Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Ilkovičova 3, 842 19 Bratislava 4

Základy webových technológií

ak. rok 2023/24, letný semester

Dokumentácia

Cvičiaci: Ing. Eduard Kuric, PhD.

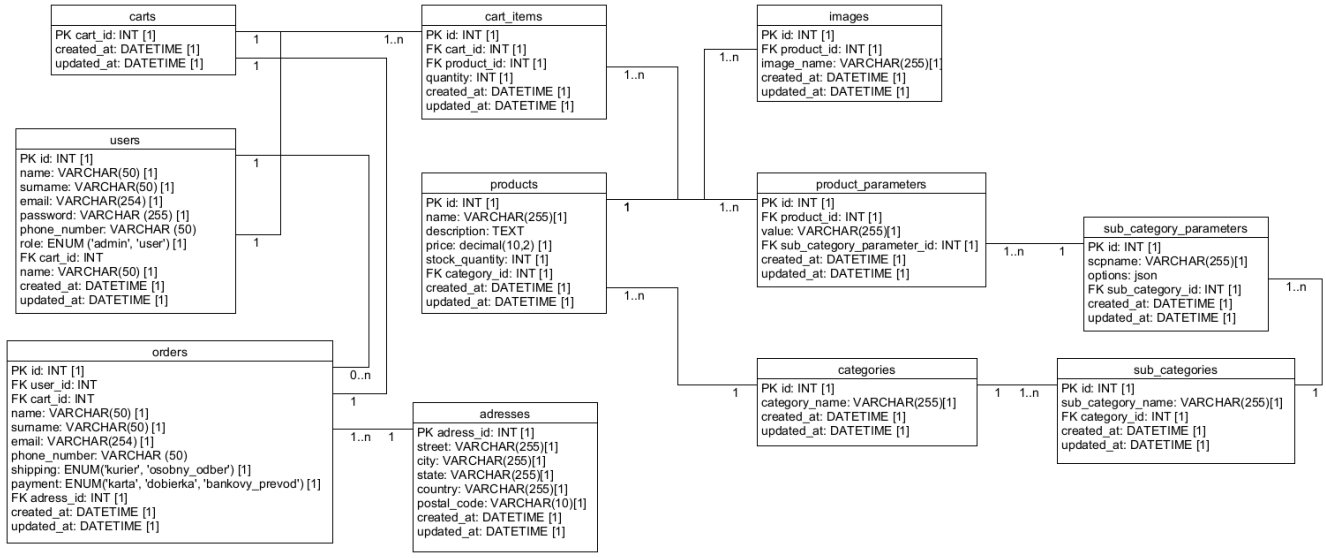
Čas cvičenia: Utorok 11:00-12:50

Vypracovali: Vojtech Babinský, Tomáš Tisovský

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



Zmeny fyzického dátového modelu:

- Do tabuľky “orders” boli pridané údaje o mene, emaili atď. objednávateľa (umožňuje sa tak objednávka aj pre neregistrovaného používateľa).
- Používateľov košík môže byž null (prázdny košík sa totiž maže)
- V tabuľke “images” sa už nepoužíva hash pre uloženie mena obrázka (pre jednoduchšiu prácu)
- Premenovanie niektorých parametrov (napr. V tabuľke “products” “product_id” na “id”)

Návrhové rozhodnutia

Knižnica Livewire – pri vytváraní nákupného košíka a všetkých funkcionalít s ním spojených (pridávanie/odoberanie produktov atď.) sme narazili na problém pri aktualizovaní údajov o košíku (hlavne pri aktualizovaní celkového súčtu). Pôvodne sme totižto zvolili prístup načítania stránky po každej zmene v košíku. Tento prístup bol neprimerane neefektívny a pomalý. Na vyriešenie problému sme použili knižnicu "livewire", ktorá poskytuje pohodlný spôsob na načítanie len zmenených elementov stránky. Výhodou je taktiež to, že knižnica používa jazyk php bez potreby použitia javascriptu. Návod na inštaláciu knižnice: https://laravel-livewire.com/docs/2.x/making-components

