Hardver Webshop

Vizsgaremek Dokumentáció

2023-2024

# Fejlesztők

Bodnár Bence

Frontend

és

Paulusz Zsombor

Backend

https://github.com/egonixaimgod/Vizsgaremek\_2024\_bbence\_pzsombor

Tartalomjegyzék

[0](#_Toc164388824)

[Fejlesztők 2](#_Toc164388825)

[Projekt bemutatása 4](#_Toc164388826)

[Miért van ár szükség 4](#_Toc164388827)

[Miért használnák felhasználók 5](#_Toc164388828)

[Felhasznált technológiák 7](#_Toc164388829)

[Keretrendszer 7](#_Toc164388830)

[Programnyelv 7](#_Toc164388831)

[Ingyenes/fizetős technológiák 8](#_Toc164388832)

[Tervek 8](#_Toc164388833)

[Adatbázis terv 9](#_Toc164388834)

[UML terv 10](#_Toc164388835)

[Folyamatábra 11](#_Toc164388836)

[Látványterv 12](#_Toc164388837)

[Program leírása 13](#_Toc164388838)

[Backend 13](#_Toc164388839)

[Osztályok 14](#_Toc164388840)

[Frontend 41](#_Toc164388841)

[Komponensek 41](#_Toc164388842)

[Service.ts fájlok 42](#_Toc164388843)

[Képek és Image Carousel 46](#_Toc164388844)

[Tárgymutató 46](#_Toc164388845)

[Általunk használt kifejezések 47](#_Toc164388846)

[Felhasznált irodalom 48](#_Toc164388847)

[Köszönet 49](#_Toc164388848)

# Projekt bemutatása

A projektünk egy számítógépes Webshop-ról szól, ahol lehet rendelni, különböző számítógép hardvereket. Ilyen például egy számítógép processzor, alaplap, memória, processzor, videókártya, esetleg egy számítógép tápegység, vagy ház. Jelenleg a webshopunkhoz 6 terméket adtunk hozzá, 2 alaplapot, 2 memória modult, illetve 2 processzort bemutatás céljából, de az adatbázisunkat fel lehet tölteni bármennyi termékkel. Ezen kívül van saját regisztrációs, illetve bejelentkező felületünk, ahol a felhasználók regisztrálni tudnak, és miután hitelesítettük adataikat, be is tudnak jelentkezni, így a megrendelt termékeik össze vannak kapcsolva a bejelentkezett e-mail címükkel, illetve felhasználónevükkel, így látjuk hogy egy adott felhasználó mit rendelt.

## Miért van ár szükség

## Miért használnák felhasználók

# Felhasznált technológiák

## Keretrendszer

Frontend:

* Angular 16.2.12

Backend:

* Laravel 10.46.0

## Programnyelv

Frontend:

* HTML
* CSS
* Typescript

Backend:

* PHP

## Ingyenes/fizetős technológiák

Frontend:

Backend:

* Insomnia, Postman: API Végpontok tesztelése

# Tervek

## Adatbázis terv

## UML terv

## Folyamatábra

## Látványterv

# Program leírása

## Backend

### Osztályok

#### Migrációs Fájlok

##### User Tábla Migráció

Ez a migrációs fájl felelős a **users** tábla létrehozásáért az adatbázisban. A **users** tábla a felhasználók adatait tárolja, és fontos szerepet tölt be a webshop felhasználói rendszerében.

###### Migráció Fájl

A migrációs fájl neve: **YYYY\_MM\_DD\_HHMMSS\_create\_users\_table.php**

###### Adatmezők

* **id**: Egyedi azonosító, automatikusan növekszik.
* **name**: A felhasználó neve.
* **email**: A felhasználó e-mail címe, egyedi az adatbázisban.
* **email\_verified\_at**: A felhasználó e-mail címének megerősítésének ideje, opcionális.
* **password**: A felhasználó jelszava.
* **address**: A felhasználó címe.
* **city**: A felhasználó városa.
* **postal\_code**: A felhasználó irányítószáma.
* **phone**: A felhasználó telefonszáma.
* **admin**: Az adminisztrátori jogosultságokat jelzi (true/false).
* **remember\_token**: Egy token a felhasználó emlékezetének megőrzésére.
* **created\_at**: A rekord létrehozásának időpontja.
* **updated\_at**: A rekord frissítésének időpontja.

###### Létrehozás és Visszavonás

* **Létrehozás**: A **Schema::create('users', function (Blueprint $table)** függvény hozza létre a **users** táblát és definiálja a fenti adatmezőket.
* **Visszavonás**: A **Schema::dropIfExists('users')** függvény törli a **users** táblát az adatbázisból.
* **Létrehozás**: A **Schema::create('users', function (Blueprint $table)** függvény hozza létre a **users** táblát és definiálja a fenti adatmezőket.
* **Visszavonás**: A **Schema::dropIfExists('users')** függvény törli a **users** táblát az adatbázisból.

Köszönöm, hogy megosztottad a második migrációs fájlt is! Íme a hozzá tartozó dokumentáció:

##### Brands Tábla Migráció

Ez a migrációs fájl felelős a **brands** tábla létrehozásáért az adatbázisban. A **brands** tábla a termékek márkáit tárolja, és fontos szerepet tölt be a webshop termékkezelő rendszerében.

###### Migráció Fájl

A migrációs fájl neve: **YYYY\_MM\_DD\_HHMMSS\_create\_brands\_table.php**

Adatmezők

* **id**: Egyedi azonosító, automatikusan növekszik.
* **name**: A márka neve.
* **created\_at**: A rekord létrehozásának időpontja.
* **updated\_at**: A rekord frissítésének időpontja.

Létrehozás és Visszavonás

* **Létrehozás**: A **Schema::create('brands', function (Blueprint $table)** függvény hozza létre a **brands** táblát és definiálja a fenti adatmezőket.
* **Visszavonás**: A **Schema::dropIfExists('brands')** függvény törli a **brands** táblát az adatbázisból.

##### Categories Tábla Migráció

Ez a migrációs fájl felelős a **categories** tábla létrehozásáért az adatbázisban. A **categories** tábla a termékek kategóriáit tárolja, és segít rendezni és csoportosítani a webshopban kínált termékeket.

###### Migráció Fájl

A migrációs fájl neve: **YYYY\_MM\_DD\_HHMMSS\_create\_categories\_table.php**

###### Adatmezők

* **id**: Egyedi azonosító, automatikusan növekszik.
* **name**: A kategória neve.
* **created\_at**: A rekord létrehozásának időpontja.
* **updated\_at**: A rekord frissítésének időpontja.

###### Létrehozás és Visszavonás

* **Létrehozás**: A **Schema::create('categories', function (Blueprint $table)** függvény hozza létre a **categories** táblát és definiálja a fenti adatmezőket.
* **Visszavonás**: A **Schema::dropIfExists('categories')** függvény törli a **categories** táblát az adatbázisból.

##### Payments Tábla Migráció

Ez a migrációs fájl felelős a **payments** tábla létrehozásáért az adatbázisban. A **payments** tábla a vásárlások fizetési adatait tárolja, és segíti nyomon követni a fizetési tranzakciók állapotát a webshopban.

###### Migráció Fájl

A migrációs fájl neve: **YYYY\_MM\_DD\_HHMMSS\_create\_payments\_table.php**

###### Adatmezők

* **id**: Egyedi azonosító, automatikusan növekszik.
* **status**: A fizetési állapot, lehet "Billed" (számlázott) vagy "Paid" (kifizetett).
* **created\_at**: A rekord létrehozásának időpontja.
* **updated\_at**: A rekord frissítésének időpontja.

