

Wichterlovo gymnázium, Ostrava-Poruba, příspěvková organizace



Marek Tyl

Systém e-shopu – webová aplikace

Maturitní práce

Ostrava

Vedoucí práce: Ing. David Hrbáč

Čestné prohlášení

Prohlašuji, že jsem tuto práci vypracoval/a samostatně s použitím literárních pramenů a informací, které cituji a uvádím v seznamu použité literatury a zdrojů informací.

V Ostravě, dne 22.4.2022

Podpis:

Obsah

Úvod.....	4
1 Použité Jazyky	5
1.1 HTML	5
1.2 CSS.....	6
1.2.1 Bootstrap.....	6
1.3 C#	7
1.3.1 Blazor	7
2 Použité programy	9
2.1 Visual Studio	9
2.2 SQL Server a SQL Server Managment Studio	9
3 Uživatelské rozhraní.....	10
3.1 Produkty.....	10
3.2 Vyhledávání.....	11
3.3 Košík.....	11
4 Funkce.....	12
4.1 Databáze	12
4.2 Vyhledávání.....	16
4.3 Zobrazení produktů	17
4.4 Košík.....	18
Závěr.....	20
Použitá Literatura	21
Seznam Obrázků	22

Úvod

Jako téma maturitní práci do informatiky jsem si vybral webovou aplikaci e-shopu. Hlavním cílem webové aplikace je umožnit návštěvníkovi prohlédnout si dostupné produkty a následně si je objednat.

Kód celé práce je dostupný na:

GitHub - tylmarek1/TM_Eshop_v1: Maturita - Eshop - v1. GitHub: Where the world builds software · GitHub [online]. Copyright © 2022 GitHub, Inc. [cit. 21.04.2022]. Dostupné z: https://github.com/tylmarek1/TM_Eshop_v1

Návod na spuštění aplikace je dostupný na:

Návod k TM Eshopu. Návod k TM Eshopu [online]. Dostupné z: <http://tylmarek.euweb.cz/>

1 Použité Jazyky

1.1 HTML

Hypertext Markup Language neboli zkráceně HTML je značkovací jazyk používaný hlavně ve webových stránkách. Jak už lze odvodit z názvu, od normálního textu se liší tím, že je provázaný hypertextovými odkazy. HTML by se dalo považovat za základní stavební kostku naprosté většiny webových stránek.

V HTML se používají různé předdefinované tagy, které mají různé významy. Tento způsob byl převzat z SGML (Standard Generalized Markup Language). Tagy jsou většinou ve dvojicích, jeden otevírající a jeden uzavírající, mezi nimi je pak text, který vkládáme na stránky. Uzavírající tag se liší pouze lomítkem před názvem. Tagy taky mohou být vnořené, například v tagu `<p></p>`, který představuje odstavec, může být vnořen tag `<i></i>`, který změní styl textu na kurzívu.

V rámci HTML jsou i pro koncového uživatele „neviditelné“ tagy, které obsahují informace pro prohlížeč.

```
1 <!DOCTYPE html>
2 <html lang="cs">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7 </head>
8 <body>
9   <nav>
10    <li>
11      <ul>Tohle je list.</ul>
12      <ul>Tohle je list.</ul>
13      <ul>Tohle je list.</ul>
14    </li>
15  </nav>
16  <div>
17    <h1>Tohle je nadpis.</h1>
18    <p>Tohle je odstavec.</p>
19    <p>V tomhle textu jsou <i>tahle slova napsaná kurzívou</i>, a tahle
    už zase ne.</p>
20    <p>V tomhle textu jsou <b>tahle slova napsaná tučně</b>, a tahle už
    zase ne.</p>
21    <p>V tomhle textu jsou <a href="cesta">tahle slova hyperlinkovým
    odkazem</a>, a tahle už zase ne.</p>
22    <img src="" alt="">
23  </div></body></html>
```

1.2 CSS

CSS neboli Kaskádové styly je jazyk používaný k úpravě vzhledu HTML stránek. Původně se vzhled stránky upravoval pomocí jazyka HTML, pak se ale z důvodu přehlednosti vyvinul jazyk CSS, aby HTML jazyk popisoval pouze obsah a strukturu a jazyk CSS popisoval vzhled stránky.

