Fullstack Academy

ECOMMERCE APLIKÁCIA

Dokumentácia

Adam Grík

ÚVOD

Táto aplikácia bola vytvorená na účel implementácie záverečného projektu v rámci FullStack Academy v spoločnosti PosAm, spol. s.r.o..

Samotná implementácia projektu sa týka e-shopu na outdoorové pomôcky. Zákazník má možnosť si vybrať z niekoľkých produktov, rozdelených do niekoľkých kategórií. Následne si môže tento vybraný tovar objednať, a to vykonaním objednávky. Po vykonaní všetkých potrebných procesov systém uloží danú objednávku do systému.

Projekt bol vytvorený na dvoch hlavných technológiách a to Spring Boot pre backend a Angular pre frontend. Databáza projektu je riešená cez MySQL. V projekte sú využité aj ďalšie vedľajšie technológie a frameworky ako napr. Bootstrap, FontAwesone či Lombok.

ANALÝZA

Požiadavky

- Zobrazenie produktov na hlavnej stránke podľa kategórií
- Zobrazenie produktov na hlavnej stránke na základe hľadaného kľúčového slova
- Zobrazenie detailu produktu
- Podpora stránkovania (voľba počtu zobrazených produktov)
- Pridanie produktu do košíka: z hlavnej stránky, z detailu produktu
- Zobrazenie obsahu košíka: pridávanie, odoberanie alebo zmazanie produktu v košíku
- Vyplnenie povinného formuláru pre objednávku: údaje o zákazníkovi, adresa doručenia atď.
- Vykonanie objednávky a uloženie objednávky do databázy

Use cases

Zobrazenie produktov podľa kategórie

Účel: Zobrazenie produktov podľa kategórie

Používateľ: zákazník

Výstup: Systém zobrazí produkty na základe kategórie ktorú si používateľ zvolí.

Postup:

1. Používateľ klikne na danú kategóriu, ktorej produkty chce zobraziť
2. Systém zobrazí produkty tej kategórie, na ktorú používateľ klikol.

Zobrazenie produktov podľa kľúčového slova

Účel: Zobrazenie produktov podľa zadaného kľúčového slova

Používateľ: zákazník

Výstup: Systém zobrazí produkty na základe kľúčového slova, ktoré používateľ zadal

Postup:

1. Používateľ napíše kľúčové slovo do textového poľa na hľadanie.
2. Systém zobrazí všetky produkty, ktoré majú vo svojom názve hľadané kľúčové slovo

Alternatívy:

1a. Používateľ zadá také kľúčové slovo ktoré sa nenachádza v názve v žiadnom produkte

2a. Systém zobrazí hlášku, že neboli nájdené žiadne produkty

Zobrazenie detailu produktu

Účel: Zobrazenie detailného náhľadu produktu (obrázok, názov, cena, popis)

Používateľ: zákazník

Výstup: Systém zobrazí detailný náhľad produktu, na ktorý používateľ klikne

Postup:

1. Používateľ klikne na názov alebo na obrázok konkrétneho produktu, ktorý chce zobraziť
2. Systém zobrazí detailný náhľad konkrétneho produktu, na ktorý požívateľ klikol.

Pridanie produktu do košíka

Účel: Pridanie konkrétneho produktu do košíka

Používateľ: zákazník

Výstup: Systém pridá do košíka konkrétny produkt a aktualizuje ikonu s nákupným košíkom

Postup:

1. Používateľ klikne na hlavnej stránke pod konkrétnym produktom na tlačidlo Pridať do košíka
2. Systém pridá daný produkt do košíka
3. Systém aktualizuje celkovú cenu košíka a celkový počet položiek v košíku

Alternatívy:

1a. Používateľ najprv klikne na detailný náhľad a v tomto detailnom náhľade klikne na Pridať do košíka.

Zobrazenie detailného náhľadu košíka

Účel: Zobrazenie detailného náhľadu košíka (produkty v košíku, počet produktov, cena atď.)

Používateľ: zákazník

Výstup: Systém zobrazí detailný náhľad obsahu čo sa nachádza v košíku

Postup:

1. Používateľ klikne na ikonu nákupného košíka
2. Systém zobrazí všetky položky v košíku, celkovú cenu všetkých položiek v košíku, celkový počet všetkých položiek v košíku

Alternatívy:

2a. V košíku sa nebude nachádzať žiaden produkt, systém vypíše hlášku, že košík je prázdny

Pridanie / odobranie produktu v košíku

Účel: Zvýšenie alebo zníženie počtu konkrétnej položky v košíku

Používateľ: zákazník

Výstup: Systém pridá alebo odoberie(zmaže) konkrétny produkt v košíku

Postup:

1. Používateľ chce zvýšiť množstvo daného produktu v košíku, tak klikne na tlačidlo +(pridať)
2. Systém zvýši množstvo daného produktu v košíku, zvýši celkovú cenu košíka a zvýši celkový počet produktov v košíku
3. Používateľ chce znížiť množstvo daného produktu v košíku, tak klikne na tlačidlo –(odobrať)
4. Systém zníži množstvo daného produktu v košíku, zníži celkovú cenu košíka a zníži celkový počet produktov v košíku

Alternatívy:

3a. Používateľ má v košíku iba jednu entitu daného produktu, ktorý chce zmazať

3b. Systém vymaže daný produkt z košíka.

4a. Košík je po odobratí produktu prázdny, systém vypíše hlášku, že košík je prázdny.

Zmazanie produktu v košíku

Účel: Zmazanie konkrétneho produktu z košíka

Používateľ: zákazník

Výstup: Systém vymaže daný produkt z košíka.

Postup:

1. Používateľ klikne na tlačidlo Vymazať pri konkrétnom produkte, ktorý chce odstrániť z košíka.
2. Systém daný produkt odstráni z košíka.