###### Létrehozás és Visszavonás

* **Létrehozás**: A **Schema::create('payments', function (Blueprint $table)** függvény hozza létre a **payments** táblát és definiálja a fenti adatmezőket.
* **Visszavonás**: A **Schema::dropIfExists('payments')** függvény törli a **payments** táblát az adatbázisból.

##### Orders Tábla Migráció

Ez a migrációs fájl felelős az **orders** tábla létrehozásáért az adatbázisban. Az **orders** tábla rögzíti a felhasználók rendeléseit a webshopban.

###### Migráció Fájl

A migrációs fájl neve: **YYYY\_MM\_DD\_HHMMSS\_create\_orders\_table.php**

###### Adatmezők

* **id**: Egyedi azonosító, automatikusan növekszik.
* **azonosito**: A rendelés egyedi azonosítója.
* **user\_id**: A rendelést leadó felhasználó azonosítója, idegen kulcs kapcsolat a **users** táblához.
* **payment\_id**: A fizetés azonosítója, idegen kulcs kapcsolat a **payments** táblához.
* **order\_date**: A rendelés dátuma.
* **created\_at**: A rekord létrehozásának időpontja.
* **updated\_at**: A rekord frissítésének időpontja.

###### Idegen Kulcsok

* **user\_id**: Az **orders** tábla **user\_id** mezője idegen kulcs kapcsolatot tart fenn a **users** táblához. Ha egy felhasználó törlődik, a rendeléshez kapcsolódó rekordok is törlődnek (CASCADE).
* **payment\_id**: Az **orders** tábla **payment\_id** mezője idegen kulcs kapcsolatot tart fenn a **payments** táblához. Ha egy fizetés törlődik, a rendeléshez kapcsolódó rekordok is törlődnek (CASCADE).

###### Létrehozás és Visszavonás

* **Létrehozás**: A **Schema::create('orders', function (Blueprint $table)** függvény hozza létre az **orders** táblát és definiálja a fenti adatmezőket és idegen kulcsokat.
* **Visszavonás**: A **Schema::dropIfExists('orders')** függvény törli az **orders** táblát az adatbázisból.

##### Order\_Items Tábla Migráció

Ez a migrációs fájl felelős az **order\_items** tábla létrehozásáért az adatbázisban. Az **order\_items** tábla rögzíti az egyes rendelésekhez tartozó termékek részleteit, mint például a rendelt termék, annak mennyisége és az adott rendeléshez való kapcsolat.

###### Migráció Fájl

A migrációs fájl neve: **YYYY\_MM\_DD\_HHMMSS\_create\_order\_items\_table.php**

###### Adatmezők

* **id**: Egyedi azonosító, automatikusan növekszik.
* **order\_id**: A rendelés azonosítója, idegen kulcs kapcsolat a **orders** táblához.
* **product\_id**: A termék azonosítója, idegen kulcs kapcsolat a **products** táblához.
* **amount**: A rendelt termék mennyisége.
* **created\_at**: A rekord létrehozásának időpontja.
* **updated\_at**: A rekord frissítésének időpontja.

###### Idegen Kulcsok

* **order\_id**: Az **order\_items** tábla **order\_id** mezője idegen kulcs kapcsolatot tart fenn a **orders** táblához. Ha egy rendelés törlődik, az ehhez kapcsolódó rekordok is törlődnek (CASCADE).
* **product\_id**: Az **order\_items** tábla **product\_id** mezője idegen kulcs kapcsolatot tart fenn a **products** táblához. Ha egy termék törlődik, az ehhez kapcsolódó rekordok is törlődnek (CASCADE).

###### Létrehozás és Visszavonás

* **Létrehozás**: A **Schema::create('order\_items', function (Blueprint $table)** függvény hozza létre az **order\_items** táblát és definiálja a fenti adatmezőket és idegen kulcsokat.
* **Visszavonás**: A **Schema::dropIfExists('order\_items')** függvény törli az **order\_items** táblát az adatbázisból.

##### Products Tábla Migráció

Ez a migrációs fájl felelős a **products** tábla létrehozásáért az adatbázisban. A **products** tábla rögzíti a webshop árucikkeinek adatait, mint például a termék neve, ára, készlete és egyéb tulajdonságok.

###### Migráció Fájl

A migrációs fájl neve: **YYYY\_MM\_DD\_HHMMSS\_create\_products\_table.php**

###### Adatmezők

* **id**: Egyedi azonosító, automatikusan növekszik.
* **category\_id**: A termék kategória azonosítója, idegen kulcs kapcsolat a **categories** táblához.
* **brand\_id**: A termék márka azonosítója, idegen kulcs kapcsolat a **brands** táblához.
* **name**: A termék neve.
* **cost**: A termék ára.
* **amount**: A termék készlete.
* **img**: A termék képének elérési útvonala.
* **description**: A termék leírása (opcionális).
* **created\_at**: A rekord létrehozásának időpontja.
* **updated\_at**: A rekord frissítésének időpontja.

###### Idegen Kulcsok

* **category\_id**: A **products** tábla **category\_id** mezője idegen kulcs kapcsolatot tart fenn a **categories** táblához. Ha egy kategória törlődik, az ehhez kapcsolódó rekordok is törlődnek (CASCADE).
* **brand\_id**: A **products** tábla **brand\_id** mezője idegen kulcs kapcsolatot tart fenn a **brands** táblához. Ha egy márka törlődik, az ehhez kapcsolódó rekordok is törlődnek (CASCADE).

###### Létrehozás és Visszavonás

* **Létrehozás**: A **Schema::create('products', function (Blueprint $table)** függvény hozza létre a **products** táblát és definiálja a fenti adatmezőket és idegen kulcsokat.
* **Visszavonás**: A **Schema::dropIfExists('products')** függvény törli a **products** táblát az adatbázisból.

#### Modelek

##### User Model

A **User** modell felelős a felhasználói adatok kezeléséért és az azokhoz tartozó műveletek végrehajtásáért.

###### Relációk

* **orders()**: A felhasználóhoz tartozó rendelések lekérését teszi lehetővé.

###### Töltött Mezők

* **name**: A felhasználó neve.
* **email**: A felhasználó e-mail címe.
* **password**: A felhasználó jelszava.
* **address**: A felhasználó címe.
* **county**: A felhasználó megyéje (opcionális).
* **city**: A felhasználó városa.
* **postal\_code**: A felhasználó irányítószáma.
* **phone**: A felhasználó telefonszáma.
* **admin**: A felhasználó adminisztrátori jogosultságát jelzi (true/false).

###### Rejtett Mezők

* **password**: A jelszó mező rejtett a szerializáció során.
* **remember\_token**: A "remember me" token rejtett a szerializáció során.

###### Típusátalakítások

* **email\_verified\_at**: Az e-mail megerősítés dátumára **datetime** típusú átalakítás.
* **password**: A jelszóra **hashed** típusú átalakítás.

###### Töltött és Rejtett Mezők Beállítása

A **$fillable** és **$hidden** tulajdonságok definiálják, hogy mely mezők töltődhetnek és melyek rejtettek a szerializáció során.

###### Laravel Sanctum API Tokenek

A **HasApiTokens** trait segítségével lehetőség van API tokenek kezelésére és generálására a felhasználóhoz.

###### Laravel Eloquent Has Factory

A **HasFactory** trait használatával lehetőség van gyártási modellek definiálására.

###### Laravel Eloquent Notifiable

A **Notifiable** trait lehetővé teszi az értesítések küldését a felhasználóhoz.

##### Brands Model

A **Brands** modell felelős a márkák kezeléséért és az azokhoz tartozó műveletek végrehajtásáért.

###### Töltött Mezők

* **name**: A márka neve.

###### Típusátalakítások

Nincsenek specifikus típusátalakítások definiálva.