Programovacie prostredie

V. Babinský - VS code/ PHPstorm

T. Tisovský - VS code

Opis implementácie

Prihlásenie

Hlavná funkcionalita prihlasovania a registrácie bola vytvorená pomocou Laravel Breeze. Pri pri plihlasovaní sa zavolá funkcia store() v AuthenticatedSessionController.php. Vstupom tejto funkcie sú prihlasovacie údaje. V tejto funkcií sa najprv zavolá funkcia authenticate(), ktorá slúži na overenie prihlasovacích údajov. Po úspešnej autentifikácii metóda pokračuje zavolaním session()->regenerate(), čo je bezpečnostné opatrenie, ktoré pomáha predchádzať útokom typu session fixation. Tento krok efektívne regeneruje session ID, čím zaisťuje, že akékoľvek staré session ID nemôže byť zneužité. Funkcia redirectBasedOnRole() slúži na presmerovanie buď na hlavnu stranku alebo na adminPanel, podľa role používatela.

```php
/**
 * Handle an incoming authentication request.
 */
public function store(LoginRequest $request): RedirectResponse
{
    $request->authenticate();

    $request->session()->regenerate();

    return $this->redirectBasedOnRole();  // Redirect user based on their role
}
```

```php
/**
 * Attempt to authenticate the request's credentials.
 *
 * @throws \Illuminate\Validation\ValidationException
 */
public function authenticate(): void
{
    $this->ensureIsNotRateLimited();

    if (! Auth::attempt($this->only('email','password'), $this->boolean('remember'))) {
        RateLimiter::hit($this->throttleKey());

        throw ValidationException::withMessages([
            'email' => trans('auth.failed'),
        ]);
    }

    RateLimiter::clear($this->throttleKey());
}
```

V tejto funkcií sa realizuje autentifikácia používateľa. Metóda ensureIsNotRateLimited() overí počet pokus prihlásenia a keď ich je viac ako 5, nastaví sa timeout na 60 sekúnd.

```php
/**
* Redirect the user based on their role.
*
* @return \Illuminate\Http\RedirectResponse
*/
protected function redirectBasedOnRole(): RedirectResponse
{
    if (Auth::user()->role === 'admin') {
        $categoryName = 'Mobilné telefóny';  // Set the default category name
        return redirect()->route('admin.products.show', ['categoryName' => $categoryName]);
    }
    return redirect()->route('dashboard');  // Redirect to the dashboard route
}
```

V tejto metóde sa realizuje presmerovanie na hlavnú stránku alebo na stránku administrátorského rozhrania podľa role používateľa. Pre adminPanel sa nastaví predvolená kategória produktov, ktoré sa zobrazia ako zoznam produktov na hlavnej stránke adminPanelu.

Zmena množstva pre daný produkt

Produkt v košíku je zobrazený pomocou livewire komponentu “live-cart-item.blade.php”. Logika je vykonávaná pomocou funkcie “changeQuantity” v kontroleri “LiveCartItem.php”. Táto funkcia je zavolaná 500ms potom ako používateľ zmení počet v elemente input. Následne sa z databázy získa počet daného produktu na sklade. Používateľ nemôže tento počet prekročiť.

```php
// ziskanie kvantity produktu
$product = Product::where('id', $this->product_id)->get()->first();

// pocet dostupnych produktov
$available_quantity = $product->stock_quantity;

$quantity_changed = false;

//zmena poctu produktu v kosiku
$quantity_diff = 0;

if (session()->has('cart')) {
    // ziskanie kosika z relacie
    $current_cart = session()->get('cart');


    // kontrola ci sa neprekrocil pocet produktov "na sklade"
    // ak sa prekrocil tak sa nastavi na max pocet produktov "na sklade"
    if ($new_quantity > $available_quantity) {
        $new_quantity = $available_quantity;
    }
```

Následne sa košík získa z session() a zmení sa pre daný produkt jeho kvantita. Ak je používateľ prihlásený tak sa zmena vykoná aj v databáze, konkrétne v tabuľke ”cart_items”.

```php
$logged_user = auth()->user();

if ($logged_user!= null){
    // zmena kvantity v databaze, konkretne v tabulke cart items
    //ziskanie zaznamu o produkte v kosiku
    $cart_item = CartItem::where('product_id',$this->product_id)
        ->where('cart_id',$logged_user->cart_id)->first();

    // zmena kvantity produktu v kosiku
    $cart_item->quantity = $new_quantity;

    //ulozenie zmien
    $cart_item->save();
}
```

Potom sa pošle event ” totalSumChanged ", ktorý zmení súčet zobrazený na nav-bare.