V CSS souboru pak můžeme navazovat na jména tagu, třídy tagu, nebo identifikátoru tagu. Lze také použít „interaktivní“ element jako například `:hover`, který se spustí až když nad objektem projedeme kurzorem.

V CSS také můžeme používat několik různých jednotek velikosti, podle toho, které se zrovna danou věc hodí. Patří mezi ně: px, em, rem, vh, vw, % a další. Jednotka px už se v dnešní době tak moc nepoužívá, protože je statická a uživateli se tak stránka může špatně zobrazit.

```
1 <div></div>
2 <div class="class1"></div>
3 <div id="id1"></div>

1 div {
2   display: flex;
3   flex-direction: column;
4   justify-content: space-around;
5   align-items: center;
6 }
7 .class1 {
8   width: 500px;
9   border: solid red 2px;
10  padding: 10px;
11 }
12 #id1 {
13   max-height: 30vh;
14 }
15 div:hover {
16   outline: solid green 1rem;
17   transform: rotate(90);
18 }
```

1.2.1 Bootstrap

Bootstrap je předepsaná knihovna v jazyce LESS, kterou můžeme připojit k našemu projektu a pak ji pomocí specifických tříd používat v HTML souboru pro úpravu vzhledu.

Bootstrap byl vyvinut pro podporu konzistence vzhledu stránek. Před přestavením Bootstrapu se pro design webu využívalo mnoho různých knihoven, což vedlo k jeho namáhavé údržbě. Proto se vývojáři Bootstrapu rozhodli vytvořit univerzální nástroj pro sjednocení kódu na svých rozhraních. Jak můžeme vidět v ukázkovém kódu, při využívání Bootstrapu upravujeme HTML jazyk pomocí předdefinovaných tříd.

```
<div class="container-fluid container-sm"></div>
```

1.3 C#

C# je vysokoúrovňový programovací jazyk. Je odvozen od jazyků C++ a Java a byl vyvinut společností Microsoft. Jazyk běží ve vývojovém prostředí .NET. Jazyk má několik základních vlastností:

- Jazyk je objektově orientovaný.
- Obsahuje jenom jednoduchou dědičnost, ale násobnou implementaci rozhraní.
- Obsahuje události.
- Obsahuje garbage collector, takže správa paměti je automatická.
- Je bezpečnější než C++.

V následné ukázce kódu vidíme nejzákladnější funkci Hello World zapsanou v jazyce C#.

```
1 using System;
2
3 namespace HelloWorld
4 {
5     class Program
6     {
7         static void Main(string[] args)
8         {
9             Console.WriteLine("Hello World!");
10        }
11    }
12 }
```

1.3.1 Blazor

Blazor je framework pro C#, který se používá na vytváření webových stránek. Jsou dva hlavní typy Blazoru: Blazor Server a Blazor WebAssembly. Blazor Server, podobně jako PHP, běží na serveru a klientovi se potom posílá pouze výsledek. Na rozdíl od toho Blazor WebAssembly funguje podobně jako JavaScript, kde se klientovi posílá celá aplikace, která

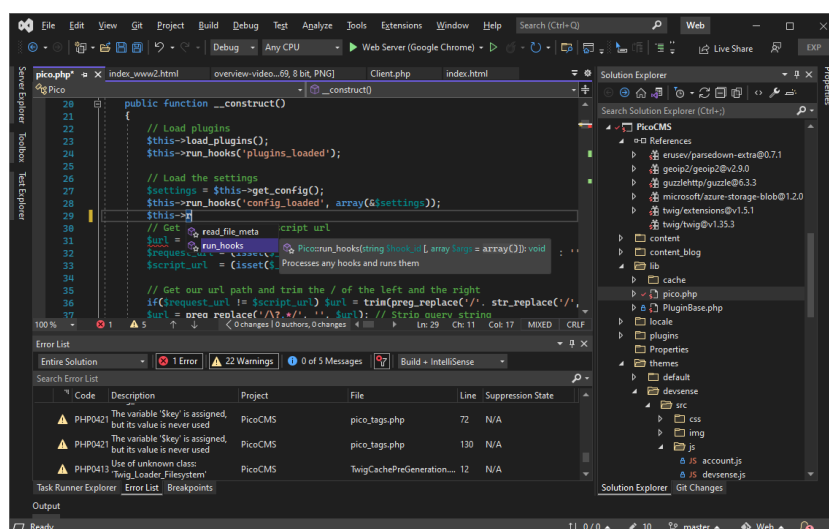
pak běží u klienta. V mém projektu jsem použil ASP.NET core hosted Blazor WebAssembly, která má i serverovou část. V následné ukázce vidíme, jak jednoduše lze HTML a C# provázat.

```
1 <p>count: @count</p>
2 <button class="btn btn-primary" @onclick="PlusCount">Plus jeden.</button>
3
4 @code {
5     private int count = 0;
6
7     private void PlusCount()
8     {
9         count++;
10    }
11 }
```


