EKONOMICKÁ UNIVERZITA V BRATISLAVE

FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 103004/I/2025/36122176496362756

ASP .NET XML webová služba poskytujúca usporiadané informácie o položkách elektronickej predajne výpočtovej techniky

Diplomová práca

2025 **Bc. Štefan Baňkos**

EKONOMICKÁ UNIVERZITA V BRATISLAVE

FAKULTA HOSPODÁRSKEJ INFORMATIKY

ASP .NET XML webová služba poskytujúca usporiadané informácie o položkách elektronickej predajne výpočtovej techniky

Diplomová práca

Študijný program: Informačný manažment

Študijný odbor: Ekonómia a manažment

Školiace pracovisko: Katedra aplikovanej informatiky

Vedúci záverečnej práce: Ing. Igor Košťál, PhD.

Bratislava 2025 **Bc. Štefan Baňkos**

ZADANIE

Ekonomická univerzita v Bratislave
Fakulta hospodárskej informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Štefan Baňkos
Študijný program: informačný manažment (Jednoodborové štúdium, inžiniersky II. st., denná forma)
Študijný odbor: ekonómia a manažment
Typ záverečnej práce: Inžinierska záverečná práca
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: ASP .NET XML webová služba poskytujúca usporiadané informácie o položkách elektronickej predajne výpočtovej techniky

Anotácia: Diplomant v práci zanalyzuje možnosti použitia XML webových služieb na sledovanie parametrov výpočtovej techniky, notebookov, stolových počítačov, tabletov a príslušenstva, v sledovanom období v elektronickom informačnom systéme elektronického obchodu a porovná ich použitie s inými spôsobmi sledovania týchto parametrov. V rámci diplomovej práce diplomant vo vybratom riadenom programovacom jazyku vytvorí ASP .NET XML webovú službu poskytujúcu prostredníctvom svojej funkcionality jej klientovi podľa vybraných kritérií usporiadané informácie o parametroch výpočtovej techniky elektronického obchodu v sledovanom období, ktorými môžu byť celkový počet jednotlivých druhov notebookov, stolových počítačov a tabletov v tomto obchode na začiatku a konci sledovaného obdobia, počty a zoznamy jednotlivých druhov výpočtovej techniky s ich základnými dátami od jednotlivých výrobcov, vybranej značky, danej kategórie, počty a zoznamy jednotlivých druhov výpočtovej techniky s ich základnými dátami na začiatku a konci sledovaného obdobia a iné ich parametre.

Vedúci: Ing. Igor Košťál, PhD.
Katedra: KAI FHI - Katedra aplikovanej informatiky
Vedúci katedry: doc. Ing. Mgr. Peter Schmidt, PhD.

Dátum zadania: 01.03.2024

Dátum schválenia: 07.03.2024 prof. Ing. Ivan Brezina, CSc.
osoba zodpovedná za realizáciu študijného programu

Čestné vyhlásenie

Čestne vyhlasujem, že záverečnú prácu som vypracoval samostatne a že som uviedol všetku použitú literatúru.

Dátum: **.....**

(podpis študenta)

Poďakovanie:

Chcel by som poďakovať vedúcemu mojej záverečnej práce Ing. Igorovi Košťálovi, PhD. za rady, ochotu a usmernenia v priebehu tvorby tejto práce.

ABSTRAKT

BAŇKOS, Štefan: ASP .NET XML webová služba poskytujúca usporiadané informácie o položkách elektronickej predajne výpočtovej techniky – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra aplikovanej informatiky. – Vedúci záverečnej práce: Ing. Igor Košťál, PhD – Bratislava: FHI EU, 2025, 69 s.

Diplomová práca sa zameriava na návrh, implementáciu a analýzu ASP.NET XML webovej služby, ktorá poskytuje štruktúrované a usporiadané informácie o produktoch elektronického obchodu špecializujúceho sa na výpočtovú techniku. V práci je riešená potreba efektívneho, prehľadného a dostupného systému pre správu údajov o notebookoch, stolových počítačoch, tabletoch a príslušenstva, pričom osobitná pozornosť je venovaná správe produktov a zákazníckych objednávok v XML formáte.

Cieľom bolo navrhnúť webovú službu, ktorá umožní centrálne uchovávať, filtrovať, triediť, pridávať a odstraňovať produkty a objednávky v rámci elektronického obchodu Elektrodynamo.sk. Okrem základných operácií poskytuje služba aj analytické funkcie – generovanie štatistík, sledovanie vývoja parametrov v čase a prehľady podľa kategórie, značky či časového obdobia. Webová služba využíva štandardy XML, SOAP a WSDL, čím zaručuje vzájomná spolupráca a kompatibilita s rôznymi platformami a systémami.

Výsledkom práce je funkčný prototyp ASP.NET XML webovej služby s navrhnutým používateľským rozhraním, ktoré demonštruje možnosti integrácie s aplikáciou cez JavaScript a AJAX. Táto práca dokazuje praktické výhody použitia XML webových služieb v e-commerce prostredí a zároveň poskytuje základ pre ďalší vývoj elektronického informačného systému.

Kľúčové slova: ASP.NET, XML webová služba, výpočtová technika, správa produktov, elektronický obchod, štatistiky, objednávky.

ABSTRACT

BAŇKOS, Štefan: ASP .NET XML web service providing sorted information about a computer electronic store items – University of Economics in Bratislava. Faculty of Economic Informatics; Department of Applied Informatics. – Supervisor of master's thesis: Ing. Igor Košťál, PhD. – Bratislava: FHI EU, 2025, 69 p.

This thesis focuses on the design, implementation and analysis of an ASP.NET XML web service that provides structured and organized information about the products of an e-commerce company specializing in computing. The thesis addresses the need for an efficient, clear and accessible system for managing data on laptops, desktops, tablets and accessories, with particular attention to the management of products and customer orders in XML format.

The aim was to design a web service that would allow central storage, filtering, sorting, adding and removing of products and orders within the Elektrodynamo.sk e-shop. In addition to basic operations, the service also provides analytical functions - generation of statistics, tracking the evolution of parameters over time and reports by category, brand or time period. The web service uses XML, SOAP and WSDL standards, thus guaranteeing interoperability and compatibility with various platforms and systems.

The result of the work is a working prototype of an ASP.NET XML web service with a proposed user interface that demonstrates the possibilities of integration with the application via JavaScript and AJAX. This work demonstrates the practical benefits of using XML web services in an e-commerce environment while providing a foundation for further development of a electronic information system.

Key words: ASP.NET, XML web service, computing, product management, e-commerce, statistics, orders.

Obsah

Úvod 12
1 Súčasný stav riešenej problematiky 13
1.1 Možnosť riešenia pomocou programu Microsoft Excel 13
1.2 Možnosť riešenia pomocou programu Microsoft Access 14
1.3 Výhody nášho riešenia 14
2 Vývoj a uplatnenie XML webových služieb v podnikových aplikáciách 15
2.1 Vývoj XML webových služieb 15
2.2 Metódy sledovania parametrov výpočtovej techniky 17
1.2.1. Tradičné metódy sledovania 17
1.2.2. XML a XML webové služby 18
1.2.3. REST API a JSON 19
1.2.4. Kombinované prístupy 19
2.3 Porovnanie XML webových služieb s inými metódami sledovania 20
1.3.1. XML webové služby vs REST API 20
1.3.2. XML webové služby vs Relačné databázy 22
2.4 Implementácia XML Webovej služby v ASP .NET 22
1.4.1. Definícia a účel XML Webovej služby v ASP .NET 23
1.4.2. Kľúčové komponenty a Protokoly 23
1.4.3. Štruktúra a organizácia webovej služby 24
1.4.4. Výhody a Nevýhody XML Webových služieb v ASP .NET 25
3 Cieľ práce 26
4 Popis vývojového prostredia 27
5 Návrh a štruktúra webovej služby 29
5.1 Metóda *AddProduct* na pridanie produktu do vstupného XML súboru 32
5.2 Metóda *GetProduct* na načítanie produktov zo vstupného XML súboru 34
5.3 Metóda *DeleteProduct* na odstránenie produktov v XML súbore 37
5.4 Metóda *GetFilteredAndSortedProducts* na filtrovanie a zoradenie produktov v XML súbore 38
5.5 Metóda *AddOrder* na pridanie objednávky do XML súboru 42

5.6 Metóda *DeleteOrder* na odstránenie objednávky v XML súbore 45
5.7 Metóda *GetAllOrders* na načítanie objednávok z XML súboru 47
5.8 Metóda *GetFilteredAndSortedOrders* na filtrovanie a zoradenie objednávok 49
6 Implementácia správy produktov a štatistík v e-shope 52
6.1 Používateľské rozhranie a navigácia 52
6.2 Filtrovanie a zoradenie produktov 53
6.3 JavaScript a AJAX pre dynamické načítanie produktov 53
6.4 Pridávanie a odstraňovanie produktov 55
6.5 Generovanie štatistík vstupného XML súboru 56
6.6 Štruktúra pre správu objednávok 56
7 Ukážka fungovania aplikácie 61
Diskusia 63
Záver 64
Zoznam použitej literatúry 66
Prílohy 69

Zoznam obrázkov

Obrázok 1 Vývojové prostredie Visual Studio 27
Obrázok 2 Inicializácia globálnych premenných s XML súbormi (Zdroj: vlastné spracovanie) ... 29
Obrázok 3 Diagram tried webovej služby 30
Obrázok 4 Kontrola existencie súboru so záznamami o produktoch (Zdroj: vlastné spracovanie) 32
Obrázok 5 Generovanie ID produktu (Zdroj: vlastné spracovanie) 33
Obrázok 6 Vytvorenie nového elementu <product> (Zdroj: vlastné spracovanie) 33
Obrázok 7 Pridanie nového elementu <product> (Zdroj: vlastné spracovanie) 34
Obrázok 8 Výpis informácie o úspešnom pridaní elementu (Zdroj: vlastné spracovanie) 34
Obrázok 9 Overenie existencie súboru so záznamami o produktoch (Zdroj: vlastné spracovanie) 35
Obrázok 10 Načítanie a transformácia inštancií uzlov triedy Product (Zdroj: vlastné spracovanie) 35
Obrázok 11 Serializácia zoznamu do formátu JSON (Zdroj: vlastné spracovanie) 36
Obrázok 12 Načítanie vstupného XML súboru (Zdroj: vlastné spracovanie) 37
Obrázok 13 Porovnanie zadaného ID s ID v XML súbore (Zdroj: vlastné spracovanie) 37
Obrázok 14 Porovnanie zadaného ID s ID v XML súbore (Zdroj: vlastné spracovanie) 38
Obrázok 15 Odstránenie elementu <product> (Zdroj: vlastné spracovanie) 38
Obrázok 16 Uloženie aktualizovaného súboru (Zdroj: vlastné spracovanie) 38
Obrázok 17 Výpis informácie o úspešnom odstránení elementu <product> (Zdroj: vlastné spracovanie) 38
Obrázok 18 Načítanie a inicializácia podmienok pre filtrovanie (Zdroj: vlastné spracovanie) 40
Obrázok 19 Parametre usporiadania záznamov produktov (Zdroj: vlastné spracovanie) 41
Obrázok 20 Výpočet štatistických údajov z vypísaných údajov produktov (Zdroj: vlastné spracovanie) 41
Obrázok 21 Vrátenie serializovanej JSON odpovede (Zdroj: vlastné spracovanie) 42
Obrázok 22 Overenie existencie XML súboru s údajmi o produktoch (Zdroj: vlastné spracovanie) 43
Obrázok 23 Odstránenie pridaného produktu do objednávky zo skladu (Zdroj: vlastné spracovanie) 43
Obrázok 24 Vytvorenie ID objednávky (Zdroj: vlastné spracovanie) 44
Obrázok 25 Vytvorenie elementu <objednávka> (Zdroj: vlastné spracovanie) 44
Obrázok 26 Pridanie a uloženie novej objednávky do XML súboru (Zdroj: vlastné spracovanie) 44
Obrázok 27 Načítanie súboru objednávok (Zdroj: vlastné spracovanie) 45
Obrázok 28 Overenie existencie objednávky vo vstupnom XML súbore (Zdroj: vlastné spracovanie) 46
Obrázok 29 Odstránenie objednávky a uloženie aktualizovaného XML súboru (Zdroj: vlastné spracovanie) 46
Obrázok 30 Overenie existencie súboru a výber elementov objednavka (Zdroj: vlastné spracovanie) 48
Obrázok 31 Vrátenie dát v JSON formáte (Zdroj: vlastné spracovanie) 48
Obrázok 32 Načítanie vstupného XML súboru objednávok (Zdroj: vlastné spracovanie) 49
Obrázok 33 Filtrovanie XML súboru objednávok na základe časového intervalu (Zdroj: vlastné spracovanie) 50

Obrázok 34 Filtrovanie XML súboru objednávok na základe kvartálneho obdobia (Zdroj: vlastné spracovanie) 50
Obrázok 35 Usporiadanie XML súboru objednávok na podľa ceny 51
Obrázok 36 Výpočet štatistických metód údajov o objednávkach (Zdroj: vlastné spracovanie) ... 51
Obrázok 37 Vytvorenie objektu <response> a serializovanie do JSON formátu (Zdroj: vlastné spracovanie) 52
Obrázok 38 Navigačné menu na prechod medzi sekciami (Zdroj: vlastné spracovanie) 52
Obrázok 39 Formulár na vstupy parametrov pre filtrovanie údajov (Zdroj: vlastné spracovanie). 53
Obrázok 40 Načítanie produktov zo servera (Zdroj: vlastné spracovanie) 54
Obrázok 41 Vytvorenie tabuľky s produktovými údajmi (Zdroj: vlastné spracovanie) 55
Obrázok 42 Formulár na pridanie produktu (Zdroj: vlastné spracovanie) 55
Obrázok 43 Funkcia na volanie vystavenej metódy na pridanie produktu (Zdroj: vlastné spracovanie) 56
Obrázok 44 Výpis štatistík o produktoch (Zdroj: vlastné spracovanie) 56
Obrázok 45 Vytvorenie formuláru na filtrovanie objednávok (Zdroj: vlastné spracovanie) 57
Obrázok 46 Vytvorenie tabuľky objednávok (Zdroj: vlastné spracovanie) 58
Obrázok 47 Funkcia na načítanie objednávok zo servera (Zdroj: vlastné spracovanie) 58
Obrázok 48 Funkcia na volanie metódy filtrovania podľa zadaných kritérií (Zdroj: vlastné spracovanie) 59
Obrázok 49 Funkcia na pridanie objednávky (Zdroj: vlastné spracovanie) 60
Obrázok 50 Sekcia objednávok zobraz v okne klienta (Zdroj: vlastné spracovanie) 61
Obrázok 51 Výpis po zadaní parametrov a zavolaní konkrétnej metódy (Zdroj: vlastné spracovanie) 62

Úvod

V súčasnosti zohrávajú elektronické informačné systémy kľúčovú úlohu v oblasti elektronického obchodu. S rastúcimi požiadavkami na transparentnosť, presnosť a efektívnosť spracovania údajov o produktoch sa zvyšuje aj potreba implementácie moderných technológií, ktoré umožňujú efektívne sledovanie a spracovanie technických parametrov tovaru. Významnou technológiou v tejto oblasti sú XML webové služby, ktoré umožňujú bezpečný a štruktúrovaný prenos dát medzi systémami prostredníctvom internetu.

Táto práca sa zameriava na možnosti využitia XML webových služieb v prostredí elektronického obchodu s výpočtovou technikou. Konkrétne sa zaoberá návrhom a implementáciou ASP.NET XML webovej služby pre podnik Elektrodynamo.sk, ktorá poskytuje svojim klientom usporiadané informácie o parametroch notebookov, stolových počítačov, tabletov a príslušenstva v sledovanom období.

Cieľom práce je analyzovať výhody a nevýhody použitia XML webových služieb v porovnaní s inými spôsobmi sledovania produktových parametrov, a jednak navrhnúť vlastné riešenie, ktoré demonštruje praktickú implementáciu tejto technológie. Vytvorená webová služba umožňuje spracovávať údaje o jednotlivých druhoch výpočtovej techniky na základe rôznych kritérií – napríklad podľa kategórie, výrobcu, značky alebo časového obdobia. Okrem základných operácií, ako je filtrovanie alebo zoradenie údajov, poskytuje služba aj analytické výstupy, ako sú počty produktov na začiatku a konci sledovaného obdobia, zoznamy produktov podľa výrobcu, či sledovanie vývoja vybraných parametrov.

Výsledkom práce bude funkčná ASP.NET XML webová služba, ktorá bude slúžiť ako súčasť elektronického informačného systému fiktívneho podniku Elektrodynamo.sk. Táto služba predstavuje praktický príklad využitia XML webových služieb v reálnom prostredí, pričom jej návrh je prispôsobený špecifikám elektronického obchodu s výpočtovou technikou.

1 Súčasný stav riešenej problematiky

V súčasnosti existuje viacero spôsobov, akými možno v podnikovej praxi sledovať a spravovať parametre výpočtovej techniky, najmä v kontexte elektronického obchodu. Medzi najčastejšie používané nástroje patria tabuľkové procesory ako Microsoft Excel, relačné databázové systémy ako Microsoft Access alebo MySQL, a moderné webové technológie, vrátane REST API služieb. Každé z týchto riešení má svoje výhody aj obmedzenia, ktoré je vhodné zvážiť pri výbere konkrétneho prístupu. V nasledujúcej časti analyzujeme jednotlivé z nich a porovnáme ich s navrhnutým riešením v tejto diplomovej práci – ASP.NET XML webovou službou vyvinutou pre podnik Elektrodynamo.sk.