Alternatívy:

2a. Daný produkt bol jediný v košíku, systém ho vymaže, a napíše hlášku, že košík je prázdny.

Zobrazenie formuláru na dokončenie objednávky

Účel: Pokračovanie v objednávke

Používateľ: zákazník

Výstup: Systém umožní používateľovi pokračovať v objednávke a zobrazí mu kontaktný formulár

Postup:

1. Používateľ klikne na Tlačidlo pokračovať v objednávke
2. Systém zobrazí kontaktný formulár pre používateľa na dokončenie objednávky

Potvrdenie kontaktného formulára

Účel: Ukončenie objednávky

Používateľ: zákazník

Výstup: Systém ukončí objednávku a uloží danú objednávku do databázy

Postup:

1. Používateľ vyplní všetky textové polia vo formulári správne a klikne na tlačidlo Objednať
2. Systém ukončí objednávku, uloží objednávku do databázy a zobrazí upozornenie že objednávka bola zaznamenaná.

Alternatívy:

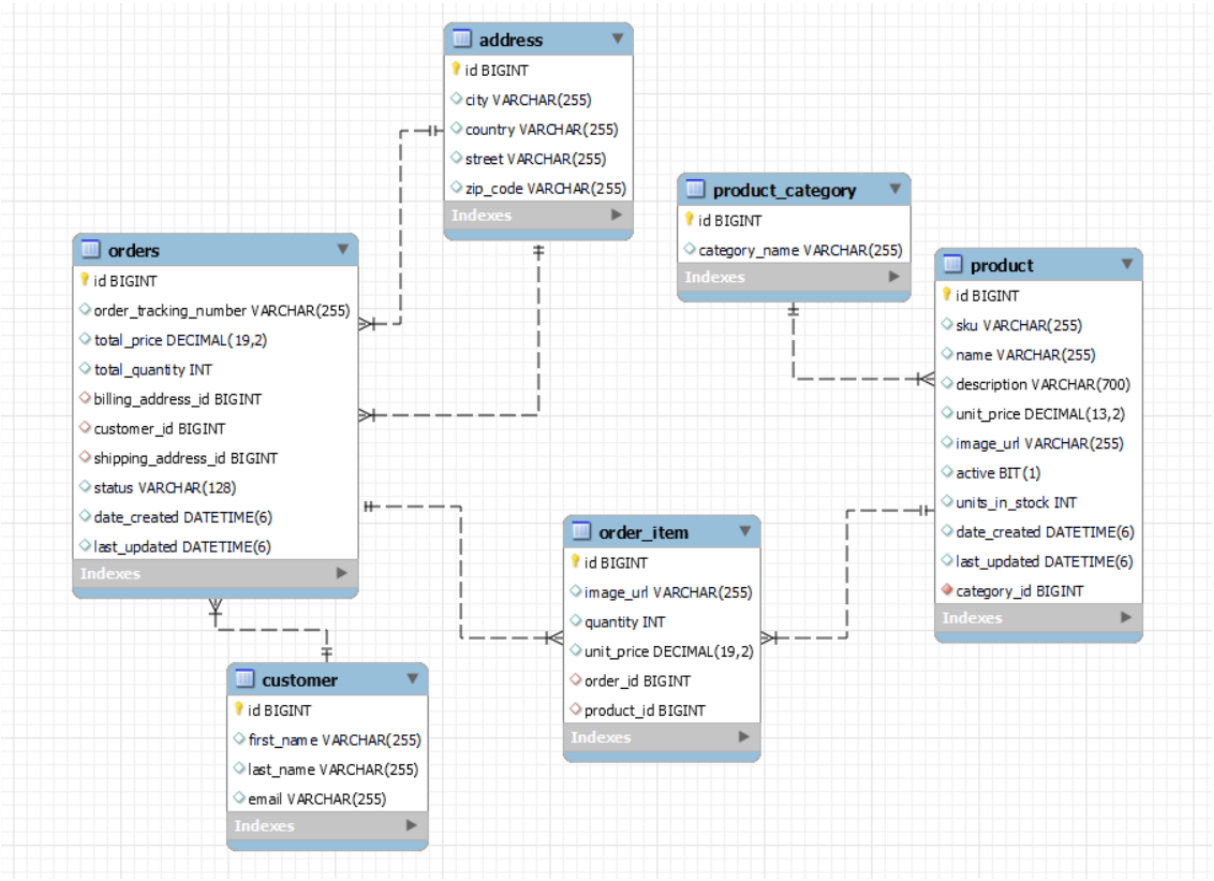
1a. Používateľ nevyplní všetky políčka, alebo vyplní niektoré políčko nesprávne a klikne na tlačidlo objednať

2a. Systém neukončí objednávku, ale vypíše chybné hlášky pod každým konkrétnym políčkom