2 Použité programy

2.1 Visual Studio

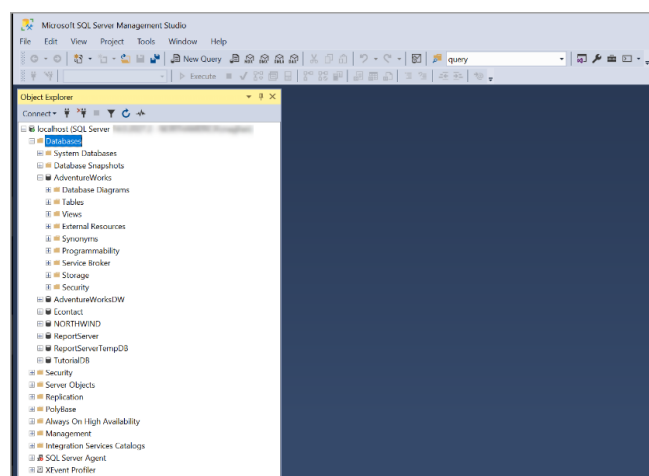
Visual Studio je vývojové prostředí vydané firmou Microsoft. Program poskytuje širokou škálu možností pro vývoj aplikací v platformě .NET, ale zároveň i v jiných platformách. Aplikace má spoustu chytrých vymožeností jako predikce kódu, nebo zvýrazňování chyb a nabídku možných řešení.



Obrázek 1: Visual Studio

2.2 SQL Server a SQL Server Management Studio

V mém projektu jsem používal SQL Server na zprostředkování databáze pro můj web a SQL Server Management Studio na úpravu databáze.