Ak bol počet kusov produktu nastavený na hodnotu 0, tak sa produkt odstráni (zo session() ak je používateľ prihlásený tak aj z databázy) a nastavením premennej “visibility” sa skryje, takže ho používateľ už nebude vidieť.

Ak sa košík vyprázdni, tak sa vymaže zo session resp. databázy.

Vyhľadávanie

Hlavná funkcionalita vyhľadávania sa nachádza v ProductController.php v metóde search(). V tejto metóde sa spracováva požiadavka na vyhľadávanie spolu s filtrovaním a preusporiadavaním produktov.

```php
public function search(Request $request)
{

    // Start the query
    $query = Product::query();
```

V metóde sa najprv vytvorí query, kotrá reprezentuje SQL dopyt nad tabuľkou products.

```php
if ($request->has('search')) {
    if(empty($request->input('search'))){
        $search = null;
    }else{
        $search = $request->input('search');
        $query->where(function ($q) use ($search) {
            $q->whereRaw('LOWER(name) LIKE ?', ["%{$search}%"])
              ->orWhereRaw('LOWER(description) LIKE ?', ["%{$search}%"]);
        });
    }
}
```

Ak sa v požiadavke nachádza atribút search, a tento atribút nie je prázdny, ku query sa pridá časť dopytu, vďaka ktorej sa z produktov vyberú len tie, ktorých hodnota products.name alebo products.description obsahuje retazec $search. Metóda LOWER() zabezpečuje aby vyhladávanie nebralo ohlad na veľké a malé písmená. V metóde search() sa k dopytu pridávajú ďalšie časti, ktoré zabezpečujú kombináciu vyhľadávania, filtrovania a zoradovania produktov.

Formulár pre odosielanie požiadavky pre vyhľadávanie sa nachádza v views/partials/navbar-main-sb.blade.php a vyzerá nasledovne:

```
<div class="col-md-4 col-12 order-3 order-md-2 pb-4">
    <form id="search-form" action="/search" method="GET">
        @csrf
        <div class="input-group p-4">
        <input type="text" class="form-control search-bar"
            aria-label="search-input"
            id="search-input"
            name="search">
            <button class="btn btn-outline-secondary" type="submit" id="search-bar-icon-btn">
                <i class="bi bi-search"></i>
            </button>
            <input type="hidden" name="categoryID" value="{{ $categoryID ?? '' }}">

        </div>
    </form>

</div>
```

Pridanie produktu do košíka

Používateľ má dve možnosti ako pridať produkt do košíka nerátajúc zmenu kvantity v košíku, ktorá je opísaná vyššie.

Prvou možnosťou je stlačenie tlačidla s ikonkou nákupného košíka, ktoré je pri každom produkte, buď na hlavnej stránke alebo pri vyhľadávaní/filtrovaní. Produkty sú reprezentované pomocou komponentu “product-card.blade.php”. Stlačením tohto tlačidla (livewire komponent “add-to-cart.blade.php”) sa produkt pridá do košíka (ak sa tam daný produkt už nachádza, tak sa pripočíta 1 kus ku kvantite v košíku). Toto je realizované zavolaním funkcie “addToCart” (súbor “AddToCart.php”).

Druhou možnosťou pridania produktu do košíka je input element na stránke detailu produktu. Tento element je súčasťou livewire komponentu “add-to-cart-quantity.blade.php”. Používateľ môže na zadanie počtu použiť tlačidlá s označením “+” resp. “-”, alebo manuálnym napísaním čísla a stlačením tlačidla “Vložiť do košíka”. (Ak používateľ zadá hodnotu, ktorá nie je validná napr. nejde o číslo, tak sa pridá automaticky 1 kus (ak je to možné)).

Keď používateľ stlačí tlačidlo “Vložiť do košíka”, vykoná sa funkcia “submit” (súbor “AddToCartQuantity.php”).

Funkcie “addToCart” a “submit” sú vo svojej podstate rovnaké, preto opíšeme len funkciu “submit”.