###### Töltött Mezők Beállítása

A **$fillable** tulajdonság definiálja, hogy mely mezők töltődhetnek fel a modellekben.

###### Laravel Eloquent Has Factory

A **HasFactory** trait segítségével lehetőség van gyártási modellek definiálására.

##### Categories Modell

A **Categories** modell felelős a kategóriák kezeléséért és az azokhoz tartozó műveletek végrehajtásáért.

###### Töltött Mezők

* **name**: A kategória neve.

###### Típusátalakítások

Nincsenek specifikus típusátalakítások definiálva.

###### Töltött Mezők Beállítása

A **$fillable** tulajdonság definiálja, hogy mely mezők töltődhetnek fel a modellekben.

###### Laravel Eloquent Has Factory

A **HasFactory** trait segítségével lehetőség van gyártási modellek definiálására.

Köszönöm a **Payment** modell kódjának megosztását! Íme a hozzá tartozó dokumentáció:

##### Payment Modell

A **Payment** modell felelős a fizetések kezeléséért és az azokhoz tartozó műveletek végrehajtásáért.

###### Töltött Mezők

* **status**: A fizetés állapota, lehet "Billed" (számlázott) vagy "Paid" (kifizetett).

###### Típusátalakítások

Nincsenek specifikus típusátalakítások definiálva.

###### Töltött Mezők Beállítása

A **$fillable** tulajdonság definiálja, hogy mely mezők töltődhetnek fel a modellekben.

###### Laravel Eloquent Has Factory

A **HasFactory** trait segítségével lehetőség van gyártási modellek definiálására.

##### Orders Modell

Az **Orders** modell felelős a rendelések kezeléséért és az azokhoz tartozó műveletek végrehajtásáért.

###### Relációk

* **orderItems()**: A rendeléshez tartozó tételek lekérését teszi lehetővé.

###### Töltött Mezők

* **azonosito**: A rendelés egyedi azonosítója.
* **user\_id**: A rendelést leadó felhasználó azonosítója.
* **payment\_id**: A fizetés azonosítója.
* **order\_date**: A rendelés dátuma.

###### Típusátalakítások

Nincsenek specifikus típusátalakítások definiálva.

###### Töltött Mezők Beállítása

A **$fillable** tulajdonság definiálja, hogy mely mezők töltődhetnek fel a modellekben.

###### Laravel Eloquent Has Factory

A **HasFactory** trait segítségével lehetőség van gyártási modellek definiálására.

###### Laravel Eloquent HasMany Reláció

A **hasMany** metódus segítségével lehetőség van a **Orders** modellek kapcsolódására más modellekhez, mint például az **OrderItems** modellekhez.

Köszönöm a **Categories** modell kódjának megosztását! Íme a hozzá tartozó dokumentáció:

##### OrderItems Modell

Az **OrderItems** modell felelős a rendelés tételeinek kezeléséért és az azokhoz tartozó műveletek végrehajtásáért.

###### Relációk

* **product()**: A rendelés tételhez tartozó termék lekérését teszi lehetővé.

###### Töltött Mezők

* **order\_id**: A rendelés azonosítója.
* **product\_id**: A termék azonosítója.
* **amount**: A termék mennyisége a rendelésben.

###### Típusátalakítások

Nincsenek specifikus típusátalakítások definiálva.

###### Töltött Mezők Beállítása

A **$fillable** tulajdonság definiálja, hogy mely mezők töltődhetnek fel a modellekben.

###### Laravel Eloquent Has Factory

A **HasFactory** trait segítségével lehetőség van gyártási modellek definiálására.

###### Laravel Eloquent BelongsTo Reláció

A **belongsTo** metódus segítségével lehetőség van az **OrderItems** modellek kapcsolódására más modellekhez, mint például a **Products** modellekhez.

##### Products Modell

A **Products** modell felelős a termékek kezeléséért és az azokhoz tartozó műveletek végrehajtásáért.

###### Relációk

* **category()**: A termékhez tartozó kategória lekérését teszi lehetővé.
* **brand()**: A termékhez tartozó márka lekérését teszi lehetővé.

###### Töltött Mezők

* **category\_id**: A termékhez tartozó kategória azonosítója.
* **brand\_id**: A termékhez tartozó márka azonosítója.
* **name**: A termék neve.
* **cost**: A termék ára.
* **amount**: A termék készlete.
* **img**: A termék képének elérési útvonala.
* **description**: A termék leírása.

###### Típusátalakítások

Nincsenek specifikus típusátalakítások definiálva.

###### Töltött Mezők Beállítása

A **$fillable** tulajdonság definiálja, hogy mely mezők töltődhetnek fel a modellekben.

###### Laravel Eloquent Has Factory

A **HasFactory** trait segítségével lehetőség van gyártási modellek definiálására.

###### Laravel Eloquent BelongsTo Relációk

A **belongsTo** metódusok segítségével lehetőség van a **Products** modellek kapcsolódására más modellekhez, mint például a kategóriákhoz és a márkákhoz.

#### Controllerek

##### UsersController

A **UsersController** osztály felelős az ügyfélkapcsolati funkciók kezeléséért és az azokhoz tartozó műveletek végrehajtásáért.

###### Metódusok

index()

* **Leírás**: Az összes felhasználó lekérdezése.
* **Visszatérési érték**: JSON formátumban visszaadja az összes felhasználót.

show($id)

* **Leírás**: Egy adott felhasználó lekérdezése az azonosító alapján.
* **Paraméterek**: **$id** - a felhasználó azonosítója.
* **Visszatérési érték**: JSON formátumban visszaadja a keresett felhasználót, hiba esetén 404-es státuszkóddal.

update(Request $request, $id)

* **Leírás**: Egy adott felhasználó adatainak frissítése.
* **Paraméterek**: **$request** - a HTTP kérés objektuma, **$id** - a felhasználó azonosítója.
* **Bemeneti adatok ellenőrzése**: Az új adatokat a megadott validációs szabályoknak megfelelően ellenőrzi.
* **Visszatérési érték**: JSON formátumban visszaadja a frissített felhasználó adatait, HTTP 200-as státuszkóddal.

destroy($id)

* **Leírás**: Egy adott felhasználó törlése.
* **Paraméterek**: **$id** - a felhasználó azonosítója.
* **Művelet**: Megkeresi és törli a megadott felhasználót az azonosító alapján.
* **Visszatérési érték**: NULL-t ad vissza, HTTP 204-es státuszkóddal.

###### Validáció

A **update()** metódusban az új adatokat a Laravel beépített validációs rendszerével ellenőrzi a megadott szabályoknak megfelelően.

##### BrandsController

A **BrandsController** osztály felelős a márkák kezeléséért és az azokhoz tartozó műveletek végrehajtásáért.

###### Metódusok

index()

* **Leírás**: Az összes márka lekérdezése.
* **Visszatérési érték**: JSON formátumban visszaadja az összes márkát.

create()

* **Leírás**: Új márka létrehozásához tartozó űrlap megjelenítése.
* **Nincs megvalósítva**: Az űrlap megjelenítése nincs implementálva ebben a kontrollerben.

store(Request $request)

* **Leírás**: Új márka létrehozása.
* **Paraméterek**: **$request** - a HTTP kérés objektuma.
* **Visszatérési érték**: JSON formátumban visszaadja az új márkát, HTTP 201-es státuszkóddal.

show($id)

* **Leírás**: Egy adott márka lekérdezése az azonosító alapján.
* **Paraméterek**: **$id** - a márka azonosítója.
* **Visszatérési érték**: JSON formátumban visszaadja a keresett márkát, hiba esetén 404-es státuszkóddal.

edit(Brands $brands)

* **Leírás**: Egy márka szerkesztési űrlapjának megjelenítése.
* **Nincs megvalósítva**: Az űrlap megjelenítése nincs implementálva ebben a kontrollerben.

update(Request $request, $id)

* **Leírás**: Egy adott márka adatainak frissítése.
* **Paraméterek**: **$request** - a HTTP kérés objektuma, **$id** - a márka azonosítója.
* **Bemeneti adatok ellenőrzése**: Az új adatokat a megadott validációs szabályoknak megfelelően ellenőrzi.
* **Visszatérési érték**: JSON formátumban visszaadja a frissített márkát, HTTP 200-as státuszkóddal.

destroy($id)

* **Leírás**: Egy adott márka törlése.
* **Paraméterek**: **$id** - a márka azonosítója.
* **Művelet**: Megkeresi és törli a megadott márkát az azonosító alapján.
* **Visszatérési érték**: NULL-t ad vissza, HTTP 204-es státuszkóddal.