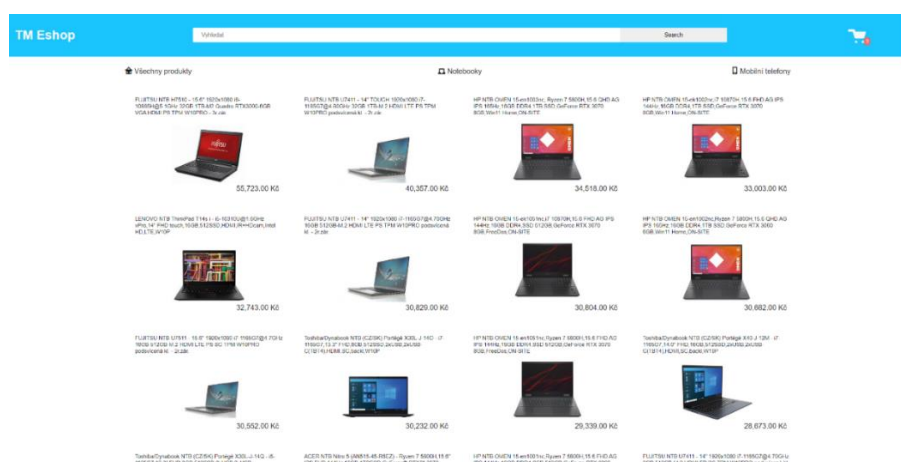


Obrázek 2: SQL Server Management Studio

3 Uživatelské rozhraní

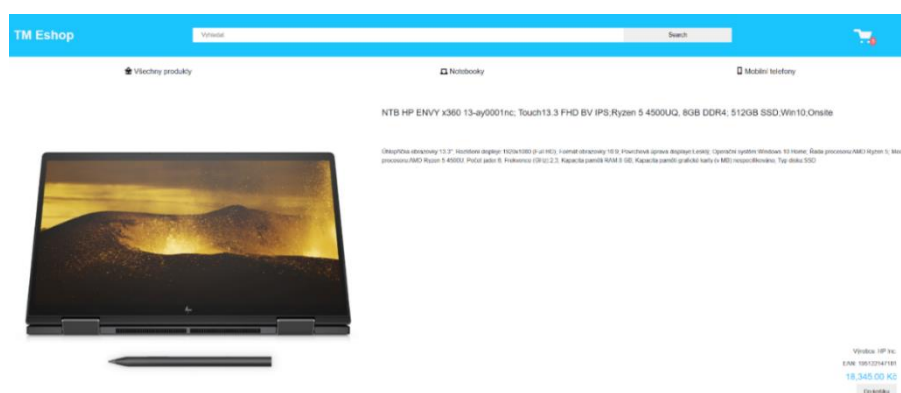
3.1 Produkty

Každý zákazník má možnost zobrazit si seznam produktů. Ať už si chce zobrazit všechny produkty najednou, nebo po kategoriích. V seznamu produktů jsou produkty seřazeny v blocích v mřížce, která má čtyři sloupce. U každého produktu vidí název produktu, obrázek a cenu. Každému produktu se pro lepší orientaci trochu změní pozadí, když po něm zákazník přejíždí kurzorem.



Obrázek 3: Seznam produktů

Poté, co zákazník na produkt klikne, dostane se na stránku s detaily daného produktu, kde si daný produkt může přidat do košíku.



Obrázek 4: Detail produktu

3.2 Vyhledávání

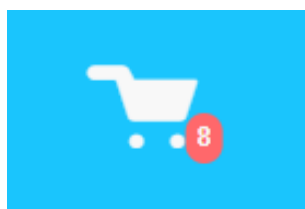
Zákazník má vždy možnost použít v záhlaví vyhledávací lištu, aby si našel produkt podle jeho jména. Po zadání klíčového slova se zákazníkovi zobrazí seznam produktů, které mají shodu klíčového slova a názvu produktu.



Obrázek 5: Vyhledávací lišta

3.3 Košík

Pokud si zákazník na stránce s detailem produktu přidá produkt do košíku, produkt se vloží do košíku a aktualizuje se indikátor u ikony košíku, aby zákazník viděl, kolik má produktů v košíku.



Obrázek 6: Ikona košíku

Po kliknutí na ikonu se zákazník dostane do košíku, kde uvidí tabulku produktů, které jsou v košíku. Produkty, které nechce může zákazník z košíku odstranit pomocí tlačítka odstranit. A pokud zákazníkovi všechno vyhovuje, může kliknout na tlačítko objednat, pak už stačí pouze zadat údaje na které to chceme poslat a objednávka se odešle. Košík se vyprázdní a zákazník může nakupovat dále.


TM Eshop

Všetchny

Search



 Všechny produkty

 Notebooky

 Mobilní telefony

Prod	Produkt	Cena	Odstřanil
1557220	FUJITSU NTB H7510 - 15.6" 1920x1080 i8-10885H@5.1GHz 32GB 1TB-M2 Quadro RTX3000-6GB VGA HDMI PS TPM W10PRO - 3x.zdr.	55,723.00 Kč	odstranit
1555362	HP NTB OMEN 15-en1003nc, Ryzen 7 5800H,15.6 QHD AG IPS 165Hz,16GB DDR4,1TB SSD,GaForce RTX 3070 8GB,Win11 Home,ON-SITE	34,518.00 Kč	odstranit
1512372	NTB HP ENVY x360 13-ay0001nc,Touch13.3 FHD BV IPS,Ryzen 5 4500U,8GB DDR4; 512GB SSD;Win10;Onsite	18,345.00 Kč	odstranit
1555364	HP NTB OMEN 15-en1051nc,Ryzen 7 5800H,15.6 FHD AG IPS 144Hz,16GB DDR4,SSD 512GB,GeForce RTX 3070 8GB,FreeDOS, ON-SITE	29,339.00 Kč	odstranit
1556872	FUJITSU NTB U7511 - 15.6" 1920x1080 i5-1135G7@4.2GHz,8GB 256GB-M.2 HDMI FP SC TOP W10PRO podsvícená kl. - 2x.zdr.	23,397.00 Kč	odstranit
1578112	Sony Xperia 5 III, 5G, zelená	19,801.00 Kč	odstranit
1574141	Realme 8 5G, 6GB/128GB, Supersonic Black	5,103.00 Kč	odstranit
1585584	Realme GT Master, 6GB/128GB, Cosmos Black	5,818.00 Kč	odstranit
		Total(8): 192,044.00 Kč	

