07T18:29:18.000000Z"
14 }
```

Felhasználó törlése

URL: DELETE /users/delete

Leírás: Egy felhasználó törlése.

Body: { "id": "integer" }

DELETE http://localhost:8000/api/users/delete Send 200 OK 169 ms 308 B

Params Body Auth Headers 4 Scripts Docs Preview Headers 7 Cookies Tests 0/0 → Mock

JSON Preview

```
1 {  
2   "id": "4"  
3 }  
  
1 {  
2   "id": 4,  
3   "name": "Nagy Sára",  
4   "email": "kata@example.com",  
5   "email_verified_at": null,  
6   "phone": "+36123456789",  
7   "address": "1234 Budapest, Admin utca 1.",  
8   "birth_date": "1990-01-01",  
9   "admin": 0,  
10  "login_counter": 0,  
11  "banning_time": null,  
12  "created_at": "2025-04-07T18:25:08.000000Z",  
13  "updated_at": "2025-04-07T18:29:18.000000Z"  
14 }
```

Új termék hozzáadása

URL: POST /addproduct

Leírás: Új termék hozzáadása.

Body: Termék adatai.

POST http://localhost:8000/api/addproduct Send 200 OK 159 ms 314 B

Params Body Auth Headers 4 Scripts Docs Preview Headers 7 Cookies Tests 0/0 → Mock Console

JSON Preview

```
1 {  
2   "name": "Sárkánygyümölcs",  
3   "price": 1200,  
4   "category": "Gyümölcs",  
5   "description": "Egzotikus, friss sárkánygyümölcs.",  
6   "unit": "db",  
7   "promotion": 0,  
8   "stock": 30,  
9   "discount_price": null,  
10  "image_url": ""  
11 }  
  
1 {  
2   "name": "Sárkánygyümölcs",  
3   "price": 1200,  
4   "category": "Gyümölcs",  
5   "description": "Egzotikus, friss sárkánygyümölcs.",  
6   "unit": "db",  
7   "promotion": 0,  
8   "stock": 30,  
9   "discount_price": null,  
10  "updated_at": "2025-04-07T18:33:21.000000Z",  
11  "created_at": "2025-04-07T18:33:21.000000Z",  
12  "id": 22  
13 }
```

Termék frissítése

URL: POST /updateproduct

Leírás: Egy termék adatainak frissítése.

Body: Termék adatai.

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:8000/api/updateproduct
- Status:** 200 OK
- Time:** 205 ms
- Size:** 30 B
- Body:** JSON
- Body Content:**

```
1 {  
2   "id": 22,  
3   "name": "Sárkánygyümölcs",  
4   "price": 1500,  
5   "category": "Gyümölcs",  
6   "description": "Friss",  
7   "unit": "db",  
8   "promotion": 1,  
9   "stock": 25,  
10  "discount_price": 1200,  
11  "image_url": ""  
12 }
```
- Preview:** "Sikeres frissítés!"

Termék törlése

URL: DELETE /productdestroy

Leírás: Egy termék törlése.

Body: { "id": "integer" }

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** http://localhost:8000/api/productdestroy
- Status:** 200 OK
- Time:** 189 ms
- Size:** 27 B
- Body:** JSON
- Body Content:**

```
1 {  
2   "id": "22"  
3 }
```
- Preview:** "Sikeres törlés!"

Fejlesztési lehetőségek

Az oldalt tovább lehet fejleszteni, hogy támogassa:

- a többnyelvűséget (angol, stb)
- több termék hozzáadását különböző kategóriákban
- közvetlen üzenetküldési lehetőséget a weboldalon keresztül
- ChatBot integrációját
- termékek szűrése különböző szempontok szerint (ár, mértékegység stb.)
- kinézet fejlesztés, hogy a weblap mindig a legjobb stílusát mutassa a vásárló felé
- keresés fejlesztése
- profilkép feltöltése a profil oldalra

Összegzés

Jelenlegi megítélésünk alapján a webshopunkat a mi elképzeléseink szerint sikerült megvalósítani, nyitott kapukat hagytunk az esetleges fejlesztésekhez. Ez a projekt nagy mennyiségben segítette a programozási tudásunk elmélyülését és a csapatban való együttműködésünket is nagy százalékban segítette. A projektet bármikor kiindulási alapnak tudjuk kezelni akár egy más projekthez vagy új munkához.

Készítők:

Frontend - Bógér Bence

Backend - Mihucz Gergő