Webáruház

Fejlesztői dokumentáció

Szlávicsek Ákos, Szücs Bence, Puskai Balázs

2025

# Bevezetés

## Projekt célja

Ez a projekt egy önállóan üzemeltethető webáruház platform, amelyet bárki használhat saját online bolt létrehozására. A rendszer különlegessége, hogy:

* **Fejlesztőbarát:** A forráskód jól strukturált, moduláris, így fejlesztők számára könnyen értelmezhető és bővíthető.
* **Testreszabhatóság:** Fejlesztők egyszerűen módosíthatják a kinézetet, funkciókat vagy akár integrálhatják más rendszerekkel.
* **Nyílt és független:** Nem kötődik egyetlen szolgáltatóhoz vagy tárhelyhez sem. A tulajdonos valóban saját webshopot futtat.

## Fő jellemzők

* Teljeskörű webáruház funkcionalitás (termékek, kategóriák, kosár, rendelés, fizetés)
* Admin felület az egyszerű kezelhetőséghez
* Plugin-alapú bővíthetőség (pl. kuponok, szállítási módok, értesítések)
* REST API
* Beépített biztonsági alapok (CSRF védelem, szerepkörök)
* Könnyű telepíthetőség: helyi fejlesztéshez, majd deploy bármilyen PHP/Nginx vagy Node.js környezetbe

## Készítők

* Szlávicsek Ákos – Frontend és UIX/UI
* Puskai Balázs – Frontend és Tesztelő
* Szücs Bence – Backend és Adatbázis kezelő

# Technológiák

## Frontend

### Programozási nyelvek

* HTML
* CSS
* EcmaScript (JavaScript)

### Keretrendszerek

* Vite
* Tailwind

## Backend

### Programozási nyelvek

* PHP

### Keretrendszerek

* Laravel (12 verzió)
* Node.js

## Adatbázis

A laravel 12 az alábbi adatbázisókat támogatja:

* MariaDB 10.3+
* MySQL 5.7+
* PostgreSQL 10.0+
* SQLite 3.26.0+
* SQL Server 2017+

Forrás: <https://laravel.com/docs/12.x/database>

## Források

* Laravel: <https://laravel.com>
* Node.js: <https://nodejs.org/en>
* Tailwind: <https://tailwindcss.com>
* Vite: <https://vite.dev>

# Modellek

User

carts()

Ez a metódus a felhasználóhoz tartozó kosarakat adja vissza, amelyeknek az állapota „empty” vagy „loaded”.

A kapcsolat típusa: hasMany (egy felhasználónak több kosara is lehet).

A metódus csak azokat a kosarakat szűri ki, amelyek aktuálisan még nem kerültek megrendelésre vagy feldolgozásra.

roles()

Ez a metódus a felhasználóhoz rendelt szerepköröket adja vissza.

A kapcsolat típusa: hasMany, mivel egy felhasználónak több szerepe is lehet (pl. admin, vásárló)

hasRole(Roles $role): bool

Ez a metódus azt vizsgálja, hogy a felhasználó rendelkezik-e egy adott szerepkörrel.

A Roles típus valószínűleg egy enumeráció vagy konstansokat tartalmazó osztály, amely felsorolja az összes lehetséges szerepet.

Paraméterek:

• Roles $role: A vizsgált szerepkör.

Visszatérési érték:

• true, ha a felhasználónak van ilyen szerepe.

• false, ha nincs.

4. totalCartPrice(): int

Ez a metódus kiszámítja a felhasználóhoz tartozó összes kosár összértékét.

A metódus feltételezi, hogy a kosár (Cart) modell rendelkezik egy totalPrice() metódussal, amely visszaadja az adott kosár értékét.

Visszatérési érték:

A kosarak árainak összege egész számként (int).

A Role modell a felhasználókhoz tartozó szerepkörök kezeléséért felelős. Egy felhasználóhoz több szerep is tartozhat, és minden szerepkör egy konkrét típust (pl. admin, vásárló, stb.) képvisel.

Alapinformációk

• Az osztály az App\Models névtérben található.

• Az Illuminate\Database\Eloquent\Model osztályból származik.

• Használja a HasFactory traitet a gyári alapú példányosításhoz.