Obrázek

Obrázek 7: Košík

4 Funkce

4.1 Databáze

V mém projektu používám databázi k uložení informací o produktech, kategoriích a objednávkách. Abychom dostali data z databáze na webovou stránku je potřeba hned několik souborů a operací. Na první pohled to vypadá jako spousta náhodných souborů, ale jakmile člověk pochopí, k čemu se který soubor používá, stává se z toho lehká skládanka. Ten postup pak lze s malými obměnami opakovat a vytváření aplikace se stává vcelku jednoduché.

První věc, kterou potřebujeme udělat, je zapsat si připojení do databáze v souboru `appsettings.json`.

```
1  "ConnectionStrings": {  
2    "DefaultConnection":  
    "server=localhost\\sqlexpress;database=EshopDB_v1;trusted_connection=true;  
    "  
3  }
```

Tohle je připojení do databáze, která nemá žádné přihlašovací údaje. Pokud bychom používali databázi, která vyžaduje přihlášení, musíme přidat další atributy.

Další věc, kterou potřebujeme vytvořit, je model dat, které budeme používat. U dat jdou nastavit různé parametry. Jdou nastavit různé typy, ať už stringy, čísla, desetinná čísla, čas a další. Taky si jde do modelu nastavit parametry jako: povinné, emailová adresa, maximální délku, nebo minimální délku. Datové modely vytváříme ve sdílené části projektu, aby byly přístupné jak v serverové, tak v klientové části projektu.

```
1 namespace TM_Eshop_v1.Shared{  
2     public class Buyer{  
3         [Key]  
4         public int BuyerId { get; set; }  
5         [Required]  
6         public string Name { get; set; } = string.Empty;  
7         [Required, EmailAddress]  
8         public string Mail { get; set; } = string.Empty;  
9         [Required]  
10        public string AdressCity { get; set; } = string.Empty;  
11        [Required]  
12        public string AdressStreet { get; set; } =  
13 string.Empty;}}
```

U modelu dat lze vytvořit i relace mezi jednotlivými modely, podle kterých se potom na sebe mohou navazovat, když voláme jednotlivé modely dat ve funkcích.

Poté pomocí Entity Frameworku vytvoříme příkazem `PM> dotnet ef migrations add DbUpdate1 migraci`, která podle datových modelů automaticky vytvoří zápis do databáze, následně použijeme příkaz `PM> dotnet ef database update` vytvoříme v databázi příslušné tabulky.

Dále je potřeba vytvořit datový kontext, abychom věděli, co z databáze nahrát. V mém projektu mám v databázi zapsané tři různé tabulky, a to jsou tabulka produktů, tabulka kategorií a tabulka objednávek.

```
1 namespace TM_Eshop_v1.Server.Data.DataContext
2 {
3     public class DataContext : DbContext
4     {
5         public DataContext(DbContextOptions<DataContext> options) :
6         base(options) { }
7         public DbSet<Product>? Products { get; set; }
8         public DbSet<Category>? Categories { get; set; }
9         public DbSet<Order>? Orders { get; set; }
10    }
```

Když máme nastavený model i připojení, můžeme vytvořit serverový servis, ve kterém bude logika zpracování dat. Každý servis má dva soubory, rozhraní a soubor s logikou. Nejdříve si v rozhraní vypíšeme funkce, které bude servis potřebovat a pak v druhém souboru definuje, co dané funkce dělají. Obvykle se rozhraní pojmenuje stejně jako soubor s logikou a přidáme před název „I“, aby program rozpoznal, že je to rozhraní. „I“ pochází z anglického interface. Například: `ProductService.cs` a `IProductService.cs`. Použití rozhraní zvyšuje bezpečnost kódu a zároveň umožní vícenásobné dědictví, které jinak není možné v C#. V příkladu uvedeném níže je funkce na získání jednoho produktu z databáze podle jeho identifikátoru.