1.1 Možnosť riešenia pomocou programu Microsoft Excel

Microsoft Excel je jedným z nástrojov pre vedenie evidencie a vykonávanie základných analýz. Jeho výhodou je nízka náročnosť, intuitívne rozhranie a možnosť rýchleho vytvárania tabuliek, grafov a filtrov. V mnohých prípadoch sa v Exceli uchovávajú informácie o produktoch, skladových zásobách či objednávkach, často manuálne zadávané alebo importované z iných zdrojov.

Hoci Excel poskytuje široké možnosti pre analýzu údajov, jeho použitie na komplexné, dlhodobé a dynamické sledovanie parametrov výpočtovej techniky má viacero nedostatkov:

- Obmedzená automatizácia – zmeny v dátach je nutné často manuálne aktualizovať.
- Riziko chýb – neexistuje ochrana pred neúmyselným prepísaním údajov.
- Zlá škálovateľnosť – pri väčšom množstve údajov a zložitejších výpočtoch sa tabuľky spomaľujú.
- Chýbajúca centralizácia – viacero kópií Excel súborov v rôznych zariadeniach môže viesť k nekonzistentnosti údajov.

Naproti tomu ASP.NET XML webová služba poskytuje centralizovaný prístup k údajom, pričom umožňuje klientom pracovať vždy s aktuálnymi dátami priamo zo servera. Všetky požiadavky sú spracované automatizovane a výsledky sú dynamicky generované bez nutnosti manuálneho zásahu.

1.2 Možnosť riešenia pomocou programu Microsoft Access

Microsoft Access predstavuje pokročilejšie riešenie než Excel, s podporou databázovej logiky, relačných väzieb a dotazovania cez SQL. Access môže byť využívaný ako nástroj na správu údajov o produktoch, zákazníkoch či objednávkach. Jeho výhodou je možnosť navrhnúť formuláre, dotazy a výstupy bez potreby rozsiahlejšieho programovania.

Obmedzenia Accessu sú však zrejmé pri pokuse o nasadenie v prostredí elektronického obchodu s potrebou prepojenia viacerých klientov:

- Slabá podpora viacerých používateľov – pri viacerých súbežných prístupoch dochádza k problémom so stabilitou.
- Zložité zdieľanie na webe – Access nie je navrhnutý pre nasadenie v online prostredí.
- Obmedzené možnosti rozšírenia – integrácia s inými systémami si vyžaduje náročné premostenie.

Navrhnutá XML webová služba tieto nedostatky prekonáva prostredníctvom webového rozhrania, ktoré je ľahko dostupné z ľubovoľného zariadenia s pripojením na internet. Okrem toho, služba poskytuje výstupy v štandardizovanom XML formáte, ktorý môže byť bez problémov spracovaný ďalšími systémami.

1.3 Výhody nášho riešenia

Navrhnutá aplikácia na báze ASP.NET XML webovej služby v prostredí Elektrodynamo.sk bude predstavovať alternatívu k riešeniam Excel a Access. V porovnaní s bežnými nástrojmi bude poskytovať:

- automatizované spracovanie požiadaviek,
- jednotné a aktuálne výstupy,
- možnosť filtrovania a triedenia produktov podľa viacerých parametrov,
- zobrazenie zmien v čase (porovnanie stavu na začiatku a na konci sledovaného obdobia),
- integráciu s webovým rozhraním,

Zároveň si zachováva zrozumiteľnosť a transparentnosť, ktorú ocenia najmä používatelia z neprogramátorského prostredia. Služba teda predstavuje efektívny nástroj pre menšie až stredne veľké podniky, ktoré chcú spravovať údaje o výpočtovej technike.

2 Vývoj a uplatnenie XML webových služieb v podnikových aplikáciách

V súčasnosti dochádza k neustálemu rozširovaniu elektronických obchodov a ich informačných systémov, ktoré zabezpečujú efektívnu evidenciu produktov, skladových zásob a parametrov jednotlivých zariadení. Webové služby sa v tejto oblasti využívajú na zabezpečenie komunikácie medzi rôznymi aplikáciami a informačnými systémami. XML webové služby patria medzi tradičné riešenia, ktoré poskytujú vzájomnú spoluprácu medzi rôznymi platformami a systémami(Curbera et al., 2002). Využívajú sa predovšetkým v podnikovej sfére, kde je dôraz kladený na formálne definované rozhrania a štandardizovanú komunikáciu(W3C, 2007).

Organizácia W3C (World Wide Web Consortium) definuje štandardy XML, SOAP a WSDL, ktoré sú nevyhnutné pre implementáciu XML webových služieb. SOAP (Simple Object Access Protocol) poskytuje mechanizmus pre prenos správ medzi klientom a serverom prostredníctvom HTTP protokolu (Gudgin et al., 2007). XML webové služby sú preto vhodné pre podnikové aplikácie, ktoré vyžadujú vysokú mieru bezpečnosti a spoľahlivosti (Papazoglou, 2003).

V oblasti elektronického obchodu sa XML webové služby používajú na poskytovanie aktuálnych informácií o produktoch, ich dostupnosti, cenách a parametroch. Tieto služby sú často používané v kombinácii s relačnými databázami, ktoré uchovávajú podrobné údaje o produktoch a ich vlastnostiach (Alonso et al. 2004). Výhodou XML webových služieb je ich schopnosť pracovať s rôznymi typmi klientov, či už ide o desktopové aplikácie, mobilné zariadenia alebo webové rozhrania (Erl, 2005).

2.1.1 Vývoj XML webových služieb

XML (eXtensible Markup Language) je značkovací jazyk, ktorý sa široko používa na reprezentáciu a prenos dát v rôznych oblastiach, vrátane webových služieb (Erl, 2005). Webové služby predstavujú spôsob výmeny dát medzi rôznymi aplikáciami a systémami

prostredníctvom internetu. XML webové služby sú špecifické tým, že využívajú XML na štandardizovaný prenos dát medzi klientom a serverom. Tento prenos prebieha väčšinou cez protokoly ako HTTP a HTTPS (Microsoft, n.d. -a).

XML webové služby sa začali vyvíjať v 90. rokoch 20. storočia, keď internet začal byť kľúčovým nástrojom na prepojenie rôznych systémov (Alonso et al., 2004). Predtým boli informačné systémy prevažne izolované, čo mohlo znamenať problémy s integráciou medzi rôznymi platformami a aplikáciami. Vznik XML, ako nezávislého formátu na výmenu dát, umožnil rozšírenie webových služieb na rôzne technologické platformy.

V roku 1998 bol vyvinutý protokol SOAP (Simple Object Access Protocol), ktorý sa stal základom pre komunikáciu medzi aplikáciami cez XML webové služby (Gudgin et al., 2007). SOAP poskytuje definovaný formát správy, ktorá je založená na XML, a umožňuje jej prenos medzi klientom a serverom prostredníctvom protokolu HTTP alebo SMTP. SOAP sa stal štandardom pre XML webové služby v začiatkoch, pretože podporoval vzájomné prepojenie medzi rôznymi platformami a aplikáciami.

V roku 2000 bola definovaná špecifikácia WSDL (Web Services Description Language), ktorá slúži na opis a definovanie webových služieb (W3C, 2001). WSDL je tiež formátovaný v XML a poskytuje presný popis toho, aké operácie sú dostupné prostredníctvom danej webovej služby, ako aj parametre a návratové hodnoty týchto operácií. Tento formát umožňuje automatické generovanie klientskych aplikácií, ktoré môžu komunikovať s webovými službami (Alonso et al., 2004).

V rovnakom období vznikla aj špecifikácia UDDI (Universal Description, Discovery, and Integration), ktorá poskytuje metódy na registráciu a vyhľadávanie webových služieb. UDDI bol základom pre vytvorenie verejných a súkromných katalógov webových služieb, čo umožnilo organizáciám nájsť a integrovať potrebné služby (W3C, 2004).

Aj keď SOAP bol spočiatku dominantným protokolom pre webové služby, v posledných rokoch sa začal presadzovať alternatívny prístup – REST (Representational State Transfer). REST využíva jednoduché HTTP požiadavky (GET, POST, PUT, DELETE) na komunikáciu medzi klientom a serverom a typicky využíva formáty ako JSON, ale aj XML. Hoci XML zostáva jedným z podporovaných formátov v rámci REST, JSON je častejšie používaný kvôli svojej ľahšej správe a rýchlejšiemu prenosu dát (Richardson & Ruby, 2007).

Aj keď REST je v súčasnosti populárnejší kvôli svojej jednoduchej implementácii a nízkej záťaži, XML webové služby na báze SOAP zostávajú v mnohých podnikových systémoch, kde je potrebná prísna štruktúra a bezpečnostné požiadavky (Taft, 2007). Webové služby implementované prostredníctvom XML sa používajú najmä v prípadoch, kde je potrebná vysoká prepojenosť medzi rôznymi platformami, ako sú systémy v bankovníctve, financiách, a veľkých korporáciách (Erl, 2005).

Výhody XML webových služieb spočívajú v ich štandardizácii, prenosnosti a schopnosti pracovať s rôznymi platformami a technológiami. XML poskytuje flexibilitu pri definovaní štruktúry dát, a preto je ideálny na prenos komplexných a hierarchických dát, akými sú informácie o produktoch v e-commerce systémoch.

Medzi hlavné výzvy patrí vyššia náročnosť na výkon pri prenose veľkých objemov dát, ako aj zložitosť pri spracovaní XML súborov. Na porovnanie, formáty ako JSON sú menej náročné na prenos a spracovanie, čo robí z REST alternatívu, ktorá je vhodnejšia pre rýchle aplikácie (Richardson & Ruby, 2007).

Budúcnosť XML webových služieb bude pravdepodobne zahŕňať vylepšenia v oblasti bezpečnosti, výkonu a prepojenia s inými technológiami. Nové prístupy, ako je GraphQL alebo WebSocket, by mohli získať na popularite, ale XML bude stále nástrojom pre komplexnejšie integrácie a väčšie systémy (Microsoft, n.d. -a).

2.2 Metódy sledovania parametrov výpočtovej techniky

Sledovanie parametrov, ako sú notebooky, stolné počítače, tablety a príslušenstvo výpočtovej techniky v elektronických obchodoch, predstavuje proces, ktorý zabezpečuje správne fungovanie a správu produktov. Tieto parametre môžu zahŕňať rôzne technické špecifikácie, ceny, stavy zásob a iné obchodné informácie. Existuje niekoľko metód sledovania týchto parametrov, ktoré sa líšia podľa typu technológie, rozsahu sledovaných údajov a spôsobu ich správy (Miller, 2019).

2.2.1 Tradičné metódy sledovania

Tradične sa parametre výpočtovej techniky sledovali pomocou relačných databáz. Tento prístup zahŕňa centralizované ukladanie informácií o produktoch do tabuliek, ktoré sa neskôr spracúvajú a analyzujú. Relačné databázy ako MySQL, PostgreSQL a Microsoft SQL

Server sú stále veľmi rozšírené v elektronických obchodoch, kde sa uchovávajú údaje o produktoch, skladových zásobách, objednávkach a predajoch (Wang & Lee, 2018).

Výhody:

- Relačné databázy umožňujú rýchly prístup k dátam a jednoduché dotazovanie cez SQL.
- Vysoká úroveň bezpečnosti a transakčnej integrity, čo je kľúčové pri správe citlivých údajov (napr. ceny, dostupnosť).
- Možnosť rozširovať databázy, ako aj robustné nástroje pre zálohovanie a obnovu dát.

Nevýhody:

- V prípade veľkých objemov dát môžu byť relačné databázy zložitejšie na spravovanie.
- Relačné databázy sú najvhodnejšie na ukladanie štruktúrovaných dát, ale pri potrebe pracovať s rôznymi formátmi (napr. neštruktúrované dáta, multimediálne súbory) môžu byť menej efektívne (Smith, 2020).

2.2.2 XML a XML webové služby

XML (eXtensible Markup Language) sa stal veľmi populárnym formátom pre prenos a ukladanie údajov v mnohých oblastiach, vrátane elektronických obchodov. XML poskytuje voľnosť pri štruktúrovaní údajov a je platformovo nezávislý, čo ho robí vhodným prostriedkom pre výmenu dát medzi rôznymi systémami (Johnson, 2017).

Výhody XML:

- XML umožňuje definovať vlastnú štruktúru údajov, čo je užitočné pri sledovaní rozmanitých parametrov výpočtovej techniky, ako sú technické špecifikácie, výrobca, typ produktu, stav zásob atď.
- XML môže byť použitý na prenos dát medzi rôznymi systémami, čo je výhodné, ak obchod používa rôzne platformy na správu dát.
- Webové služby založené na XML (ako je SOAP) umožňujú efektívnu výmenu údajov medzi rôznymi aplikáciami, čím uľahčujú správu produktov v online obchode (Harris, 2019).

Nevýhody XML:

- Pri spracovaní veľkých objemov dát môže byť XML menej efektívne v porovnaní s inými formátmi, ako je JSON, ktorý je menej náročný na pamäť a rýchlosť spracovania.
- Na prácu s XML sa vyžaduje určitá úroveň technickej zdatnosti, najmä pri tvorbe komplexných štruktúr a ich validácii (Lee, 2021).

2.2.3 REST API a JSON

S rastúcim záujmom o moderné webové aplikácie sa čoraz viac začína používať REST (Representational State Transfer) na komunikáciu medzi aplikáciami, pričom sa na prenos dát stále častejšie používa formát JSON (JavaScript Object Notation). JSON je ľahší, rýchlejší a menej náročný na spracovanie ako XML, čo ho robí vhodným formátom v mnohých systémoch (Thomas & Walker, 2020).

Výhody REST a JSON:

- JSON je ľahší na spracovanie a menej náročný na šírku pásma, čo z neho robí efektívny formát pre webové aplikácie.
- V mobilných aplikáciách je REST s JSON často voľbou číslo jeden, pretože poskytuje jednoduchý a rýchly prístup k dátam.
- REST umožňuje lepšiu škálovateľnosť aplikácií a flexibilitu pri práci s rôznymi typmi požiadaviek (GET, POST, PUT, DELETE).

Nevýhody:

- Neponúka rovnakú úroveň bezpečnosti a transakčnej integrity ako SOAP, čo môže byť problém v prípadoch, kde je potrebná prísna kontrola prenosu dát.
- JSON nie je vhodný pre prenos veľmi komplexných alebo hierarchických dát, ktoré by bolo lepšie reprezentovať pomocou XML (Brown, 2019).

2.2.4 Kombinované prístupy

Mnohé elektronické systémy využívajú kombinované prístupy na sledovanie parametrov výpočtovej techniky. Tento prístup zahŕňa kombináciu XML webových služieb,

REST API, IoT senzorov a ďalších technológií, čím sa zabezpečuje efektívna a lepšia správa produktových informácií v rôznych formátoch a z rôznych zdrojov (Smith et al., 2021).

Výhody kombinovaných prístupov:

- Kombinovaním viacerých technológií sa dosahuje širšia funkčnosť a lepšia adaptabilita na rôzne situácie.
- Sledovanie a správa dát pomocou viacerých prístupov umožňuje získanie komplexného prehľadu o výpočtovej technike v online obchode.

Nevýhody:

- Integrácia rôznych technológií a systémov môže byť technicky náročná a môže si vyžadovať dodatočné investície do vývoja a údržby (Johnson & Miller, 2021).

2.3 Porovnanie XML webových služieb s inými metódami sledovania

Sledovanie parametrov výpočtovej techniky v elektronických obchodoch, ako sú notebooky, stolné počítače, tablety a príslušenstvo, si vyžaduje rôzne metódy na správu a prenos dát. XML webové služby, ako technológia, ktorá umožňuje výmenu a správu týchto dát medzi rôznymi aplikáciami a systémami, sa často porovnávajú s inými modernými metódami sledovania, ako sú REST API, relačné databázy, IoT technológie a využívanie formátu JSON (Alshuqayran & Shamkhi, 2022). Každá z týchto metód má svoje výhody a nevýhody v závislosti od konkrétneho použitia, typu aplikácie a požiadaviek na výkon, flexibilitu a bezpečnosť (Erl, 2005).

2.3.1 XML webové služby vs REST API

XML webové služby (SOAP) a REST API umožňujú prenos a spracovanie dát medzi servermi a klientmi. Hlavný rozdiel medzi nimi spočíva v protokole a štruktúre (Fielding, 2000).

Štruktúra dát:

XML webové služby (SOAP) používajú XML na štruktúrovanie a prenos dát. XML poskytuje presnú a rozsiahlu definíciu štruktúry údajov, čo je ideálne pre aplikácie, ktoré vyžadujú vysokú bezpečnosť a prísnu kontrolu nad prenosom dát.

REST API využíva jednoduchšie formáty ako JSON alebo XML na prenos dát, pričom JSON je v súčasnosti častejšie používaný kvôli svojej jednoduchej štruktúre a menšej náročnosti na spracovanie.

Výkon:

SOAP webové služby sú vo všeobecnosti náročnejšie na spracovanie a výkon, pretože každý prenos dát musí byť spracovaný v úplnej XML štruktúre, čo môže byť náročné pri veľkých objemoch dát.