###### Validáció

A **update()** metódusban az új adatokat a Laravel beépített validációs rendszerével ellenőrzi a megadott szabályoknak megfelelően.

##### CategoriesController

A **CategoriesController** osztály felelős a kategóriák kezeléséért és az azokhoz tartozó műveletek végrehajtásáért.

###### Metódusok

index()

* **Leírás**: Az összes kategória lekérdezése.
* **Visszatérési érték**: JSON formátumban visszaadja az összes kategóriát.

store(Request $request)

* **Leírás**: Új kategória létrehozása.
* **Paraméterek**: **$request** - a HTTP kérés objektuma.
* **Visszatérési érték**: JSON formátumban visszaadja az új kategóriát, HTTP 201-es státuszkóddal.

show($id)

* **Leírás**: Egy adott kategória lekérdezése az azonosító alapján.
* **Paraméterek**: **$id** - a kategória azonosítója.
* **Visszatérési érték**: JSON formátumban visszaadja a keresett kategóriát, hiba esetén 404-es státuszkóddal.

update(Request $request, $id)

* **Leírás**: Egy adott kategória adatainak frissítése.
* **Paraméterek**: **$request** - a HTTP kérés objektuma, **$id** - a kategória azonosítója.
* **Bemeneti adatok ellenőrzése**: Az új adatokat a megadott validációs szabályoknak megfelelően ellenőrzi.
* **Visszatérési érték**: JSON formátumban visszaadja a frissített kategóriát, HTTP 200-as státuszkóddal.

destroy($id)

* **Leírás**: Egy adott kategória törlése.
* **Paraméterek**: **$id** - a kategória azonosítója.
* **Művelet**: Megkeresi és törli a megadott kategóriát az azonosító alapján.
* **Visszatérési érték**: NULL-t ad vissza, HTTP 204-es státuszkóddal.

###### Validáció

A **update()** metódusban az új adatokat a Laravel beépített validációs rendszerével ellenőrzi a megadott szabályoknak megfelelően.

##### PaymentController

A **PaymentController** osztály felelős a fizetési módok kezeléséért és az azokhoz tartozó műveletek végrehajtásáért.

###### Metódusok

index()

* **Leírás**: Az összes fizetési mód lekérdezése.
* **Visszatérési érték**: JSON formátumban visszaadja az összes fizetési módot.

show($id)

* **Leírás**: Egy adott fizetési mód lekérdezése az azonosító alapján.
* **Paraméterek**: **$id** - a fizetési mód azonosítója.
* **Visszatérési érték**: JSON formátumban visszaadja a keresett fizetési módot, hiba esetén 404-es státuszkóddal.

###### Validáció

A **PaymentController** osztályban nincs validáció, mivel csak lekérdezéseket hajt végre.

##### OrdersController

A **OrdersController** osztály felelős a rendelések kezeléséért és az azokhoz tartozó műveletek végrehajtásáért.

###### Metódusok

index()

* **Leírás**: Az összes rendelés lekérdezése.
* **Visszatérési érték**: JSON formátumban visszaadja az összes rendelést.

placeOrder(Request $request)

* **Leírás**: Új rendelés felvétele.
* **Paraméterek**: **$request** - a HTTP kérés objektuma.
* **Bemeneti adatok ellenőrzése**: Ellenőrzi a beérkező adatokat, hogy megfelelnek-e a validációs szabályoknak.
* **Lépések**:
  1. Validálja a beérkező adatokat.
  2. Létrehozza az új rendelést.
  3. Létrehozza a rendeléshez tartozó tétel(ek)et.
* **Visszatérési érték**: JSON formátumban visszaadja az új rendelést, HTTP 201-es státuszkóddal.

showOrders($perPage = 10)

* **Leírás**: A bejelentkezett felhasználó rendeléseinek lekérdezése.
* **Paraméterek**: **$perPage** - oldalankénti elemek száma (opcionális, alapértelmezett érték: 10).
* **Visszatérési érték**: JSON formátumban visszaadja a bejelentkezett felhasználó rendeléseit.

updateOrder(Request $request, $order\_id)

* **Leírás**: Egy rendelés frissítése.
* **Paraméterek**: **$request** - a HTTP kérés objektuma, **$order\_id** - a frissítendő rendelés azonosítója.
* **Bemeneti adatok ellenőrzése**: Ellenőrzi a beérkező adatokat, hogy megfelelnek-e a validációs szabályoknak.
* **Lépések**:
  1. Validálja a beérkező adatokat.
  2. Frissíti a rendelés adatait, ha szükséges.
  3. Frissíti a rendeléshez tartozó tételek(ek)et, ha szükséges.
* **Visszatérési érték**: JSON formátumban visszaadja a frissített rendelést, HTTP 201-es státuszkóddal.

deleteOrder($id)

* **Leírás**: Egy rendelés törlése.
* **Paraméterek**: **$id** - a törlendő rendelés azonosítója.
* **Művelet**: Megkeresi és törli a megadott rendelést az azonosító alapján.
* **Visszatérési érték**: NULL-t ad vissza, HTTP 204-es státuszkóddal.

###### Validáció

* A **placeOrder()** és **updateOrder()** metódusokban az új adatokat a Laravel beépített validációs rendszerével ellenőrzi a megadott szabályoknak megfelelően.

##### OrderItemsController

Az **OrderItemsController** osztály felelős a rendelés tétel(ek) kezeléséért és az azokhoz tartozó műveletek végrehajtásáért.

###### Metódusok

index()

* **Leírás**: Az összes rendelés tétel(ek) lekérdezése.
* **Visszatérési érték**: JSON formátumban visszaadja az összes rendelés tétel(ek)et.

store(Request $request)

* **Leírás**: Új rendelés tétel(ek) felvétele.
* **Paraméterek**: **$request** - a HTTP kérés objektuma.
* **Bemeneti adatok ellenőrzése**: Ellenőrzi a beérkező adatokat, hogy megfelelnek-e a validációs szabályoknak.
* **Lépések**:
  1. Validálja a beérkező adatokat.
  2. Létrehozza az új rendelés tétel(ek)et.
* **Visszatérési érték**: JSON formátumban visszaadja az új rendelés tétel(ek)et, HTTP 201-es státuszkóddal.

show($id)

* **Leírás**: Egy adott rendelés tétel(ek) lekérdezése az azonosító alapján.
* **Paraméterek**: **$id** - a rendelés tétel(ek) azonosítója.
* **Visszatérési érték**: JSON formátumban visszaadja a keresett rendelés tétel(ek)et, hiba esetén 404-es státuszkóddal.

showOrderItems($order\_id)

* **Leírás**: Egy adott rendeléshez tartozó tétel(ek) lekérdezése az order\_id alapján.
* **Paraméterek**: **$order\_id** - a rendelés azonosítója.
* **Visszatérési érték**: JSON formátumban visszaadja a keresett rendeléshez tartozó tétel(ek)et, hiba esetén 404-es státuszkóddal.