• Az adatbázisban az elsődleges kulcs (primaryKey) mező a user\_id.

• Az időbélyeg mezők (created\_at, updated\_at) nem kerülnek automatikusan kezelésre, mivel a $timestamps tulajdonság false.

Kitölthető mezők ($fillable)

• user\_id: A felhasználó azonosítója, akinek a szerepköre van tárolva.

• role: A szerepkör típusa, amelyet az App\Roles enumeráció reprezentál.

Típuskonverziók ($casts)

• A role mező az App\Roles enum típusra van konvertálva, így a szerepkör típusát típusbiztosan lehet használni a kódban.

Kapcsolatok

• user():

Egy belongsTo kapcsolat a User model felé.

Ez biztosítja, hogy a szerepkörhöz tartozó felhasználó elérhető legyen.

Product modell leírása

A Product modell a webshop termékeit reprezentálja. Ez a modell felelős egy adott termék összes fontos tulajdonságának és kapcsolódó adatainak kezeléséért az adatbázisban.

Alapinformációk

• Az osztály az App\Models névtérben található.

• Az Illuminate\Database\Eloquent\Model osztályból származik.

• Használja a HasFactory traitet, amely lehetővé teszi teszteléshez vagy seed-eléshez gyárak használatát.

• A modell nem használ időbélyeg mezőket (created\_at, updated\_at), mivel a $timestamps tulajdonság false.

• Használja a SoftDeletes traitet, amely lehetővé teszi a „puha törlést” (azaz a rekord nem törlődik fizikailag, csak egy deleted\_at mező kerül kitöltésre).

Kitölthető mezők ($fillable)

A következő mezők adhatók tömegesen a modellhez:

• title: A termék neve.

• slug: A termék URL-ben használt egyedi azonosítója.

• description: A termék részletes leírása.

• price: A termék ára (valószínűleg egész szám vagy decimális típus).

• category\_id: A termékhez tartozó kategória azonosítója.

• picture: A termékhez tartozó kép (pl. fájlnév vagy URL).

• summary: A termék rövid leírása.

• quantity: A készleten lévő mennyiség.

• in\_stock: Boolean vagy logikai érték, amely azt jelzi, hogy van-e raktáron.

• visible: Boolean, hogy megjelenjen-e a vásárlók számára.

Kapcsolatok

• category():

Egy belongsTo kapcsolat a Category modellhez.

Ez a metódus azt határozza meg, hogy a termék mely kategóriához tartozik.

• discount():

Egy hasOne kapcsolat a Discount modellhez.

Ez azt jelenti, hogy egy termékhez egy kedvezmény tartozhat (vagy egyáltalán nem tartozik).

OrderInfo modell leírása

Az OrderInfo modell a felhasználók megrendeléseihez tartozó számlázási és szállítási adatokat tárolja. Ez az adatmodell kapcsolódik a rendelés leadása során megadott információkhoz, mint például a vásárló neve, címe, fizetési módja és elérhetőségei.

Alapinformációk

• Az osztály az App\Models névtérben található.

• Az Illuminate\Database\Eloquent\Model osztályból származik.

• Használja a HasFactory traitet, amely lehetővé teszi a modellek gyári példányosítását (pl. teszteléshez vagy adatfeltöltéshez).

⸻

Kitölthető mezők ($fillable)

A modell az alábbi mezőket engedi tömegesen kitölteni:

• user\_id: A felhasználó azonosítója, aki leadta a rendelést.

• name: A vásárló neve.

• address: A szállítási cím.

• postal\_code: A szállítási irányítószám.

• city: A város, ahová a rendelést szállítani kell.

• phone: A vásárló telefonszáma.

• email: A vásárló e-mail címe.

• billing\_name: A számlázási név.

• billing\_address: A számlázási cím.

• billing\_postal\_code: A számlázási irányítószám.

• billing\_phone: A számlázási telefonszám.

• billing\_email: A számlázási e-mail cím.

• payment\_method: A választott fizetési mód (pl. bankkártya, utánvét, stb.).

• payment\_status: A fizetés állapota (pl. „fizetve”, „függőben”, stb.).

• payment\_transaction\_id: A fizetéshez kapcsolódó tranzakció azonosítója (pl. a bank vagy szolgáltató által generált ID).