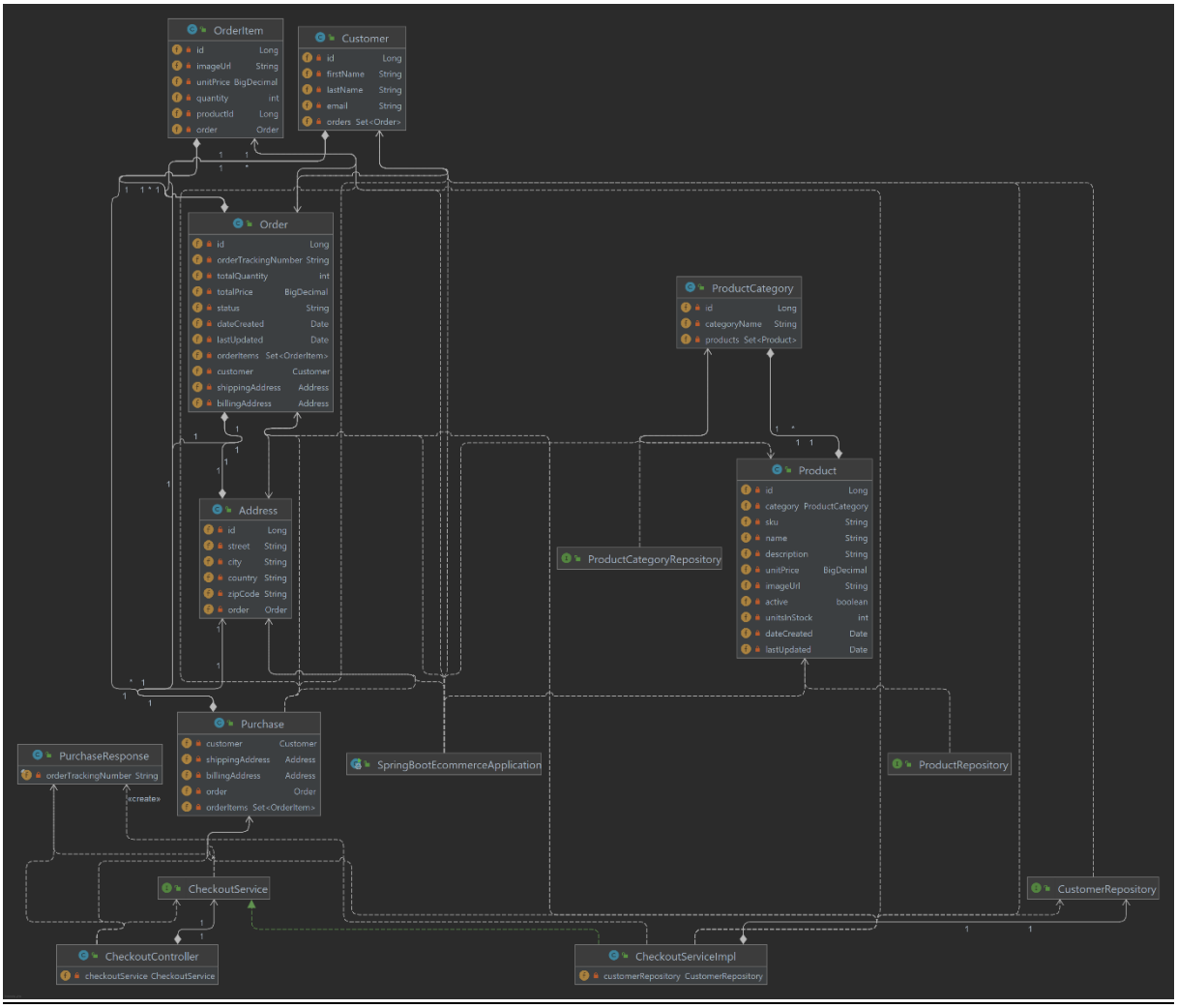
2b. Nastane neočakávaná chyba, a systém vypíše upozornenie že sa vyskytla chyba.

