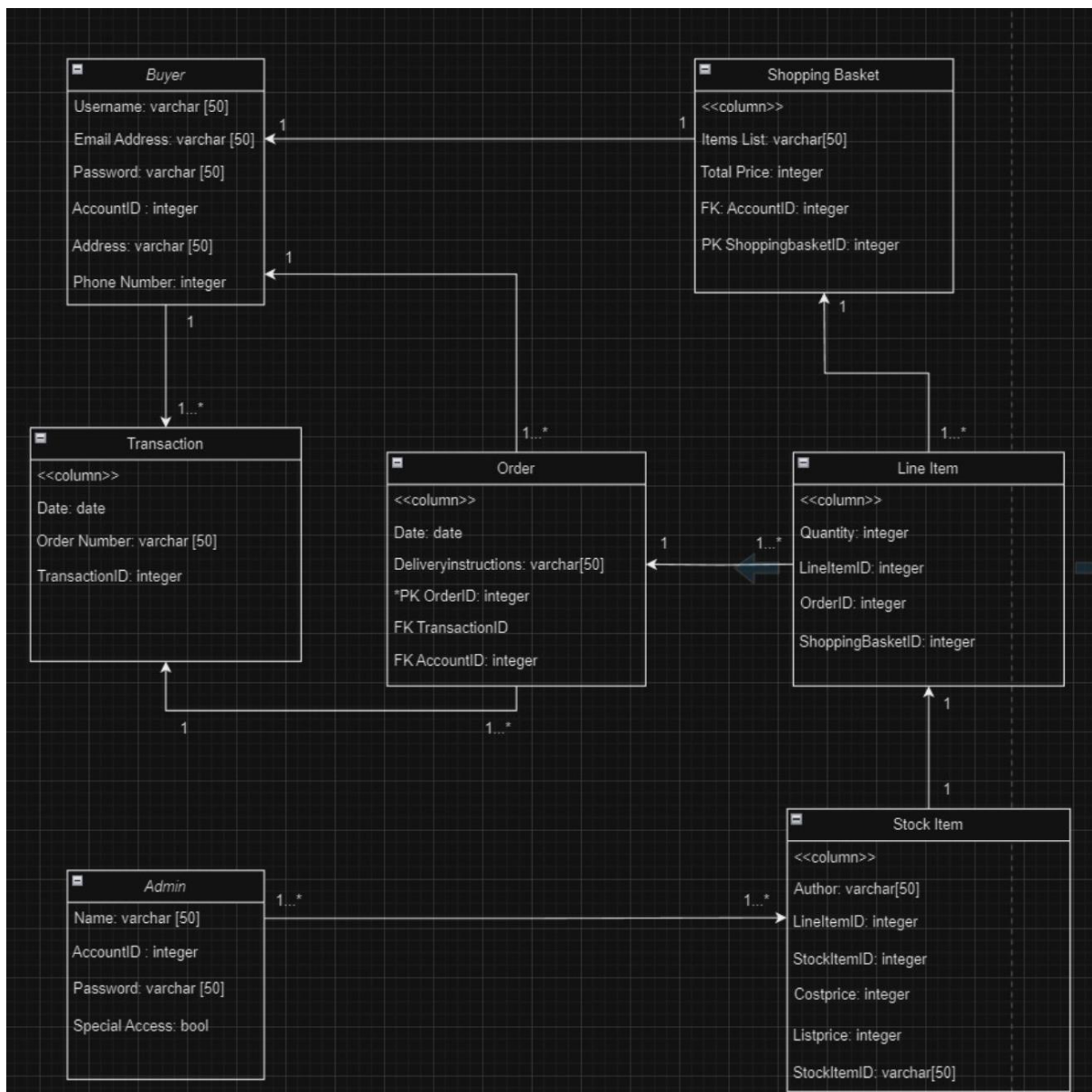


MuscleHustle

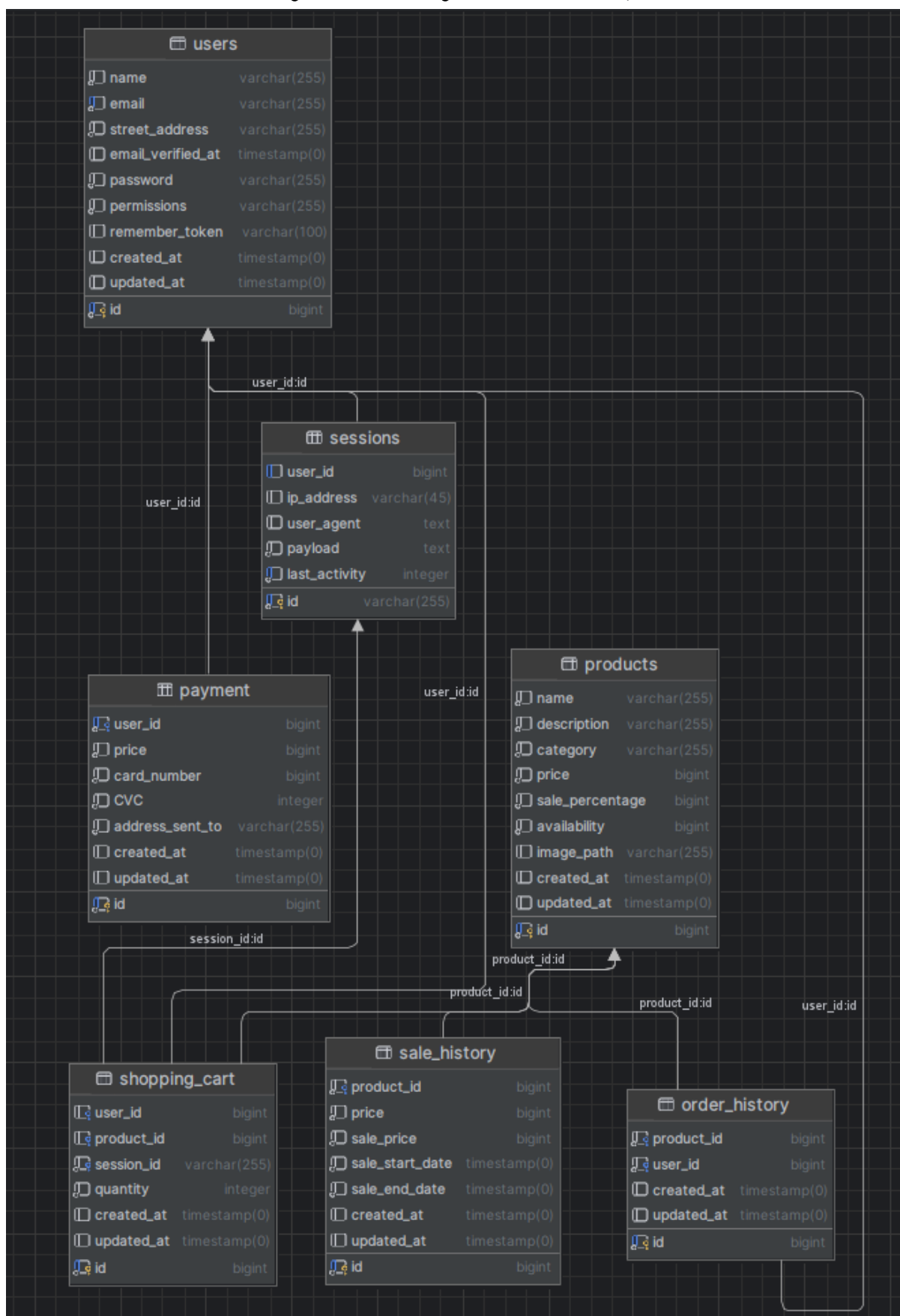
Zadanie

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.