1. Na začiatku prebehne validácia vstupu, či naozaj ide o celé číslo.

```php
// vpodstate sanitacia vstupu
// ak zadal pouzivatel string tak prednastavena hodnota bude 1
$this->quantity = (int)$this->quantity;
if (!filter_var($this->quantity, filter: FILTER_VALIDATE_INT)) {
    $this->quantity = 1;
}
```

2. Následne sa zistí z databázy počet kusov produktu na sklade a získajú sa údaje o používateľovi (ak je prihlásený).
3. Potom je na rade kontrola či už v session jestvuje košík (ak nie je to to isté len sa vytvorí košík v session a v prípade prihláseného používateľa aj v databáze). Košík je v session reprezentovaný ako pole kde kľúče sú id produktov a hodnoty sú polia s uloženou kvantitou, menom obrázka, cenou a menom produktu).
4. Potom sa skontroluje, či by sa neprekročil počet produktov na sklade, (ak áno pridá sa maximum, čo sa môže pridať). Vzápätí sa zmení počet v košíku v session a ak je používateľ prihlásený tak aj v databáze pre konkrétny záznam v tabuľke “cart_items”.

```php
if (session()->has('cart')) {
    // ziskanie kosika z relacie
    $current_cart = session()->get('cart');

    // kontrola ci uz je produkt v kosiku
    if (array_key_exists($this->product_id, $current_cart)) {

        // kontrola ci sa neprekrocil pocet produktov "na sklade"
        // ak bol prekroceny tak sa prida max produktov kolko sa moze pridat
        if ($current_cart[$this->product_id]['quantity'] + $this->quantity > $available_quantity) {
            $this->quantity = $available_quantity - $current_cart[$this->product_id]['quantity'];
        }
        //zmena kvantity pre zaznam v kosiku
        $current_cart[$this->product_id] = ['quantity' => $this->quantity + $current_cart[$this->product_id]['quantity'],
            'image' => $this->image, 'price' => $this->price, 'name' => $this->name];
        $product_added = true;

        if ($logged_user != null) {
            // zmena kvantity v databaze, konkretne v tabulke cart items

            // ziskanie zaznamu o produkte v kosiku
            $cart_item = CartItem::where('product_id', $this->product_id)
                ->where('cart_id', $logged_user->cart_id)->first();

            // zmena kvantity produktu v kosiku
            $cart_item->quantity = $this->quantity + $current_cart[$this->product_id]['quantity'];

            //ulozenie zmien
            $cart_item->save();
        }
```

Nasledujúci obrázok, ilustruje vytvorenie košíka a záznamov o produktoch v ňom, v prípade, že košík nejestvoval a používateľ je prihlásený (Viď. Bod 3).

```php
if ($logged_user != null) {

    //vytvorenie zaznamu v tabulke carts
    $new_user_cart = new Cart();
    $new_user_cart->save();

    //pridanie kosika pouzivatelovi
    $logged_user->cart_id = $new_user_cart->id;
    $logged_user->save();

    // vytvorenie zaznamu v databaze, konkretne v tabulke cart_items
    $cart_item = new CartItem();
    $cart_item->cart_id = $logged_user->cart_id;
    $cart_item->product_id = $this->product_id;
    $cart_item->quantity = $this->quantity;

    //ulozenie noveho zaznamu
    $cart_item->save();
}
```

Stránkovanie

Stránkovanie bolo vyriešené pomocou vstavanej funkcie Laravel paginate():

```php
// Pagination
$products = $query->paginate(24);
```

Táto funkcia berie ako parameter počet produktov, ktoré sa zobrazia na jednej stránke.

Vo view sa potom stránkovanie použije nasledovne:

```php
<div class="pagination justify-content-center">
    {{ $products->links() }}
</div>
```

Pre upravenie vzhľadu lišty s číslami stránok som upravil šablóny stránkovania v Laraveli. Tieto šablony som stiahol pomocou príkazu : `php artisan vendor:publish --tag=laravel-pagination`. Tento príkaz skopíruje šablóny stránkovania do resources/views/vendor/pagination.