Nejdříve si zavedeme neznámou datového kontextu a poté vztahujeme z databáze podle datového modelu a přidáváme filtr na identifikátor, následně už stačí jen vzít v potaz možnost prázdné odpovědi a máme funkci hotovou.

```

1 namespace TM_Eshop_v1.Server.Services.ProductService
2 {
3     public class ProductService : IProductServiceInterface
4     {
5         private readonly DataContext _context;
6
7         public ProductService(DataContext context)
8         {
9             _context = context;
10        }
11
12        public async Task<ServiceResponse<Product>> GetProductAsync(int
    proId)
13        {
14            var response = new ServiceResponse<Product>();
15            var product = await _context.Products.FindAsync(proId);
16            if (product == null)
17            {
18                response.Success = false;
19                response.Message = "Wrong path or maybe error.";
20            }
21            else
22            {
23                response.Data = product;
24            }
25            return response;
26        }

```

Posledním krokem na serverové části je vytvoření kontroléru, který vytvoří, nebo dostává Http žádost. Například, pokud získáváme data, používáme metodu [HttpGet] a pokud ukládáme data, používáme metodu [HttpPost]. Existují dva způsoby, jak vytvářet kontrolery. Jedním způsobem je do kontroléru zahrnout i logiku a už není potřeba žádného servisu. Tento způsob, ale není doporučovaný, protože se kontrolér nestává znovupoužitelný a my bychom jich museli vytvářet zbytečně pořád více a zpomalovali bychom naši aplikaci. Proto je lepší všechnu logiku vytvořit v servisu a pak postavit kontrolér, který pouze vytváří http žádosti.

```

1 namespace TM_Eshop_v1.Server.Controllers
2 {
3     [Route("api/[controller]")]
4     [ApiController]
5     public class ProductController : ControllerBase
6     {
7         private readonly IProductServiceInterface _productService;
8
9         public ProductController(IProductServiceInterface
    productService)
10        {
11            _productService = productService;
12        }

```

```

13
14     [HttpGet("getallproducts")]
15     public async Task<ActionResult<ServiceResponse<List<Product>>>>
    GetProducts()
16     {
17         var result = await _productService.GetProductsAsync();
18         return Ok(result);
19     }
20     [HttpGet("{proId}")]
21     public async Task<ActionResult<ServiceResponse<Product>>>
    GetProduct(int proId)
22     {
23         var result = await _productService.GetProductAsync(proId);
24         return Ok(result);
25     }
26 ...

```

Tímhle krokem jsme dokončili serverovou část, všechno máme připravené a můžeme se přesunout na klientovou část.

4.2 Vyhledávání

Pro vyhledávání si vytvoříme dva soubory, klientový servis, který se bude dovolávat HTTP žádosti ze serverového kontroléru, a komponent, který se bude zobrazovat na webu.

Jako první vytvoříme servis, který je zase stvořený ze dvou souborů, a to z rozhraní a soubor s logikou. V mém případě dostávám HTTP žádost s listem produktů, které budou vyfiltrované pomocí klíčového slova, až tuto funkci v zavoláme v komponentu.

```
1 namespace TM_Eshop_v1.Client.Services.ProductService{
2     public class ProductService : IProductServiceInterface
3     {
4         private readonly HttpClient _http;
5
6         public ProductService(HttpClient http) {_http = http;}
7         public List<Product> Products { get; set; } = new
8 List<Product>();
9         public string Message { get; set; } = "Produkty se načítají...";
10        public event Action ProductChanged;
11        public async Task Search(string searchString) {
12            var result = await
13            _http.GetFromJsonAsync<ServiceResponse<List<Product>>>
14            ($"api/product/search/{searchString}");
15            if (result != null && result.Data != null)
16                Products = result.Data;
17            if (Products.Count == 0)
18                Message = "Tady nic není...";
19            ProductChanged?.Invoke(); }
```