UML diagram (1.fáza)



Relačný dátový model (2.fáza)



- V prvej fáze sme nevedeli, ako bude vyzerat' naša databáza, preto to bol len prvotný nápad. Nakoniec náš dátový model v 2. fáze vyzerá úplne inak.
- V prvej fáze sme mali tabuľky Buyer a Admin, avšak teraz sme spravili tabuľku USER, v ktorej sú všetci používatelia, a povolenia (permissions) sa nachádzajú ako údaj v tejto tabuľke (USER/ADMIN).
- Tabuľky STOCK ITEM a LINE ITEM z prvej fázy sme spojili do tabuľky PRODUCTS, v ktorej sa nachádzajú všetky potrebné informácie o ňom.
- TRANSACTION sme zmenili na PAYMENT, v ktorom sa nachádzajú informácie o kreditnej karte, cenu, ID transakcie atď. Info o kreditnej karte sa ukladá iba, ak užívateľ klikne "remember my card".
- SHOPPING BASKET sme zmenili na SHOPPING CART, a pridali sme "quantity", teda počet, koľko kusov z daného produktu máme v košíku.

Programátorské prostredie

Pre vývoj webových aplikácií využívame kombináciu Visual Studio Code (VS Code) a XAMPP. Jeho flexibilita, rozšíriteľnosť pomocou extensions robia z VS Code obľúbenú voľbu pre vývojárov. XAMPP je jednoduchý spôsob, ako lokálne spustiť webový server a databázu, čo umožňuje rýchle a jednoduché testovanie a vývoj webových aplikácií. Táto kombinácia poskytuje efektívne a pohodlné prostredie pre vývoj a testovanie aplikácií založených na Laravel frameworku.

Návrhové rozhodnutia

V našom projekte sme využili externú knižnicu noUiSlider. Využívame ju pri zobrazovaní filtra podľa ceny v katalógu.

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/noUiSlider/14.6.3/nouislider.min.js"></script>
```

Implementácia noUiSlider poskytuje užívateľom jednoduchý spôsob nastavenia cenového intervalu pomocou intuitívneho posuvníka, ktorý umožňuje filtrovanie produktov podľa ceny.

Implementácia

Zmena množstva pre daný produkt

Hlavná funkcia využívaná v zmene počtu produktov v košíku, je funkcia updateCart v shopping_cart.js

```
function updateCart() {
  // Fetch all quantity input fields
  const quantityInputs = document.querySelectorAll('.quantity-input');
  const productQuantities = {};

  // Iterate over each quantity input field
```

```

quantityInputs.forEach(input => {
  const productId = input.dataset.productId;
  const quantity = parseInt(input.value);
  productQuantities[productId] = quantity;
});

// Send an AJAX request to update cart quantities
fetch('/update-cart', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-token"]').getAttribute('content')
  },
  body: JSON.stringify(productQuantities)
})
.then(response => {
  console.log('Response:', response);
  if (response.ok) {
    // Cart updated successfully
    // Reload the page to reflect the changes
    location.reload();
  } else {
    // Cart update failed
    console.log('Failed to update cart');
  }
})
.catch(error => {
  console.error('Error:', error);
  console.log('An error occurred while updating cart');
});
}

```

Táto funkcia sa spúšťa pri stlačení tlačidla update cart na stránke košíka. Funkcia vyextrahuje z vstupných polí pri produktoch ich chcený počet, ktorý potom nastaví v cykle každému produktu. Potom posielam tzv. AJAX požiadavku serveru, ktorý potom aktualizuje databázu v CartControlleri, ktorý potom pošle odpoveď. Tam využívam funkciu update cart:

```

public function updateCart(Request $request)
{
  // Log the request data
  Log::info('Request Data:', $request->all());

  // Retrieve the authenticated user
  $user = auth()->user();

  // Retrieve the updated product quantities from the request
  $productQuantities = $request->all();

  // Loop through each product ID and quantity and update or delete the corresponding cart item
  foreach ($productQuantities as $productId => $quantity) {

```

```

// Find the cart item associated with the user and the product ID
$cartItem = Cart::where('user_id', $user->id)
    ->where('product_id', $productId)
    ->first();

if ($cartItem) {
    if ($quantity == 0) {
        // If quantity is 0, delete the cart item
        $cartItem->delete();
    } else {
        // Update the quantity of the cart item
        $cartItem->quantity = $quantity;
        $cartItem->save();
    }
}
}
}

```

UpdateCart() zistí user ID (predtým pošle informácie o požiadavku do Logu), a podľa ID nájde všetky produkty v košíku. Každému produktu potom zmení príslušnú kvantitu a záznamy uloží.

Prihlásenie

Na prihlásenie využívam AuthenticatedSessionController. Funkcia create() zobrazuje login stránku.