update(Request $request, $id)

* **Leírás**: Egy rendelés tétel(ek) frissítése.
* **Paraméterek**: **$request** - a HTTP kérés objektuma, **$id** - a frissítendő rendelés tétel(ek) azonosítója.
* **Bemeneti adatok ellenőrzése**: Ellenőrzi a beérkező adatokat, hogy megfelelnek-e a validációs szabályoknak.
* **Lépések**:
  1. Validálja a beérkező adatokat.
  2. Frissíti a rendelés tétel(ek)et.
* **Visszatérési érték**: JSON formátumban visszaadja a frissített rendelés tétel(ek)et, HTTP 200-as státuszkóddal.

destroy($id)

* **Leírás**: Egy rendelés tétel(ek) törlése.
* **Paraméterek**: **$id** - a törlendő rendelés tétel(ek) azonosítója.
* **Művelet**: Megkeresi és törli a megadott rendelés tétel(ek)et az azonosító alapján.
* **Visszatérési érték**: NULL-t ad vissza, HTTP 204-es státuszkóddal.

###### Validáció

* A **store()** és **update()** metódusokban az új adatokat a Laravel beépített validációs rendszerével ellenőrzi a megadott szabályoknak megfelelően.

##### ProductsController

Az **ProductsController** osztály felelős a termékek kezeléséért és az azokhoz tartozó műveletek végrehajtásáért.

###### Metódusok

index()

* **Leírás**: Az összes termék lekérdezése.
* **Visszatérési érték**: JSON formátumban visszaadja az összes terméket.

store(Request $request)

* **Leírás**: Új termék felvétele.
* **Paraméterek**: **$request** - a HTTP kérés objektuma.
* **Bemeneti adatok ellenőrzése**: Nincs külön validáció, minden bemenet elfogadásra kerül.
* **Lépések**:
  1. Létrehozza az új terméket.
* **Visszatérési érték**: JSON formátumban visszaadja az új terméket, HTTP 201-es státuszkóddal.

show($id)

* **Leírás**: Egy adott termék lekérdezése az azonosító alapján.
* **Paraméterek**: **$id** - a termék azonosítója.
* **Visszatérési érték**: JSON formátumban visszaadja a keresett terméket, hiba esetén 404-es státuszkóddal.

update(Request $request, $id)

* **Leírás**: Egy termék frissítése.
* **Paraméterek**: **$request** - a HTTP kérés objektuma, **$id** - a frissítendő termék azonosítója.
* **Bemeneti adatok ellenőrzése**: Ellenőrzi a beérkező adatokat, hogy megfelelnek-e a validációs szabályoknak.
* **Lépések**:
  1. Validálja a beérkező adatokat.
  2. Frissíti a terméket.
* **Visszatérési érték**: JSON formátumban visszaadja a frissített terméket, HTTP 200-as státuszkóddal.

destroy($id)

* **Leírás**: Egy termék törlése.
* **Paraméterek**: **$id** - a törlendő termék azonosítója.
* **Művelet**: Megkeresi és törli a megadott terméket az azonosító alapján.
* **Visszatérési érték**: NULL-t ad vissza, HTTP 204-es státuszkóddal.

###### Validáció

* A **update()** metódusban az új adatokat a Laravel beépített validációs rendszerével ellenőrzi a megadott szabályoknak megfelelően.

##### AuthController

Az **AuthController** osztály felelős az autentikációs funkciók és a felhasználói profilkezelés végrehajtásáért.

###### Metódusok

register(Request $request)

* **Leírás**: Új felhasználó regisztrációja.
* **Paraméterek**: **$request** - a HTTP kérés objektuma.
* **Bemeneti adatok ellenőrzése**: A beérkező adatokat validálja a megadott szabályoknak megfelelően.
* **Lépések**:
  1. Létrehozza az új felhasználót a beérkező adatok alapján.
* **Visszatérési érték**: JSON formátumban visszaadja az új felhasználót és egy üzenetet, hogy a regisztráció sikeres volt.

login(Request $request)

* **Leírás**: Felhasználó bejelentkeztetése.
* **Paraméterek**: **$request** - a HTTP kérés objektuma.
* **Bemeneti adatok ellenőrzése**: A beérkező adatokat validálja a megadott szabályoknak megfelelően.
* **Lépések**:
  1. Ellenőrzi a felhasználónév és a jelszó párosítását.
  2. Ha a bejelentkezés sikeres, JWT tokent hoz létre a felhasználó számára.
* **Visszatérési érték**: JSON formátumban visszaadja a JWT tokent és a felhasználót, ha a bejelentkezés sikeres volt.

profile(Request $request)

* **Leírás**: Az aktuális felhasználó profiljának lekérése.
* **Paraméterek**: **$request** - a HTTP kérés objektuma.
* **Működés**: Ellenőrzi, hogy a felhasználó be van-e jelentkezve. Ha igen, visszaadja az aktuális felhasználó adatait.
* **Visszatérési érték**: JSON formátumban visszaadja az aktuális felhasználó adatait.

logout(Request $request)

* **Leírás**: Felhasználó kijelentkeztetése.
* **Paraméterek**: **$request** - a HTTP kérés objektuma.
* **Működés**: Ellenőrzi, hogy a felhasználó be van-e jelentkezve. Ha igen, törli a felhasználó összes aktív tokent.
* **Visszatérési érték**: JSON formátumban visszaadja a kijelentkezés sikerét.

updateProfile(Request $request)

* **Leírás**: Felhasználó profiljának frissítése.
* **Paraméterek**: **$request** - a HTTP kérés objektuma.
* **Működés**: Ellenőrzi, hogy a felhasználó be van-e jelentkezve. Ha igen, validálja a beérkező adatokat, majd frissíti a felhasználó adatait.
* **Visszatérési érték**: JSON formátumban visszaadja a frissített felhasználó adatait.

deleteProfile(Request $request)

* **Leírás**: Felhasználó profiljának törlése.
* **Paraméterek**: **$request** - a HTTP kérés objektuma.
* **Működés**: Ellenőrzi, hogy a felhasználó be van-e jelentkezve. Ha igen, törli a felhasználó adatait.
* **Visszatérési érték**: JSON formátumban visszaadja a törlés sikerét.

###### Validáció

* Az **updateProfile()** metódusban a beérkező adatokat a Laravel beépített validációs rendszerével ellenőrzi a megadott szabályoknak megfelelően.

#### Végpontok

##### Admin végpontok

Az itt látható kódrészlet az adminisztrátor jogosultságokkal rendelkező felhasználók számára elérhető API végpontokat definiálja. Itt csak az adminisztrátorok hajthatnak végre műveleteket, például márkák, kategóriák, termékek, rendelési tételek és felhasználók kezelése terén.

Ezen végpontok csak akkor lesznek elérhetők, ha a felhasználó autentikálva van Sanctum segítségével és rendelkezik adminisztrátori jogosultságokkal. A **auth:sanctum** middleware biztosítja az autentikációt, míg az **admin** middleware az adminisztrátori jogosultságok ellenőrzését végzi.