Základné filtrovanie

Filtrovanie je riešené už v spomínanej metóde search() v ProductController. V tejto metóde sa realizuje filtrovanie podľa kategórie, minimálnej a maximálnej ceny, podľa farby a značky produktu.

```php
// Filter by category
if ($request->has('categoryID')) {
    if(empty($request->input('categoryID'))){
        $categoryID = null;
    }else{
        $categoryID = $request->input('categoryID');
        $query->where('category_id', $categoryID);
        $category_name = Category::where('id', $categoryID)->first()->category_name;
    }
}
```

```php
$minimumPrice = $request->input('minimum_price');
if ($minimumPrice !== null) {
    $minimumPrice = (float) $minimumPrice;  // Cast to float to ensure numeric comparison
    $query->where('price', '>=', $minimumPrice);
}
```

```php
$maximumPrice = $request->input('maximum_price');
if ($maximumPrice !== null) {
    $maximumPrice = (float) $maximumPrice;  // Cast to float to ensure numeric comparison
    $query->where('price', '<=', $maximumPrice);
}
```

```php
// Filter by brands if provided
if ($request->has('brands')) {
    $brands = $request->input('brands');  // Brands is an array
    $query->whereHas('parameters', function ($q) use ($brands) {
        $q->whereIn('value', $brands);
    });
}
```

```php
// Filter by colors if provided
if ($request->has('colors')) {
    $colors = $request->input('colors');  // Colors is an array
    $query->whereHas('parameters', function ($q) use ($colors) {
        $q->whereIn('value', $colors);
    });
}
```

Vo View sa všetky filtre nachádzaju v jednom formuláry filter-form v searchResults.blade.php.

Zoradovanie

Takisto v rámci metódy search() bolo riešené zoraďovanie, ak sa v požiadavke nachádzal atribút sort, ku query sa pridala časť orderBy, ktorá slúži na zoraďovanie. Podľa toho či hodnota sort je price_asc alebo price_desc realizuje sa zoradenie od najnižšej ceny po najvyššiu respektíve opačne.

```php
if ($request->has('sort')) {
    $sort = $request->input('sort');
    if ($sort === 'price_asc') {
        $query->orderBy('price', 'asc');
    } elseif ($sort === 'price_desc') {
        $query->orderBy('price', 'desc');
    } else {{
        $query->orderBy('price', 'asc');
    }
    }
}
```

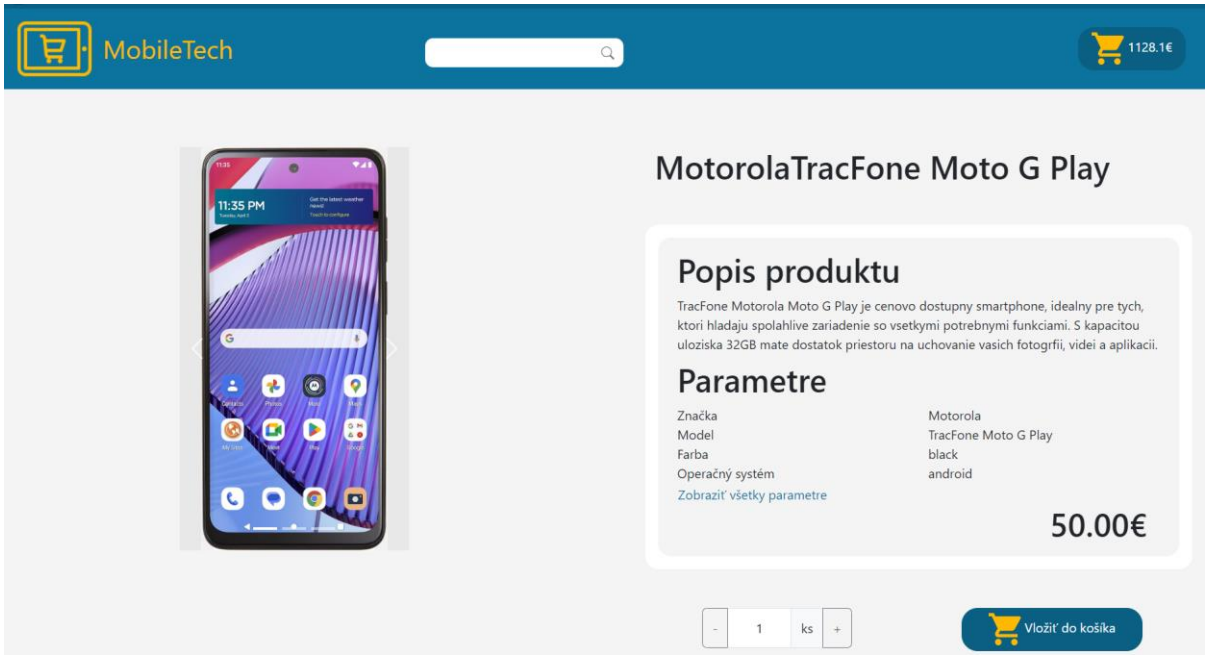
Vo view sa input pre sort realizuje pomocou hidden input.