```

<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Http\Requests\Auth\LoginRequest;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\View\View;

class AuthenticatedSessionController extends Controller
{
    /**
     * Display the login view.
     */
    public function create(): View
    {
        return view('auth.login');
    }

    /**
     * Handle an incoming authentication request.
     */
}

```

```

*/
public function store(LoginRequest $request): RedirectResponse
{
    $request->authenticate();

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

    return redirect()->intended(route('dashboard', absolute: false));
}

/**
 * Destroy an authenticated session.
 */
public function destroy(Request $request): RedirectResponse
{
    Auth::guard('web')->logout();

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

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

    return redirect('/');
}

// In your AuthController
public function checkAuthStatus()
{
    // Check if the user is authenticated
    if (auth()->check()) {
        return response()->json(['loggedIn' => true]);
    } else {
        return response()->json(['loggedIn' => false]);
    }
}
}

```

Funkcia store() využíva authenticate() ktorá spracováva autentifikačnú logiku na základe prihlasovacích údajov poslaných v požiadavke. Potom aktualizuje session token a užívateľa prehodí na dashboard.

Vyhľadávanie a filtrovanie

Vyhľadávanie podľa mena v katalógu produktov zaručuje javascript funkcia createFilteredProducts() nachádzajúca sa v catalog.js, ktorá filtruje podľa zadaných požiadaviek

```

function createFilteredProducts() {
    let filteredProducts = [];
    const minPriceValue = parseFloat(minPrice.textContent.slice(1)); // Extract the minimum price value
    const maxPriceValue = parseFloat(maxPrice.textContent.slice(1)); // Extract the maximum price value

```

```

const saleCheckbox = document.getElementById('ch1'); // Get the sale checkbox element
const selectedCategory = categoryFilter.value; // Get the selected category from the dropdown menu
const searchFilter = document.getElementById('search-filter').value.trim().toLowerCase(); // Get the search filter input value

productsData.forEach((product) => {
  // Calculate the price to be used for filtering
  let priceToUse = product.price; // Default to normal price
  if (product['sale_percentage'] > 0) {
    // If the product is on sale, calculate the discounted price
    const discountedPrice = (product.price - (product.price * product['sale_percentage'] / 100));
    priceToUse = discountedPrice;
  }

  // Check if the sale checkbox is checked and if the product has a sale greater than 0
  // Check if the product's category matches the selected category or if no category is selected
  // Also, check if the product name contains the search filter string
  if ((!saleCheckbox.checked || (saleCheckbox.checked && product['sale_percentage'] > 0)) &&
    priceToUse >= minPriceValue && priceToUse <= maxPriceValue &&
    (selectedCategory === '' || product.category === selectedCategory) &&
    (product.name.toLowerCase().includes(searchFilter))) {
    filteredProducts.push(product);
  }
});

return filteredProducts;
}

```

V tejto funkcii zistíme cenový interval v ktorom sa majú produkty pohybovať, zisťuje či hľadáme iba zľavnené produkty a berie text z vyhľadávacieho vstupu. Potom prechádzame produktami, zistíme či nie je zľavnený, ak hej počítame s touto cenou. Potom skontrolujeme, či je stlačené tlačidlo zobrazujúce zľavnené produkty, ak nie alebo ak je a produkt je v zľave a zároveň sa nachádza v cenovom intervale a obsahuje reťazec vyextrahovaný z vyhľadávacieho vstupu a produkt je vybratej kategórie produkt pridá do poľa filtrovaných produktov. Toto pole funkcia vracia.

Pridanie produktu do košíka

O pridávanie produktu do košíka sa stará funkcia v CartControllery addToCart().