Dále pak vytvoříme komponent `Search.razor`, ve kterém bude HTML část a C# část. V HTML části vytvoříme `<input>` tag do kterého bude uživatel zapisovat klíčové slovo, které bude chtít vyhledat a tlačítko, jehož kliknutím potvrdí svůj input a spustí danou funkci. Ještě to můžeme obalit do blokového elementu `<div>`, aby se nám stránka lépe graficky upravovala. V C# části potom zavedeme funkci, která se spouští tlačítkem. Jediné, co funkce musí dělat, je zavolat si příslušný servis a dosadit do něho klíčové slovíčko.

```
1 @inject NavigationManager NavigationManager
2 @inject IProductServiceInterface ProductService
3
4 <div class="searchdiv">
5     <div class="searchinputdiv">
6         <input type="search" id="searchbarid" placeholder="Vyhledat"
7         @bind-value="searchString" @bind-value:event="oninput"
8         @onkeyup="Keyboard" @ref="userSearch" />
9     </div>
10    <div class="buttondiv">
```



```

9         <button @onclick="SearchPro">Search</button>
10     </div>
11 </div>

12 @code {
13     private string searchString = string.Empty;
14     protected ElementReference userSearch;
15     public void SearchPro() {
16         if (searchString == string.Empty) {
17             NavigationManager.NavigateTo($"{products}");
18         }
19         else {
20             NavigationManager.NavigateTo($"{products}/search/{searchString}");
21         }
22     }
23 }

```

Pak už jenom jen vložíme `<Search />` do našeho kódu a máme hotovo.

4.3 Zobrazení produktů

U zobrazení produktů postupujeme podobně jako u vyhledávání. V tomto případě však buď požádáme o všechny produkty, v případě, že zobrazujeme seznam všech produktů, anebo je filtrujeme podle kategorie nebo identifikátoru, abych dostali buď produkty z jedné kategorie, nebo pouze jeden produkt.

Jako první si zase vytvoříme servis.

```

1 namespace TM_Eshop_v1.Client.Services.ProductService {
2     public class ProductService : IProductServiceInterface {
3         private readonly HttpClient _http;
4         public ProductService(HttpClient http) { _http = http; }
5         public List<Product> Products { get; set; } = new
6         List<Product>();
7         public string Message { get; set; } = "Produkty se načítají...";
8         public event Action ProductChanged;
9         public async Task<ServiceResponse<Product>> GetProduct(int proId)
10         {
11             var result = await
12             _http.GetFromJsonAsync<ServiceResponse<Product>>
13             ($"api/product/{proId}");
14             return result;
15         }
16         public async Task GetProducts(string? catUrl = null) {
17             var results = catUrl == null?
18             await
19             _http.GetFromJsonAsync<ServiceResponse<List<Product>>>
20             ("api/product/getallproducts"):
21             await
22             _http.GetFromJsonAsync<ServiceResponse<List<Product>>>
23             ($"api/product/category/{catUrl}");
24             if (results != null && results.Data != null)
25                 Products = results.Data;
26             ProductChanged.Invoke();
27         }
28     }
29 }

```

Potom to můžeme hezky pomocí `foreach()` cyklu vyobrazit v komponentu `ProductList.razor`. A poté vložit na webovky jako `<ProductList />`.

```
@foreach (var product in ProductService.Products)
{
    <a href="product/@product.ProId">
        <h4>@product.Name</h4>
        
        <p>@product.Price Kč</p>
    </a>
}
```

4.4 Košík

S košíkem začneme pracovat v momentu, kdy se zákazník rozhodne něco přidat do košíku. V rámci košíku jsou tři hlavní operace. Přidání produktu do košíku, zobrazení produktu v košíku a vytvoření objednávky.

Když přidáváme produkty do košíku, vezmeme si identifikátor produktu a zapíšeme ho do `localStorage` prohlížeče. To je úložiště v prohlížeči, které zůstane zachováno i po restartu prohlížeče, zákazník proto bude mít produkty v prohlížeči i když další den znovu navštíví e-shop.