REST API je efektívnejšie v prípade jednoduchých a ľahkých požiadaviek, najmä v prípade prenosu menších objemov dát v reálnom čase.

Bezpečnosť:

SOAP ponúka mechanizmy pre zabezpečenie komunikácie, ako je WS-Security, ktorý umožňuje šifrovanie a autentifikáciu.

REST API je menej zamerané na bezpečnosť na úrovni protokolu a viac závisí na implementácii zabezpečenia cez HTTPS a OAuth.

Prepojiteľnosť:

SOAP je viac prepojiteľný s rôznymi platformami a technológiami, najmä v prípade komplexnejších podnikových systémov, ktoré musia komunikovať s inými systémami.

REST API je flexibilnejšie, ale môže mať obmedzenia v prípade komunikácie medzi systémami s odlišnými technickými požiadavkami.

2.3.2 XML webové služby vs Relačné databázy

Relačné databázy (napr. MySQL, PostgreSQL, Microsoft SQL Server) sú tradičným spôsobom ukladania a spravovania štruktúrovaných dát. V porovnaní s XML webovými službami sa zameriavajú na iný aspekt správy a prenosu dát (Stonebraker & Moore, 2010).

Ukladanie dát:

Poskytujú nástroj na efektívne ukladanie veľkých objemov štruktúrovaných dát, ako sú informácie o produktoch v elektronických obchodoch. Sú ideálne pre veľké množstvá záznamov, ktoré je potrebné efektívne triediť a spracovávať pomocou SQL dotazov.

XML webové služby na druhej strane neukladajú dáta, ale poskytujú mechanizmus na ich prenos medzi rôznymi systémami, čo znamená, že XML môže byť použitý na prenos dát medzi rôznymi databázami alebo aplikáciami.

Výhody a nevýhody:

Relačné databázy ponúkajú vysoký výkon a rýchle vyhľadávanie štruktúrovaných dát, ale neponúkajú priamy spôsob prenosu dát medzi rôznymi aplikáciami alebo systémami. Pre tento účel je potrebné využiť iné technológie, ako sú webové služby.

XML webové služby sú silné v oblasti komunikácie medzi rôznymi aplikáciami, ale nie sú vhodné na ukladanie veľkých objemov dát alebo na ich analýzu priamo v databáze. Pre tento účel je potrebné kombinovať XML webové služby s relačnými databázami alebo inými formami ukladania dát.

2.4 Implementácia XML Webovej služby v ASP .NET

V prostredí ASP .NET, vytvorenie XML webovej služby zahŕňa vytvorenie serverového rozhrania, ktoré poskytuje prístup k rôznym funkciám aplikácie cez štandardizovaný protokol (väčšinou HTTP) a formát (väčšinou XML). Tento prístup umožňuje výmenu dát medzi rôznymi systémami a platformami, čo je kľúčové pre prepojenie v distribuovaných aplikáciách (Microsoft, 2023). V tejto časti sa zameriame na teoretický popis implementácie XML webovej služby v prostredí ASP .NET.

2.4.1 Definícia a účel XML Webovej služby v ASP .NET

XML webová služba je typ webovej aplikácie, ktorá poskytuje metódy pre vzdialený prístup cez internet alebo sieť, pričom spracováva požiadavky a odpovedá v štruktúrovanej forme. Webové služby sú založené na otvorených štandardoch ako WSDL pre popis služieb a SOAP pre prenos správ (W3Schools, 2023). V ASP .NET sa tieto služby implementujú pomocou šablóny ASMX, ktorá poskytuje jednoduché rozhranie na vytvorenie webových služieb v jazyku C# alebo VB.NET (Microsoft Learn, 2023).

XML webové služby v ASP .NET umožňujú aplikáciám poskytovať funkcionalitu, ako je získavanie informácií o produktoch, spracovanie objednávok a interakcia s inými systémami v elektronickom obchode. Tieto služby sú obzvlášť užitočné pre systémy, ktoré potrebujú byť kompatibilné s rôznymi platformami a systémami, ako aj pre aplikácie, ktoré musia efektívne komunikovať so vzdialenými servermi.

2.4.2 Kľúčové komponenty a protokoly

Na to, aby webová služba v ASP .NET fungovala efektívne, je nevyhnutné pochopiť niekoľko základných komponentov a protokolov, ktoré sa používajú na jej implementáciu:

WSDL je XML formát, ktorý popisuje dostupné operácie webovej služby, štruktúru vstupov a výstupov a spôsoby pripojenia. Tento popis je automaticky generovaný a poskytuje klientom informácie o tom, ako sa môžu pripojiť a komunikovať s webovou službou (W3Schools, 2023).

SOAP je protokol, ktorý umožňuje prenos správ medzi serverom a klientom. Využíva XML na štruktúrované požiadavky, odpovede a umožňuje komunikáciu cez rôzne transportné protokoly Mozilla Developer Network, 2023).

UXML (Universal XML): Jednoduchý formát na prenos a uchovávanie údajov v XML formáte medzi serverom a klientom, umožňujúci štandardizovaný spôsob prenosu dát (Developer.com, 2021).

2.4.3 Štruktúra a organizácia webovej služby

V prostredí ASP .NET sa webová služba implementuje pomocou súborov ASMX a .cs (pre C#), ktoré definujú logiku webovej služby. Základné kroky implementácie sú nasledovné:

- Vytvorenie projektu webovej služby: Začína sa vytvorením ASP.NET projektu v prostredí Visual Studio, kde si používateľ vyberie šablónu pre webovú službu. Tento projekt bude obsahovať všetky potrebné komponenty na definovanie a správu metód webovej služby.
- Definovanie metód a operácií webovej služby: Každá metóda, ktorá má byť prístupná klientskej aplikácii, musí byť označená atribútom *[WebMethod]*. Tento atribút zabezpečuje, že metóda bude dostupná cez HTTP požiadavky a bude môcť komunikovať s inými aplikáciami.
- WSDL popisuje, aké metódy sú dostupné a aký formát dát sa očakáva pri komunikácii s webovou službou. ASP .NET automaticky generuje tento popis pre všetky webové služby.

Prenos dát medzi klientom a serverom prebieha prostredníctvom SOAP správ, ktoré sú zabalené v XML formáte. Klient posiela požiadavky vo formáte XML na server, ktorý ich spracuje a odpovedá tiež vo formáte XML. Tento prenos je štandardizovaný, čo znamená, že klient a server môžu komunikovať bez ohľadu na technológie, ktoré používajú (W3C, 2023).

Zabezpečenie webových služieb je nevyhnutné, najmä pri prenose citlivých informácií, ako sú osobné údaje používateľov alebo finančné transakcie. ASP .NET poskytuje niekoľko spôsobov, ako zabezpečiť komunikáciu s webovou službou:

- Pre šifrovanú komunikáciu sa používa HTTPS, ktoré zabezpečuje, že prenos dát medzi klientom a serverom je šifrovaný a chránený pred zachytávaním.
- Webová služba môže implementovať rôzne metódy autentifikácie (napr. pomocou tokenov alebo certifikátov), aby zabezpečila prístup k citlivým dátam len oprávneným používateľom.

- WS-Security: Tento protokol sa používa na implementáciu bezpečnostných mechanizmov priamo do SOAP správ, vrátane podpory šifrovania a podpisovania správ (OASIS, 2022).

2.4.4 Výhody a Nevýhody XML Webových služieb v ASP .NET

Výhody:

Prepojiteľnosť: Webové služby podporujú rôzne platformy a technológie, čo znamená, že aplikácie môžu komunikovať medzi sebou bez ohľadu na jazyk alebo operačný systém.

Škálovateľnosť: Webové služby umožňujú škálovanie aplikácie, pretože každý klient môže využiť rovnakú službu bez potreby inštalácie dodatočného softvéru.

Štandardizácia: Použitie XML a SOAP znamená, že komunikácia je štandardizovaná a široko podporovaná.

Nevýhody:

Výkon: Práca s XML a SOAP môže byť náročná na výpočtové zdroje, čo môže mať vplyv na výkon aplikácie, najmä pri práci s veľkými objemami dát.

Zložitosť implementácie: Zatiaľ čo implementácia webových služieb v ASP .NET je jednoduchá, implementácia pokročilých bezpečnostných mechanizmov a správne spracovanie chýb si vyžaduje podrobné technické znalosti.

3 Cieľ práce

Cieľom diplomovej práce je analýza možnosti využitia XML webových služieb na sledovanie parametrov výpočtovej techniky v kontexte elektronického obchodu, konkrétne pre produkty ako notebooky, stolné počítače, tablety a príslušenstvo. Práca sa zameriava na vytvorenie efektívneho riešenia na sledovanie týchto parametrov v elektronickom informačnom systéme počas vybraného obdobia pomocou rôznych webových metód pre tento proces.

V rámci práce vytvoríme ASP.NET XML webovú službu, ktorá poskytne klientovi usporiadané informácie o parametroch výpočtovej techniky v sledovanom období. Medzi týmito informáciami bude zahrnutý celkový počet jednotlivých druhov zariadení, ako sú notebooky, stolné počítače, tablety, v obchode na začiatku a na konci sledovaného obdobia. Zároveň sa zameriame na analýzu počtov a zoznamov produktov podľa jednotlivých druhov zariadení, kde budú zahrnuté základné údaje od výrobcov ako sú cena, technické parametre, dostupnosť na sklade, ako aj ďalšie relevantné parametre. Služba tiež umožní poskytovať informácie o produktoch vybranej značky alebo kategórie podľa rôznych filtrov.

Pri vývoji XML webovej služby budeme používať ASP .NET, konkrétne framework ASP .NET Web API, ktorý umožňuje jednoduchú tvorbu webových služieb s podporou XML formátu. Služba bude schopná poskytovať štatistiky o produktoch a objednávkach, filtrovať a zoradiť údaje na základe rôznych kritérií. Tieto metódy budú implementované priamo v backendovej logike webovej služby.

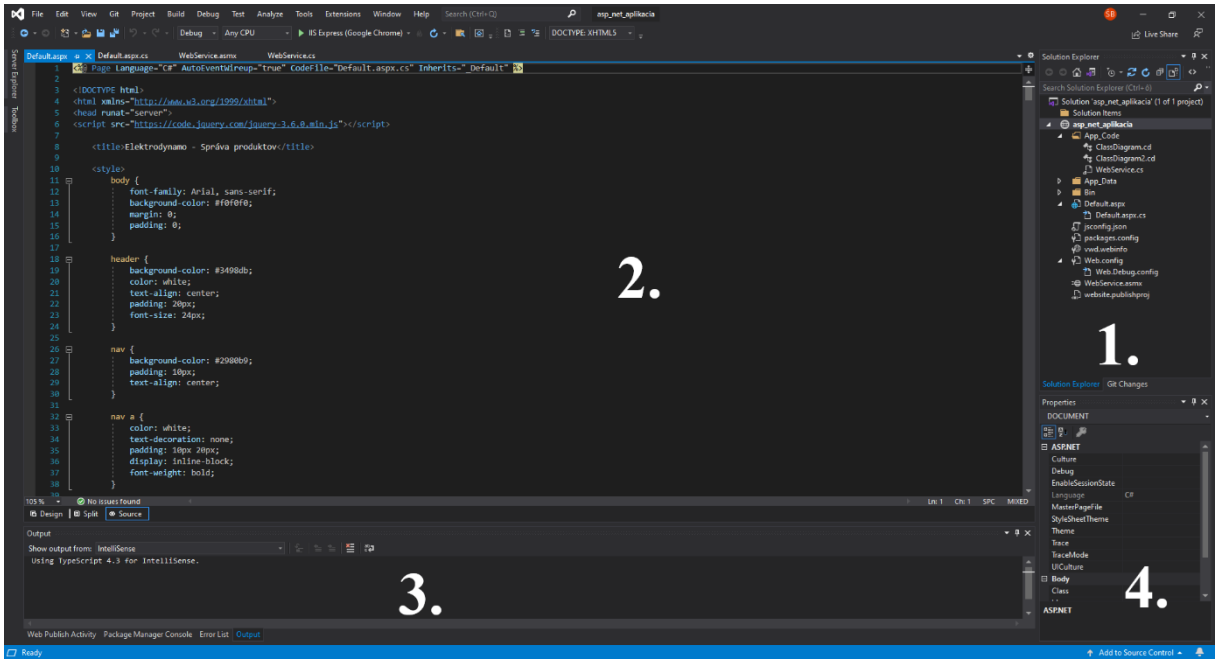
Na manipuláciu s údajmi budeme používať XML súbory, kde každý produkt bude reprezentovaný samostatným elementom. Tieto XML súbory budú obsahovať informácie ako ID produktu, názov, značka, model, cena, stav skladu, externý sklad a popis. Tento spôsob ukladania a organizácie údajov umožní flexibilné a efektívne spracovanie produktových informácií, pričom sa zachová prehľadnosť a jednoduchosť pri práci s dátami.

Webová služba bude obsahovať rôzne metódy pre manipuláciu s údajmi, ako sú pridávanie nových produktov, odstraňovanie zastaralých alebo neaktuálnych produktov, aktualizovanie informácií o produktoch a generovanie štatistických prehľadov. Tieto prehľady budú zahŕňať analýzu dát podľa rôznych kritérií, ako sú cena, množstvo, rok výroby, kvartál, atď.

4 Popis vývojového prostredia

Vývoj webovej služby na sledovanie výpočtovej techniky v elektronickom obchode si vyžaduje vhodné technologické zázemie a vývojové prostredie, ktoré zabezpečí spoľahlivosť a integráciu s ďalšími nástrojmi. Pre tieto účely bolo zvolené vývojové prostredie Microsoft Visual Studio, konkrétne vo verzii Visual Studio 2019, ktoré predstavuje jedno z najrozšírenejších a najkomplexnejších integrovaných vývojových prostredí (IDE). Výber Visual Studia ako hlavného nástroja pre vývoj aplikácie bol odôvodnený najmä jeho podporou pre technológiu ASP.NET, funkcionalitou, možnosťou práce s XML, intuitívnym editorom kódu a možnosťami pre ladenie, testovanie a správu projektu v jednotnom prostredí.

Visual Studio poskytuje vývojárom infraštruktúru na tvorbu riešení v jazyku C# s použitím súborov XML ako zdroja dát. V tejto práci Visual Studio zohráva kľúčovú úlohu – od návrhu štruktúry XML súborov, tvorby tried, vývoja metód v rámci webovej služby, až po nasadenie a testovanie služby v prostredí IIS Express. Prvým krokom bolo vytvorenie nového projektu typu „ASP.NET Web Service (.asmx)“. Po zvolení šablóny Visual Studio automaticky vygenerovalo základnú štruktúru projektu, ktorá obsahuje preddefinovaný súbor *WebService.asmx* a hlavný zdrojový súbor *WebService.cs*, v ktorom boli definované všetky potrebné metódy.



Obrázok 1 Vývojové prostredie Visual Studio

Rozhranie Visual Studia pozostáva z viacerých panelov, pričom najdôležitejšie z nášho hľadiska vývoja boli nasledovné:

1. **Solution Explorer**: slúžil na navigáciu medzi jednotlivými súbormi, triedami a referenciami. Umožnil prístup k XML súborom, zdrojovým kódom, konfiguračným súborom a testovacím súborom.
2. **Code Editor:** hlavný priestor pre písanie a úpravu kódu, ktorý disponuje inteligentným dopĺňaním kódu, zvýrazňovaním syntaxe, blokmi kódu a integrovanou kontrolou chýb.
3. **Output Window a Error List:** nástroje na diagnostiku počas vývoja. Zobrazovali chyby pri kompilácii, varovania, výstupy z ladenia a hlásenia zo systému.
4. **Properties Window:** umožňovalo meniť vlastnosti komponentov, objektov a súborov bez nutnosti zásahov do samotného kódu.

V prostredí Visual Studio boli zároveň vytvorené dve pomocné triedy – *Product* a *Order*, ktoré slúžili ako dátové modely. Tieto triedy boli implementované ako C# objekty s vlastnosťami zodpovedajúcimi jednotlivým XML elementom ako napr. <*name*>, <*brand*>, <*price*>, <*timestamp*> atď. V prostredí Visual Studio sme definovali tieto triedy vo forme *public class*, pričom jednotlivé vlastnosti boli deklarované pomocou typov *string, decimal, DateTime a int*. Práca s XML bola zabezpečená cez triedy ako *XmlDocument, XmlElement, XmlNodeList a XPath*, ktoré boli integrované priamo do logiky webovej služby.

Samotné metódy webovej služby boli definované ako vystavené metódy s atribútom *[WebMethod],* čo Visual Studio automaticky vyhodnocuje ako súčasť zverejneného rozhrania služby. Každá metóda bola naprogramovaná tak, aby najprv načítala XML súbor (pomocou *XmlDocument.Load()*), následne aplikovala filtrovanie, triedenie alebo iné operácie a výsledok serializovala späť do objektov C# a XML reťazcov.

Webovú službu sme spúšťali pomocou IIS Express, čo umožnilo testovať funkčnosť služby ešte pred jej nasadením na server. Visual Studio zároveň vygenerovalo WSDL súbor, ktorý slúži ako dokumentácia a rozhranie pre klientov, ktorí chcú službu využívať.