```

public function addToCart($productId)
{
  $cartItem = new Cart();
  // Check if the user is authenticated
  if (auth()->check()) {
    // If the user is authenticated, use their user ID
    $userId = auth()->id();
    $sessionId = null; // No need to store session ID for authenticated users
    $cartItem->user_id = $userId;
  } else {

```

```

    // If the user is not authenticated, use the session ID
    $userId = null; // No need to store user ID for unauthenticated users
    $sessionId = Session::getId();
    $cartItem->session_id = $sessionId;
}

// Create a new cart item
$cartItem->product_id = $productId;
// Add any other relevant fields such as quantity, etc.
$cartItem->save();

// Redirect back to the previous page
return back();
}

```

Táto funkcia skontroluje, či je užívateľ zaregistrovaný alebo nie, ak áno, funkcia hodí do košíka produkt s user_id, ak nie tak session_id.

Stránkovanie

Stránkovanie je v podstate zahrnuté vo viacerých javascriptových funkciách
generateProductItems():

```

function generateProductItems() {
    // Clear products before generating new
    const productsContainer = document.querySelector('.products');
    productsContainer.innerHTML = ""; // Clear the inner HTML of the products container

    // Filter products by price interval and then sort it
    let filteredProducts = createFilteredProducts();
    filteredProducts = sortProducts(filteredProducts);

    // Calculate the starting and ending index for the current page
    const startIndex = (currentPage - 1) * productsPerPage;
    const endIndex = Math.min(startIndex + productsPerPage, productsData.length);

    // Generate product items for the current page
    for (let i = startIndex; i < endIndex; i++) {
        if (filteredProducts.length > i) {
            const product = filteredProducts[i];
            const productItem = document.createElement('div');
            productItem.classList.add('product');

            let priceDisplay = `${product.price}$`; // Default price display

            // If the product has a sale, calculate the discounted price and display it along with the original price
            if (product['sale_percentage'] > 0) {
                const discountedPrice = (product.price - (product.price * product['sale_percentage'] / 100)).toFixed(2);

```



```

        priceDisplay = `${product.price} ${discountedPrice}`;
        // Add the sale.png image to the product item
        const saleImage = document.createElement('img');
        saleImage.src = 'icons/sale_icon.png';
        saleImage.alt = 'Sale';
        saleImage.classList.add('sale-image');
        productItem.appendChild(saleImage);
    }

    // Include product name and price display in the product item
    productItem.innerHTML += `
        <a onclick="openProductPage('${product.name}', '${product.image_path}', ${product.price})">
            
        </a>
        <p>${product.name}</p>
        <p>${priceDisplay}</p>
        <button onclick="openProductPage('${product.name}', '${product.image_path}',
${product.price})">Info</button>
    `;
    productsContainer.appendChild(productItem); // Append the product item to the products container
}
}
updatePageCounter(filteredProducts); // Update the page counter after generating product items
}

```

Táto funkcia má na starosti zobrazovanie produktov na stránke. Filtrované produkty zoradí podľa požiadaviek užívateľa (podľa ceny alebo abecedy). Funkcia začne produkovať produkty od startIndexu, ktorý je určený podľa aktuálnej stránky a generuje HTML kód pre produkt. Stránka je staticky nastavená aby produkovala 6 produktov na jednu stranu. Na zobrazenie ďalších stránok nám slúžia navigačné tlačidlá, ktoré spúšťajú funkcie showPreviousProducts() a showNextProducts()

```

function showPreviousProducts() {
    currentPage = Math.max(currentPage - 1, 1);
    generateProductItems(); // Regenerate product items for the current page
}

function showNextProducts() {
    currentPage = Math.min(currentPage + 1, Math.ceil(createFilteredProducts().length / productsPerPage));
    generateProductItems(); // Regenerate product items for the current page
}

```

Funkcie prestavia currentPage a znovu spustia funkciu generovania produktov. V navigačnom paneli máme taktiež counter, ktorý nám ukazuje na ktorej strane produktov sme. Na to slúži funkcia updatePageCounter().