Diagramy

Databáza

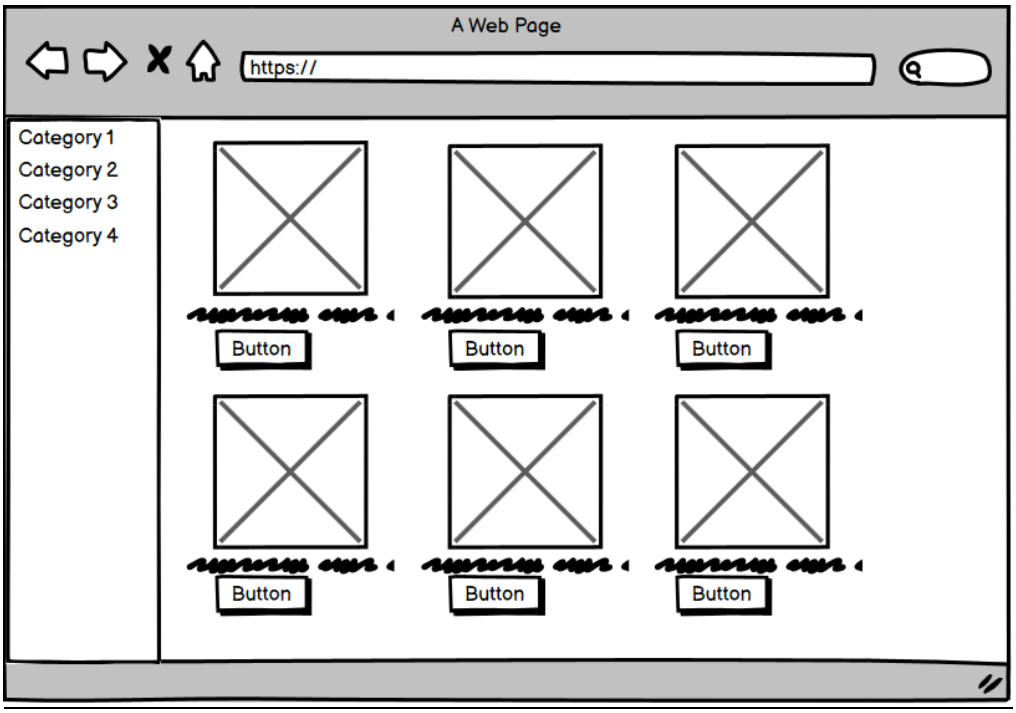


Spring Boot aplikácia

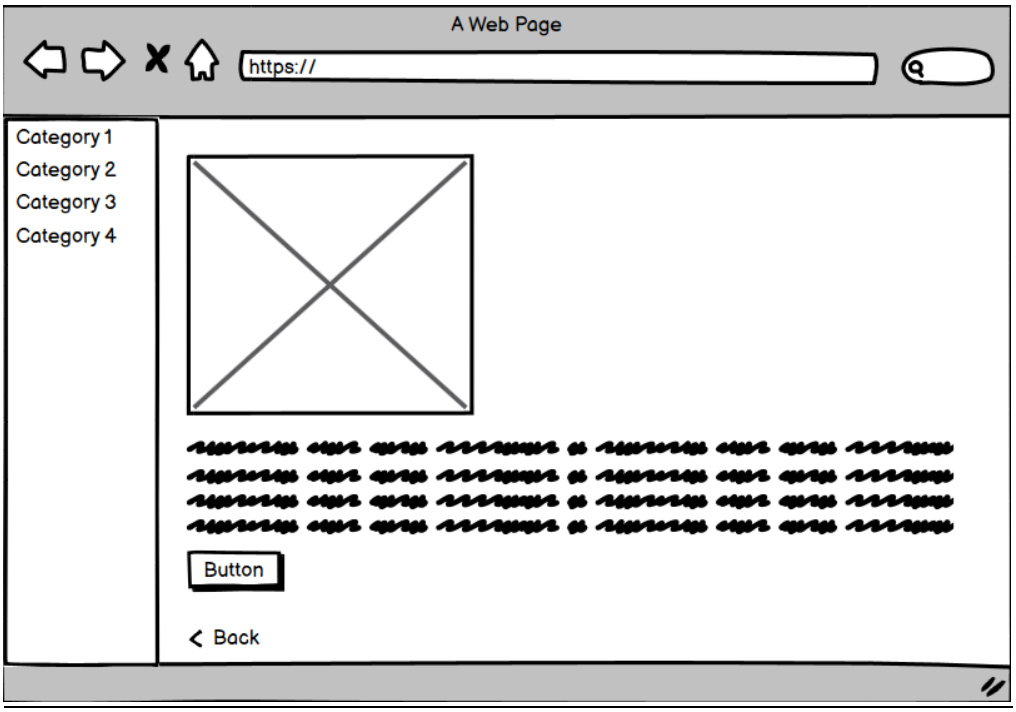


Wireframe návrhy

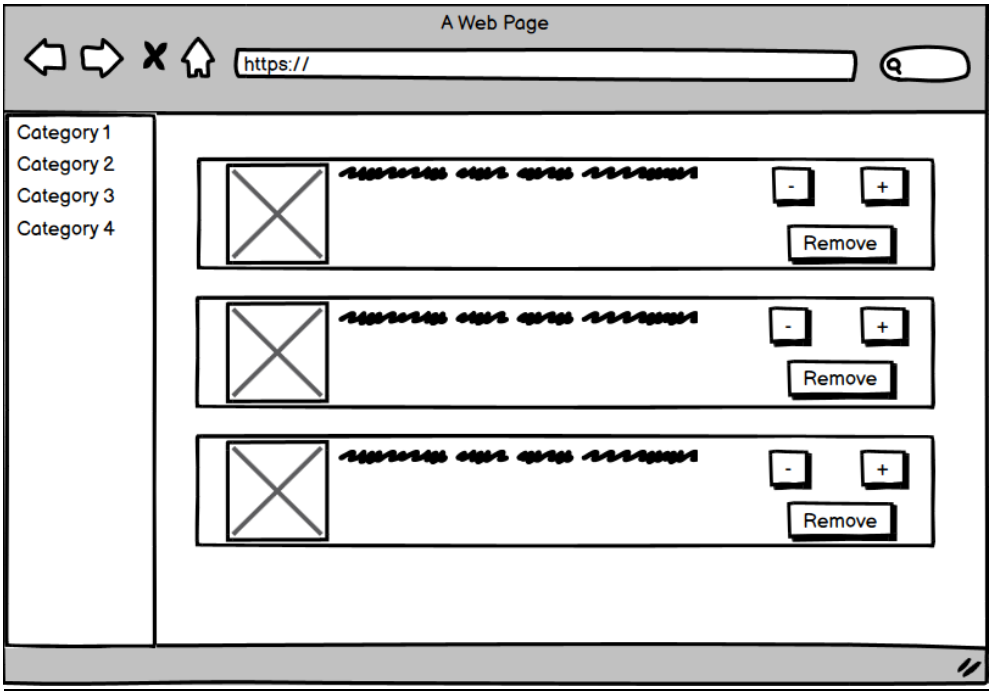
Hlavná stránka



Detail produktu



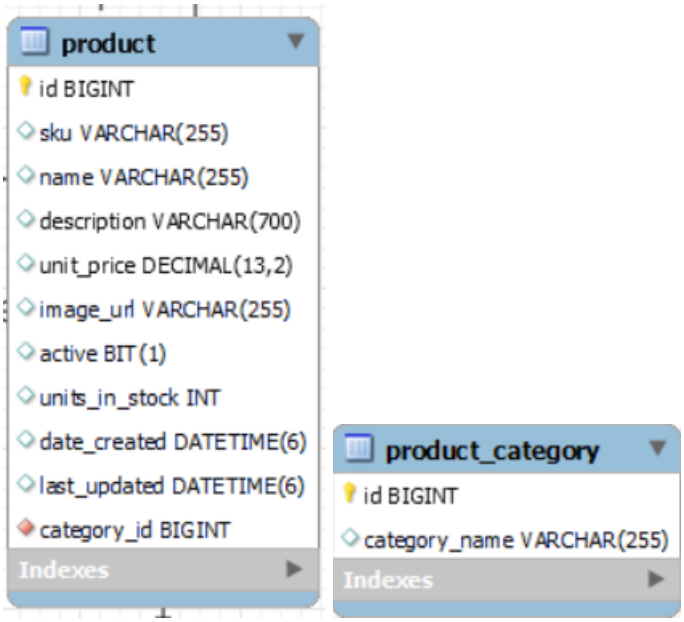
Detail košíka



PRIBLIŽNÝ IMPLEMENTAČNÝ POSTUP

Inicializovanie databázy

- Tabuľky: product, product-category



Vytvorenie tabuliek v databáze pre produkty, a kategórie produktov.