Pri tvorbe používateľského rozhrania pre správu výpočtovej techniky bola aplikácia navrhnutá tak, aby bola schopná komunikovať s webovou službou prostredníctvom JavaScriptu a AJAX volaní.

Na strane frontendu boli testované jednoduché HTML a JavaScriptové komponenty, ktoré využívali odpovede zo služby vo formáte XML a následne ich zobrazovali na stránke.

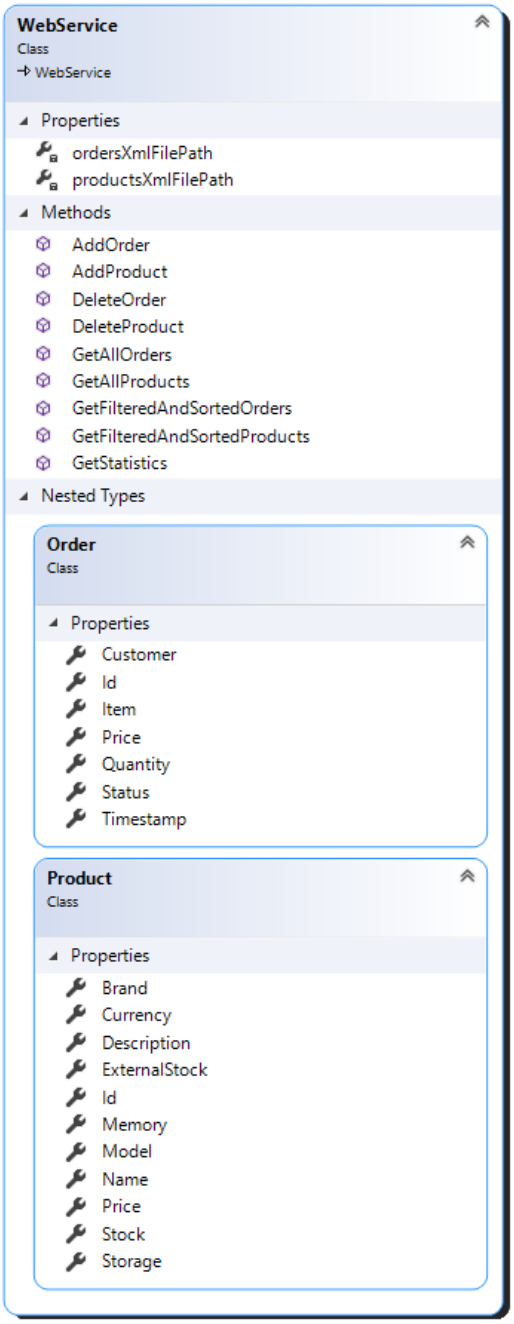
Celkovo bolo Visual Studio vhodným nástrojom počas celej realizácie projektu. Poskytlo stabilné prostredie, ktoré umožnilo efektívny vývoj, testovanie a ladenie webovej služby, ako aj údržbu projektu počas jeho rozširovania. Vďaka integrácii všetkých nástrojov a intuitívnemu prostrediu bolo možné sústrediť sa na samotnú logiku aplikácie bez ďalších technických komplikácií.

5 Návrh a štruktúra webovej služby

V rámci návrhu aplikačnej logiky ASP.NET XML Web Service pre elektronický obchod Elektrodynamo.sk bol vytvorený objektovo orientovaný model reprezentovaný diagramom tried, ktorý znázorňuje vzťahy medzi kľúčovými entitami systému. Tento diagram slúži ako vizuálna reprezentácia dátovej vrstvy a vychádza z analýzy XML štruktúry produktového súboru *xmlko.xml* a súboru objednávok *elektrodynamo_objednavky.xml*. Tieto súbory sú deklarované v globálnych premenných v rámci súboru WebService.cs:

```
private string productsXmlFilePath => HttpContext.Current.Server.MapPath
("~/App_Data/xmlko.xml");
private string ordersXmlFilePath => HttpContext.Current.Server.MapPath
("~/App_Data/elektrodynamo_objednavky.xml");
```

Obrázok 2 Inicializácia globálnych premenných s XML súbormi (Zdroj: vlastné spracovanie)



Obrázok 3 Diagram tried webovej služby

Na najvyššej úrovni sa nachádza trieda *WebService*, ktorá predstavuje hlavnú implementačnú triedu ASP.NET webovej služby. Táto trieda obsahuje množinu verejne dostupných metód ([*WebMethod*]), ktoré zabezpečujú komunikáciu medzi klientskou aplikáciou a serverom. Všetky metódy spracovávajú údaje uložené v XML súboroch pomocou pomocných tried a LINQ dotazov.

Diagram ďalej obsahuje triedu *Product*, ktorá reprezentuje jeden produkt v elektronickom obchode. Táto trieda obsahuje nasledovné vlastnosti:

- Id (string): identifikátor produktu,
- Name (string): názov produktu,
- Brand (string): značka výrobcu produktu,
- Model (string): označenie konkrétneho modelu,
- Memory (string): kapacita operačnej pamäte,
- Storage (string): veľkosť interného úložiska,
- Price (decimal): cena produktu,
- Currency (string): mena produktu,
- Stock (int): množstvo produktov dostupných na sklade,
- ExternalStock (int): množstvo produktov dostupných v externom sklade,
- Description (string): popis produktov.

Trieda *Product* je využívaná vo viacerých metódach triedy *WebService* na čítanie, filtrovanie, úpravu a export údajov o produktoch.

Druhou kľúčovou entitou je trieda *Order*, ktorá reprezentuje jednu zákaznícku objednávku. Jej vlastnosti sú nasledovné:

- Id (string): identifikátor objednávky (vo formáte YYMMDDNN),
- Timestamp (DateTime): dátum a čas uskutočnenia objednávky,
- CustomerName (string): meno zákazníka,
- CustomerEmail (string): e-mailová adresa zákazníka,
- CustomerPhone (string): telefónne číslo zákazníka,
- ProductId (string): identifikátor objednaného produktu,
- Quantity (int): počet kusov objednaného produktu,
- TotalPrice (decimal): celková cena objednávky.

Objekty typu *Order* sú využívané pri spracovaní a evidencii zákazníckych objednávok, pričom údaje sú ukladané do samostatného XML súboru.

Triedy *Product* a *Order* sú navzájom nezávislé, avšak logicky prepojené prostredníctvom *ProductId*, ktoré sa vyskytuje ako atribút objednávky. Týmto spôsobom je zabezpečené oddelenie údajov o produktoch a o zákazníkoch pri zachovaní funkčnej väzby medzi nimi.

Obidve dátové triedy sú jednoduché dátové objekty (DTO), ktoré slúžia ako nosiče informácií medzi aplikačnými vrstvami. Všetky vlastnosti majú verejný prístup a sú implementované ako automaticky generované vlastnosti, čo zjednodušuje serializáciu a deserializáciu údajov z XML súborov.

Diagram tried zároveň vizualizuje závislosť medzi triedou *WebService* a dátovými triedami *Product* a *Order,* kde *WebService* obsahuje viacero metód, ktoré buď vracajú zoznamy inštancií týchto tried, alebo ich priamo manipulujú.

Tento objektový model bol navrhnutý s cieľom zabezpečiť maximálnu zrozumiteľnosť a udržateľnosť kódu, ako aj kompatibilitu s ďalšími systémami, ktoré by mohli v budúcnosti pristupovať k dátam pomocou rovnakého webového rozhrania. Zároveň umožňuje rozšírenie o ďalšie vlastnosti produktov či objednávok bez zásadných zmien v architektúre služby.

5.1 Metóda *AddProduct* na pridanie produktu do vstupného XML súboru

Metóda *AddProduct* slúži na pridanie nového produktu do systému. Táto metóda je základnou súčasťou správy produktov v rámci ASP.NET XML webovej služby. Cieľom je umožniť používateľom alebo administrátorom pridávať nové položky do online obchodu, čím zabezpečuje, že zoznam dostupných produktov bude aktuálny a bude sa dať spravovať v reálnom čase. Tento proces zahŕňa vytvorenie nového záznamu v XML súbore, ktorý obsahuje všetky relevantné údaje o produkte.

Metóda sa začína kontrolou, či súbor produktov (xmlko.xml) existuje. Ak tento súbor ešte neexistuje, vytvorí sa nový dokument XML s koreňovým elementom <*store*>, ktorý obsahuje podelement <*products*> a ktorý je predpripravený na pridávanie produktov. Tento krok zabezpečuje, že aj pri prvej interakcii s webovou službou, kde ešte neexistuje žiadny produkt, sa vytvorí potrebná štruktúra XML pre ďalšie operácie.

```
if (!File.Exists(productsXmlFilePath))
{
        XDocument newDoc = new XDocument(new XElement("store",
        new XElement("products")));
        newDoc.Save(productsXmlFilePath);
}
```

Obrázok 4 Kontrola existencie súboru so záznamami o produktoch (Zdroj: vlastné spracovanie)

Týmto spôsobom sa zabezpečuje, že ak súbor neexistuje, bude automaticky vytvorený s počiatočnou štruktúrou, čo zjednodušuje ďalšiu manipuláciu s údajmi.

Po tejto kontrole sa súbor načíta pomocou triedy *XDocument*, ktorá umožňuje manipuláciu s XML dokumentmi. Ďalej sa vyhľadá element <*products*>, do ktorého sa pridajú nové produkty. Tieto produkty sú reprezentované elementom <*product*>, ktorý obsahuje podrobnosti o konkrétnom produkte. Každý nový produkt je vybavený identifikátorom (ID), ktorý sa generuje automaticky na základe aktuálneho dátumu a počtu produktov pridaných v daný deň.

```
string today = DateTime.Now.ToString("yyMMdd");
int countToday = root.Elements("product")
        .Where(p => p.Element("id")?.Value.StartsWith(today) ??
        false).Count();
string newId = today + countToday.ToString("D2");
```

Obrázok 5 Generovanie ID produktu (Zdroj: vlastné spracovanie)

Tento identifikátor je založený na aktuálnom dátume vo formáte "yyMMdd", nasledovanom číslom, ktoré sa zvyšuje pre každý nový produkt pridaný v ten deň. Tento spôsob generovania ID zaručuje, že každý produkt bude mať jedinečný a predvídateľný identifikátor.

Po vygenerovaní ID je produkt pridaný do XML dokumentu ako nový element <*product*>, ktorý obsahuje všetky potrebné informácie, ako sú názov, značka, model, technické špecifikácie (pamäť, úložisko), cena, mena, skladové množstvo a popis. Tieto údaje sú získané z parametrov metódy *AddProduct*.

```
XElement newProduct = new XElement("product",
   new XElement("id", newId),
   new XElement("name", name),
   new XElement("brand", brand),
   new XElement("model", model),
   new XElement("memory", memory),
   new XElement("storage", storage),
   new XElement("price", price),
   new XElement("currency", currency),
   new XElement("stock", stock),
   new XElement("external_stock", externalStock),
   new XElement("description", description)
);
```

Obrázok 6 Vytvorenie nového elementu <product> (Zdroj: vlastné spracovanie)

Po pridaní nového produktu do XML dokumentu sa dokument uloží do pôvodnej cesty (*productsXmlFilePath*), čím sa zmeny zaznamenajú.

```
root.Add(newProduct);
doc.Save(productsXmlFilePath);
```

Obrázok 7 Pridanie nového elementu <product> (Zdroj: vlastné spracovanie)

Metóda sa končí návratom správy, ktorá potvrdzuje, že produkt bol úspešne pridaný, a obsahuje ID nového produktu.

```
return "Produkt bol úspešne pridaný s ID: " + newId;
```

Obrázok 8 Výpis informácie o úspešnom pridaní elementu (Zdroj: vlastné spracovanie)

Celkovo metóda *AddProduct* umožňuje spravovať produktový inventár v rámci online obchodu. Použitím automatického generovania týchto identifikátorov na základe aktuálneho dátumu je možné udržiavať prehľad o produktoch, ktoré boli pridané počas rôznych dní. Tým sa zabraňuje problémom s duplikovaním identifikátorov a zabezpečuje sa konzistentnosť údajov.

Tento prístup je užitočný, keď je potrebné pridať nový produkt do systému v reálnom čase, bez toho, aby sa manipulovalo s databázami alebo inými externými systémami. XML súbor poskytuje spôsob uchovávania týchto údajov, čím sa zabezpečuje ich dostupnosť a spravovateľnosť aj bez potreby databázového systému.

5.2 Metóda *GetProduct* na načítanie produktov zo vstupného XML súboru

Jednou z ďalších metód v rámci vytvorenej ASP.NET XML Webovej služby je metóda *GetAllProducts()*. Jej hlavným účelom je sprístupniť úplný zoznam všetkých produktov uložených v externom XML súbore. Tento zoznam slúži ako vstupný bod pre ďalšiu prácu so systémom, vrátane filtrovania, analýz, objednávok alebo pre jednoduché zobrazenie dostupných položiek vo webovom rozhraní.

Z hľadiska technickej realizácie je označená atribútmi *[WebMethod]* a *[ScriptMethod(ResponseFormat = ResponseFormat.Json)]*, ktoré zabezpečujú jednak to, že metóda je dostupná cez HTTP ako webová služba, a zároveň že odpoveď vráti vo formáte JSON. Práve JSON je v súčasnosti štandardný dátový formát pre komunikáciu medzi backendom a frontendom, najmä pri aplikáciách, ktoré využívajú AJAX alebo JavaScript.

```
if (!File.Exists(productsXmlFilePath))
{
    return "[]";
}
```

Obrázok 9 Overenie existencie súboru so záznamami o produktoch (Zdroj: vlastné spracovanie)

Tu sa overuje, či súbor, ktorý má obsahovať údaje o produktoch, skutočne existuje na určenom mieste. Cesta k súboru je uložená v premennej *productsXmlFilePath*, ktorá je definovaná ako privátne pole triedy a smeruje k súboru *xmlko.xml*. Tento súbor obsahuje štruktúrované dáta o produktoch, kde každý produkt je zapísaný ako samostatný XML element <*product*>. V prípade, že súbor nie je nájdený – napríklad pri chybe nasadenia, nesprávnej ceste alebo poškodení dát – metóda bezpečne ukončí svoju činnosť a vráti prázdne pole vo formáte JSON. Tým sa predchádza výnimkám a zabezpečuje sa funkčnosť systému.

Ak súbor existuje, pokračuje sa jeho načítaním pomocou knižnice *System.Xml.Linq* a konkrétne objektu *XDocument*. Ide o nástroj na prácu s XML dátami v rámci platformy .NET, ktorý umožňuje manipulovanie s XML štruktúrami.

Po úspešnom načítaní dokumentu sa metóda zameriava na extrahovanie jednotlivých produktov. Na tento účel sa využíva LINQ to XML – kombinácia deklaratívneho zápisu (známeho z databázových dotazov) s možnosťou pracovať priamo nad XML stromom.

Pomocou metódy *Descendants("product")* sa získavajú všetky uzly <*product*> nachádzajúce sa v dokumente:

```
XDocument doc = XDocument.Load(productsXmlFilePath);
var products = doc.Descendants("product")
.Select(p => new Product{
        Id = (string)p.Element("id"),
        Name = (string)p.Element("name"),
        Brand = (string)p.Element("brand"),
        Model = (string)p.Element("model"),
        Memory = (string)p.Element("memory"),
        Storage = (string)p.Element("storage"),
        Price = (string)p.Element("price"),
        Currency = (string)p.Element("currency"),
        Stock = (string)p.Element("stock"),
        ExternalStock = (string)p.Element("external_stock"),
        Description = (string)p.Element("description")
}).ToList();
```

Obrázok 10 Načítanie a transformácia inštancií uzlov triedy Product (Zdroj: vlastné spracovanie)

Každý uzol <*product*> sa následne pomocou metódy *Select()* transformuje na inštanciu triedy *Product*, ktorá reprezentuje dátový model produktu v rámci systému. Trieda

Product je definovaná s verejnými vlastnosťami, ktoré zodpovedajú jednotlivým XML elementom.

Každé z týchto polí sa načítava ako *string* a spracováva sa s cieľom zachovať flexibilitu pri ďalšej serializácii. Týmto spôsobom sa XML dokument pretransformuje na zoznam inštancií triedy *Product*, s ktorým sa dá ďalej pohodlne pracovať v C#.

Následne je zoznam produktov serializovaný do formátu JSON pomocou *JavaScriptSerializer*:

```
return new JavaScriptSerializer().Serialize(products);
```

Obrázok 11 Serializácia zoznamu do formátu JSON (Zdroj: vlastné spracovanie)

Výsledok celej operácie sa pripraví pre front-end rozhranie, ktoré si môže pomocou AJAX požiadavky vyžiadať zoznam produktov a následne ho vizuálne zobraziť. JSON formát je zároveň dostatočne univerzálny na to, aby ho bolo možné spracovať aj v prípade, že by webová služba bola integrovaná do externých aplikácií, napríklad mobilných alebo desktopových klientov.

Z technického hľadiska tak metóda *GetAllProducts*() zabezpečuje nasledujúce:

- Overenie existencie XML súboru s dátami – minimalizuje výskyt chýb pri čítaní.
- Načítanie dát zo súboru pomocou LINQ to XML – moderný, prehľadný a efektívny prístup k XML.
- Konverzia dát do dátovej triedy *Product* – zabezpečuje typovú bezpečnosť a konzistenciu.
- Serializácia do formátu JSON – univerzálny výstup pre rôzne frontendové technológie.