Order modell leírása

Az Order modell a webshopban leadott megrendeléseket reprezentálja. Ez a modell tárolja a megrendeléshez tartozó felhasználói, kosár- és rendelési információkat, valamint a megrendelés állapotát és időpontját.

Alapinformációk

• Az osztály az App\Models névtérben található.

• Az Illuminate\Database\Eloquent\Model osztályból származik.

• Használja a HasFactory traitet, amely támogatja a gyári példányosítást adatfeltöltéshez vagy teszteléshez.

• Az időbélyegek (created\_at, updated\_at) nincsenek automatikusan kezelve, mivel a $timestamps tulajdonság false.

Típuskonverziók ($casts)

• status: A rendelés állapota az App\OrderStatus enum osztály segítségével kerül tárolásra és visszaadásra típusbiztos módon.

Kitölthető mezők ($fillable)

• user\_id: A rendelést leadó felhasználó azonosítója.

• cart\_id: Az ehhez a rendeléshez tartozó kosár azonosítója.

• status: A rendelés aktuális állapota (pl. függőben, feldolgozás alatt, kiszállítva stb.).

• ordered\_at: A megrendelés időpontja.

• order\_info\_id: A rendeléshez tartozó számlázási és szállítási adatok azonosítója.

Kapcsolatok

• user():

Egy belongsTo kapcsolat a User modellhez.

Ez a kapcsolat határozza meg, melyik felhasználóhoz tartozik a megrendelés.

• cart():

Egy belongsTo kapcsolat a Cart modellhez.

A megrendelés a leadáskori kosár tartalmát tárolja ezzel a kapcsolattal.

Discount modell leírása

A Discount modell a termékekhez tartozó kedvezményeket kezeli. Ezen a modellen keresztül adhatók meg, hogy egy adott termékre mekkora százalékos kedvezmény érvényes, és milyen időintervallumban.

Alapinformációk

• Az osztály az App\Models névtérben található.

• Az Illuminate\Database\Eloquent\Model osztályból származik.

• Használja a HasFactory traitet, így támogatja a factory-alapú tesztelést és seedelést.

• Az időbélyegek (created\_at, updated\_at) nincsenek kezelve, mivel a $timestamps tulajdonság false.

• Az elsődleges kulcs a product\_id mező.

• Az elsődleges kulcs nem autoinkrementált ($incrementing = false), és típusa egész szám ($keyType = 'int').

Kitölthető mezők ($fillable)

• product\_id: A termék azonosítója, amelyhez a kedvezmény tartozik.

• percentage: A kedvezmény mértéke százalékban (pl. 15).

• starts\_at: A kedvezmény kezdő időpontja (dátum vagy időbélyeg).

• ends\_at: A kedvezmény lejárati időpontja.

Kapcsolatok

• product():

Egy belongsTo kapcsolat a Product modellhez.

Ez biztosítja, hogy a kedvezmény a megfelelő termékhez legyen társítva.

Coupon modell leírása

A Coupon modell a webshopban felhasználható kuponkódokat reprezentálja. A kuponok meghatározott százalékos kedvezményt biztosítanak egy időszakon belül, ha a vásárló megadja a kuponkódot.

Alapinformációk

• Az osztály az App\Models névtérben található.

• Az Illuminate\Database\Eloquent\Model osztályból származik.

• Használja a HasFactory traitet, amely támogatja a modellek példányosítását teszteléshez vagy seed-eléshez.

• Az időbélyegek automatikus kezelése ki van kapcsolva, mivel a $timestamps tulajdonság értéke false.

Kitölthető mezők ($fillable)

• code: A kupon egyedi azonosító kódja, amelyet a felhasználó megad.

• percentage: A kedvezmény mértéke százalékban (pl. 10).

• starts\_at: A kupon érvényességének kezdődátuma.

• ends\_at: A kupon lejárati dátuma.

Metódusok

• isValid(): bool

Ez a metódus ellenőrzi, hogy a kupon jelenleg érvényes-e.

Akkor érvényes, ha a jelenlegi idő (now()) a starts\_at és ends\_at közé esik.

Visszatérési érték:

• true, ha a kupon most felhasználható.

