EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 103004/I/2023/421000214401

ASP .NET XML webová služba poskytujúca usporiadané informácie o klientoch domova sociálnych služieb

Diplomová práca

2023 **Bc. Martin Lukáč**

EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

ASP .NET XML webová služba poskytujúca usporiadané informácie o klientoch domova sociálnych služieb

Diplomová práca

Študijný program: Informačný manažment

Študijný odbor: Ekonómia a manažment

Školiace pracovisko: Katedra aplikovanej informatiky

Vedúci záverečnej práce: Ing. Igor Košťál, PhD.

Bratislava 2023 **Bc. Martin Lukáč**

Ekonomická univerzita v Bratislave
Fakulta hospodárskej informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Martin Lukáč
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 klientoch domova sociálnych služieb

Anotácia: Diplomant v práci zanalyzuje možnosti použitia XML webových služieb na sledovanie parametrov klientov domova sociálnych služieb vo vybratom období v elektronickom informačnom systéme a porovná ich použitie s doterajším spôsobom sledovania týchto parametrov vo vybraných domovoch sociálnych služieb. 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 klientov domova sociálnych služieb v sledovanom období, ktorými môžu byť celkový počet klientov v tomto domove na začiatku a konci sledovaného
obdobia, počty a zoznamy klientov s ich základnými dátami v jednotlivých kategóriách imobility, zdravotného stavu a veku
na začiatku a konci sledovaného obdobia a iné ich parametre.

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

Dátum zadania: 25.10.2021

Dátum schválenia: 17.04.2023

prof. Ing. Ivan Brezina, CSc.
osoba zodpovedná za realizáciu študijného programu

Čestné vyhlásenie

Čestne vyhlasujem, že záverečnú prácu s názvom: ASP .NET XML webová služba poskytujúca usporiadané informácie o klientoch domova sociálnych služieb som vypracoval samostatne. Použitú literatúru som uviedol na príslušnom mieste a neporušil som autorský zákon.

Dátum: 28.04.2023 .....

Martin Lukáč

Poďakovanie

Chcel by som sa poďakovať Ing. Igorovi Košťálovi, PhD. za stanovenie praktickej a zaujímavej témy a osobné odborné konzultácie, ktoré prispeli ku skvalitneniu záverečnej práce a naplneniu cieľov práce.
Ďakujem.

ABSTRAKT

LUKÁČ, Martin: *ASP .NET XML webová služba poskytujúca usporiadané informácie o klientoch domova sociálnych služieb.* – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; katedra aplikovanej informatiky. – Vedúci záverečnej práce: Ing. Igor Košťál, PhD. – Bratislava: FHI EU, 2023, 68 strán.

Cieľom záverečnej práce je zanalyzovať možnosti použitia XML webových služieb na sledovanie parametrov klientov domova sociálnych služieb vo vybranom období v elektronickom informačnom systéme a porovnať ich použitie s doterajším spôsobom sledovania týchto parametrov vo vybraných domovoch sociálnych služieb. Ďalším cieľom bolo vytvorenie ASP .NET XML webovú službu poskytujúcu prostredníctvom svojej funkcionality jej klientovi podľa vybraných kritérií usporiadané informácie o parametroch klientov domova sociálnych služieb v sledovanom období. Práca je rozdelená do štyroch kapitol. Obsahuje 38 obrázkov a 2 prílohy. Prvá kapitola je venovaná teoretickému zobrazeniu tematiky webových služieb. Definovali sme v nej pojmy potrebné na pochopenie tematiky webových služieb a webového klienta. V ďalšej časti sme charakterizovali ciele, metodiku a metódy práce. V tretej kapitole sme sa zamerali na analýzu výsledkov našej diplomovej práce. Výsledkom riešenia diplomovej práce je ASP .NET XML webová služba, ktorá poskytuje svojmu klientovi podľa vybraných kritérií usporiadané informácie o parametroch klientov domova sociálnych služieb a zároveň porovnanie tohto riešenia s už existujúcim riešením zavedeným v domove sociálnych služieb, ktoré sme bližšie popísali vo štvrtej kapitole. V piatej a poslednej kapitole sme sa zamerali na diskusiu, kde sme zhrnuli prínos nášho riešenia, no taktiež poukázali na možné zlepšenia.

Kľúčové slová: ASP .NET, webová služba, XML, ASMX, C sharp, spracovanie údajov klientov domova sociálnych služieb, filtrovanie dát klientov

ABSTRACT

LUKÁČ, Martin: *ASP .NET XML Web service that provides sorted information about social services home clients.* – University of Economics in Bratislava. Faculty of Economic Informatics; Department of Applied Informatics. – Head of thesis: Ing. Igor Košťál, PhD. – Bratislava: FHI EU, 2023, 68 pages.

The aim of the final thesis is to analyze the possibilities of using XML web services to monitor the parameters of clients of the social services home in the selected period in the electronic information system and to compare their use with the current way of monitoring these parameters in selected social service homes. Another goal was the creation of an ASP .NET XML web service providing, through its functionality, to its web client, according to selected criteria, organized information about the parameters of the social services home clients in the monitored period. The work is divided into four chapters. Contains 38 images and 2 attachments. The first chapter is devoted to the theoretical presentation of the topic of web services. In it, we defined the terms necessary to understand the topic of web services and the web client. In the next chapter, we characterized the goals, methodology and work methods. In the third chapter, we focused on the analysis of the results of our final thesis. The result of the thesis is an ASP .NET XML web service, which provides information about the parameters of the social services home clients organized according to selected criteria to its web client, as well as a comparison of our solution with an already existing solution used in a social services home which we described in more detail in the fourth chapter. In the fifth and last chapter, we focused on the discussion, where we summarized the contribution of our solution, but also pointed out possible improvements.

