Základy webových technológií

**Zadanie:**

**Semestrálny projekt - elektronický obchod**

Jana Puchein  
Matej Kotúč

Obsah

[**1.** **Zadanie** 2](#_Toc197720787)

[**2.** **Diagram fyzického dátového modelu** 3](#_Toc197720788)

[**3.** **Návrhové rozhodnutia** 3](#_Toc197720789)

[**2.** **Snímky obrazoviek** 11](#_Toc197720790)

[**1.** **Detail produktu**  11](#_Toc197720791)

[**2.** **Prihlásenie/registrácia** 11](#_Toc197720792)

[**3.** **Landing page** 11](#_Toc197720793)

[**4.** **Home page** 12](#_Toc197720794)

[**5.** **Košík** 13](#_Toc197720795)

[**6.** **Checkout** 13](#_Toc197720796)

[**7.** **Potvrdenie objednávky** 14](#_Toc197720797)

[**8.** **Admin page** 14](#_Toc197720798)

[**9.** **Wireframes** 15](#_Toc197720799)

# **Zadanie**

Vytvorte webovú aplikáciu - eshop, ktorá komplexne rieši nižšie definované prípady použitia vo vami zvolenej doméne (napr. elektro, oblečenie, obuv, nábytok). Presný rozsah a konkretizáciu prípadov použitia si dohodnete s vašim vyučujúcim.

# **Diagram fyzického dátového modelu**

Tento diagram zobrazuje návrh databázovej štruktúry pre náš e-shop. Popisuje nasledujúce entity a ich vzťahy:

1. **users** – Užívatelia systému so základnými údajmi (meno, email, heslo, rola).
2. **sessions** – Sledovanie prihlásených relácií používateľov podľa session ID.
3. **products** – Produkty, ktoré obsahujú informácie ako názov, popis, cena, farba a väzba na výrobcu.
4. **manufacturers** – Výrobcovia produktov, uložené sú názvy a logá.
5. **categories** a **category\_product** – Produkty sú zaradené do kategórií cez pomocnú tabuľku na mnoho-na-mnoho vzťah.
6. **images** – Obrázky produktov s odkazom na URL a typ obrázka.
7. **size\_stocks** – Zásoby produktov podľa veľkostí.
8. **cart\_items** – Položky v nákupnom košíku, viazané na session alebo používateľa, obsahujú produkt, množstvo a veľkosť.
9. **orders** – Objednávky priradené k používateľovi alebo session, so stavom objednávky.
10. **order\_items** – Položky v objednávke s produktom, veľkosťou a množstvom.

Všetky entity sú prepojené cudzími kľúčmi a majú správnu väzbu medzi užívateľmi, produktmi, košíkom a objednávkami.

Obrázok, na ktorom je opálenie, štvorec

Obsah vygenerovaný umelou inteligenciou môže byť nesprávny.

# **Návrhové rozhodnutia**

Počas návrhu aplikácie sme prijali niekoľko dôležitých rozhodnutí s cieľom zjednodušiť vývoj, udržať čitateľnosť kódu a zároveň zabezpečiť požadovanú funkcionalitu systému:

Pri vývoji aplikácie sme sa rozhodli použiť **Laravel framework**, ktorý nám poskytuje robustnú podporu pre prácu s databázou (migrácie, seedery), routing, autentifikáciu a organizáciu projektu podľa **MVC architektúry**. Laravel výrazne urýchľuje vývoj a zjednodušuje správu backendu. Použili sme **migrácie a seedery**, aby sme vedeli ľahko definovať a naplniť databázu s testovacími údajmi počas vývoja a testovania.

Na frontend sme zvolili **Bootstrap**, ktorý nám umožnil rýchlo vytvoriť responzívne a konzistentné používateľské rozhranie bez potreby vlastného CSS frameworku.

Pri návrhu databázy sme vychádzali z požiadaviek e-shopu, pričom sme zohľadnili entities ako produkty, objednávky, kategórie, výrobcu a stav zásob. Všetky tieto entity sú navzájom prepojené pomocou relačných väzieb. Roly sme riešili jednoduchým **atribútom role v tabuľke používateľov**, ktorý určuje, či ide o bežného používateľa alebo administrátora. **Nepoužili sme zložité oprávnenia (authorization gates alebo policies)**, nakoľko pre náš účel postačovalo rozlíšiť iba tieto dve základné roly a riadiť prístup jednoducho pomocou kontroléra.

# **Opis implementácie vybraných use cases**

1. **Zmena množstva produktu v košíku**

Používateľ má možnosť zmeniť množstvo konkrétneho produktu v nákupnom košíku (napr. ak si chce objednať viac alebo menej kusov daného produktu v konkrétnej veľkosti).