• false, ha már lejárt vagy még nem aktív.

Category modell leírása

A Category modell a webshop termékkategóriáit reprezentálja. Ez a modell lehetővé teszi a termékek kategorizálását, valamint a szülő-gyerek kategóriák kezelését.

Alapinformációk

• Az osztály az App\Models névtérben található.

• Az Illuminate\Database\Eloquent\Model osztályból származik.

• Használja a HasFactory traitet, amely a gyári alapú példányosítást segíti elő.

• Használja a SoftDeletes traitet, amely biztosítja, hogy a rekordok nem kerülnek törlésre, hanem csak „puha törlés” történik (deleted\_at mező kitöltése).

• A modell a kategóriák nevét, leírását, tageket és szülő kategóriát tárolja.

Kitölthető mezők ($fillable)

A következő mezők adhatók tömegesen a modellhez:

• name: A kategória neve.

• description: A kategória leírása.

• tags: A kategóriahoz tartozó címkék, melyek tömbként tárolódnak.

• parent\_id: A szülő kategória azonosítója (ha van).

Típuskonverziók ($casts)

• tags: A tags mezőt tömb típusra konvertálja, lehetővé téve, hogy a címkék tömbként legyenek tárolva és visszaadva.

Kapcsolatok

• parent():

Egy belongsTo kapcsolat a Category modellhez.

Ez határozza meg, hogy egy kategória melyik szülő kategóriához tartozik.

• products():

Egy hasMany kapcsolat a Product modellhez.

Egy kategóriához több termék is tartozhat, és a products() metódus visszaadja a kategóriához tartozó összes terméket.

CartItem modell leírása

A CartItem modell egy adott kosárba tartozó terméket reprezentál. Ez a modell tartalmazza a termék azonosítóját, a kosár azonosítóját és a termék mennyiségét.

Alapinformációk

• Az osztály az App\Models névtérben található.

• Az Illuminate\Database\Eloquent\Model osztályból származik.

• Használja a HasFactory traitet, amely támogatja a gyári példányosítást (pl. teszteléshez vagy seedeléshez).

• A $timestamps tulajdonság értéke false, ami azt jelenti, hogy az időbélyegek (pl. created\_at, updated\_at) nem lesznek automatikusan kezelve.

• A modell a kosárban lévő termékek mennyiségét és azok kapcsolódó információit tárolja.

Kitölthető mezők ($fillable)

A következő mezők adhatók tömegesen a modellhez:

• product\_id: A termék azonosítója, amely a kosárhoz tartozik.

• cart\_id: A kosár azonosítója, amelyhez a termék tartozik.

• quantity: A kosárban lévő termék mennyisége.

Kapcsolatok

• product():

Egy belongsTo kapcsolat a Product modellhez.

Ez határozza meg, hogy egy kosárban lévő termékhez melyik termék tartozik.

Cart modell leírása

A Cart modell a felhasználók bevásárló kosarait reprezentálja. A kosár tárolja azokat az információkat, amelyek szükségesek a termékek kosárba helyezéséhez, azok állapotának kezeléséhez és a kosár összesített árának kiszámításához.

Alapinformációk

• Az osztály az App\Models névtérben található.

• Az Illuminate\Database\Eloquent\Model osztályból származik.

• Használja a HasFactory traitet, amely lehetővé teszi a gyári alapú példányosítást (pl. teszteléshez vagy seedeléshez).

• Az időbélyegek automatikus kezelése ki van kapcsolva, mivel a $timestamps tulajdonság értéke false.

Kitölthető mezők ($fillable)

A következő mezők adhatók tömegesen a modellhez:

• user\_id: A kosárhoz tartozó felhasználó azonosítója.

• cart\_id: A kosár egyedi azonosítója.

• status: A kosár állapota (pl. üres, betöltött, lezárt).

Kapcsolatok

• user():

Egy belongsTo kapcsolat a User modellhez.

Ez biztosítja, hogy a kosár egy adott felhasználóhoz tartozik.

• items():

Egy hasMany kapcsolat a CartItem modellhez.

A kosár több terméket is tartalmazhat, és a items() metódus visszaadja a kosárban lévő összes terméket.