Tabuľka product: id, sku (unikátne identifikačné číslo), name (názov produktu), descritpion(popis produktu), unit_price (cena), image_url(obrázok), active, units_in_stock (počet položiek na sklade), date_created, date_updated, category_id

Tabuľka product_category: id, category_name (názov kategórie)

Inicializovanie Spring Boot aplikácie

- Inicializovanie Spring boot aplikácie pre backend
- Vytvorenie entít (product, product_category)
- Vytvorenie repozitárov
- Vzťah medzi product a product_category – ManyToOne, OneToMany

Inicializovanie Angular projektu

- Vytvorenie componentov, ktoré sú prepojené so Spring Boot aplikáciou
- Vytvorenie componenty product-list, ktorá zobrazuje producty na hlavnej stránke
- Vytvorenie typescript classy product, bude obsahovať premenné také ako sú v databáze
- Vytvorenie service ProductService, ktorá bude volať REST APIs
- Vytvoriť REST klienta – HttpClient (HttpClientModule)
- Vytvorenie ProductList component, ktorá bude čítať údaje z ProductService

HTML template

- Integrovať HTML template do app.component.html
- Pridanie CSS štýlov
- Vytvorenie product-list-grid-component.html
- Pridanie obrázkov produktov

Zobraziť produkty podľa kategórie

- Angular Routing – aktualizuje iba sekciu stránky na základe danej linky
- Definovať Routes na zobrazenie produktov podľa category id +
- Pridanie Router do importov (na základe našich routes)
- Definovať router-outlet (RouterModule)
- Pridanie router-outlet do app.component.html
- Nastavenie RouterLinks na odovzdanie category_id do param (v menu sidebar)

```
- <a style="text-decoration: none;" routerLink="/category/1"
  routerLinkActive="active-link">Laná</a>
```

- Aktualizovať ProductListComponent, tak aby čítal id param

```
- currentCategoryId: number | undefined;
- this.currentCategoryId = Number(this.route.snapshot.paramMap.get('id'));
```

- Aktualizovať Spring Boot aplikáciu tak aby vracala produkty na základe category id

```
- Page<Product> findByCategoryId(@RequestParam("id") Long id, Pageable
  pageable);
```

- Query metóda, (SELECT * FROM product where category_id = ?)
- Aktualizovať ProductService, tak aby volala novú URL na základe metódy vyššie

```
- getProductList(theCategoryId: number): Observable<Product[]> {
-
-     // need to build URL based on category id
-     const searchUrl =
  `${this.baseUrl}/search/findByCategoryId?id=${theCategoryId}`;
-
-     return this.getProducts(searchUrl);
-   }
-
- private getProducts(searchUrl: string): Observable<Product[]> {
-     return this.httpClient.get<GetResponse>(searchUrl).pipe(
-       map(response => response._embedded.products)
-     );
-   }
```

Hľadať produkty podľa názvu

- Aktualizovať Spring Boot aplikáciu tak aby vracal produkty na základe product name

```
- Page<Product> findByNameContaining(@RequestParam("name") String name,
  Pageable pageable);
```

- Vytvoriť nový component search

- Pridať novú route pre hľadanie

```
- // route for search by keyword
-   {path: 'search/:keyword', component: ProductListComponent}
```

- Pridať event binding
- Aktualizovať SearchComponent.html

```
- <input #myInput type="text" placeholder="Zadaj názov produktu..."
  class="au-input au-input-xl"
-     (keyup.enter)="doSearch(myInput.value)" />
-
-   <button (click)="doSearch(myInput.value)" class="au-btn-submit">
-     Hľadať
-   </button>
```

- Definovať metódu v SearchComponent.ts

```
- doSearch(value: string) {
-     console.log(`value=${value}`);
-     this.router.navigateByUrl(`/search/${value}`);
```

```
-     }
```

- Aktualizovať ProductListComponent.ts

```
-   // normal view of products  by category
-     handleListProducts() {
-       // check if "id" parameter is available
-       const hasCategoryId: boolean =
    this.route.snapshot.paramMap.has('id');
-
-       if(hasCategoryId) {
-         // read id paramameter
-         // convert string into number
-         this.currentCategoryId =
    Number(this.route.snapshot.paramMap.get('id'));
-       }
-       else {
-         // not category id available .. dafault to category id 1
-         this.currentCategoryId = 1;
-       }
```

- Aktualizovať metódu listProducts v ProductListComponent.ts

```
-   // assign results to the product arrray
-     listProducts() {
-
-       // if route has parameter keyword
-       this.searchMode = this.route.snapshot.paramMap.has('keyword');
-
-       if(this.searchMode) {
-         this.handleSearchProducts();
-       }
-       else{
-       this.handleListProducts();
-       }
-     }
-
```

- Vytvoriť metódu handleSearchProducts
- Definovať metódu searchProducts v ProductService

```
-   searchProducts(theKeyword: string): Observable<Product[]> {
-
-       // need to build URL based on keyword
-       const searchUrl =
    `${this.baseUrl}/search/findByNameContaining?name=${theKeyword}`;
-
-       return this.getProducts(searchUrl);
-     }
-
```

```
-    private getProducts(searchUrl: string): Observable<Product[]> {
-       // map JSON data from Spring Data REST to Product Array
-       return this.httpClient.get<GetResponse>(searchUrl).pipe(
-         map(response => response._embedded.products)
-       );
-     }
```