Z pohľadu architektúry systému je táto metóda dôležitá ako základ pre všetky operácie pracujúce s produktmi. Používateľské rozhranie sa vďaka nej môže načítať už pri prvom otvorení stránky s kompletným zoznamom produktov. Metóda je tiež opakovane použiteľná – či už ide o zobrazenie v základnom výpise, pri filtrovaní alebo ďalších nadstavbových funkciách systému.

Výhodou takejto implementácie je jej jednoduchá rozšíriteľnosť – napríklad ak by bolo potrebné do produktov pridať nové atribúty, stačí rozšíriť triedu *Product* a aktualizovať mapovanie v metóde *Select()*.

Okrem toho sa JSON výstup dá použiť v JavaScripte alebo frameworkoch ako React, Vue či Angular. Vďaka tomu môže webová služba *GetAllProducts*() slúžiť nielen ako súčasť interného systému obchodu, ale aj ako otvorené API pre partnerov, dodávateľov alebo analytické nástroje.

5.3 Metóda *DeleteProduct* na odstránenie produktov v XML súbore

Účelom tejto metódy je umožniť odstránenie konkrétneho produktu z XML súboru na základe jeho unikátneho identifikátora (ID). Táto metóda prijíma ako parameter jediné ID produktu, ktorý má byť odstránený.

Prvým krokom v metóde je načítanie XML súboru, ktorý obsahuje zoznam produktov. Na tento účel sa používa trieda *XDocument* z knižnice LINQ to XML. Tento dokument predstavuje celú štruktúru XML súboru a poskytuje možnosti na jeho analýzu a manipuláciu.

Premenná *productsXmlFilePath* obsahuje cestu k súboru, ktorý je načítaný metódou *XDocument.Load.*

```
XDocument doc = XDocument.Load(productsXmlFilePath);
```

Obrázok 12 Načítanie vstupného XML súboru (Zdroj: vlastné spracovanie)

Ďalším krokom je vyhľadanie konkrétneho produktu podľa jeho ID. Pomocou LINQ na XML sa vyhľadá prvý element <product>, ktorý obsahuje element <id> s hodnotou zodpovedajúcou parametru id, ktorý bol odoslaný do metódy. Tento krok je vykonaný pomocou metódy FirstOrDefault, ktorá vráti prvý výskyt alebo null, ak produkt neexistuje.

```
XElement product = doc.Descendants("product").FirstOrDefault(o =>
(string)o.Element("id") == id);
```

Obrázok 13 Porovnanie zadaného ID s ID v XML súbore (Zdroj: vlastné spracovanie)

Ak sa produkt nenájde, premenná product bude null a metóda vráti správu o tom, že produkt nebol nájdený. Tento krok zaručuje, že sa vyhne pokusu o odstránenie neexistujúceho produktu, čím sa predchádza potenciálnym chybám.

Ak produkt neexistuje, metóda ihneď vracia textovú správu, ktorá informuje používateľa o tom, že produkt s daným ID neexistuje v zozname:

```
if (product == null)
{
    return "Produkt nebol nájdený.";
}
```

Obrázok 14 Porovnanie zadaného ID s ID v XML súbore (Zdroj: vlastné spracovanie)

Tento krok je veľmi dôležitý, pretože zabraňuje neúspešným pokusom o úpravu XML súboru bez toho, aby produkt bol nájdený.

Ak sa produkt s daným ID nájde, môže byť odstránený. Na tento účel sa používa metóda Remove, ktorá odstráni celý element <*product*> zo zoznamu produktov v XML dokumente. Tento krok je dôležitý pre aktualizáciu obsahu XML súboru na základe požiadaviek používateľa.

```
product.Remove();
```

Obrázok 15 Odstránenie elementu <*product*> (Zdroj: vlastné spracovanie)

Po úspešnom odstránení produktu z XML dokumentu je potrebné uložiť zmeny do pôvodného súboru. Na tento účel sa opäť využíva metóda Save triedy XDocument, ktorá prepíše existujúci súbor s novým obsahom, ktorý už neobsahuje odstránený produkt.

```
doc.Save(productsXmlFilePath);
```

Obrázok 16 Uloženie aktualizovaného súboru (Zdroj: vlastné spracovanie)

Týmto spôsobom sa zabezpečí, že všetky zmeny budú trvalo zaznamenané v XML súbore.

Na záver metóda vráti textovú správu, ktorá informuje používateľa o úspešnom odstránení produktu.

```
return "Produkt bol úspešne odstránený.";
```

Obrázok 17 Výpis informácie o úspešnom odstránení elementu <product> (Zdroj: vlastné spracovanie)

5.4 Metóda *GetFilteredAndSortedProducts* na filtrovanie a zoradenie produktov v XML súbore

Metóda *GetFilteredAndSortedProducts()* je určená na získavanie a zobrazenie zoznamu produktov zo súboru XML na základe rôznych filtrov a parametrov zoradenia. Táto metóda poskytuje klientom možnosť filtrovania produktov podľa rôznych kritérií, ako sú názov, značka, model, cena, dostupnosť a iné, a zároveň umožňuje zoradiť produkty podľa

rôznych atribútov (napríklad názvu alebo ceny) a v rastúcom alebo klesajúcom poradí. Okrem samotného zoznamu produktov metóda generuje aj štatistické informácie o produktoch, ako sú priemerná cena, maximálna a minimálna cena a rozsah ceny.

Metóda akceptuje nasledujúce parametre:

- name (string): Názov produktu, ktorý sa má vyhľadať. Ak je hodnota null, filter podľa názvu sa neaplikuje.
- brand (string): Značka produktu, ktorý sa má vyhľadať. Ak je hodnota null, filter podľa značky sa neaplikuje.
- model (string): Model produktu, ktorý sa má vyhľadať. Ak je hodnota null, filter podľa modelu sa neaplikuje.
- minPrice (string): Minimálna cena produktu. Ak je hodnota null, filter podľa minimálnej ceny sa neaplikuje.
- maxPrice (string): Maximálna cena produktu. Ak je hodnota null, filter podľa maximálnej ceny sa neaplikuje.
- stock (string): Stav skladu (napr. "dostupný", "nedostupný"). Ak je hodnota null, filter podľa stavu skladu sa neaplikuje.
- sortBy (string): Určuje, podľa ktorého atribútu budú produkty zoradené. Predvolená hodnota je "name", čo znamená zoradenie podľa názvu.
- ascending (string): Určuje, či bude zoradenie vzostupné alebo zostupné. Predvolená hodnota je "true" (vzostupné).

Tieto parametre umožňujú používateľovi filtrovať a zoradiť produkty podľa rôznych kritérií.

Prvým krokom je načítanie XML dokumentu obsahujúceho všetky produkty, ktorý sa nachádza na definovanej ceste. Táto operácia načíta XML dokument do objektu *XDocument*. Na tomto dokumente sa následne vykonávajú všetky operácie filtrovania, výberu a zoradenia produktov.

Produkty sú filtrované pomocou metódy *Where(),* ktorá umožňuje definovať podmienky pre vyhľadávanie v XML dokumente:

```
XDocument doc = XDocument.Load(productsXmlFilePath);
var products = doc.Descendants("product")
    .Where(p => (string.IsNullOrEmpty(name) ||
    ((string)p.Element("name")).Contains(name)) &&
    (string.IsNullOrEmpty(brand) || ((string)p.Element("brand")).Contains(brand)) &&
    (string.IsNullOrEmpty(model) || ((string)p.Element("model")).Contains(model)) &&
    (string.IsNullOrEmpty(minPrice) || decimal.TryParse((string)p.Element("price"),
    out decimal min) && min >= decimal.Parse(minPrice)) &&
    (string.IsNullOrEmpty(maxPrice) || decimal.TryParse((string)p.Element("price"),
    out decimal max) && max <= decimal.Parse(maxPrice)) &&
    (string.IsNullOrEmpty(stock) || ((string)p.Element("stock")).Contains(stock)))
    .Select(p => new Product
    {
        Id = (string)p.Element("id"),
        Name = (string)p.Element("name"),
        Brand = (string)p.Element("brand"),
        Model = (string)p.Element("model"),
        Memory = (string)p.Element("memory"),
        Storage = (string)p.Element("storage"),
        Price = (string)p.Element("price"),
        Currency = (string)p.Element("currency"),
        Stock = (string)p.Element("stock"),
        ExternalStock = (string)p.Element("external_stock"),
        Description = (string)p.Element("description")
    }).ToList();
```

Obrázok 18 Načítanie a inicializácia podmienok pre filtrovanie (Zdroj: vlastné spracovanie)

Podmienky sa používajú pre každý filter (napr. name, brand, minPrice, atď.), pričom sa kontroluje, či daný filter nie je prázdny alebo null. Ak je filter prázdny, daný produkt nebude filtrovaný podľa tohto kritéria. Ak filter obsahuje hodnotu, produkt musí spĺňať podmienku, že daný atribút (napríklad názov) obsahuje zadaný reťazec. Pre cenu sa používa konverzia na decimal a overenie, či cena spĺňa zadané hranice.

Po filtrovaní produktov sa záznamy zoradia podľa zvoleného parametra. Ak je zadaný parameter *sortBy* na hodnotu "price", produkty sa zoradia podľa ceny, pričom sa použije hodnoty *decimal.TryParse()* na bezpečné spracovanie ceny. Ak je parameter *sortBy* na hodnotu "name", produkty sa zoradia podľa názvu. Pre zoradenie sa používa metóda *OrderBy*() alebo *OrderByDescending(),* ktorá zabezpečuje vzostupné alebo zostupné zoradenie na základe parametra *ascending*:

```
if (sortBy == "price")
{
    products = isAscending ? products.OrderBy(p => decimal.TryParse(p.Price, out decimal
    price) ? price : 0).ToList() :
    products.OrderByDescending(p => decimal.TryParse(p.Price, out decimal price) ?
    price :0).ToList();
}
else
{
    products = isAscending ? products.OrderBy(p => p.Name).ToList() :
    products.OrderByDescending(p => p.Name).ToList();
}
```

Obrázok 19 Parametre usporiadania záznamov produktov (Zdroj: vlastné spracovanie)

Týmto spôsobom sa produkty zoradia podľa ceny alebo názvu, pričom sa zohľadňuje aj to, či je zoradenie vzostupné alebo zostupné.

Po filtrovaní a zoradení produktov metóda vypočíta štatistické údaje o týchto produktoch:

```
var statistics = new
{
    TotalProducts = products.Count,
    AveragePrice = products.Count > 0 ? products.Average(p =>
    decimal.TryParse(p.Price,out decimal price) ? price : 0) : 0,
    MaxPrice = products.Count > 0 ? products.Max(p => decimal.TryParse(p.Price,
    out decimal price) ? price : 0) : 0,
    MinPrice = products.Count > 0 ? products.Min(p => decimal.TryParse(p.Price,
    out decimal price) ? price : 0) : 0,
    PriceRange = products.Count > 0 ? products.Max(p => decimal.TryParse(p.Price,
    out decimal price) ? price : 0) –
    products.Min(p => decimal.TryParse(p.Price, out decimal price) ? price : 0) : 0
};
```

Obrázok 20 Výpočet štatistických údajov z vypísaných údajov produktov (Zdroj: vlastné spracovanie)

Tento objekt obsahuje nasledovné štatistiky:

- TotalProducts: Počet produktov po filtrovaní.
- AveragePrice: Priemerná cena produktov.
- MaxPrice: Maximálna cena produktu.
- MinPrice: Minimálna cena produktu.
- PriceRange: Rozdiel medzi maximálnou a minimálnou cenou.

Tieto štatistiky môžu byť užitočné pre administrátorov alebo používateľov, ktorí sa zaujímajú o cenovú analýzu alebo prehľad o produktových kategóriách.

Na záver metóda vytvorí objekt, ktorý obsahuje zoznam produktov spolu so štatistikami, a tento objekt sa serializuje do formátu JSON:

```
var response = new
{
    Products = products,
    Statistics = statistics
};
return new JavaScriptSerializer().Serialize(response);
```

Obrázok 21 Vrátenie serializovanej JSON odpovede (Zdroj: vlastné spracovanie)

Metóda vráti JSON odpoveď, ktorá obsahuje dva hlavné elementy: *Products* (zoznam produktov) a *Statistics* (štatistické údaje o produktoch).

5.5 Metóda *AddOrder* na pridanie objednávky do XML súboru

Metóda *AddOrder*() slúži na pridanie novej objednávky do systému. Okrem pridania objednávky táto metóda tiež zabezpečuje aktualizáciu skladových zásob, keď je produkt zakúpený. Ak je produkt na sklade, metóda ho odstráni zo skladu a následne pridá objednávku do XML súboru s objednávkami. Ak produkt nie je nájdený alebo sú súbory neexistujú, metóda vráti príslušné chybové hlásenie.

Metóda AddOrder() akceptuje:

- customer (string): Meno zákazníka, ktorý vykonal objednávku.
- productId (string): ID produktu, ktorý zákazník objednal.
- quantity (int): Množstvo objednaných kusov produktu.
- price (string): Cena produktu.
- status (string): Stav objednávky (napr. "vybavená", "čaká na potvrdenie").

Tieto parametre slúžia na pridanie novej objednávky a aktualizáciu informácií o produkte a jeho dostupnosti v sklade.

Na začiatku metódy je vykonaná kontrola, či existujú požadované XML súbory – súbor so zoznamom produktov (*productsXmlFilePath*) a súbor s objednávkami (*ordersXmlFilePath*). Ak niektorý z týchto súborov neexistuje, metóda vráti chybu:

```
if (!File.Exists(productsXmlFilePath) ||
!File.Exists(ordersXmlFilePath))
{
    return "Jeden alebo oba XML súbory neexistujú.";
}
XDocument productDoc = XDocument.Load(productsXmlFilePath);
XElement product = productDoc.Descendants("product").FirstOrDefault(p =>
(string)p.Element("id") == productId);

if (product == null)
{
    return "Produkt nebol nájdený.";
}
string productName = (string)product.Element("name");
string productBrand = (string)product.Element("brand");
string productModel = (string)product.Element("model");
```

Obrázok 22 Overenie existencie XML súboru s údajmi o produktoch (Zdroj: vlastné spracovanie)

Táto kontrola je dôležitá na to, aby sa zabránilo ďalšiemu spracovaniu, ak neexistujú potrebné údaje o produktoch alebo objednávkach.

Následne sa načíta XML dokument so zoznamom produktov a hľadá sa produkt, ktorého ID zodpovedá ID poskytnutému v parametroch. Ak produkt nie je nájdený, metóda vráti chybovú správu. Tento krok zabezpečuje možnosť vytvorenia objednávky iba na existujúci záznam produktu.

Ak je produkt nájdený, metóda získava základné informácie o produkte (názov, značka, model), ktoré sa následne použijú pri vytváraní objednávky. Tieto informácie sú potrebné na pridanie položky do objednávky.

Po získaní informácií o produkte sa produkt odstráni zo skladu, čím sa znižuje jeho dostupnosť. Tento krok sa vykoná pred pridaním objednávky, pretože sa predpokladá, že produkt bol zakúpený:

```
product.Remove();
productDoc.Save(productsXmlFilePath);
```

Obrázok 23 Odstránenie pridaného produktu do objednávky zo skladu (Zdroj: vlastné spracovanie)

Týmto sa aktualizuje XML súbor s produktmi, pričom produkt, ktorý bol objednaný, je trvalo odstránený zo zoznamu dostupných produktov.

Ďalej sa načíta XML dokument s objednávkami a pripraví sa nová objednávka. ID novej objednávky je generované automaticky na základe aktuálneho dátumu a počtu objednávok vytvorených v daný deň:

```
XDocument orderDoc = XDocument.Load(ordersXmlFilePath);
XElement root = orderDoc.Root;
string today = DateTime.Now.ToString("yyMMdd");
int countToday = root.Elements("objednavka").Where(o =>
o.Element("id")?.Value.StartsWith(today) ?? false).Count();
string newId = today + countToday.ToString("D2");
```

Obrázok 24 Vytvorenie ID objednávky (Zdroj: vlastné spracovanie)

Následne sa vytvorí nová položka objednávky, ktorá obsahuje všetky potrebné informácie vrátane ID, časového označenia, údajov o zákazníkovi, zakúpenom produkte, množstve, cene a stave objednávky:

```
XElement newOrder = new XElement("objednavka",
    new XElement("id", newId),
    new XElement("timestamp", DateTime.Now.ToString("yyyy-MM-ddTHH:mm:ss")),
    new XElement("zakaznik", customer),
    new XElement("polozka",
        new XElement("name", productName),
        new XElement("brand", productBrand),
        new XElement("model", productModel)
    ),
    new XElement("mnozstvo", quantity),
    new XElement("cena", price),
    new XElement("stav", status)
);
```

Obrázok 25 Vytvorenie elementu <objednávka> (Zdroj: vlastné spracovanie)

Tento XML element je pridaný do koreňového elementu dokumentu objednávok a následne je dokument uložený s novou objednávkou:

```
root.Add(newOrder);
orderDoc.Save(ordersXmlFilePath);
return $"Objednávka bola úspešne pridaná s ID: {newId}, produkt {productName} bol odstránený zo skladu.";
```

Obrázok 26 Pridanie a uloženie novej objednávky do XML súboru (Zdroj: vlastné spracovanie)

Po pridaní objednávky metóda vráti správu o úspechu, vrátane ID novej objednávky a informácie o tom, že produkt bol odstránený zo skladu.