**a) Prijatie a validácia požiadavky**

Funkcia updateQuantity je súčasťou CartController a prijíma POST požiadavku obsahujúcu:

* product\_id – ID produktu
* size – veľkosť produktu
* amount – nové požadované množstvo

Laravel validátor skontroluje, že:

* product\_id existuje v databáze produktov
* size je celé číslo
* amount je celé číslo väčšie alebo rovné 1

#### **b) Vyhľadanie položky v košíku**

Funkcia vyhľadá zodpovedajúcu položku v košíku (tabuľka cart\_items), ktorá zodpovedá danému produktu a veľkosti, podľa user\_id alebo session\_id.

Ak sa takáto položka v košíku nenájde, používateľ dostane chybové hlásenie:

**c) Kontrola dostupnosti skladu**

Systém skontroluje, koľko kusov daného produktu vo zvolenej veľkosti je momentálne na sklade (z tabuľky size\_stocks).

$stockItem = SizeStock::where('product\_id', $validated['product\_id'])

->where('size', $validated['size'])

->first();

Ak používateľ zadá vyššie množstvo, ako je aktuálne dostupné, zobrazí sa mu chybová správa:

**d) Aktualizácia množstva**

Ak je zadané množstvo v poriadku, hodnota v stĺpci amount v tabuľke cart\_items sa aktualizuje a uloží:

$cartItem->amount = $validated['amount'];

$cartItem->save();  
  
**e) Spätná väzba používateľovi**

Po úspešnej zmene sa používateľovi zobrazí hláška o úspechu.

1. **Prihlásenie**

Funkcia: login(Request $request) overuje prihlasovacie údaje používateľa (email a heslo) a prihlási ho do systému, ak sú údaje správne.

$credentials = $request->validate([

'email' => 'required|email',

'password' => 'required|min:8',

]);

Používateľ odosiela email a heslo cez prihlasovací formulár.

Systém najprv overí, či bol email vyplnený správne a heslo má aspoň 8 znakov.

Ak validácia zlyhá, Laravel automaticky vráti chyby do predchádzajúcej stránky.

$credentials = $request->validate([

'email' => 'required|email',

'password' => 'required|min:8',

]);  
  
Funkcia Auth::attempt() overí, či existuje používateľ s daným emailom a či heslo zodpovedá tomu, ktoré je uložené v databáze. Ak je overenie úspešné, používateľ je autentifikovaný.

if (Auth::attempt($credentials))

**b) Presmerovanie podľa role používateľa**

$user = Auth::user();

if ($user && $user->role === 'admin') {

return redirect('/admin');

}

return redirect()->intended('/');

Získa prihláseného používateľa.Ak má používateľ rolu admin, presmeruje ho do administrátorskej časti (/admin) -v opačnom prípade ho presmeruje na pôvodnú stránku, ktorú chcel navštíviť (intended).

Ak prihlasovanie zlyhá (zlé údaje), systém vráti používateľa späť na formulár s chybovou správou a zachová vstupy.

return back()->withErrors([

'email' => 'Login not successful',

])->withInput();

1. **Vyhľadávanie a pridanie produktu do košíka**

**1. Vyhľadávanie produktu**

Vyhľadávanie produktov je implementované v triede BrowsingController, najmä v metóde show(string $category).

* **Používateľ navštívi** stránku kategórie (napr. /category/obuv), pričom môže použiť aj parametre v URL:

subcategory – filtrovanie podľa podkategórie,

term – vyhľadávanie podľa názvu produktu alebo výrobcu (fulltext vyhľadávanie pomocou ILIKE).

* **Query builder** načítava produkty, ktoré:

patria do danej kategórie (a voliteľne aj podkategórie),

zodpovedajú hľadanému výrazu (term) podľa názvu produktu alebo výrobcu.

* **Okrem samotného** zoznamu produktov sa pripravujú aj:

dostupní výrobcovia, farby, veľkosti (zo stock položiek),

minimálna a maximálna cena v rámci výsledkov.

Výsledné dáta sú odovzdané do view browsing.category-view, ktoré zobrazí filtrovaný zoznam produktov s možnosťou ďalšieho vyhľadávania a filtrovania.

1. **Pridanie produktu do košíka**

Pridávanie produktu do košíka sa realizuje v CartController cez metódu add(Request $request).

**Postup:**

1. **Validácia vstupu**:

product\_id, amount a size sú požadované a overuje sa ich platnosť.

1. **Identifikácia používateľa**:

Použije sa ID prihláseného používateľa (Auth::id()), alebo fallback na ID session pre neprihlásených.