Metódusok

• isOpen(): bool:

Ez a metódus azt vizsgálja, hogy a kosár nyitott-e (azaz üres vagy betöltött állapotban van-e).

Visszatérési érték:

• true, ha a kosár nyitott (üres vagy betöltött),

• false, ha a kosár lezárt.

• addItem($product, int $quantity = 1):

Ez a metódus hozzáad egy terméket a kosárhoz a megadott mennyiséggel.

Paraméterek:

• $product: A hozzáadni kívánt termék.

• $quantity: A hozzáadni kívánt mennyiség (alapértelmezett érték: 1).

• close():

A kosár állapotát lezártra állítja, és elmenti a módosítást.

• totalPrice():

A kosár összesített árát számolja ki az összes kosárban lévő termék árából és mennyiségéből.

Visszatérési érték:

A kosár összesített ára.

# Kontrollerek

AuthController modell leírása

Az AuthController osztály a felhasználói hitelesítéshez tartozó vezérlő. A felhasználók regisztrálásáért, bejelentkezéséért és kijelentkezéséért felelős metódusokat tartalmaz.

Alapinformációk

• Az osztály az App\Http\Controllers névtérben található.

• A Controller osztályból származik.

• A vezérlő a User, Cart és CartStatus modellekkel dolgozik a felhasználói regisztrációk és bejelentkezések kezelésében.

Metódusok

1. showLogin()

Ez a metódus visszaadja a bejelentkezési űrlap nézetet.

Visszatérési érték:

view('auth.login'): A resources/views/auth/login.blade.php fájlt jeleníti meg.

2. showRegister()

Ez a metódus visszaadja a regisztrációs űrlap nézetet.

Visszatérési érték:

view('auth.register'): A resources/views/auth/register.blade.php fájlt jeleníti meg.

3. register(Request $request)

A regisztrációs metódus, amely a felhasználó által kitöltött adatokat validálja és új felhasználót hoz létre.

• Validálja a beérkező adatokat, mint név, email, és jelszó.

• A jelszót hash-eli és eltárolja az adatbázisban.

• A sikeres regisztráció után átirányítja a felhasználót a bejelentkezési oldalra.

Paraméterek:

• $request: Az HTTP kérés, amely a felhasználó adatait tartalmazza.

Visszatérési érték:

redirect()->route('login'): A felhasználót a bejelentkezési oldalra irányítja egy sikeres regisztrációs üzenettel.

4. login(Request $request)

A bejelentkezési metódus, amely a felhasználó által megadott hitelesítő adatokat validálja, és ha azok helyesek, bejelentkezteti a felhasználót.

• Validálja az email és jelszó mezőket.

• Ha a felhasználó hitelesítése sikeres, létrehozza a felhasználóhoz tartozó kosarat a Cart modellel, az EMPTY státusszal.

Paraméterek:

• $request: Az HTTP kérés, amely tartalmazza az email és jelszó adatokat.

Visszatérési érték:

redirect()->route('home'): A felhasználót átirányítja a főoldalra, ha a bejelentkezés sikeres.

5. logout()

A kijelentkezési metódus, amely kilépteti a felhasználót és visszairányítja őt a bejelentkezési oldalra.

Visszatérési érték:

redirect()->route('login'): A felhasználót a bejelentkezési oldalra irányítja.

A CartController osztály felelős a webshopban a kosarak kezeléséért. Ez a vezérlő lehetővé teszi a rendelési kosarak létrehozását, módosítását és megjelenítését, valamint a kosárhoz tartozó termékek kezelését. A kosár azonosítja a felhasználó vásárlási folyamatát, és segíti az adminisztrátorokat a rendelés kezelésében.

Fő funkciók:

• index(): Az összes rendelést lekérdezi, és az adminisztrációs felületen listázza.

• show(): Egy rendeléshez tartozó kosár és annak elemeit jeleníti meg.

• addItem(): Hozzáad egy terméket a felhasználó kosarához, ellenőrizve, hogy a felhasználó be van-e jelentkezve, és ha nincs kosár, akkor létrehoz egy újat.

A CategoryController osztály a kategóriák kezelésére szolgál a webshop admin felületén. A kontroller különböző metódusokkal biztosítja a kategóriák létrehozását, módosítását, megjelenítését, törlését, és keresését.