Tento krok zabezpečuje, že zákazník bude informovaný o úspechu operácie a bude mať prehľad o tom, že objednávka bola spracovaná.

Metóda *AddOrder*() zabezpečuje pridanie novej objednávky do systému a aktualizáciu skladových zásob. Jej hlavné kroky zahŕňajú:

- Kontrolu existencie XML súborov s produktmi a objednávkami.
- Načítanie informácií o produkte a kontrolu jeho existencie v sklade.
- Odstránenie produktu zo skladu po vytvorení objednávky.

- Generovanie unikátneho ID objednávky na základe dátumu a počtu objednávok v ten deň.
- Vytvorenie a pridanie objednávky do XML súboru s objednávkami.

Vrátenie správy o úspechu, ktorá obsahuje ID objednávky a informácie o tom, že produkt bol odstránený zo skladu.

Metóda poskytuje dôležitú funkcionalitu pre správu objednávok v e-commerce systéme a zároveň umožňuje efektívne spravovať skladové zásoby.

5.6 Metóda *DeleteOrder* na odstránenie objednávky v XML súbore

Účelom tejto metódy je odstránenie konkrétnej objednávky na základe jej ID, pričom sa zabezpečuje, že záznam o objednávke je trvalo odstránený zo systému.

Celý proces zahŕňa niekoľko krokov, ako je načítanie dát z XML súboru, vyhľadanie objednávky, jej odstránenie a uloženie aktualizovaného stavu do súboru. V tomto popise vysvetlíme, čo sa deje v jednotlivých krokoch implementácie.

Tento súbor obsahuje informácie o všetkých objednávkach v XML formáte. Premenná *ordersXmlFilePath* obsahuje cestu k tomuto súboru. Tento súbor je otvorený v režime čítania, čo umožňuje načítanie celého XML stromu do pamäte. Použitím knižnice LINQ sa uľahčuje manipulácia s XML dátami.

Po načítaní dokumentu nasleduje vyhľadanie konkrétnej objednávky. Používa sa LINQ dotaz, ktorý prehľadáva všetky elementy <*objednavka*>:

```
XDocument doc = XDocument.Load(ordersXmlFilePath);
XElement order = doc.Descendants("objednavka").FirstOrDefault(o =>
(string)o.Element("id") == id);
```

Obrázok 27 Načítanie súboru objednávok (Zdroj: vlastné spracovanie)

Tento dotaz prehľadáva všetky elementy <*objednavka*> v XML dokumente a vyhľadáva prvý výskyt, kde sa hodnota ID objednávky zhoduje s hodnotou parametra id, ktorý je odoslaný do metódy. Ak je objednávka nájdená, objekt order bude obsahovať referenciu na tento element. Ak sa objednávka nenájde, objekt *order* bude null.

Predtým, ako sa objednávka odstráni, je nevyhnutné skontrolovať, či objednávka naozaj existuje:

```
if (order == null)
{
    return "Objednávka nebola nájdená.";
}
```

Obrázok 28 Overenie existencie objednávky vo vstupnom XML súbore (Zdroj: vlastné spracovanie)

Tento krok je dôležitý, pretože ak by objednávka neexistovala, nemohli by sme ju odstrániť a preto je potrebné informovať používateľa o tom, že objednávka nebola nájdená. Tento krok tiež zabraňuje chybám v kóde, ako sú pokusy o operácie na neexistujúcich objektoch, ktoré by mohli viesť k výnimkám.

Ak bola objednávka nájdená, je na rade jej odstránenie zo zoznamu. Používa sa metóda *Remove*() zo štandardnej knižnice LINQ.

Metóda *Remove*() odstráni daný element z XML dokumentu. Tento krok je nevyhnutný pre zabezpečenie toho, že objednávka bude úplne vymazaná zo systému, a to nielen z pamäte, ale aj zo súboru, ktorý obsahuje trvalé údaje o objednávkach. Po tomto kroku sa objednávka už nenachádza v XML dokumente.

Po odstránení objednávky z XML dokumentu je potrebné tento dokument uložiť späť na disk:

```
order.Remove();
doc.Save(ordersXmlFilePath);
return "Objednávka bola úspešne odstránená.";
return "Objednávka nebola nájdená.";
```

Obrázok 29 Odstránenie objednávky a uloženie aktualizovaného XML súboru (Zdroj: vlastné spracovanie)

Metóda *Save*() ukladá zmeny vykonané na XML dokumente. Bez tohto kroku by sa zmena neprejavila v súbore, a teda by sa po ukončení aplikácie stratila. Uloženie zabezpečí, že systém bude udržiavať správny stav a že objednávka bude trvalo odstránená zo všetkých ďalších operácií. Na konci metódy je vrátená správa, ktorá informuje používateľa o výsledku operácie. Tento text je odoslaný ako odpoveď na požiadavku. Informuje používateľa o tom, že požadovaná objednávka bola úspešne odstránená zo systému. Ak objednávka neexistovala, vráti sa iná správa. Používateľ má spätnú väzbu o výsledku svojej akcie.

Metóda zabezpečuje:

- Načítanie XML dokumentu: Dokument so všetkými objednávkami je načítaný z disku do pamäte, kde sa s ním bude manipulovať.
- Vyhľadanie objednávky podľa ID: Metóda vyhľadáva objednávku, ktorá má zhodné ID s požiadavkou.
- Kontrola existencie objednávky: Ak objednávka neexistuje, operácia je prerušená a používateľ je informovaný o chybe.
- Odstránenie objednávky: Ak objednávka existuje, je z XML dokumentu odstránená.
- Uloženie zmeneného dokumentu: Po odstránení objednávky je dokument uložený s novým stavom.

5.7 Metóda *GetAllOrders* na načítanie objednávok z XML súboru

Cieľom metódy je poskytnúť prístup k údajom o objednávkach v XML formáte, ktoré sú následne prevedené do formátu JSON.

Podobne ako v metóde *GetAllProducts*, metóda *GetAllOrders* začína kontrolou, či súbor s objednávkami, ktorý je špecifikovaný v ceste *ordersXmlFilePath*, skutočne existuje. Ak súbor neexistuje, metóda vráti prázdny JSON zoznam ([]), čím informuje volajúcu aplikáciu, že v systéme nie sú žiadne objednávky alebo, že došlo k chybe pri načítaní súboru.

Ak súbor existuje, metóda pokračuje načítaním XML dokumentu pomocou triedy *XDocument*. Táto trieda umožňuje jednoducho pracovať so štruktúrovanými XML dátami a poskytuje množstvo metód na manipuláciu s jednotlivými elementmi XML dokumentu.

Ďalej metóda využíva LINQ na výber všetkých elementov typu *objednavka* z XML dokumentu. Každý z týchto elementov predstavuje jednu objednávku. Pomocou metódy *Select* vytvárame zoznam objektov typu *Order*, kde pre každý element objednávky mapujeme príslušné hodnoty na vlastnosti objektu *Order*.

```
if (!File.Exists(ordersXmlFilePath))
{
    return "[]";
}
XDocument doc = XDocument.Load(ordersXmlFilePath);
var orders = doc.Descendants("objednavka").Select(o => new Order
    {
        Id = (string)o.Element("id"),
        Timestamp = (string)o.Element("timestamp"),
        Customer = (string)o.Element("zakaznik"),
        Item = (string)o.Element("polozka"),
        Quantity = (int)o.Element("mnozstvo"),
        Price = (string)o.Element("cena"),
        Status = (string)o.Element("stav")}
).ToList();
```

Obrázok 30 Overenie existencie súboru a výber elementov objednavka (Zdroj: vlastné spracovanie)

Po načítaní všetkých objednávok a ich mapovaní do objektov typu Order sa tieto objekty serializujú do JSON formátu pomocou *JavaScriptSerializer*.

```
return new JavaScriptSerializer().Serialize(orders);
```

Obrázok 31 Vrátenie dát v JSON formáte (Zdroj: vlastné spracovanie)

Načítanie XML súboru a následné spracovanie pomocou LINQ je veľmi efektívne, pokiaľ ide o prácu s relatívne malými až stredne veľkými súbormi. Tento prístup zaručuje rýchlu odozvu aj pri väčšom počte objednávok, pretože sú všetky operácie optimalizované pre výkon.

Použitie LINQ na výber a manipuláciu s XML dokumentmi umožňuje jednoducho pracovať s dátami bez potreby ručného parsovania XML súboru. Toto výrazne zjednodušuje prácu a znižuje množstvo kódu, ktorý by bol potrebný pri manuálnom spracovaní XML dát.

Táto metóda môže byť ďalej prispôsobená podľa potrieb. Napríklad, môžeme pridať rôzne filtre pre výber objednávok alebo upraviť spôsob, akým sú objednávky spracovávané a zobrazené na klientskej strane. Tento prístup poskytuje veľkú flexibilitu v rôznych scenároch, kde potrebujeme pracovať s objednávkami.

Pri čítaní dát z XML je potrebné zabezpečiť, aby všetky hodnoty v dokumente boli validné a odpovedali očakávanému formátu. To znamená, že by sme mali pridať kontrolu typov a validáciu, aby sa predišlo nekompletným alebo neplatným dátam.

Metóda *GetAllOrders* poskytuje spôsob zobrazenia všetkých objednávok z XML súboru vo formáte JSON. Tento prístup je výhodný najmä pre webové aplikácie, ktoré

potrebujú rýchlo spracovať a zobraziť dáta o objednávkach. Metóda môže byť ďalej rozšírená a vylepšená podľa potreby aplikácie a konkrétnych požiadaviek.

5.8 Metóda *GetFilteredAndSortedOrders* na filtrovanie a zoradenie objednávok

Metóda *GetFilteredAndSortedOrders* spracováva požiadavky na získanie objednávok z XML súboru, pričom tieto objednávky môžu byť filtrované podľa rôznych kritérií (ako je zákazník, stav objednávky, dátum, štvrťrok a rok) a zoradené podľa rôznych parametrov (ako je cena, množstvo, zákazník, stav objednávky alebo časová pečiatka). V tejto metóde sa využíva technológia LINQ na manipuláciu s XML dokumentom a JSON formát na vrátenie výsledkov.

Metóda načíta XML súbor pomocou triedy *XDocument* a extrahuje všetky objednávky (<*objednavka*>). Parametre, ako sú *customer* a *status*, sa používajú na filtrovanie údajov o objednávkach.

```
if (!File.Exists(ordersXmlFilePath))
{
    return "[]";
}
XDocument doc = XDocument.Load(ordersXmlFilePath);
var orders = doc.Descendants("objednavka").Where(o =>
(string.IsNullOrEmpty(customer) || ((string)o.Element("zakaznik")).Contains(customer))
&& (string.IsNullOrEmpty(status) || ((string)o.Element("stav")).Contains(status))).Select(o
=> new Order
{
    Id = (string)o.Element("id"),
    Timestamp = (string)o.Element("timestamp"),
    Customer = (string)o.Element("zakaznik"),
    Item = o.Element("polozka") != null ? o.Element("polozka").Value : "",
    Quantity = (int)o.Element("mnozstvo"),
    Price = (string)o.Element("cena"),
    Status = (string)o.Element("stav")
}).ToList();
```

Obrázok 32 Načítanie vstupného XML súboru objednávok (Zdroj: vlastné spracovanie)

Tento kód využíva metódu Descendants na prehľadávanie všetkých objednávok v XML dokumente. Ak sú parametre ako *customer* alebo *status* nevyplnené, tieto kritériá sa ignorujú. Inak sa objednávky filtrujú na základe zadaných hodnôt.

Ak sú zadané parametre *minDate* alebo *maxDate*, metóda filtruje objednávky podľa týchto dátumov. Používa sa metóda *DateTime.Parse* na konverziu časových pečiatok na dátumy a potom sa porovnávajú s hodnotami *minDate* a *maxDate*.

```
if (!string.IsNullOrEmpty(minDate) || !string.IsNullOrEmpty(maxDate))
{
    orders = orders.Where(o => (string.IsNullOrEmpty(minDate) ||
    DateTime.Parse(o.Timestamp) >= DateTime.Parse(minDate)) &&
    (string.IsNullOrEmpty(maxDate) || DateTime.Parse(o.Timestamp) <=
    DateTime.Parse(maxDate))).ToList();
}
```

Obrázok 33 Filtrovanie XML súboru objednávok na základe časového intervalu (Zdroj: vlastné spracovanie)

Ak sú parametre *quarter* a *year* zadané, metóda najprv vyberie začiatok a koniec príslušného kvartálu a potom filtruje objednávky, ktoré sa zhodujú s rokom a mesiacom zadaného kvartálu.

```
if (!string.IsNullOrEmpty(quarter) && !string.IsNullOrEmpty(year) && int.TryParse(year,
out int yearInt))
{
    int qStartMonth = 0, qEndMonth = 0;
    switch (quarter)
    {
    case "Q1": qStartMonth = 1; qEndMonth = 3; break;
    case "Q2": qStartMonth = 4; qEndMonth = 6; break;
    case "Q3": qStartMonth = 7; qEndMonth = 9; break;
    case "Q4": qStartMonth = 10; qEndMonth = 12; break;
    default: return "[]"; // Neplatný kvartál
    }
    orders = orders.Where(o =>
    {
    DateTime orderDate = DateTime.Parse(o.Timestamp);
    return orderDate.Year == yearInt && orderDate.Month >= qStartMonth &&
    orderDate.Month <= qEndMonth;
    }).ToList();
}
```

Obrázok 34 Filtrovanie XML súboru objednávok na základe kvartálneho obdobia (Zdroj: vlastné spracovanie)

Tento kód implementuje filtrovanie objednávok podľa zadaného kvartálu. Pre každý kvartál sa určí rozsah mesiacov, ktoré doň patria (napr. Q1 je od januára do marca).

Metóda umožňuje zoradenie objednávok podľa rôznych kritérií: cena, množstvo, zákazník, stav alebo časová pečiatka. Na základe parametra *sortBy* sa vyberie, aké kritérium sa použije na zoradenie. Zoradenie môže byť buď vzostupné, alebo zostupné, čo je riadené parametrom *ascending*.

```
if (sortBy.ToLower() == "price")
{
    orders = isAscending ? orders.OrderBy(o =>
    decimal.TryParse(o.Price.Replace(",",".").Split(' ')[0],
    System.Globalization.NumberStyles.Any,
    System.Globalization.CultureInfo.InvariantCulture, out decimal price) ? price : 0).
    ToList(): orders.OrderByDescending(o => decimal.TryParse(o.Price.Replace(",", ".").
    Split(' ')[0], System.Globalization.NumberStyles.Any,
    System.Globalization.CultureInfo.InvariantCulture, out decimal price) ? price : 0).
    ToList();
}
```

Obrázok 35 Usporiadanie XML súboru objednávok na podľa ceny

V tomto úseku kódu sa objednávky zoradia podľa ceny. Na správnu interpretáciu ceny sa používa *decimal.TryParse*, pričom cena je predtým spracovaná (napr. odstránením nežiaducich znakov). Podobne sa realizuje zoradenie podľa množstva, zákazníka, stavu alebo časovej pečiatky:

Na konci metódy sa vypočíta niekoľko štatistík, ktoré poskytujú prehľad o objednávkach. Tieto štatistiky zahŕňajú:

- Celkový počet objednávok (*TotalOrders*),
- Celkový príjem (*TotalRevenue*),
- Priemerná hodnota objednávky (*AverageOrderValue*),
- Celkové množstvo predaných položiek (*TotalQuantitySold*),
- Priemerné množstvo na objednávku (*AverageQuantityPerOrder*),
- Maximálne a minimálne množstvo (*MaxQuantity*, *MinQuantity*)

Štatistiky sú vypočítané ako agregované hodnoty na základe filtrovaných a zoradených objednávok.

```
var statistics = new
{
    TotalOrders = orders.Count,
    TotalRevenue = orders.Sum(o => o.Quantity * (decimal.TryParse
    (o.Price, out decimal price) ? price : 0)),
    AverageOrderValue = orders.Count > 0 ? orders.Average(o => o.Quantity *
    (decimal.TryParse(o.Price, out decimal price) ? price : 0)) : 0,
    TotalQuantitySold = orders.Sum(o => o.Quantity),
    AverageQuantityPerOrder = orders.Count > 0 ? orders.Average(o => o.Quantity) : 0,
    MaxQuantity = orders.Count > 0 ? orders.Max(o => o.Quantity) : 0,
    MinQuantity = orders.Count > 0 ? orders.Min(o => o.Quantity) : 0
    };
```

Obrázok 36 Výpočet štatistických metód údajov o objednávkach (Zdroj: vlastné spracovanie)

Na záver sa vytvorí objekt, ktorý obsahuje zoznam objednávok a štatistiky, ktoré sú následne serializované do JSON formátu pomocou *JavaScriptSerializer*.

```
var response = new
{
    Orders = orders,
    Statistics = statistics
};
return new JavaScriptSerializer().Serialize(response);
```

Obrázok 37 Vytvorenie objektu <response> a serializovanie do JSON formátu (Zdroj: vlastné spracovanie)

Metóda poskytuje API na filtrovanie a zoradenie objednávok podľa rôznych kritérií. Početné možnosti filtrovania podľa zákazníka, stavu objednávky, dátumu a kvartálu umožňujú flexibilnú analýzu údajov.