- Aplikovanie podmienky ak nie je nájdený žiadny produkt

```
-    <!-- if products empty then display a message-->
-           <div *ngIf="products?.length == 0" class="alert alert-warning
    col-md-12" role="alert">
-             Neboli nájdené žiadne produkty
-           </div>
```

Detailný náhľad produktu

- Vytvoriť novú component ProductDetails
- Pridať novú Route na základe product id

```
-   // route for view product details
-     {path: 'products/:id', component: ProductDetailsComponent},
```

- Pridať routerLink do productListComponent

```
-    <a routerLink="/products/{{ tempProduct.id }}">
-                 <!--routerLink="/products/{{ tempProduct.id }}"-->
-                 <img src="{{ tempProduct.imageUrl }}" class="img-
    responsive">
-               </a>
```

- Vytvoriť metódu v productDetailsComponent.ts, ktorá bude čítať dáta z productService

```
-   handleProductDetails() {
-       // get the "id" param string, convert string to a number using the
    "+" symbol
-       //const theProducts: number =
    +this.route.snapshot.paramMap.get('id');
-       const theProductId: number =
    Number(this.route.snapshot.paramMap.get('id'));
-
-       this.productService.getProduct(theProductId).subscribe(
-         data => {
-           this.product = data;
-         }
-       )
-     }
```

- Vytvoriť novú metódu v ProductService getProduct, ktorá bude vracať objekt Product a JSON dáta konvertuje na Product objekt

```
-   // return observable Product
-     getProduct(theProductId: number): Observable<Product> {
-
-       // need to build URL based on product id
-       const productUrl = `${this.baseUrl}/${theProductId}`;
-
-       // map the JSON data from Spring Data REST to Product Array
-       return this.httpClient.get<Product>(productUrl);
-     }
-
```

- Aktualizovať ProductDetailsComponent.html

Pagination

- Inštalovať ng-boostrap , pridať NgbModuel do Imports
- Aktualizovať getResponse interface v ProductService, tak aby čítal z dáta z JSON pre pagination

```
interface GetResponse {

  _embedded: {
    products: Product[];
  }
  page: {
    size: number;
    totalElements: number;
    totalPages: number;
    number: number;
  }
}
```

- Vytvoriť novú metódu v ProductService, ktorá bude vytvárať ProductList na základe počtu zobrazených produktov

```
-    getProductListPaginate(thePage: number,
-                           thePagesize: number,
-                           theCategoryId: number): Observable<GetResponse>
     {
-
-       // need to build URL based on category id, page, and size
-       const searchUrl =
     `${this.baseUrl}/search/findByCategoryId?id=${theCategoryId}` +
     `&page=${thePage}&size=${thePagesize}`;
-
-       return this.httpClient.get<GetResponse>(searchUrl);
-     }
```

- Aktualizovať ProductListComponent.ts

```
// new properties for pagination
  thePageNumber: number = 1;
  thePageSize: number = 5;
  theTotalElements: number = 0;
```

- Aktualizovať metódu handleListProducts v ProductListComponent.ts
- Vytvoriť novú metódu processResult()

```
-   processResult() {
-       // map JSON data to the actual properties
-       return (data: { _embedded: { products: Product[]; }; page: { number:
    number; size: number; totalElements: number; }; }) => {
-         this.products = data._embedded.products;
-         this.thePageNumber = data.page.number + 1;
-         this.thePageSize = data.page.size;
-         this.theTotalElements = data.page.totalElements;
-       };
-
```

- Vytvoriť novú metódu v ProductService.ts

```
-   searchProductsPaginate(thePage: number,
-       thePagesize: number,
-       theKeyword: string): Observable<GetResponse> {
-
-         // need to build URL based on keyword, page, and size
-         const searchUrl =
    `${this.baseUrl}/search/findByNameContaining?name=${theKeyword}` +
    `&page=${thePage}&size=${thePagesize}`;
-
-         return this.httpClient.get<GetResponse>(searchUrl);
-     }
```

- Vytvoriť novú metódu v ProductListComponent.ts

```
-   // view products by search
-     handleSearchProducts() {
-
-       const theKeyword: string =
    this.route.snapshot.paramMap.get('keyword') || '{}';
-
-       // if we have a different keyword than previous
-       //then set thePageNumber to 1
-
-       if(this.previousKeyword != theKeyword) {
-         this.thePageNumber = 1;
-       }
-
-       this.previousKeyword = theKeyword;
-
```

```
-       console.log(`keyword=${theKeyword},
    thepagenumber=${this.thePageNumber}`);
-
-       // now search for the products using keyword
-       this.productService.searchProductsPaginate(this.thePageNumber - 1,
    this.thePageSize, theKeyword).subscribe(this.processResult());
-     }
```

Pridanie položiek do košíka n.1

- Vytvoriť novú component CartStatusComponent
- Vytvoriť HTML template pre CartStatusComponent (Font Awesome Icon)
- Pridanie "click handler" pre pridanie do košíka

```
-   <button (click)="addToCart(tempProduct)" class="btn btn-primary btn-
    sm">Pridať do košíka</button>
```

- Pridať metódu addToCart() do ProductListComponent

```
addToCart(theProduct: Product) {
    console.log(`Adding to cart: ${theProduct.name},
${theProduct.unitPrice}`);

    // TODO ... do real workk
    const theCartItem = new CartItem(theProduct);

    this.cartService.addToCart(theCartItem);
  }
```

Pridanie položiek do košíka n.2

- Vytvoriť triedu CartItem

```
-   export class CartItem {
-
-     id: string;
-     name: string;
-     imageUrl: string;
-     unitPrice: number;
-
-     quantity: number;
-
```

- Vytvoriť CartService.ts

```
-   export class CartService {
-
-     cartItems: CartItem[] = [];
-
-     totalPrice: Subject<number> = new BehaviorSubject<number>(0);
-     totalQuantity: Subject<number> = new BehaviorSubject<number>(0);
```