1. **Overenie skladovej dostupnosti**:

Skontroluje sa, či produkt v danej veľkosti existuje a koľko kusov je skladom (SizeStock).

Ak požadované množstvo presahuje dostupné kusy, používateľ je informovaný o chybe.

1. **Vloženie alebo aktualizácia položky v košíku**:

Ak už produkt v rovnakej veľkosti existuje v košíku, množstvo sa navýši.

Inak sa vytvorí nová položka (CartItem).

1. **Uloženie a presmerovanie**:

Položka sa uloží do databázy.

Používateľ je presmerovaný späť s úspešnou hláškou.

1. **Stránkovanie**

Obrázok, na ktorom je text, snímka obrazovky, písmo, dizajn

Obsah vygenerovaný umelou inteligenciou môže byť nesprávny.

* 1. **Backend (Kontrolér – IndexController)**

V metóde show sa z databázy načítajú produkty prislúchajúce k danej kategórii a použije sa metóda simplePaginate(16), ktorá zabezpečuje stránkovanie:  
Používateľ pri prezeraní kategórie uvidí vždy 16 produktov na stránku. Pomocou navigačných tlačidiel "Prev" a "Next" môže jednoducho prechádzať medzi stránkami bez zobrazenia konkrétnych čísiel strán.

$products = $categoryProds->products()->with('images')->simplePaginate(16);

simplePaginate(16) načíta **16 produktov na jednu stranu**.

simplePaginate je jednoduchšia verzia paginate, ktorá nezobrazuje celkový počet strán (vhodné pre výkon). Výsledok sa odovzdá do view:

return view('index.category-view', compact('products', 'category'));

**2. Frontend (Blade šablóna – stránkovanie)**

Paginácia sa vykresľuje pomocou vlastného Blade kódu:

Prev a Next odkazy sa zobrazujú podľa toho, či existuje predchádzajúca alebo nasledujúca strana.

Aktuálne číslo stránky sa zobrazuje medzi nimi.

Trieda disabled zneaktívni tlačidlá na krajoch, keď nie je kam ísť.

@if ($paginator->hasPages())

    <nav class="my-4">

        <ul class="pagination justify-content-center mb-0">

            <li class="page-item {{ $paginator->onFirstPage() ? 'disabled' : '' }}">

                @if ($paginator->onFirstPage())

                    <span class="page-link">Prev</span>

                @else

                    <a class="page-link shadow-none" href="{{ $paginator->previousPageUrl() }}">Prev</a>

                @endif

            </li>

            <li class="page-item disabled">

                <span class="page-link text-dark px-3">

                    Page {{ $paginator->currentPage() }}

                </span>

            </li>

            <li class="page-item {{ $paginator->hasMorePages() ? '' : 'disabled' }}">

                @if ($paginator->hasMorePages())

                    <a class="page-link shadow-none" href="{{ $paginator->nextPageUrl() }}">Next</a>

                @else

                    <span class="page-link">Next</span>

                @endif

            </li>

        </ul>

    </nav>

@endif

1. **Základné filtrovanie**

Na filtrovanie produktov sa využíva knižnica Livvewire, ktorá umožňuje vytvárať dynamické používateľské rozhrania. Funguje na princípe jednoduchých fetch požiadaviek, ktoré zabezpečujú interakciu medzi frontendom a backendom.

Komponenty Livewire obsahujú render metódu, ktorá sa opakovane vykonáva pri každej zmene stavu. Všetky zmeny v premenných Livewire automaticky spúšťajú požiadavky na server, kde sa komponent znovu vykreslí a odošle sa iba potrebná časť HTML späť do prehliadača. To zabezpečuje dynamické filtrovanie bez potreby refresh celej stránky.

## **Livewire kontroler** ProductFilters

**Premenné, ktoré filtrujú produkty:**

public $brand = []; // výber značiek (výrobcov)

public $size = []; // výber veľkostí

public $color = []; // výber farieb

public $price; // max. cena

public $sortPrice; // zoradenie podľa ceny (asc/desc)

Tieto premenné sú naviazané na jednotlivé komponenty v product-filters.blade.php view pomocou atribútu *wire.*

**Aplikovanie jednotlivých filtrov:**

$products = $productsQuery

->when($this->price, fn ($query) => $query->where('price','<=',$this->price))

->when($this->color, fn ($query) => $query->whereIn('color',$this->color))

->when($this->size, fn ($query) =>

$query->whereHas('stock', fn ($q) => $q->whereIn('size',$this->size)))

->when($this->brand, fn ($query) =>

$query->whereHas('manufacturer', fn ($q) => $q->whereIn('name',$this->brand)))