Rövid leírás a metódusokról:

• index():

Lekéri az adatbázisból a fő és alternatív kategóriákat, majd megjeleníti azokat az admin felületen.

• create():

Jelenleg nem tartalmaz funkciót, de elvileg a kategória létrehozásához szükséges űrlap lenne itt kezelve.

• store():

A StoreCategoryRequest validálása után egy új kategóriát hoz létre az adatbázisban. Ha van megadva szülő kategória, azt is hozzárendeli.

• show():

Ez a metódus a kategória megjelenítéséhez lenne használva, de jelenleg nem tartalmaz funkciót.

• edit():

A kategória szerkesztésére szolgáló űrlapot adja vissza, a kategória adataival és a szülő kategóriák listájával.

• update():

A kategória adatainak frissítésére szolgál. A validált adatokat menti az adatbázisba, és ha szükséges, frissíti a szülő kategóriát.

• destroy():

A megadott kategóriát törli az adatbázisból.

• search():

Keresést végez a fő és alternatív kategóriák között, és azokat listázza, amelyek nevében szerepel a keresett kifejezés.

A DiscountController osztály felelős a kedvezmények kezeléséért az admin felületen. A kontroller biztosítja a kedvezmények listázását, létrehozását, frissítését, törlését és keresését.

Főbb metódusok és funkciók:

1. index():

• Lekérdezi az összes kedvezményt és azokhoz tartozó termékeket, majd azokat oldalanként (5-ös laponként) jeleníti meg az admin felületen.

• Ezen kívül lekérdezi a kuponokat is, és azokat is ugyanígy megjeleníti.

2. create():

• Visszaadja a kedvezmény létrehozásához szükséges űrlapot.

3. store():

• A validált adatokat elmenti az adatbázisba, új kedvezményt hoz létre.

• A sikeres mentést követően visszairányítja az admin felületre a kedvezmények listájához egy sikerüzenettel.

4. edit():

• Megjeleníti a kedvezmény szerkesztési űrlapját a kiválasztott kedvezmény adataival.

5. update():

• A kedvezmény adatait frissíti a validált űrlapon keresztül.

• A frissítés után az admin felületre irányítja a felhasználót sikerüzenettel.

6. destroy():

• A megadott kedvezményt törli az adatbázisból, és visszairányítja az admin felületre sikerüzenettel.

7. search():

• A keresett kifejezés alapján keres kedvezményeket és kuponokat.

• A Discount és a Coupon modelleken végzi a keresést, és a megfelelő találatokat jeleníti meg az admin felületen.

Ez a kontroller tehát teljes körű kedvezménykezelést biztosít az adminisztrátorok számára, beleértve a keresést, létrehozást, frissítést és törlést.

A DisplayProductsController osztály a termékek megjelenítéséért felelős vezérlő. Az osztályban az alábbi metódusok találhatók:

index()

Ez a metódus lekéri a kedvezményes termékeket és a fő kategóriákat az adatbázisból.

• $discountedProducts:

A discounted\_products táblából véletlenszerűen 8 kedvezményes terméket kér le. A limit(8) biztosítja, hogy legfeljebb 8 terméket kapjunk vissza.

• $mainCategories:

A categories táblából véletlenszerűen 4 fő kategóriát kér le, ahol a parent\_id értéke null, vagyis a fő kategóriák.

Ezután a metódus visszaadja a home nézetet, átadva a lekért adatokat (discountedProducts és mainCategories) a nézetnek.

Funkció

Ez a vezérlő a webshop főoldalához kapcsolódó adatokat szolgáltat, mint a kedvezményes termékek és a kategóriák megjelenítése.

A SearchController osztály a termékek kereséséért felelős a webshopban. Ez a vezérlő lehetővé teszi a látogatók számára, hogy különféle szűrési feltételek alapján (kulcsszó, árintervallum, kategória) keressenek a termékek között.

index(Request $request)

Ez az egyetlen metódus az osztályban, és a következőket végzi:

Szűrési lehetőségek:

• Kulcsszavas keresés (search):

A title, description, summary és slug mezőkre keres, részleges egyezéssel (LIKE).