- Pridať metódu addToCart v CartService

```
addToCart(theCartItem: CartItem) {
    // check if we already have the item in our cart
    let alreadyExistsInCart: boolean = false;
    let existingCartItem: CartItem = undefined;

    if (this.cartItems.length > 0) {
      // find the item in the cart based on item id

      existingCartItem = this.cartItems.find(tempCartItem => tempCartItem.id
=== theCartItem.id);

      // check if we found it
      alreadyExistsInCart = (existingCartItem != undefined);
    }
```

- Vytvoriť metódu computeCartTotals() v CartService

```
-   computeCartTotals() {
-
-       let totalPriceValue: number = 0;
-       let totalQuantityValue: number = 0;
-
-       for (let currentCartItem of this.cartItems) {
-         totalPriceValue += currentCartItem.quantity *
    currentCartItem.unitPrice;
-         totalQuantityValue += currentCartItem.quantity;
-       }
-
-       // publish the new values ... all subscribers will receive the new
    data
-       this.totalPrice.next(totalPriceValue);
-       this.totalQuantity.next(totalQuantityValue);
-
-       // log cart data just for debugging purposes
-       this.logCartData(totalPriceValue, totalQuantityValue);
-     }
```

- Upraviť metódu addToCart() v ProductListComponent,

```
    const theCartItem = new CartItem(theProduct);

    this.cartService.addToCart(theCartItem);
```

- Upraviť CartStatusComponent, tak aby čítala údaje z CartService

```
-   updateCartStatus() {
-
-       // subscribe to the cart totalPrice
-       this.cartService.totalPrice.subscribe(
-         data => this.totalPrice = data
-       );
-
-       // subscribe to the cart totalQuantity
-       this.cartService.totalQuantity.subscribe(
-         data => this.totalQuantity = data
-       );
-     }
```

- Aktualizovať CartStatusComponent.html, tak aby zobrazil celkovú cenu a celkový počet položiek

Pridanie položky do košíka z detailného náhľadu produktu

- Pridanie "click handler" pre pridanie do košíka v ProductDetailsComponent
- Vytvorenie metódy addToCart() v ProductDetailsComponent

```
-   addToCart() {
-
-       console.log(`Adding to cart: ${this.product.name},
    ${this.product.unitPrice}`);
-       const theCartItem = new CartItem(this.product);
-       this.cartService.addToCart(theCartItem);
-     }
-
```

Detailný náhľad produktov v košíku

- Vytvoriť novú component CartDetailsComponent
- Pridať novú route pre CartDetailsComponent

```
-   {path: 'cart-details', component: CartDetailsComponent},
```

- Pridať routerLink do v CartStatusComponent.html

- Vytvoriť novú metódu listCartDetails() v CartDetailsComponent

```
-   listCartDetails() {
-
-       // get a handle to the cart items
-       this.cartItems = this.cartService.cartItems;
-
-       // subscribe to the cart totalPrice
```

```
-       this.cartService.totalPrice.subscribe(
-         data => this.totalPrice = data
-       );
-
-       // subscribe to the cart totalQuantity
-       this.cartService.totalQuantity.subscribe(
-         data => this.totalQuantity = data
-       );
-
-       // compute cart total price and quantity
-       this.cartService.computeCartTotals();
-     }
```

- Vytvoriť HTML template pre CartDetailsComponent.html

- Pridať “increment”, “decrement” a “remove” button do CartDetailsComponent.html (FontAwesome icon)

- Vytvoriť metódu incrementQuantity() v CartDetailsComponent

```
-   incrementQuantity(theCartItem: CartItem) {
-       this.cartService.addToCart(theCartItem);
-     }
```

- Vytvoriť novú metódy decrementQuantity() a remove() v CartDetailsComponent

```
-   decrementQuantity(theCartItem: CartItem) {
-       this.cartService.decrementQuantity(theCartItem);
-     }
-
-     remove(theCartItem: CartItem) {
-       this.cartService.remove(theCartItem);
-     }
```

- Vytvoriť metódu decrementQuantity() v CartService

```
-   decrementQuantity(theCartItem: CartItem) {
-
-       theCartItem.quantity--;
-
-       if(theCartItem.quantity === 0) {
-         this.remove(theCartItem);
-       }
-       else{
-         this.computeCartTotals();
-       }
-     }
```

- Vytvoriť metódu remove() v CartService

```
- remove(theCartItem: CartItem) {
-
-     // get index of item in the array
-     const itemIndex = this.cartItems.findIndex(tempCartItem =>
  tempCartItem.id === theCartItem.id);
-
-     // if found, remove the item from the array at the given index
-     if(itemIndex > -1) {
-       this.cartItems.splice(itemIndex, 1);
-
-       this.computeCartTotals();
-     }
-   }
```

Formulár pre objednávku

- Vytvoriť CheckoutComponent
- Pridať novú route pre CheckoutComponent

```
- {path: 'checkout', component: CheckoutComponent},
```

- Vytvoriť nový “button” , pre pokračovanie v objednávke

```
-  <p><a routerLink="/checkout" class="btn btn-primary">Pokračovať v
  objednávke</a></p>
```

- Vytvoriť FormGroups pre formuláre v CheckoutComponent.ts

- Vytvoriť metódu onSubmit() v CheckoutComponent.ts – zatiaľ len console.log

- Vytvoriť HTML template, v CheckoutComponent.html

- Pridať “event handler” pre metódu onSubmit() v CheckoutComponent.html

Ošetrenie dátumov expirácie kreditnej karty

- Vytvoriť DatesFormService
- Pridať metódy do DatesFormService
- Upraviť CheckoutComponent.ts, tak aby čítala údaje z DatesFormService

```
- const startMonth: number = new Date().getMonth() + 1;
-     console.log("startMonth:" + startMonth);
-
-     this.DatesFormService.getCreditCardMonths(startMonth).subscribe(
-       data => {
-         console.log("Retrieved credit card months: " +
  JSON.stringify(data));
-         this.creditCardMonths = data;
```

```
-
-         }
-       );
-
-       // populate credit card years
-       this.DatesFormService.getCreditCardYears().subscribe(
-         data => {
-           console.log("Retrieved credit card years: " +
    JSON.stringify(data));
-           this.creditCardYears = data;
-         }
-       )
```