* **Brands Controller:**
  + **/addBrand**: Új márka hozzáadása.
  + **/updateBrand/{id}**: Márka frissítése az azonosító alapján.
  + **/deleteBrand/{id}**: Márka törlése az azonosító alapján.
* **Categories Controller:**
  + **/addCategory**: Új kategória hozzáadása.
  + **/updateCategory/{id}**: Kategória frissítése az azonosító alapján.
  + **/deleteCategory/{id}**: Kategória törlése az azonosító alapján.
* **Products Controller:**
  + **/products**: Új termék létrehozása.
  + **/products/{id}**: Termék frissítése az azonosító alapján.
  + **/products/{id}**: Termék törlése az azonosító alapján.
* **OrderItems Controller:**
  + **/order\_items**: Összes rendelési tétel lekérése.
* **Orders Controller:**
  + **/orders**: Összes rendelés lekérése.
* **Users Controller:**
  + **/users**: Összes felhasználó lekérése.
  + **/users/{id}**: Egy felhasználó részletes adatainak lekérése az azonosító alapján.
  + **/updateUser/{id}**: Felhasználó frissítése az azonosító alapján.
  + **/deleteUser/{id}**: Felhasználó törlése az azonosító alapján.

##### User végpontok

Az itt látható kód egy olyan végpontcsoportot definiál, amely az autentikált felhasználók számára elérhető. Ezek a végpontok a felhasználói profil kezelését és a rendelésekkel kapcsolatos műveleteket teszik lehetővé.

Ezen végpontok csak akkor lesznek elérhetők, ha a felhasználó autentikálva van Sanctum segítségével. Az autentikációt a **auth:sanctum** middleware biztosítja.

* **Felhasználói profil kezelése:**
  + **/auth/logout**: Kijelentkezés.
  + **/auth/profile**: Felhasználói profil lekérése.
  + **/auth/deleteProfile**: Felhasználói profil törlése.
  + **/auth/updateProfile**: Felhasználói profil frissítése.
* **Rendelések kezelése:**
  + **/placeOrder**: Új rendelés leadása.
  + **/showOrders**: Felhasználó összes rendelésének lekérése.
  + **/showOrderItems/{id}**: Egy rendeléshez tartozó összes tétel lekérése.
  + **/updateOrder/{id}**: Rendelés frissítése.
  + **/deleteOrder/{id}**: Rendelés törlése.
* **OrderItems Controller:**
  + **/order\_items**: Összes rendelési tétel lekérése.
  + **/order\_items/{id}**: Egy rendelési tétel részleteinek lekérése az azonosító alapján.
  + **/order\_items/{id}**: Rendelési tétel frissítése az azonosító alapján.
  + **/order\_items/{id}**: Rendelési tétel törlése az azonosító alapján.
* **Orders Controller:**
  + **/orders**: Összes rendelés lekérése.
  + **/orders/{id}**: Egy rendelés részleteinek lekérése az azonosító alapján.
  + **/orders/{id}**: Rendelés frissítése az azonosító alapján.
  + **/orders/{id}**: Rendelés törlése az azonosító alapján.
* **Users Controller:**
  + **/users/{id}**: Felhasználó frissítése az azonosító alapján.
  + **/users/{id}**: Felhasználó törlése az azonosító alapján.

##### Autentikáció nélküli végpontok

Ezek a végpontok az autentikáció nélkül elérhető funkciókat teszik elérhetővé a felhasználók számára.

Ezeket a végpontokat az autentikáció nélkül elérhető szolgáltatásokhoz, például a termékek, kategóriák vagy márkák megtekintéséhez használhatják a felhasználók. Az autentikációs middleware-t nem alkalmazzák ezeken a végpontokon, így a felhasználók szabadon hozzáférhetnek hozzájuk az autentikáció nélkül is.

* **Authentikáció:**
  + **/auth/register**: Regisztráció.
  + **/auth/login**: Bejelentkezés.
* **Brands Controller:**
  + **/brands**: Összes márka lekérése.
  + **/brands/{id}**: Egy márka részleteinek lekérése az azonosító alapján.
* **Categories Controller:**
  + **/categories**: Összes kategória lekérése.
  + **/categories/{id}**: Egy kategória részleteinek lekérése az azonosító alapján.
* **Payment Controller:**
  + **/payment**: Összes fizetési mód lekérése.
  + **/payment/{id}**: Egy fizetési mód részleteinek lekérése az azonosító alapján.
* **Products Controller:**
  + **/products**: Összes termék lekérése.
  + **/products/{id}**: Egy termék részleteinek lekérése az azonosító alapján.

#### Middleware-k

##### IsAdmin Middleware

Az **IsAdmin** middleware felelős az adminisztrátori jogosultságok ellenőrzéséért egy bejövő kérés esetén. Ennek a middleware-nek a használata során ellenőrizzük, hogy a felhasználó rendelkezik-e adminisztrátori jogosultsággal. Ha igen, engedélyezzük a kérés feldolgozását, különben pedig visszautasítjuk a hozzáférést.

###### Működés

Az **IsAdmin** middleware a következőképpen működik:

* Ha a bejelentkezett felhasználó rendelkezik adminisztrátori jogosultsággal, akkor a kérés folytatódik és a hozzáférés engedélyezett.
* Ha a felhasználó nem rendelkezik adminisztrátori jogosultsággal, akkor egy 403-as státuszkóddal ellátott választ kap, ami azt jelzi, hogy a felhasználó nem rendelkezik megfelelő jogosultságokkal.

Ez a middleware hasznos biztonsági réteget képezhet egy alkalmazásban, amely korlátozza a kritikus műveletek végrehajtását csak az engedélyezett felhasználók számára.

###### Használat

Az **IsAdmin** middleware-t úgy használhatjuk, hogy regisztráljuk az alkalmazásunk middleware csoportjába, majd hozzárendeljük a megfelelő útvonalakhoz vagy kontrollerekhez.

Példa a middleware regisztrálására a **app/Http/Kernel.php** fájlban:

protected $routeMiddleware = [

// …

’admin’ => \App\Http\Middleware\IsAdmin::class,

];

Ezután az adminisztrátori jogosultságot ellenőrizhetjük az útvonalaknál vagy a kontrollereknél a **admin** kulcsszó használatával:

Route::middleware(’admin’)->get(’admin/dashboard’, function () {

// …

});

Ez biztosítja, hogy csak az adminisztrátorok férjenek hozzá az adott útvonalakhoz vagy kontrollerekhez.

#### Seeder-ek

##### BrandsSeeder

A **BrandsSeeder** osztály felelős a márkák létrehozásáért az adatbázisban. Ezek a márkák segítenek az alkalmazásban található termékek azonosításában és csoportosításában.

###### Létrehozott márkák

1. **Intel**: Az Intel egy világszerte elismert vállalat, amely processzorokat és egyéb számítógépes hardvereket gyárt.
2. **Amd**: Az AMD egy másik nagyobb szereplő a számítógépes hardverpiacon, amely szintén processzorokat és más alkatrészeket gyárt.
3. **G.Skill**: A G.Skill egy olyan márka, amely a számítógép-memóriamodulokra (RAM) specializálódott, köztük különféle sebességekkel és kapacitásokkal rendelkező memóriamodulokat kínál.
4. **Gigabyte**: A Gigabyte egy olyan vállalat, amely széleskörű számítógépes alkatrészeket kínál, ideértve az alaplapokat, videokártyákat és egyéb perifériákat.
5. **MSI**: Az MSI egy másik olyan vállalat, amely számítógépes alkatrészeket gyárt, beleértve az alaplapokat, videokártyákat és egyéb hardvereket.

###### Használat

A **BrandsSeeder** osztály a Laravel **db:seed** parancsának használatával futtatható, vagy más inicializálási folyamatok részeként. A **run** metódusában létrehozott márkák az alkalmazás számára biztosítják az alapvető márkák kezdeti adatokkal való feltöltését az adatbázisban.

##### CategoriesSeeder

A **CategoriesSeeder** osztály felelős a kategóriák létrehozásáért az adatbázisban. Ezek a kategóriák segítenek csoportosítani a termékeket az alkalmazásban.