```

function updatePageCounter(filteredProducts) {
    const totalPages = Math.max(Math.ceil(filteredProducts.length / productsPerPage), 1);
    const pageCounter = document.getElementById('page-counter');
}

```

```
pageCounter.textContent = `Page ${currentPage} of ${totalPages}`;  
}
```

Tá vypočíta celkový počet strán podľa produktov a údaje vypíše.

Stránky

Úvodná stránka (main page)

[Log in](#) [Register](#)

MUSCLE HUSTLE

CATALOG

Created by: Sebastián Lener, Dominik Zafovič

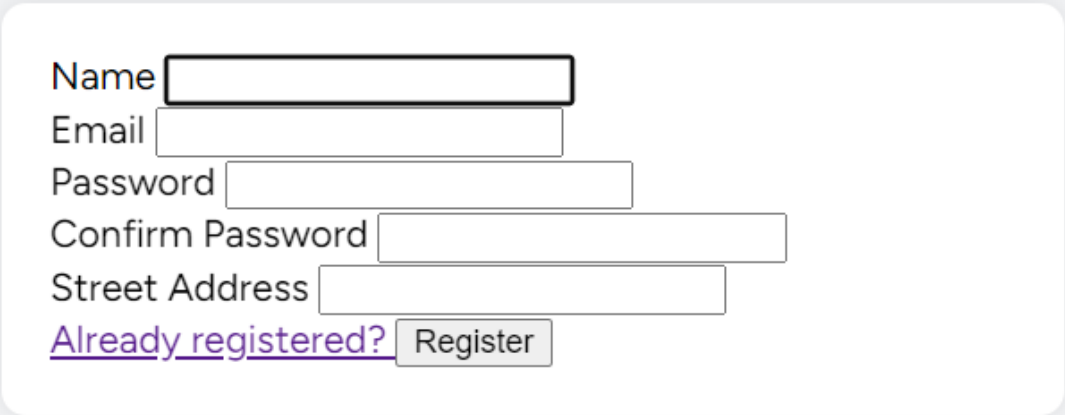
- V strede úvodnej stránky je veľký názov nášho e-shopu. V ľavom hornom rohu vidíme možnosti [Log in](#) (prihlásiť sa) a [Register](#) (vytvoriť nový účet). Pri kliknutí nás presmerujú na ich priradenú stránku.
- V dolnej časti obrazovky vidíme nápis “Created by...”, ktorý hovorí o autoroch tejto stránky (teda nás).
- Ako posledné sa tu nachádza tlačidlo “CATALOG”, ktoré nás presmeruje do katalógu s vecami, ktoré ponúkame

Log in

A light gray rounded rectangle containing a login form. It has two text input fields: one for 'Email' and one for 'Password'. Below the password field is a button labeled 'Log in'.

- Stránka na prihlásenie obsahuje textové polia, do ktorých je potrebné zadať svoj e-mail, ktorým sa užívateľ registroval a heslo.
- Následne klikne na Log In a prihlási ho.

Register

A light gray rounded rectangle containing a registration form. It has five text input fields: 'Name', 'Email', 'Password', 'Confirm Password', and 'Street Address'. Below the 'Confirm Password' field is a link labeled 'Already registered?' and a button labeled 'Register'.

- Stránka na registráciu nového účtu obsahuje textové polia, do ktorých je potrebné zadať svoje celé meno, e-mail adresu, heslo, potvrdenie hesla (zopakovať heslo) a ulicu, s ktorou bude spojený užívateľ (napr. pri objednávkach).
- Následne klikne na Register a vytvorí sa nový účet.