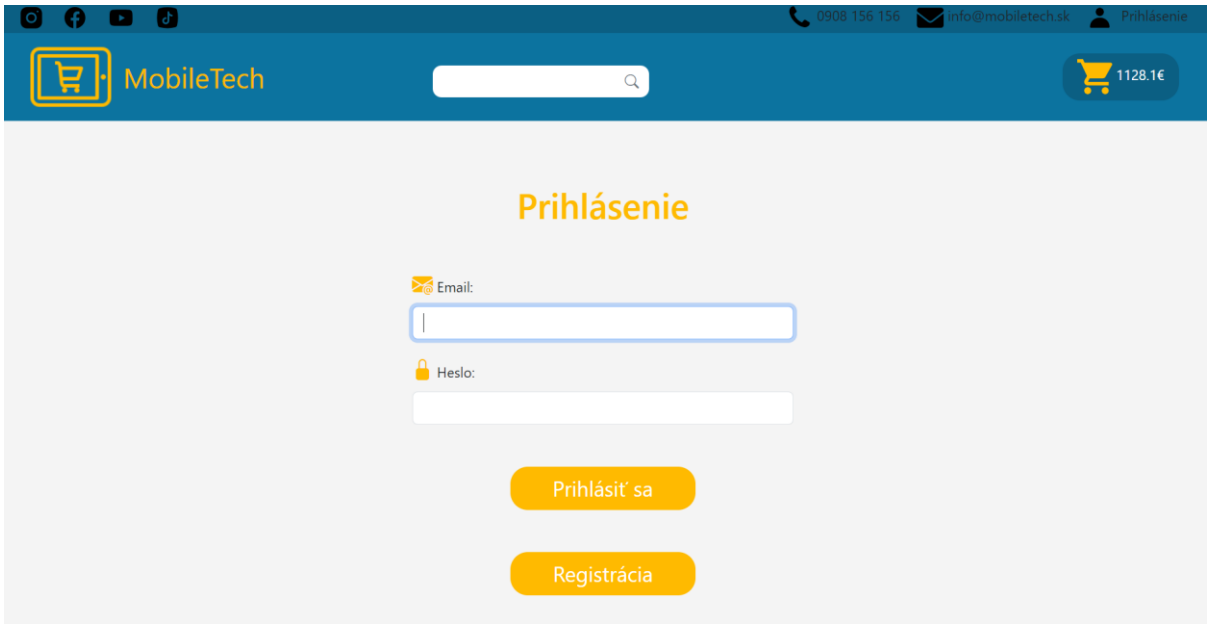
```html
<div class="col-sm-3 col-6 text-center">
  <button type="button" class="btn sort-btn" data-sort="price_asc">Od najlacnejšieho</button>
</div>
<div class="col-sm-3 col-6 text-center">
  <button type="button" class="btn sort-btn" data-sort="price_desc">Od najdrahšieho</button>
</div>
<div class="col-sm-3 col-6 text-center">
  <button type="button" class="btn" data-sort="newest">Najnovšie</button>
</div>
<div class="col-sm-3 col-6 text-center">
  <button type="button" class="btn" data-sort="discount">Výška zľavy</button>
</div>
<input type="hidden" id="sort-select" name="sort" value="{{ $sort ?? '' }}">
```