Toho dosáhneme pomocí funkce v servisu.

```
1 public async Task AddToCart(CartItem cartItem)
2     {
3         var cart = await
4         _localStorage.GetItemAsync<List<CartItem>>("cart");
5         if (cart == null) { cart = new List<CartItem>(); }
6         cart.Add(cartItem);
7         await _localStorage.SetItemAsync("cart", cart);
8         OnChange.Invoke();
9     }
```

Poté na zobrazení produktů v košíku budeme potřebovat dvě funkce. Jednu, která získá list identifikátorů z `localStorage` a druhou, která vytáhne z databáze produkty se shodujícími se identifikátory a všechny je pomocí funkce `foreach()` zobrazí v tabulce. Pokud budeme chtít produkty z košíku vymazat, budeme potřebovat ještě třetí funkci, která z `localStorage` vymaže určitý identifikátor.

```

1 public async Task<List<CartResponse>> GetCartProducts()
2     {
3         var cartItems = await
4         _localStorage.GetItemAsync<List<CartItem>>("cart");
5         var response = await
6         _http.PostAsJsonAsync("api/cart/products", cartItems);
7         var cartProducts = await
8         response.Content.ReadFromJsonAsync<ServiceResponse<List<CartResponse>>>();
9         return cartProducts.Data;
10    }

```

Na vytvoření objednávky, budeme potřebovat získat data o zákazníkovi z formuláře, identifikátory produktů z localStorage a aktuální čas, který získáme pomocí jednoduché funkce. Poté to všechno odeslat do databáze. A nakonec vymazat localStorage.

```

1 public async Task CreateOrder(Order order)
2     {
3         var result = await
4         _http.PostAsJsonAsync("api/order/createorder", order);
5         var response = await
6         result.Content.ReadFromJsonAsync<List<Order>>();
7         Orders = response;
8         await _localStorage.ClearAsync();
9         _navigationManager.NavigateTo("", forceLoad: true);
10    }

```

Závěr

Maturitní práce se mi povedla. Na stánkách funguje prohlížení produktů, vyhledávání produktů, i funkce košíku a objednávek. Do budoucnosti by aplikaci šlo vylepšit přidáním uživatelů a plateb, nebo taky tmavé verze stránky pro pohodlnější sledování během nočních hodin. Stránky jsou kompatibilní se všemi nejpoužívanějšími prohlížeči a zároveň mají jednoduchý a přehledný vzhled, aby měl zákazník příjemný zážitek. Během vytváření projektu jsem si velmi prohloubil své znalosti C# a určitě s ním budu chtít programovat i v budoucnosti.

Použitá Literatura

1. Hypertext Markup Language. *Wikipedie* [online]. [cit. 2022-04-21]. Dostupné z: https://cs.wikipedia.org/wiki/Hypertext_Markup_Language
2. Kaskádové styly. *Wikipedie* [online]. [cit. 2022-04-21]. Dostupné z: https://cs.wikipedia.org/wiki/Kask%C3%A1dov%C3%A9_styly
3. Bootstrap. *Wikipedie* [online]. [cit. 2022-04-21]. Dostupné z: <https://cs.wikipedia.org/wiki/Bootstrap>
4. C Sharp. *Wikipedie* [online]. [cit. 2022-04-21]. Dostupné z: https://cs.wikipedia.org/wiki/C_Sharp
5. C Sharp. *VSB* [online]. [cit. 2022-04-21]. Dostupné z: <http://www.cs.vsb.cz/behalek/vyuka/pcsharp/text.pdf>
6. Visual Studio. *Wikipedia* [online]. [cit. 2022-04-21]. Dostupné z: https://cs.wikipedia.org/wiki/Microsoft_Visual_Studio
7. CHAN, Jamie. C#: Learn C# in One Day and Learn It Well. C# for Beginners with Hands-on Project. Amazon [ebook]. Kindle Store: LCF Publishing, 2015, 20.10.2015 [cit. 2022-04-21].
8. ENGSTRÖM, Jimmy. Web Development with Blazor: A hands-on guide for .NET developers to build interactive UIs with C#. Amazon [kniha]. India: Packt Publishing Limited, 2021, 18.06.2021 [cit. 2022-04-21].

Seznam Obrázků

Obrázek 1: Visual Studio	9
Obrázek 2: SQL Server Managment Studio.....	9
Obrázek 3: Seznam produktů	10
Obrázek 4: Detail produktu	10
Obrázek 5: Vyhledávací lišta	11
Obrázek 6: Ikona košíku	11
Obrázek 7: Košík.....	11