6 Implementácia správy produktov a štatistík v e-shope

V tejto kapitole sa budeme zaoberať popisom implementácie správy produktov a generovania štatistík v rámci webovej aplikácie pre e-shop Elektrodynamo. Implementácia vyžaduje interakciu medzi frontendovou časťou aplikácie a backendovou logikou spracovávajúcou dáta.

6.1 Používateľské rozhranie a navigácia

Používateľské rozhranie (UI) aplikácie je navrhnuté tak, aby umožnilo jednoduchú navigáciu medzi rôznymi sekciami aplikácie, ako sú zobrazenie produktov, pridávanie nových produktov, ich odstraňovanie, a generovanie štatistík. Hlavná stránka obsahuje navigačné odkazy, ktoré umožňujú prechod medzi týmito sekciami bez nutnosti opustenia stránky.

```
<nav>
    <a href="javascript:void(0)" onclick="showTab('home')">Domov</a>
    <a href="javascript:void(0)" onclick="showTab('view')">Zobraziť produkty</a>
    <a href="javascript:void(0)" onclick="showTab('add')">Pridať produkt</a>
    <a href="javascript:void(0)" onclick="showTab('delete')">Odstrániť produkt</a>
    <a href="javascript:void(0)" onclick="showTab('orders')">Objednávky</a>
</nav>
```

Obrázok 38 Navigačné menu na prechod medzi sekciami (Zdroj: vlastné spracovanie)

Každá sekcia je zobrazená v samostatnom div-elemente s id podľa názvu sekcie. Dynamicky sa zobrazuje obsah na základe interakcie používateľa s navigačnými odkazmi.

Tento prístup minimalizuje potrebu prechádzania medzi rôznymi stránkami, čo zlepšuje používateľskú skúsenosť.

6.2 Filtrovanie a zoradenie produktov

Aplikácia umožňuje filtrovanie produktov podľa rôznych parametrov, ako sú názov, cena, dostupnosť na sklade a ďalšie. Používateľ môže zadať hodnoty pre tieto filtre v textových poliach a výberových zoznamoch. Nasledujúca časť kódu definuje prvky *input* a *select* pre filtrovanie produktov.

```
<label for="productSearchInput">Hľadať podľa názvu, značky alebo modelu:</label>
<input type="text" id="productSearchInput" runat="server" />
<label for="minProductPrice">Cena od:</label>
<input type="number" id="minProductPrice" placeholder="Min cena">
<label for="maxProductPrice">Cena do:</label>
<input type="number" id="maxProductPrice" placeholder="Max cena">
<label for="productNamesSelect">Názov produktu:</label>
<select id="productNamesSelect">
    <option value="">Vyberte názov produktu</option>
</select>

<label for="sortBySelect">Zoradiť podľa:</label>
<select id="sortBySelect">
    <option value="name">Názov</option>
    <option value="price">Cena</option>
</select>
```

Obrázok 39 Formulár na vstupy parametrov pre filtrovanie údajov (Zdroj: vlastné spracovanie)

6.3 JavaScript a AJAX pre dynamické načítanie produktov

Pre dynamické načítanie produktov podľa filtrovaných a zoradených údajov je použitý JavaScript s AJAX technológiou. Tento prístup umožňuje načítať dáta zo servera a zobraziť ich na stránke bez nutnosti jej znovu načítania. Nasledujúci kód ukazuje spôsob, akým sú spracované zadané filtre a ako sa produkty načítajú pomocou AJAX požiadavky.

```
$("#filterProductsBtn").click(function () {
    filters.name = $("#productSearchInput").val();
    filters.minPrice = $("#minProductPrice").val();
    filters.maxPrice = $("#maxProductPrice").val();
    loadProducts(filters);  // Načítame produkty s aktuálnymi hodnotami filtrov
});

function loadProducts(filters) {
    $.ajax({
    type: "POST",
    url: "WebService.asmx/GetFilteredAndSortedProducts",
    data: JSON.stringify(filters),
    contentType: "application/json; charset=utf-8",
    dataType: "json",
    success: function (response) {
    var data = response.d;

    if (typeof data === "string") {
    data = JSON.parse(data);
    }

    var products = data.Products;
    var statistics = data.Statistics;
```

Obrázok 40 Načítanie produktov zo servera (Zdroj: vlastné spracovanie)

Nasledujúci skript posiela filter do webového servisu, ktorý následne vráti produkty podľa zadaných kritérií. Tieto produkty sú potom zobrazené v tabuľke na stránke:

```
// Generovanie tabuľky s produktmi
    var table = "<table><thead><tr>" +

"<th>ID</th><th>Názov</th><th>Značka</th><th>Model</th><th>Pamäť</th>" +
"<th>Úložisko</th><th>Cena</th><th>Mena</th><th>Sklad</th><th>Externý
sklad</th>" + "<th>Popis</th></tr></thead><tbody>";

    $.each(products, function (i, p) {
    table += "<tr>" +
        "<td>" + p.Id + "</td>" +
        "<td>" + p.Name + "</td>" +
        "<td>" + p.Brand + "</td>" +
        "<td>" + p.Model + "</td>" +
        "<td>" + p.Memory + "</td>" +
        "<td>" + p.Storage + "</td>" +
        "<td>" + p.Price + "</td>" +
        "<td>" + p.Currency + "</td>" +
        "<td>" + p.Stock + "</td>" +
        "<td>" + p.ExternalStock + "</td>" +
        "<td>" + p.Description + "</td>" +
        "</tr>";
    });
    table += "</tbody></table>";
    $("#productContainer").html(table);

    // Zobrazenie štatistických údajov
    $("#totalProducts").text("Celkový počet produktov: " + statistics.TotalProducts);
    $("#averagePrice").text("Priemerná cena: " + statistics.AveragePrice.toFixed(2) + " " +
    products[0].Currency);
    },
    error: function (xhr, status, error) {
    console.error("Chyba pri načítaní produktov: ", error);
    $("#productContainer").html("<p>Nastala chyba pri načítaní produktov.</p>");
    }
    });
}
```

Obrázok 41 Vytvorenie tabuľky s produktovými údajmi (Zdroj: vlastné spracovanie)

6.4 Pridávanie a odstraňovanie produktov

Aplikácia tiež umožňuje pridávať nové produkty a odstraňovať existujúce produkty. Pre túto funkcionalitu je vytvorený jednoduchý formulár, v ktorom používateľ zadáva údaje o produkte, ako sú názov, značka, model, cena a ďalšie.

```
<div class="form-container">
    <label for="txtName">Názov produktu:</label>
    <input type="text" id="txtName" />
</div>
<!-- ďalšie polia pre značku, model, cenu, popis, atď. -->
<button type="button" onclick="addProduct()">Pridať produkt</button>
```

Obrázok 42 Formulár na pridanie produktu (Zdroj: vlastné spracovanie)

Funkcia *addProduct*() následne posiela tieto údaje cez AJAX do webového servisu, ktorý vykoná operáciu pridania produktu do XML súboru.

```
function addProduct() {
    var name = document.getElementById("txtName").value;
    var brand = document.getElementById("txtBrand").value;
    var xhr = new XMLHttpRequest();
    xhr.open("POST", "webservice.asmx/AddProduct", true);
    xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    var params = "name=" + encodeURIComponent(name) +
        "&brand=" + encodeURIComponent(brand) +
        "&model=" + encodeURIComponent(model) +
        "&price=" + encodeURIComponent(price);
       xhr.onreadystatechange = function () {
          if (xhr.readyState == 4 && xhr.status == 200) {
          alert("Produkt bol úspešne pridaný!");
    }
    };
xhr.send(params);
}
```

Obrázok 43 Funkcia na volanie vystavenej metódy na pridanie produktu (Zdroj: vlastné spracovanie)

6.5 Generovanie štatistík vstupného XML súboru

Na generovanie štatistík o produktoch aplikácia vypočíta základné metriky, ako je celkový počet produktov, priemerná cena, najvyššia cena a najnižšia cena. Tieto údaje sa zobrazujú v sekcii štatistík, čo poskytuje používateľovi prehľad o stave produktov v obchode.

```
$("#totalProducts").text("Celkový počet produktov: " + statistics.TotalProducts);
$("#averagePrice").text("Priemerná cena: " + statistics.AveragePrice.toFixed(2) + " " +
products[0].Currency);
```

Obrázok 44 Výpis štatistík o produktoch (Zdroj: vlastné spracovanie)

6.6 Štruktúra pre správu objednávok

V prvej časti sa nachádza HTML kód, ktorý slúži ako základ pre zobrazenie a správu objednávok. Táto časť obsahuje formuláre na filtrovanie objednávok, zobrazenie tabuľky s objednávkami, ale aj formuláre na pridanie a odstránenie objednávok.

```
<!-- Objednávky -->
<div id="orders" class="tab-content hidden">
    <h3>Objednávky</h3>
    <label for="searchCustomer">Hľadať zákazníka:</label>
    <input type="text" id="searchCustomer" placeholder="Zadajte meno...">

        <label for="minPrice">Cena od:</label>
        <input type="number" id="minPrice" placeholder="Min cena">

        <label for="maxPrice">Cena do:</label>
        <input type="number" id="maxPrice" placeholder="Max cena">

        <label for="startDate">Od dátumu:</label>
        <input type="date" id="startDate">

        <label for="itemFilter">Druh produktu:</label>
        <select id="itemFilter">
           <option value="">Všetky</option>
        </select>

        <label for="quarterSelect">Kvartál:</label>
        <select id="quarterSelect">
           <option value="">Vyberte kvartál</option>
           <option value="1">1. kvartál</option>
           <option value="2">2. kvartál</option>
           <option value="3">3. kvartál</option>
           <option value="4">4. kvartál</option>
        </select>

        <label for="yearSelect">Rok:</label>
        <select id="yearSelect">
           <option value="">Vyberte rok</option>
        </select>

        <button id="filterBtn">Filtrovať</button>
```

Obrázok 45 Vytvorenie formuláru na filtrovanie objednávok (Zdroj: vlastné spracovanie)

```
<table id="ordersTable" border="1">
    <thead>
        <tr>
        <th data-column="Id">ID </th>
        <th data-column="Timestamp">Timestamp</th>
        <th data-column="Customer">Zákazník</th>
        <th data-column="Item">Položka</th>
        <th data-column="Quantity">Množstvo</th>
        <th data-column="Price">Cena</th>
        <th data-column="Status">Stav</th>
    </tr>
</thead>
 </table>
        <div id="orderSummary">
          <p><strong>Počet objednávok:</strong> <span id="totalOrders">0</span></p>
          <p><strong>Predané kusy:</strong> <span id="totalQuantity">0</span></p>
          <p><strong>Celková cena:</strong> <span id="totalPrice">0</span> €</p>
          <p><strong>Priemerná cena:</strong> <span id="averagePrice">0</span> €</p>
        </div>
    </div>
```

Obrázok 46 Vytvorenie tabuľky objednávok (Zdroj: vlastné spracovanie)

V tejto HTML štruktúre môžeme vidieť rôzne vstupné prvky, ako sú textové polia, výberové boxy a tlačidlá, ktoré slúžia na filtrovanie objednávok podľa rôznych kritérií ako zákazník, cena, dátum, produkt, kvartál a rok. Zároveň je tu aj tabuľka, ktorá zobrazuje výsledky filtrovania v prehľadnej forme. Ďalej je zobrazený súhrn objednávok, ktorý zobrazuje základné štatistiky o filtrovaných objednávkach, ako je počet objednávok, predané množstvo, celková cena a priemerná cena.

JavaScript je v tejto aplikácii použitý na manipuláciu, interakciu s webovým servisom a správu dynamických zmien v rozhraní bez nutnosti obnovovať stránku. Celý proces je realizovaný pomocou *jQuery*, čo výrazne zjednodušuje volania AJAX na získavanie a odosielanie dát na server. Základná štruktúra JavaScriptu je nasledovná:

```
function loadOrders() {
    $.ajax({
    type: "POST",
    url: "webservice.asmx/GetAllOrders",
    contentType: "application/json; charset=utf-8",
    dataType: "json",
    success: function (response) {
        orders = JSON.parse(response.d);
        populateItemFilter(orders);
        populateYearSelect(orders);
        filterOrders();
    },
    error: function (xhr, status, error) {
    console.error("Chyba pri načítaní objednávok:", error);
}});}
```

Obrázok 47 Funkcia na načítanie objednávok zo servera (Zdroj: vlastné spracovanie)

Na začiatku sa volá funkcia *loadOrders*(), ktorá pomocou AJAX volania načíta všetky objednávky z webovej služby. Tieto objednávky sú následne spracované a zobrazené na stránke.

Funkcia *loadOrders*() vykoná POST požiadavku na server a načíta všetky objednávky, ktoré následne spracuje a zobrazí na stránke. Okrem toho sa vyplnia výberové polia, ako je filter produktov a rok.

Po získaní objednávok zo servera, je možné tieto objednávky filtrovať podľa rôznych kritérií, ako sú zákazník, cena, dátum, produkt, kvartál a rok. Filtrovanie je realizované funkciou *filterOrders*(), ktorá prechádza zoznamom objednávok a na základe kritérií zadaných používateľom v HTML formulároch vykoná príslušnú filtráciu.

```
function filterOrders() {
    let searchCustomer = $("#searchCustomer").val().toLowerCase();
    let minPrice = parseFloat($("#minPrice").val()) || 0;
    let maxPrice = parseFloat($("#maxPrice").val()) || Infinity;
    let startDate = $("#startDate").val();
    let selectedItem = $("#itemFilter").val();
    let selectedQuarter = $("#quarterSelect").val();
    let selectedYear = $("#yearSelect").val();
    let filteredOrders = orders.filter(order => {
    let price = parseFloat(order.Price.replace(",", "."));
    let timestamp = new Date(order.Timestamp);
    let matchCustomer = order.Customer.toLowerCase().includes(searchCustomer);
    let matchPrice = price >= minPrice && price <= maxPrice;
    let matchDate = startDate ? timestamp >= new Date(startDate) : true;
    let matchItem = selectedItem ? order.Item === selectedItem : true;
    let matchQuarter = selectedQuarter ? (Math.floor((timestamp.getMonth() + 3) / 3) ==
    selectedQuarter) : true;
    let matchYear = selectedYear ? timestamp.getFullYear() == selectedYear : true;

    return matchCustomer && matchPrice && matchDate && matchItem &&
        matchQuarter && matchYear;
});
    sortOrders(filteredOrders);
    displaySummary(filteredOrders);
}
```

Obrázok 48 Funkcia na volanie metódy filtrovania podľa zadaných kritérií (Zdroj: vlastné spracovanie)

Táto funkcia filtruje objednávky na základe hodnôt zadaných používateľom v príslušných filtroch. Potom je výsledok zoradený a zobrazený na stránke.

```
function addOrder() {
    var customer = document.getElementById('txtCustomer').value;
    var productId = document.getElementById('txtProduct').value;
    var quantity = parseInt(document.getElementById('txtQuantity').value, 10);
    var price = document.getElementById('txtPrice').value;
    var status = document.getElementById('txtStatus').value;

    $.ajax({
    type: "POST",
    url: "webservice.asmx/AddOrder",
    data: JSON.stringify({
        customer: customer,
        productId: productId,
        quantity: quantity,
        price: price,
        status: status
    }),
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: function (response) {
        document.getElementById('response').innerText = response.d;
    },
        error: function (xhr) {
        document.getElementById('response').innerText = "Chyba: " + xhr.responseText;
    }
});
}
```

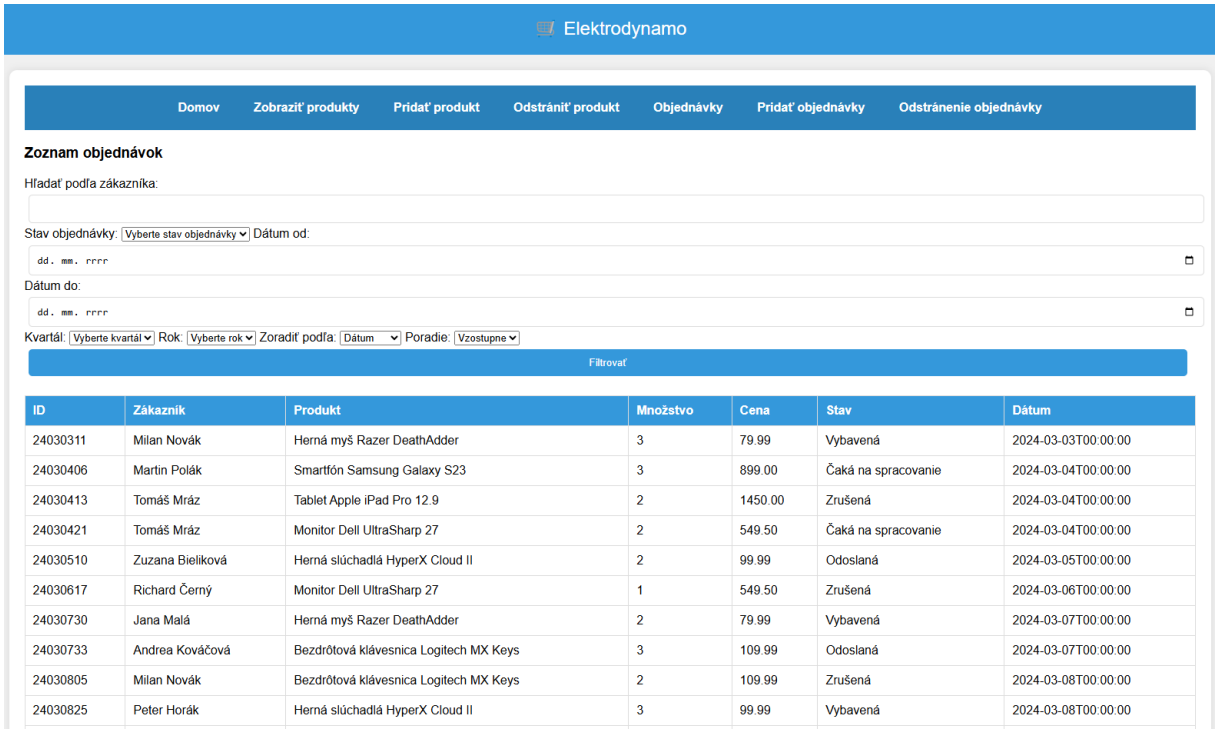
Obrázok 49 Funkcia na pridanie objednávky (Zdroj: vlastné spracovanie)

Webový servis implementovaný v C# poskytuje všetky potrebné metódy na získanie, pridanie a odstránenie objednávok. Metódy ako *GetAllOrders*, *AddOrder*, a *DeleteOrder* sa volajú z frontendu pomocou AJAX požiadaviek a umožňujú serveru spracovať dáta a poskytnúť požadované výsledky. Tieto metódy budú detailne popísané v ďalšej časti diplomovej práce.