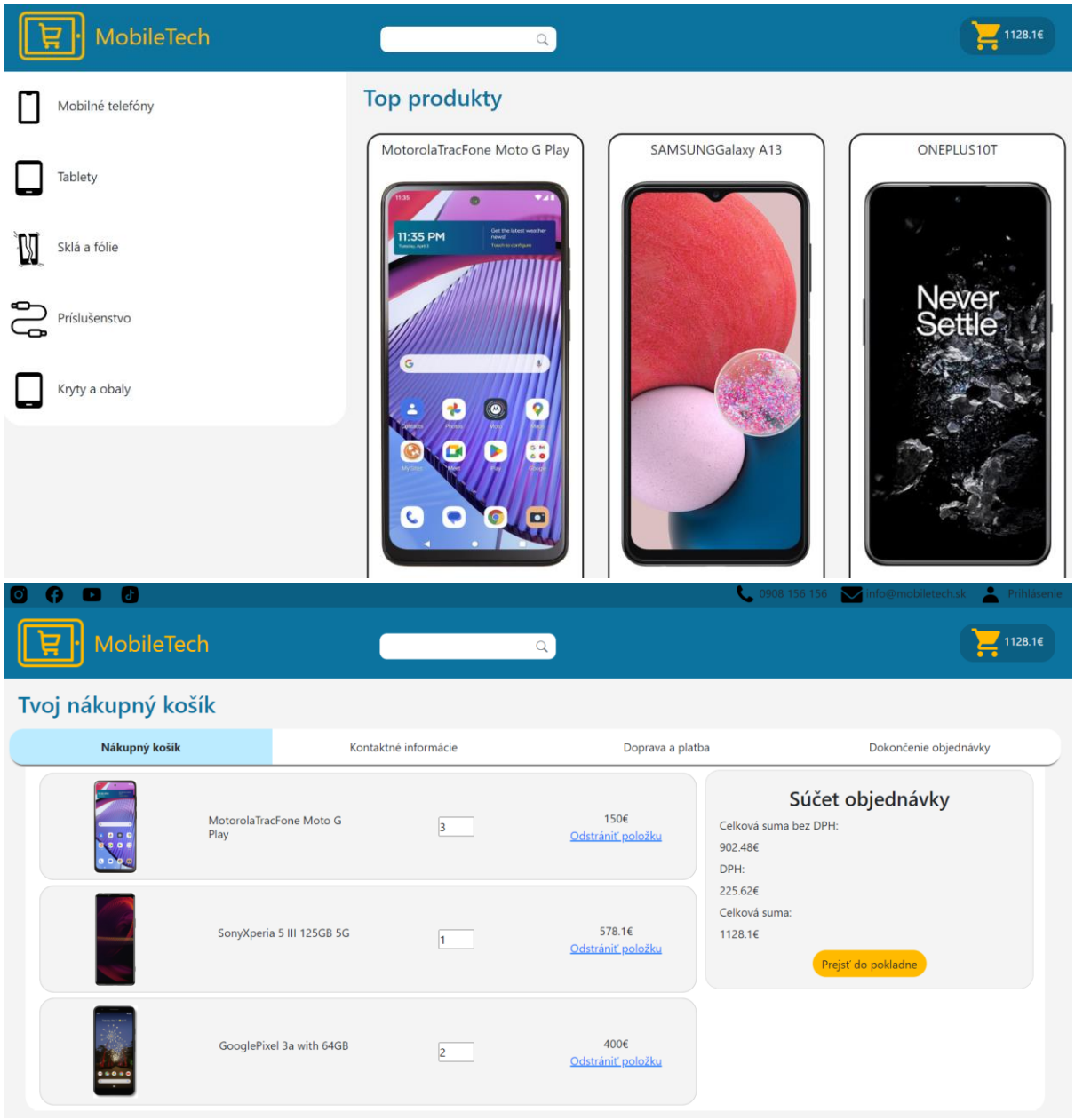
Po kliknutí na niektorý zo buttonov s triedou sort-btn sa nastaví hodnota tohto inputu pomocou nasledovného JavaScript kódu:

```javascript
$('.sort-btn').on('click', function() {
    var sort = $(this).data('sort');

    $('#sort-select').val(sort);

    $('#filter-form').submit();
});
```

Snímky obrazoviek







Inštalácia a spustenie

1. Spustite nasledujúci príkaz na inštaláciu závislostí PHP uvedených v súbore composer.json:

   ```
   composer install
   ```

2. Skopírujte súbor .env.example na nový súbor .env, ktorý bude obsahovať vaše lokálne nastavenia:

```
copy .env.example .env # Pre Windows
```

```
cp .env.example .env # Pre Unix/Linux/Mac
```

3. Upravte .env súbor, aby zodpovedal vašim databázovým nastaveniam:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=your_database_name
DB_USERNAME=your_database_username
DB_PASSWORD=your_database_password
```

4. Na nastavenie databázových tabuliek a vykonanie počiatočného nastavenia spustite:

```
php artisan migrate
```

5. Pre spustenie seeders použite nasledovný príkaz:

```
php artisan db:seed
```