• Minimum ár (min\_price):

Csak azokat a termékeket hozza vissza, amelyek ára nem kisebb, mint a megadott minimum.

• Maximum ár (max\_price):

Csak azokat a termékeket hozza vissza, amelyek ára nem nagyobb, mint a megadott maximum.

• Kategóriák (categories):

A category\_id mező alapján szűri azokat a termékeket, amelyek a megadott kategóriák valamelyikébe tartoznak.

Lekérdezések és adatok:

• A találati lista 16 termékenként lapozható.

• A lekérdezés paraméterei (search, ár, kategóriák) megmaradnak a lapozásnál (appends).

• Lekérdezi a fő- és alkategóriákat (main\_categories, alt\_categories) a nézethez.

Nézet:

• A products.index nézetet tölti be, amely a keresési eredményeket és a kategória listákat jeleníti meg.

A UserController osztály az admin felületen kezeli a felhasználókat. Lehetővé teszi a felhasználók listázását, keresését, módosítását, törlését, jelszó-visszaállító e-mail küldését, valamint a törölt felhasználók visszaállítását vagy végleges törlését.

Metódusok leírása

index(Request $request)

• Kilistázza az aktív felhasználókat.

• Ha keresési kifejezés van megadva, akkor név vagy email alapján szűr.

• 10-esével lapozható lista.

update(UpdateUserRequest $request, User $user)

• Frissíti a megadott felhasználó nevét, e-mail címét és jelszavát (ha lett küldve új).

• A jelszó titkosítva kerül eltárolásra.

• Átirányít a felhasználói listára egy sikerüzenettel.

destroy(User $user)

• “Puha törléssel” törli a felhasználót (nem véglegesen).

• Átirányít a listára sikerüzenettel.

editEmail(User $user)

• Visszaadja a nézetet a felhasználó e-mail címének szerkesztéséhez.

newPassword(User $user)

• Elküldi a jelszó-visszaállító emailt a felhasználónak a sendPasswordReset() metóduson keresztül.

• Visszairányít a felhasználói listára.

forceDelete($user)

• Véglegesen töröl egy már törölt (trashed) felhasználót az adatbázisból.

• Visszairányít a törölt felhasználók listájára sikerüzenettel.

restore($user)

• Visszaállít egy korábban törölt felhasználót.

• Visszairányít a törölt felhasználók listájára sikerüzenettel.

trashed(Request $request)

• Kilistázza a „puha törléssel” eltávolított (archivált) felhasználókat.

• Támogatja a név vagy e-mail szerinti keresést.

• 10-esével lapozható lista.

A ProductController a webshop adminisztrációs felületén kezeli a termékeket. Lehetővé teszi a termékek létrehozását, szerkesztését, megjelenítését, törlését, keresését, valamint JSON-kimenet előállítását API-k számára.

Működő metódusok

index()

• Lekérdezi az összes terméket kategória- és kedvezménykapcsolatokkal együtt, és 10-esével lapozza.

• A termékeket megjeleníti az admin.products.products.index nézetben.

create()

• Megjeleníti az új termék létrehozására szolgáló űrlapot, a kategóriák listájával együtt.

store(StoreProductRequest $request)

• Validálja és elmenti az új terméket az adatbázisba.

• Feltölti a képet, ha meg lett adva.

• A sikeres létrehozás után visszairányít a terméklista oldalra.

show($id)

• Lekéri a megadott ID alapján a terméket, és megjeleníti a products.show nézetben.

edit(Product $product)

• Megjeleníti a kiválasztott termék szerkesztő űrlapját, a kategóriák listájával együtt.

update(UpdateProductRequest $request, Product $product)

• Frissíti a megadott termék adatait a validált értékekkel.

• Frissíti a képet, ha új fájl lett feltöltve.

• A frissítés után visszairányít a terméklistához.

destroy(Product $product)

• Törli a termékhez tartozó kedvezményt, majd magát a terméket is.

• A törlés után visszairányít a terméklistára.

search()

• Kulcsszó alapján keres a termékek címében vagy a hozzájuk tartozó kategóriák nevében.

• Az eredményeket a szokásos admin listanézetben jeleníti meg, 10-es lapozással.

display($slug)

• A megadott slug alapján megkeresi a terméket, és megjeleníti a products.show nézetben.

getAllProducts()

• Lekéri az összes terméket az adatbázisból és JSON formátumban visszaadja (pl. API-használathoz).