###### Létrehozott kategóriák

1. **Processor**: Ez a kategória a processzorokat tartalmazza, amelyek az informatikai eszközök egyik alapvető elemei.
2. **Memory**: A "Memory" kategória a memóriamodulokat vagy RAM (Random Access Memory) eszközöket foglalja magában, amelyek az adatok ideiglenes tárolására szolgálnak a számítógépben.
3. **Motherboard**: Ez a kategória az alaplapokat vagy az alaplaphoz kapcsolódó kiegészítő eszközöket tartalmazza. Az alaplap az egyik legfontosabb hardveralkatrész a számítógépben, mivel összeköti a különböző alkatrészeket és biztosítja az adatok továbbítását.

###### Használat

A **CategoriesSeeder** osztály a Laravel **db:seed** parancsának használatával futtatható, vagy más inicializálási folyamatok részeként. A **run** metódusában létrehozott kategóriák az alkalmazás számára biztosítják az alapvető kategóriák kezdeti adatokkal való feltöltését az adatbázisban.

##### PaymentSeeder

A **PaymentSeeder** osztály felelős a fizetési módok létrehozásáért az adatbázisban. Ezek a fizetési módok segítenek az alkalmazásban a rendelések fizetésének kezelésében és kategorizálásában.

###### Létrehozott fizetési módok

1. **Boltban**: Ez a fizetési mód azt jelenti, hogy a vásárlók személyesen fizetnek a boltban, amikor átveszik a megrendelt termékeket.
2. **Háznál**: Ez a fizetési mód azt jelenti, hogy a vásárlók a szállítási szolgáltatás által szállított termékek kiszállításakor fizetnek.

###### Használat

A **PaymentSeeder** osztály a Laravel **db:seed** parancsának használatával futtatható, vagy más inicializálási folyamatok részeként. A **run** metódusában létrehozott fizetési módok az alkalmazás számára biztosítják az alapvető fizetési módok kezdeti adatokkal való feltöltését az adatbázisban.

##### ProductsSeeder

A **ProductsSeeder** osztály felelős különböző termékek létrehozásáért és feltöltéséért az adatbázisba. Ezek a termékek reprezentálják a boltban elérhető különböző hardverkomponenseket, például processzorokat, memóriákat és alaplapokat.

###### Létrehozott termékek

1. **i7-11700k**
   * Kategória: Processzor
   * Márka: Intel
   * Ár: 117000 Ft
   * Készleten lévő mennyiség: 0
   * Kép: .\assets\images\proci1.jpg
   * Leírás: faszfasz
2. **Ryzen 5 5600x**
   * Kategória: Processzor
   * Márka: Amd
   * Ár: 60000 Ft
   * Készleten lévő mennyiség: 0
   * Kép: .\assets\images\proci2.jpg
   * Leírás: faszfasz
3. **16GB DDR4 3200MHz CL16 Trident Z RGB**
   * Kategória: Memória
   * Márka: G.Skill
   * Ár: 25000 Ft
   * Készleten lévő mennyiség: 0
   * Kép: .\assets\images\memoria1.jpg
   * Leírás: faszfasz
4. **16GB DDR4 3733MHz Kit Aorus RGB**
   * Kategória: Memória
   * Márka: Gigabyte
   * Ár: 26000 Ft
   * Készleten lévő mennyiség: 0
   * Kép: .\assets\images\memoria2.jpg
   * Leírás: faszfasz
5. **B550 Gaming X V2**
   * Kategória: Alaplap
   * Márka: Gigabyte
   * Ár: 45000 Ft
   * Készleten lévő mennyiség: 0
   * Kép: .\assets\images\alaplap1.jpg
   * Leírás: faszfasz
6. **MSI MPG Z590 GAMING PLUS**
   * Kategória: Alaplap
   * Márka: MSI
   * Ár: 90000 Ft
   * Készleten lévő mennyiség: 0
   * Kép: .\assets\images\alaplap2.jpg
   * Leírás: faszfasz

###### Használat

A **ProductsSeeder** osztály a Laravel **db:seed** parancsának használatával futtatható, vagy más inicializálási folyamatok részeként. A **run** metódusában létrehozott termékek az alkalmazás számára biztosítják az alapvető termékek kezdeti adatokkal való feltöltését az adatbázisban.

##### DatabaseSeeder Documentation

1. **Create Admin User:**
   * Creates an admin user with the following credentials:
     + Name: admin
     + Email: admin@example.com
     + Password: admin
     + Address: ''
     + City: ''
     + Postal Code: 0
     + Phone: ''
     + Admin: true
2. **Seed Other Tables:**
   * Calls the seeder classes responsible for seeding other tables:
     + **BrandsSeeder**: Seeds brands data.
     + **CategoriesSeeder**: Seeds categories data.
     + **PaymentSeeder**: Seeds payment methods data.
     + **ProductsSeeder**: Seeds products data.

###### Usage:

The **DatabaseSeeder** class is executed when running the **db:seed** Artisan command. It initializes the database with essential initial data, including an admin user and various seed data for other tables.

## Frontend

#### about

#### cart

#### choose

#### footer

#### header

#### home

#### login

#### myorders

#### product-edit

#### products

#### profile

#### profile-edit

#### signup

#### app

### Komponensek

#### About Komponens

##### about.component.html

* Az **about.component.html** fájl tartalmazza a "Rólunk" oldal HTML kódját, amely a PC Webshop bemutatását és köszönetét tartalmazza a felhasználóknak.

##### about.component.ts

* Az **about.component.ts** fájl egy üres TypeScript osztályt tartalmaz, amelynek nincs saját logikája vagy adattagja, mivel a komponens statikus tartalmat jelenít meg.

#### Cart Komponens

##### cart.component.html

* Az **cart.component.html** fájl tartalmazza a kosár komponens HTML kódját, amely megjeleníti a felhasználó kosarában lévő termékeket és az összesített árat.

##### cart.component.ts

* Az **cart.component.ts** fájlban található osztály felelős a kosár komponens logikájáért. Ez a logika magában foglalja a kosár elemeinek betöltését, a képfájl útvonalak generálását, a termékek eltávolítását a kosárból, az összesített ár kiszámítását és a továbblépés műveletét.
* A kosár elemek dinamikusan generált képeinek eléréséhez használja a **generateImagePath** metódust, amely az adott termék nevéből állítja elő az elérési útvonalat.

###### Metódusok Leírása

generateImagePath(productName: string): string

Ez a metódus felelős a termék nevéből generált képfájl elérési útvonalának létrehozásáért. A **productName** paraméterként kapja meg a termék nevét, majd a metódus visszaadja a megfelelő elérési útvonalat a képfájlhoz. Például, ha a **productName** értéke "laptop", akkor az elérési útvonal a **assets/images/laptop.jpg** lesz.

removeFromCart(item: any): void

Ez a metódus felelős egy adott termék eltávolításáért a kosárból. Paraméterként megkapja a törölni kívánt terméket (**item**), majd azonosítja annak indexét a **cartItems** tömbben és eltávolítja azt. Ezután újraszámolja az összesített árat a kosárban található termékek alapján.

calculateTotalPrice(): void

Ez a metódus felelős az összesített ár kiszámításáért a kosárban található termékek alapján. Végigmegy a **cartItems** tömbön, és minden termék árát és mennyiségét figyelembe véve összeadja azokat. Az eredményt a **totalPrice** változóban tárolja.

next(): void

Ez a metódus egy műveletet hajt végre, amikor a felhasználó tovább lép a következő lépéshez a vásárlási folyamatban. Jelenleg csak egy üzenetet jelenít meg egy felugró ablakban, amely figyelmezteti a felhasználót, hogy válasszon szállítási módot.