->when($this->sortPrice, fn ($query) => $query->orderBy('price', $this->sortPrice))

->with('images')

->paginate(16);

* **Resetovanie stránkovania pri zmene filtra**

public function updatedBrand() { $this->resetPage(); }

public function updatedColor() { $this->resetPage(); }

public function updatedSize() { $this->resetPage(); }

public function updatedPrice() { $this->resetPage(); }

Tento kúsok kódu zabezpečuje zavolanie render funkcie po zmene premenných ovládaných livewire komponentami:

<input type="checkbox" wire:model.lazy='size' value="{{ $size }}">

<input type="checkbox" wire:model.lazy='color' value="{{ $color }}">

<input type="checkbox" wire:model.lazy="brand" value="{{ $manufacturer }}">

<input type="range" wire:model.lazy="price" min="{{ $priceMin }}" max="{{ $priceMax }}" class="slider" id="priceRange" style="accent-color: #8e5bf3b5;">

* **Zoradenie produktov podľa ceny**

<button wire:click="sortByPrice" class="filterBtn">

Sort by Price

      @if ($sortPrice === 'asc')

      ↑

  @elseif ($sortPrice === 'desc')

      ↓

@endif

</button>

Tento button volá nasledovnú funkciu, ktorá v backende mení zoradenie produktov:

public function sortByPrice()

{

$this->sortPrice = $this->sortPrice === 'asc' ? 'desc' : 'asc';

$this->resetPage();

}

A následne aplikácia vo funkcií render pomocou productQuery:

->when($this->sortPrice, fn ($query) => $query->orderBy('price', $this->sortPrice))->with('images')->paginate(16);

# **Snímky obrazoviek**

## **Detail produktu Obrázok, na ktorom je text, ochrana hlavy, snímka obrazovky, šiltovka Obsah vygenerovaný umelou inteligenciou môže byť nesprávny.**

## **Prihlásenie/registrácia**Obrázok, na ktorom je text, snímka obrazovky, softvér, počítačová ikona Obsah vygenerovaný umelou inteligenciou môže byť nesprávny.**Obrázok, na ktorom je snímka obrazovky, text, softvér, dizajn Obsah vygenerovaný umelou inteligenciou môže byť nesprávny.**

## **Landing page**

Obrázok, na ktorom je ošatenie, ľudská tvár, text, chlap

Obsah vygenerovaný umelou inteligenciou môže byť nesprávny.

## **Home page**

Obrázok, na ktorom je text, módny doplnok, snímka obrazovky, ochrana hlavy

Obsah vygenerovaný umelou inteligenciou môže byť nesprávny.

## **Košík**

Obrázok, na ktorom je text, softvér, multimediálny softvér, počítačová ikona

Obsah vygenerovaný umelou inteligenciou môže byť nesprávny.

## **Checkout**

Obrázok, na ktorom je text, softvér, multimediálny softvér, počítačová ikona

Obsah vygenerovaný umelou inteligenciou môže byť nesprávny.

## **Potvrdenie objednávky**

Obrázok, na ktorom je text, snímka obrazovky, softvér

Obsah vygenerovaný umelou inteligenciou môže byť nesprávny.

## **Admin page**

Obrázok, na ktorom je text, snímka obrazovky, softvér, počítačová ikona

Obsah vygenerovaný umelou inteligenciou môže byť nesprávny.

## **Wireframes**

Obrázok, na ktorom je text, snímka obrazovky, diagram, plán

Obsah vygenerovaný umelou inteligenciou môže byť nesprávny.

# **Záver**

V rámci tohto projektu sme úspešne navrhli a implementovali e-shopovú webovú stránku pomocou frameworku **Laravel**, s dôrazom na prehľadnú architektúru, responzívny dizajn a funkčnosť. Na vývoj frontend časti sme použili **HTML**, **CSS**, **Bootstrap** a **JavaScript**, čo nám umožnilo vytvoriť moderné a používateľsky prívetivé rozhranie. Backend bol postavený pomocou **PHP** a Laravelových komponentov ako sú **routy**, **kontroléry**, **modely** a **Livewire** pre interaktívne prvky bez potreby načítania celej stránky.

Pre správu databázy sme využili **migrácie** a **seederov**, ktoré nám umožnili jednoducho vytvárať tabuľky, testovacie dáta a zabezpečiť konzistentnosť vývoja. Implementované boli základné funkcionality ako filtrovanie produktov, stránkovanie, triedenie podľa ceny, a zobrazenie detailu produktu. To všetko nám poskytuje skvelý základ pre ďalšie vylepšenia a reálne nasadenie.