- Je tu aj “Already registered?” odkaz, ktorý presmeruje užívateľa na log in page.

Dashboard

You're logged in as ADMIN!

Dominik Zatovie
dominik.zatovic@gmail.com

[EDIT Profile](#)

[Log Out](#)

[MAIN PAGE](#)

- Dashboard stránka obsahuje meno a email užívateľa, ktorý je prihlásený. Vľavo hore je napísané “You're logged in!” pre užívateľa, a pre admina s dodatkom “as ADMIN”.
- V strede stránky je “EDIT Profile”, ktorý presmeruje užívateľa na stránku pre editovanie profilu
- Následne Log Out tlačidlo, pomocou ktorého sa vie užívateľ odhlásiť
- MAIN PAGE tlačidlo, ktoré nás presmeruje späť na úvodnú stránku

Edit profile

Profile

Profile Information

Update your account's profile information and email address.

Name

Email

Update Password

Ensure your account is using a long, random password to stay secure.

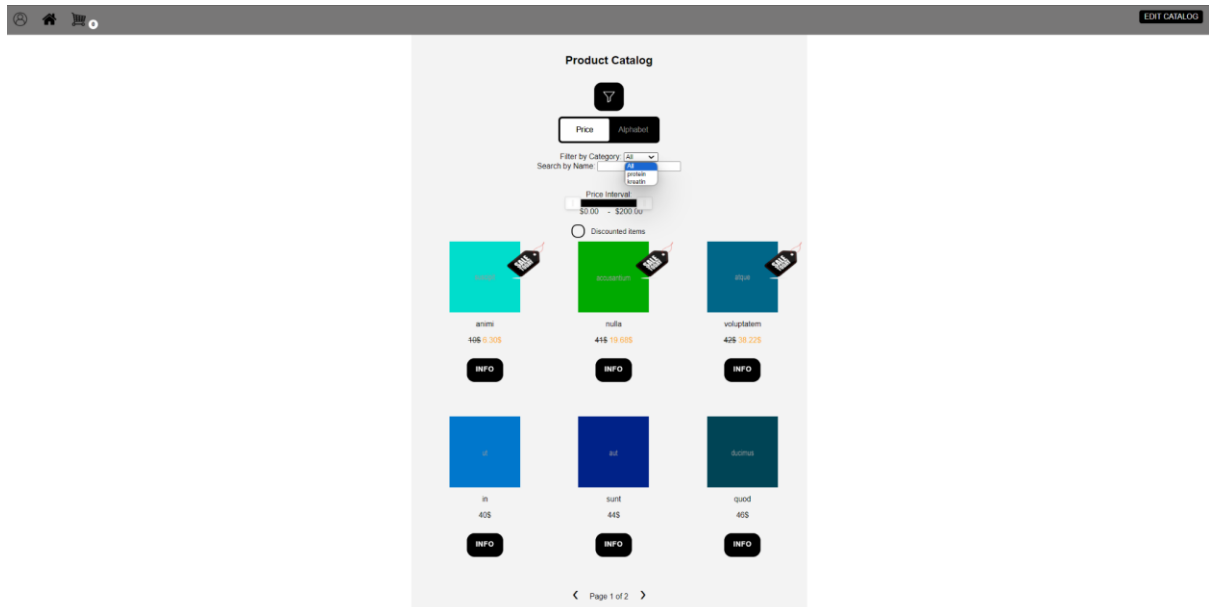
Current Password

New Password

Confirm Password

Na tejto stránke si vieme zmeniť údaje o našom účte, ako meno, e-mailovú adresu, heslo (treba zadať staré pre zmenu).

Catalog



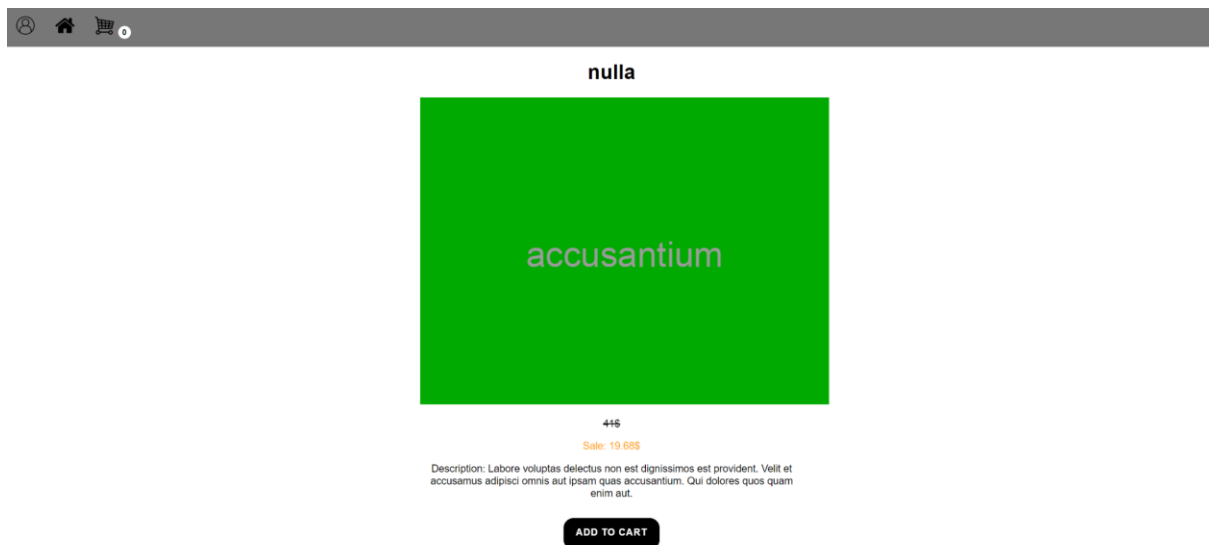
- Toto je stránka katalógu. V hornej lište sa nachádzajú 3 tlačidlá:
 - Main page
 - Catalog
 - Nákupný košík
- V strede obrazovky je katalóg s produktami a nad ním filter. Vieme filtrovať podľa ceny, abecedy, názvu produktu, ale aj cenového intervalu (po zadaní hodnot treba stlačiť tlačidlo filter (pod nápisom Product Catalog)
- Taktiež vieme filtrovať podľa kategórie produktu (stačí vybrať kategóriu)
- Je tu zahrnuté aj stránkovanie, teda produkty sa nachádzajú na viac ako jednej strane a vieme medzi nimi prepínať.
- Pod každým produktom je napísaná jeho cena (Ak je v zľave, je žltým a pri obrázku produktu je obrázok SALE TODAY) a tlačidlo INFO, ktoré nám otvorí detail produktu s možnosťou pridania do košíku.
- Ak sme prihlásený ako admin, tak v pravom hornom rohu je "EDIT CATALOG", ktorý nás presmeruje na stránku katalógu pre admina.

Catalog (Admin)