- Pridať polia rokov a mesiacov do CheckoutComponent.html

- Pridať “event binding” do CheckoutComponent.ts

```
-   <select formControlName="expirationYear"
    (change)="handleMonthsAndYears()">
```

- Vytvoriť handleMonthsAndYears() v CheckouComponent.ts

```
-   handleMonthsAndYears() {
-       const creditCardFormGroup =
    this.checkoutFormGroup.get('creditCard');
-       const currentYear: number = new Date().getFullYear();
-       const selectedYear: number =
    Number(creditCardFormGroup.value.expirationYear);
-
-       // if the current year eqauls the selected year, then start with the
    current month
-       let startMonth: number;
-
-       if(currentYear === selectedYear) {
-         startMonth = new Date().getMonth() + 1;
-       }
-       else {
-         startMonth = 1;
-       }
-
-       this.DatesFormService.getCreditCardMonths(startMonth).subscribe(
-         data => {
-           console.log("Retrieved credit card months: " +
    JSON.stringify(data));
-           this.creditCardMonths = data;
-         }
-       )
-     }
```

Validácia objednávkového formulára

- Požiadavky: všetky políčka musia byť povinné, email adresa musí mať správny formát, číslo kreditnej karty musí obsahovať 16 čísel, bezpečnostný kód kreditnej karty musí obsahovať 3 čísla
- Špecifikovať pravidlá pre validáciu v CheckoutComponent.ts

```
-   customer: this.formBuilder.group({
-           firstName: new FormControl('', [Validators.required,
    Validators.minLength(2), MyOwnValidators.notOnlyWhitespace]),
-           lastName: new FormControl('', [Validators.required,
    Validators.minLength(2), MyOwnValidators.notOnlyWhitespace]),
-           email: new FormControl('', [Validators.required,
    Validators.pattern('^[a-z0-9._%+-]+@[a-z0-9.-]+\\.[a-z]{2,4}$')])
-         }),
```

- Vytvorenie “getterov” kvôli získaniu prístupu v FormControl

```
-   get firstName() { return
    this.checkoutFormGroup.get('customer.firstName'); }
-     get lastName() { return
    this.checkoutFormGroup.get('customer.lastName'); }
-     get email() { return this.checkoutFormGroup.get('customer.email'); }
```

- Pridanie error správ do HTML v CheckoutComponent.html

```
-   <div *ngIf="firstName.invalid && (firstName.dirty || firstName.touched)"
-                     class="alert alert-danger mt-1">
-
-                     <div *ngIf="firstName.errors.required ||
    firstName.errors.notOnlyWhitespace">
-                       Toto políčko je povinné
-                     </div>
-
-                     <div *ngIf="firstName.errors.minlength">
-                       Toto políčko musí obsahovať minimálne 2 znaky
-                     </div>
-                   </div>
```

- Pridať “event handler” na kontrolu validácie v metóde onSubmit() v CheckoutComponent.ts

```
-   onSubmit() {
-       console.log("HAndling the submit button");
-
-       if(this.checkoutFormGroup.invalid) {
-         this.checkoutFormGroup.markAllAsTouched();
-         return;
-       }
```

Vytvorenie vlastnej validácie

- Ošetrenie prípad ak užívateľ napíše iba voľné miesto
- Vytvoriť novú triedu MyOwnValidator
- Vytvorenie metódy v MyOwnValidators.ts, ktorá zistí či daný formulár obsahuje alebo neobsahuje iba prázdne miesta

```
-   static notOnlyWhitespace(control: FormControl): ValidationErrors {
-
-       // check if string only contains whitespace
-       if((control.value != null) && (control.value.trim().length === 0)) {
-
-         // invalid, return error object
-         return {'notOnlyWhitespace': true};
-       }
-       else {
-         // valid, return null
-         return null;
-       }
-     }
```

Zhrnutie objednávky

- Vytvoriť novú metódu reviewCartDetails() v CheckoutComponent.ts

```
-   reviewCartDetails() {
-
-       // subscribe to cartService.totalQuantity
-       this.cartService.totalQuantity.subscribe(
-         totalQuantity => this.totalQuantity = totalQuantity
-       );
-
-       // subscribe to cartService.totalPrice
-       this.cartService.totalPrice.subscribe(
-         totalPrice => this.totalPrice = totalPrice
-       );
-
-     }
```

- Zmeniť Subject na BehaviorSubject v CartService.ts

Uloženie objednávky do databázy – Back end

- Vytvoriť skript do databázy
- Vytvoriť entity tabuliek v Spring Boot aplikácií
- Vytvoriť “data transfer” objekty – Purchase, PurchaseResponse
- Vytvoriť CustomerRepository
- Vytvoriť CheckoutService

- Vytvoriť novú triedu CheckoutServiceImpl
- Vytvoriť novú metódu generateOrderTrackingNumber() v CheckouServiceImpl. Ktorá bude generovať unikátne číslo objednávky
- Vytvoriť metódu placeOrder() v CheckouServiceImpl

Uloženie objednávky do databázy – Front end

- Vytvoriť základné triedy: Customer, Order, OrderItem, Adress, Purchase
- Vytvoriť service CheckoutService
- Aktualizovať metódu onSubmit() v ChekoutComponent.ts