### Service.ts fájlok

#### AuthService

Ez a szolgáltatás felelős a felhasználók autentikációjáért és profilkezeléséért az Angular webshop alkalmazásban.

##### Metódusok

###### login(userData: any): void

Ez a metódus lehetővé teszi a felhasználók bejelentkeztetését a megadott felhasználói adatok alapján. Az adatokat a backend szerverrel történő HTTP POST kéréssel küldi el a következő végpontra: **/api/auth/login**.

* **Paraméterek:**
  + **userData** (any): Egy objektum, amely tartalmazza a bejelentkezéshez szükséges felhasználói adatokat, például felhasználónév és jelszó.

###### updateProfile(userData: any): Observable<any>

Ez a metódus lehetővé teszi a felhasználói profil frissítését a megadott felhasználói adatok alapján. Az adatokat a backend szerverrel történő HTTP PUT kéréssel küldi el a következő végpontra: **/api/auth/updateProfile**.

* **Paraméterek:**
  + **userData** (any): Egy objektum, amely tartalmazza a frissíteni kívánt felhasználói adatokat, például név, e-mail cím stb.
* **Visszatérési érték:** Egy Observable, amely a frissítés eredményét tartalmazza.

###### deleteProfile(): Observable<any>

Ez a metódus lehetővé teszi a felhasználói profil törlését. Az adatokat a backend szerverrel történő HTTP DELETE kéréssel küldi el a következő végpontra: **/api/auth/deleteProfile**.

* **Visszatérési érték:** Egy Observable, amely a törlés eredményét tartalmazza.

###### getProfile(): Observable<any>

Ez a metódus lehetővé teszi a felhasználói profil lekérését a bejelentkezett felhasználóhoz tartozó adatokkal. Az adatokat a backend szerverrel történő HTTP GET kéréssel kéri le a következő végpontra: **/api/auth/profile**.

* **Visszatérési érték:** Egy Observable, amely a felhasználó profilját tartalmazza.

##### Tulajdonságok

* **isLoggedIn**: Logikai érték, amely azt jelzi, hogy a felhasználó be van-e jelentkezve.
* **isAdmin**: Logikai érték, amely azt jelzi, hogy a felhasználó adminisztrátor-e.
* **userData**: Egy objektum, amely a bejelentkezett felhasználó adatait tárolja.
* **token**: A bejelentkezett felhasználóhoz tartozó token.

##### Konstruktor

###### constructor(http: HttpClient)

Az **AuthService** szolgáltatás konstruktora, amelynek a **HttpClient** injektált függősége van. Ez az osztály használja a HttpClient modult az HTTP kérések végrehajtásához a backend szerverrel.

#### BaseService

Ez a szolgáltatás felelős az alkalmazás termékekkel kapcsolatos alapvető műveleteinek végrehajtásáért.

##### Metódusok

###### getProducts(): Observable<any>

Ez a metódus lehetővé teszi az összes termék lekérését a szerverről a következő végpontra: **/api/products**.

* **Visszatérési érték:** Egy Observable, amely az összes terméket tartalmazza.

###### setSelectedProduct(product: Product): void

Ez a metódus beállítja a kiválasztott terméket a paraméterként megadott termékkel.

* **Paraméterek:**
  + **product** (Product): A kiválasztott termék.
* **Visszatérési érték:** Nincs.

###### getSelectedProduct(): Product

Ez a metódus visszaadja a jelenleg kiválasztott terméket.

* **Visszatérési érték:** A kiválasztott termék.

##### Tulajdonságok

* **host**: A backend szerver címe, ahol a termékadatokhoz való hozzáférés történik.
* **selectedProduct**: A jelenleg kiválasztott termék.

##### Konstruktor

###### constructor(http: HttpClient)

A **BaseService** szolgáltatás konstruktora, amelynek a **HttpClient** injektált függősége van. Ez az osztály használja a HttpClient modult az HTTP kérések végrehajtásához a backend szerverrel.

#### CartService

Ez a szolgáltatás felelős a felhasználó kosarában lévő termékek kezeléséért az Angular webshop alkalmazásban.

##### Metódusok

###### addToCart(item: any, quantity: number = 1): void

Ez a metódus hozzáad egy elemet a kosárhoz vagy növeli annak mennyiségét, ha az már létezik a kosárban.

* **Paraméterek:**
  + **item** (any): A hozzáadandó elem.
  + **quantity** (number): Az elem mennyisége (alapértelmezetten 1).
* **Visszatérési érték:** Nincs.

###### getCartItems(): any[]

Ez a metódus visszaadja a kosárban lévő összes elemet.

* **Visszatérési érték:** Egy tömb, amely tartalmazza a kosárban lévő összes elemet.

###### removeFromCart(index: number): void

Ez a metódus eltávolít egy elemet a kosárból a megadott index alapján.

* **Paraméterek:**
  + **index** (number): Az eltávolítandó elem indexe a kosárban.
* **Visszatérési érték:** Nincs.

###### clearCart(): void

Ez a metódus kiüríti a kosarat, eltávolítva minden elemet belőle.

* **Visszatérési érték:** Nincs.

##### Tulajdonságok

* **cartItems**: Egy tömb, amely a kosárban lévő elemeket tárolja.

##### Konstruktor

###### constructor()

A **CartService** szolgáltatás konstruktora.

#### OrderService

Ez a szolgáltatás felelős a rendelések kezeléséért az Angular webshop alkalmazásban.

##### Metódusok

###### order(userData: any): Observable<boolean>

Ez a metódus lehetővé teszi a felhasználók számára, hogy rendeléseket küldjenek be a megadott felhasználói adatok alapján. Az adatokat a backend szerverrel történő HTTP POST kéréssel küldi el a következő végpontra: **/api/placeOrder**.

* **Paraméterek:**
  + **userData** (any): Az új rendelés adatai.
* **Visszatérési érték:** Egy Observable, amely jelzi a rendelés sikerességét (**true** ha sikeres, **false** ha sikertelen).

###### getOrders(): Observable<any[]>

Ez a metódus lehetővé teszi a felhasználók számára, hogy lekérjék a korábban leadott rendeléseket. A rendeléseket a backend szerverrel történő HTTP GET kéréssel kéri le a következő végpontra: **/api/showOrders**.

* **Visszatérési érték:** Egy Observable, amely a rendeléseket tartalmazó tömböt adja vissza.

##### Tulajdonságok

* **userData**: Objektum, amely a rendeléshez kapcsolódó felhasználói adatokat tárolja.

##### Konstruktor

###### constructor(http: HttpClient, authService: AuthService)

Az **OrderService** szolgáltatás konstruktora, amelynek a **HttpClient** és az **AuthService** injektált függőségei vannak. Az **AuthService** segítségével az **OrderService** hozzáférhet a felhasználó autentikációs adataihoz.

### Képek és Image Carousel

Az alkalmazásban használt képek és az image carousel funkció részletes leírása.

#### Képek

Az alkalmazás megjelenítéséhez használt képek az "assets" mappában találhatók. Ezek a képek különböző részein jelennek meg, például a termékek megjelenítésénél vagy az image carouselban.

#### Image Carousel

Az alkalmazás főoldalán egy image carousel található, amely több képet jelenít meg. Ezek a képek olyan termékeket vagy promóciókat mutatnak be, amelyekkel a felhasználókat megcélozzuk.

Az image carousel automatikusan váltja a képeket egy adott időközönként, hogy vonzóbbá és dinamikusabbá tegye az oldal megjelenését.

Az image carouselban használt képek ugyanazon "assets" mappában találhatók, mint az alkalmazás többi képei, így könnyen hozzáférhetők és cserélhetők.

# Tárgymutató

## Általunk használt kifejezések

# Felhasznált irodalom

# Köszönet