# Enum

Roles enum

A Roles enum a felhasználói szerepköröket határozza meg.

Ez az alkalmazásban a jogosultságkezelés alapját képezi.

Lehetséges értékek:

• ADMIN: rendszergazda jogosultságokkal rendelkező felhasználó

• USER: normál felhasználó

OrderStatus enum

Az OrderStatus enum a megrendelések állapotát reprezentálja.

Minden rendelés egy adott állapotban van, amely meghatározza a feldolgozás vagy teljesítés fázisát.

Lehetséges értékek:

• PENDING: várakozó állapot, a rendelés még nem lett feldolgozva

• PROCESSING: a rendelés feldolgozás alatt van

• SHIPPED: a rendelés kiszállítás alatt van

• DELIVERED: a rendelés kézbesítve lett

• CANCELLED: a rendelést törölték

Metódus:

• isFinal(): bool

Igaz értéket ad vissza, ha az állapot végleges (DELIVERED vagy CANCELLED).

CartStatus enum

A CartStatus enum a vásárlói kosarak aktuális állapotát írja le.

Ez segít eldönteni, hogy egy kosár még szerkeszthető-e, vagy már lezárt.

Lehetséges értékek:

• EMPTY: a kosár üres

• LOADED: a kosárba lettek téve termékek

• CHECKED\_OUT: a kosarat véglegesítették, fizetésre került

• CLOSE: a kosár lezárt állapotba került (például teljesítés után)

Metódus:

• isFinal(): bool

Igaz értéket ad vissza, ha az állapot CHECKED\_OUT.

# Web.php

Publikus felület (USER)

• Főoldal:

GET / → DisplayProductsController@index – megjeleníti a főoldalt kedvezményes termékekkel.

• Termék megtekintése:

GET /product/{slug} → ProductController@display – megjelenít egy adott terméket slug alapján.

• Keresés termékek között:

GET /products → SearchController@index – termékek keresése szűrőkkel.

Hitelesítés

• Bejelentkezés:

GET /login → AuthController@showLogin

POST /login → AuthController@login

• Regisztráció:

GET /register → AuthController@showRegister

POST /register → AuthController@register

• Kijelentkezés:

POST /logout → AuthController@logout

Kosárkezelés

• Termék hozzáadása kosárhoz:

POST /addItem → CartController@addItem

• Felhasználói kosár oldal:

GET /cart – csak bejelentkezett felhasználó láthatja.

• Checkout oldal:

GET /checkout – megrendelési folyamat kezdete.

Felhasználói nézetek

• GET /user → felhasználói adatok megjelenítése

• GET /userOrder → rendelési előzmények

• GET /user/userAddress → szállítási cím megjelenítése

• GET /editData → felhasználói adatok szerkesztése

• GET /editAddress → cím szerkesztése

Admin felület (ADMIN)

Kategóriák kezelése

• Route::resource('/admin/products/categories', CategoryController::class)

• POST /admin/products/categories/search → keresés kategóriák között

Termékek kezelése

• GET /admin/products/products/search → keresés termékek között

• Route::resource('/admin/products/products', ProductController::class)

Kedvezmények és kuponok

• GET /admin/products/discounts/search

• Route::resource('admin/products/discounts', DiscountController::class)

• Route::resource('/admin/products/coupons', CouponController::class)

Felhasználók kezelése

• GET /admin/users/users → felhasználók listázása

• GET /admin/users/users/search → keresés

• GET /admin/users/users/{user}/edit → e-mail szerkesztés

• PUT /admin/users/users/{user} → adatok frissítése

• DELETE /admin/users/users/{user} → törlés

• POST /admin/users/users/newPassword/{user} → új jelszó küldése

• PUT /admin/users/users/restore/{user} → visszaállítás törlésből

• DELETE /admin/users/users/forcedelete/{user} → végleges törlés

• GET /admin/users/users/trashed → törölt felhasználók listázása

Rendelések és kosarak kezelése

• Route::resource('/admin/orders/carts', CartController::class)

• POST /admin/orders/orders/store → új rendelés létrehozása