Prostredníctvom HTML, JavaScriptu a jQuery sa poskytuje používateľské rozhranie pre filtrovanie, pridávanie a odstraňovanie objednávok. ASP.NET Web Service slúži ako backend na spracovanie požiadaviek a manipuláciu s údajmi objednávok, čo poskytuje robustný základ pre efektívnu správu objednávok v elektronickom obchode.

7 Ukážka fungovania aplikácie

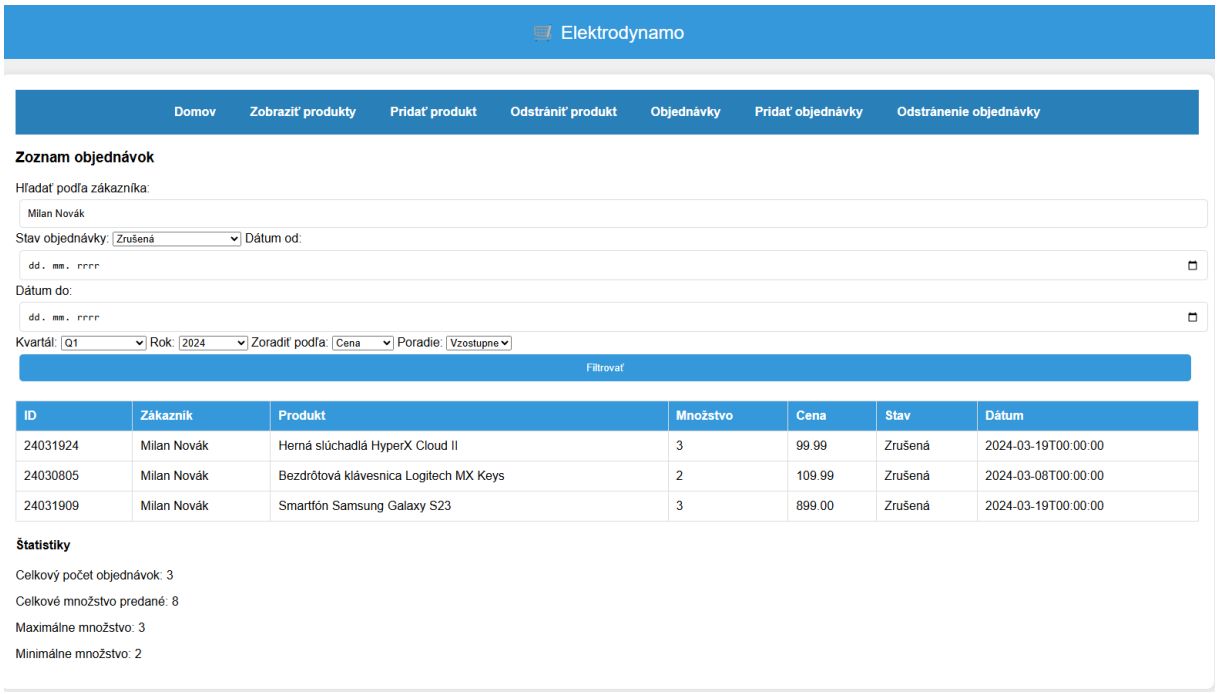
Na ilustráciu praktického využitia vytvorenej ASP.NET XML webovej služby uvádzame konkrétny príklad použitia jej funkcionalít z prostredia používateľského rozhrania. V tomto prípade po zadaní vstupných parametrov je zavolaná vystavená metóda *GetFilteredAndSortedOrders*, do ktorej sú vložené údaje zadané užívateľom.



Obrázok 50 Sekcia objednávok zobraz v okne klienta (Zdroj: vlastné spracovanie)

V rámci tohto zobrazenia je možné použiť viacero filtrov, ako napríklad vyhľadávanie podľa mena zákazníka, stavu objednávky, dátumu alebo časového obdobia (štvrťrok, rok). Následne je možné výsledky zoradiť podľa vybraného atribútu (napr. podľa ceny, množstva alebo dátumu).

Na obrázku je znázornený prípad, kedy používateľ zadal meno „Milan Novák“ a zvolil stav objednávky „Zrušená“. Po potvrdení filtrovania sa webová služba dotazom v XML formáte spojí s príslušným XML súborom elektrodynamo_objednavky.xml a vráti všetky zodpovedajúce záznamy.



Obrázok 51 Výpis po zadaní parametrov a zavolaní konkrétnej metódy (Zdroj: vlastné spracovanie)

Vo výsledku sa zobrazujú tri zrušené objednávky daného zákazníka:

- Herné slúchadlá HyperX Cloud II (3 ks, 99.99 €)
- Bezdrôtová klávesnica Logitech MX Keys (2 ks, 109.99 €)
- Smartfón Samsung Galaxy S23 (3 ks, 899.00 €)

Pod výpisom objednávok sa dynamicky aktualizujú aj štatistiky. V danom prípade:

- Celkový počet objednávok: 3
- Celkové množstvo predané (v zmysle položiek v objednávkach): 8 kusov
- Maximálne množstvo (v jednej objednávke): 3
- Minimálne množstvo: 2

Tento príklad demonštruje praktické využitie funkcionalít služby – od interaktívneho filtrovania dát až po výpočet štatistických ukazovateľov v reálnom čase. Webová služba tak umožňuje rýchle a prehľadné spracovanie objednávok v e-shope.

Diskusia

V rámci realizácie našej ASP.NET XML webovej služby pre elektronický obchod Elektrodynamo.sk sme sa stretli s viacerými praktickými výzvami, ktoré vyplynuli najmä z požiadaviek na prácu s dátami vo formáte XML, ich štruktúrované spracovanie a následnú vizualizáciu vo webovom prostredí. Počas vývoja sme sa rozhodli zachovať čo najväčšiu transparentnosť dát, preto sme uprednostnili použitie XML.

Jednou z najdôležitejších oblastí diskusie je samotná efektivita XML pri uchovávaní a filtrovaní dát. XML súbory ako *xmlko.xml* a *elektrodynamo_objednavky.xml* sa ukázali ako vhodné pre menšie objemy údajov, avšak pri väčšom počte produktov a objednávok by sa mohol prejaviť pokles výkonu. Napriek tomu však poskytli výhodu v podobe jednoduchej čitateľnosti a možnosti manuálneho zásahu bez potreby špeciálnych nástrojov.

Z hľadiska užívateľskej skúsenosti by sme do budúcnosti odporúčali rozšíriť funkcionalitu webovej aplikácie o možnosti autentifikácie, pokročilé vizualizačné prehľady a optimalizované rozhranie pre mobilné zariadenia. Tiež by bolo zaujímavé preskúmať prechod z XML na hybridné riešenie, kde by sa často meniace údaje spracúvali databázovo a statické informácie ostali v XML.

Celkovo považujeme náš návrh služby za funkčný základ pre online obchod, ktorý efektívne využíva technológie XML a ASP.NET na spracovanie a prezentáciu údajov o výpočtovej technike. Výsledky práce ukazujú, že aj bez použitia plnohodnotnej databázy je možné vytvoriť riešenie s dobrým pomerom medzi jednoduchosťou a výkonom.

Záver

Cieľom tejto diplomovej práce bolo navrhnúť a implementovať funkčnú ASP.NET XML webovú službu, ktorá umožňuje sledovanie parametrov výpočtovej techniky v prostredí elektronického obchodu Elektrodynamo.sk. V úvode sme si stanovili vytvoriť riešenie, ktoré poskytne používateľovi prehľadné a usporiadané informácie o produktoch, ako sú notebooky, stolové počítače, tablety a príslušenstvo, s možnosťou ich filtrovania, triedenia a analýzy v čase. Zároveň sme chceli overiť vhodnosť XML webových služieb pre takéto účely v porovnaní s inými spôsobmi sledovania údajov, ako sú Excel, Access či REST API rozhrania.

Po realizácii vývoja a testovaní služby môžeme konštatovať, že hlavné ciele práce boli splnené. Webová služba bola navrhnutá modulárne a s dôrazom na prehľadnosť, pričom umožňuje:

- dynamické filtrovanie a triedenie produktov podľa typu, značky, kategórie a ďalších parametrov,
- zobrazenie produktových údajov od jednotlivých výrobcov a ich základných vlastností,
- porovnanie stavu zásob na začiatku a na konci sledovaného obdobia,
- vyhodnotenie štatistických ukazovateľov,

Služba tak poskytuje rozšíriteľný základ pre sledovanie produktových údajov v rámci elektronického obchodu, s potenciálom ďalšieho rozvoja napríklad smerom k integrácii s databázovým systémom alebo s inými informačnými systémami.

V rámci porovnania s alternatívnymi riešeniami sa potvrdilo, že XML webové služby ponúkajú kompromis medzi zrozumiteľnosťou, štruktúrovanosťou a rozšíriteľnosťou. Na rozdiel od statických nástrojov ako Excel či Access, poskytujú možnosť centrálneho, online prístupu k údajom, pričom sú zároveň menej náročné na nasadenie než plnohodnotné REST API systémy. To z nich robí riešenie pre malé a stredné podniky, ktoré potrebujú spravovať technické dáta efektívne, bez potreby rozsiahleho vývoja.

Na záver môžeme konštatovať, že navrhnutá aplikácia spĺňa počiatočné zadanie diplomovej práce a zároveň otvára priestor pre ďalšie vylepšenia – napríklad doplnenie

autentifikácie používateľov, vizualizáciu dát vo forme grafov alebo prepojenie s ďalšími zdrojmi údajov. Diplomová práca tak ukazuje, že aj technológie založené na XML a ASP.NET môžu byť v súčasnosti plnohodnotným a praktickým riešením v oblasti podnikových informačných systémov.

Zoznam použitej literatúry

- Alonso, G., Casati, F., Kuno, H., & Machiraju, V. (2004). Web services: Concepts, architectures and applications. Springer.
- Alshuqayran, M., & Shamkhi, A. (2022). Comparison of SOAP web services and RESTful APIs. Journal of Software Engineering, 15(2), 65–82.
- Brown, R. (2019). REST API and JSON: Modern technologies for web applications. WebTech Publishers.
- C# Corner. (n.d.). Working with XML in C#. dostupné online 15.1.2025, https://www.c-sharpcorner.com/article/working-with-xml-in-C-Sharp/
- Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., & Weerawarana, S. (2002). Unraveling the Web services web: An introduction to SOAP, WSDL, and UDDI. IEEE Internet Computing, 6(2), 86–93. https://doi.org/10.1109/4236.991449
- Erl, T. (2005). Service-oriented architecture: Concepts, technology, and design. Prentice Hall.
- Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures (Doctoral dissertation, University of California, Irvine). https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
- Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J. J., Nielsen, H. F., & Karmarkar, A. (2007). SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). World Wide Web Consortium (W3C). https://www.w3.org/TR/soap12-part1/
- Harris, S. (2019). Using XML for data management in electronic commerce. Journal of Web Services, 12(3), 45–59.
- IBM. (n.d.). What are web services? dostupné online 6.2.2025, https://www.ibm.com/docs/en/was/9.0.5?topic=SSAW57_9.0.5/com.ibm.websphere.nd.multiplatform.doc/ae/cwebs_webserv.html
- IBM. (n.d.). What is a relational database? dostupné online 26.1.2025, https://www.ibm.com/topics/relational-databases
- Johnson, M. (2017). The role of XML in online data exchange. International Journal of Information Systems, 29(1), 34–48.
- Johnson, M., & Miller, L. (2021). Integration of IoT and web services for product data management. Journal of Modern Technology, 19(4), 72–84.
- Lee, D. (2021). Challenges in XML data processing. Computing Technologies, 35(2), 80–92.
- Learn Microsoft. (n.d.). ASP.NET Web Forms. dostupné online 16.1.2025, https://learn.microsoft.com/en-us/aspnet/web-forms/
- Microsoft. (n.d.-b). What is REST?. dostupné online 18.1.2025, https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design
- Microsoft Learn. (n.d.). ASP.NET Overview. dostupné online 13.1.2025, https://learn.microsoft.com/en-us/aspnet/overview/

- Microsoft Learn. (n.d.). What is a Database? dostupné online 13.1.2025, https://learn.microsoft.com/en-us/sql/relational-databases/databases/databases
- Microsoft Learn. (n.d.). XML in C#. dostupné online 6.2.2025, https://learn.microsoft.com/en-us/dotnet/standard/data/xml/
- Microsoft Learn. (2021, September 15). Interoperating with ASMX Web Services - WCF. Retrieved from https://learn.microsoft.com/en-us/dotnet/framework/wcf/samples/interoperating-with-asmx-web-services
- Microsoft Learn. (2022, May 8). Write a web service by using Visual C# .NET. Retrieved from https://learn.microsoft.com/en-us/troubleshoot/developer/visualstudio/csharp/language-compilers/write-web-service
- Miller, J. (2019). Data tracking and management in e-commerce. E-commerce Insights, 10(5), 102–115.
- OASIS Open. (2006, February 1). Web Services Security v1.1. Retrieved from https://www.oasis-open.org/standard/wssv1-1/
- Oracle. (n.d.). Introduction to XML. dostupné online 18.1.2025, https://docs.oracle.com/javase/tutorial/jaxp/intro/index.html
- Papazoglou, M. P. (2003). Service-oriented computing: Concepts, characteristics and directions. In Proceedings of the Fourth International Conference on Web Information Systems Engineering (pp. 3–12). https://doi.org/10.1109/WISE.2003.1254461
- Red Hat. (n.d.). What is an API? dostupné online 17.3.2025, https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces
- Richardson, L., & Ruby, S. (2007). RESTful Web Services. O'Reilly Media.
- Smith, A. (2020). Relational databases for e-commerce businesses. Database Management, 14(1), 53–64.
- Smith, A., Walker, B., & Thomas, T. (2021). The future of data management in e-commerce. Journal of Digital Commerce, 26(2), 121–135.
- TutorialsPoint. (n.d.). ASP.NET Web Services. dostupné online 6.2.2025, https://www.tutorialspoint.com/asp.net/asp.net_web_services.htm
- W3C. (2001). Web Services Description Language (WSDL) 1.1. https://www.w3.org/TR/wsdl
- W3C. (2004). UDDI Version 3.0.2. https://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm
- W3C. (2007). Web services architecture. World Wide Web Consortium. https://www.w3.org/TR/ws-arch/
- W3C. (n.d.). Web Services Architecture. dostupné online 3.12.2024, https://www.w3.org/TR/ws-arch/
- W3Schools. (n.d.). RESTful Web Services. dostupné online 12.3.2025, https://www.w3schools.com/xml/xml_services.asp
- W3Schools. (n.d.). SOAP Web Services. dostupné online 21.3.2025, https://www.w3schools.com/xml/xml_soap.asp

- W3Schools. (n.d.). WSDL Tutorial. dostupné online 13.3.2025, https://www.w3schools.com/xml/xml_wsdl.asp
- W3Schools. (n.d.). XML Web Services. dostupné online 20.4.2025, https://www.w3schools.com/xml/xml_services.asp
- Wang, Y., & Lee, S. (2018). Database systems in modern online stores. International Journal of Computer Science, 16(4), 33–45.
- World Wide Web Consortium. (n.d.). SOAP Version 1.2. dostupné online 18.3.2025, from https://www.w3.org/TR/soap12-part1/
- World Wide Web Consortium. (n.d.). Web Services Description Language (WSDL) 1.1. dostupné online 13.3.2025, https://www.w3.org/TR/wsdl

Prílohy

Zdrojový kód obsahujúci webovú službu a klienta sa nachádza v adresári *asp_net_aplikacia*, ktorý je obsiahnutý v archíve BAŇKOS_asp_net_aplikacia_príloha_k_DP.zip. Ďalej je vložený návod k spusteniu aplikácie s názvom BAŇKOS_navod_na_spustenie_ASP_NET_aplikacie_priloha_k_DP.pdf.