Toto je stránka pre katalóg admina. Obsahuje tie isté elementy ako obyčajný katalóg, ale navyše aj pridanie nového produktu a miesto INFO pri produktoch je EDIT, kde vieme zmeniť informácie pre konkrétny produkt.

Product details



Táto stránka sa objaví, keď užívateľ klikne na “INFO” pri produkte. Ukáže nám to podrobnosti o danom produkte, ako aj možnosť pridania ho do košíka.

Edit product



Edit Product Details

Product Name:	<input type="text" value="null"/>
Product Price:	<input type="text" value="41"/>
Product Sale:	<input type="text" value="52"/>
Product Image:	<input type="text" value="https://via.placeholder.com/"/>
Product Category:	<input type="text" value="icecream"/>
Product Description:	<input type="text" value="Labore voluptas delectus re"/>
Product Availability:	<input type="text" value="44"/>

SAVE CHANGES

DELETE PRODUCT

Táto stránka sa zobrazí, ak ADMIN klikne na EDIT pri produkte. Vidíme aktuálne data o produkte, ktoré vieme prepísať a následne kliknúť na SAVE CHANGES a uložia sa. Taktiež je tu aj “DELETE PRODUCT”, ktorým konkrétny produkt vymažeme z našej stránky a databázy.

Shopping cart

Shopping Cart

possimus
19\$
1

vel
76\$
Sale: 30.00\$
1
Update Cart

Total price: 49\$

Payment form:

1. ☐ via Card
2. ☒ Payment on delivery

Shipping:

1. ☒ MuscleHustle account address
2. ☐ Custom address

Checkout

Clear Cart

- Tu vidíme stránku nákupného košíku. Užívateľ vidí všetky produkty, ktoré si doň dal, a vie im zmeniť množstvo podľa potreby. Ak užívateľ zadá počet “0”, tak sa produkt vymaže z košíku. Pre uloženie zmien je potrebné stlačiť “Update cart”.
- Následne treba vybrať spôsob platby, (pri Card sa zobrazia ďalšie polia pre vpísanie čísla karty, CVC..)
- Ďalej treba vybrať kam sa majú produkty doručiť, buď na adresu,
- ktorá bola zadaná pri registrácii, alebo Custom address (iná).
- Následne treba stlačiť Checkout, čím ukončíme objednávku.
- Tlačidlo Clear Cart vymaže celý košík (ak treba)