Key words: ASP .NET, web service, XML, ASMX, C sharp, data processing of social services home clients, filtering client data

Obsah

Zoznam obrázkov

Úvod

Domovy sociálnych služieb poskytujú klientom zabezpečenie ubytovania a starostlivosť o ich zdravie a komfort. Pre účely sledovania parametrov klientov v týchto domovoch a poskytovania potrebných informácií je dôležité mať k dispozícii efektívny a spoľahlivý systém. V súčasnosti sa stále častejšie využívajú XML webové služby ako prostriedok na poskytovanie dát a informácií medzi rôznymi aplikáciami.

V diplomovej práci sme sa zamerali na predstavenie možností využitia ASP .NET XML webových služieb na sledovanie parametrov klientov domova sociálnych služieb a porovnali ich použitie s doterajším spôsobom sledovania týchto parametrov. V práci sme navrhli a vytvorili ASP .NET XML webovú službu, ktorá poskytuje svojmu HTML klientovi usporiadané informácie o parametroch klientov domova sociálnych služieb v sledovanom období.

Prvá kapitola našej diplomovej práce sa venuje vysvetleniu teoretických pojmov potrebných na lepšie pochopenie tematiky webových služieb. Popisuje webové služby a výhody ich použitia, konkrétne sa zameriava na ASP.NET XML webové služby a ich vlastnosti. Ďalej sa zaoberá XML, HTTP, SOAP, REST a WSDL, ktoré sú neoddeliteľnou súčasťou tejto problematiky. Ako posledné sme sa zamerali na vysvetlenie pojmov potrebných na lepšie pochopenie tematiky webového klienta, ako HTML, CSS a AJAX.

Druhá kapitola diplomovej práce popisuje cieľ, metodiku práce a metódy skúmania. V tretej kapitole sú uvedené výsledky práce, vrátane štruktúry projektu, použitých knižníc a kľúčových členských metód webovej služby. Ďalej sa venuje vedľajším členským metódam a HTML klientom. Následne sme v štvrtej porovnali naše riešenie s existujúcim riešením domova sociálnych služieb XYZ a v poslednej kapitole sme načali diskusiu, ako by bolo možné naše riešenie ešte vylepšiť a zároveň predviedli jeho využitie v praxi. Nami navrhnuté riešenie je dôležitým krokom v zlepšovaní sledovania parametrov klientov v domovoch sociálnych služieb a môže pomôcť zlepšiť kvalitu starostlivosti o klientov v budúcnosti.

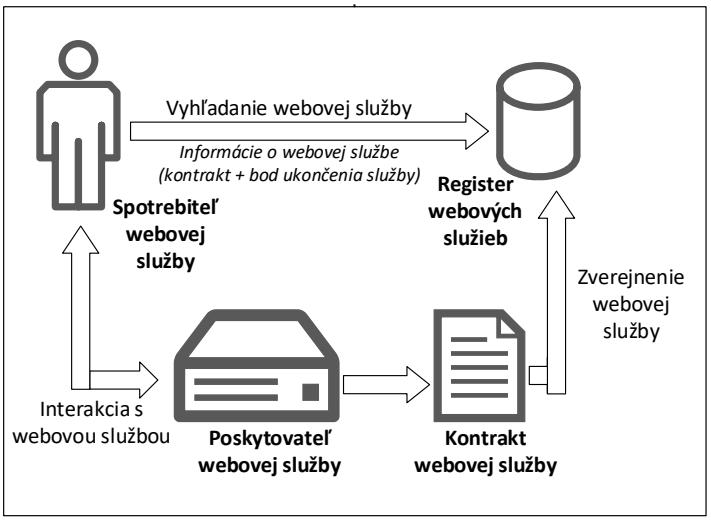
1 Použité technológie

V tejto kapitole sme sa zamerali na vysvetlenie základných pojmov potrebných na pochopenie tematiky. Definovali sme pojmy ako webová služba a objasnili rozdiel medzi webovou službou a webovou aplikáciou. Vysvetlili sme výraz ASP .NET a súbory ASMX, ktoré ASP .NET používa. Ďalej sme sa venovali vysvetleniu pojmov XML, HTTP, SOAP, REST a WSDL. Následne sme definovali výrazy použité pri tvorbe webového klienta.

1.1 Webová služba

Webová služba je implementácia architektúry orientovanej na služby (SOA). SOA je architektonický štýl vývoja softvérových aplikácií, založený na vytváraní softvérových služieb dostupných cez sieť. Služba je dobre definovaná, samostatná funkcia, ktorá sa nespolieha na kontext alebo stav iných služieb. Ako prvé, poskytovateľ služieb implementuje službu špecifikovanú v zmluve o poskytovaní služieb.

Servisná zmluva obsahuje rozhranie služby, popis služby a zásady služby. Zmluvu o poskytovaní služieb zverejňuje poskytovateľ služieb v registri služieb, ktorému sa hovorí aj zoznam dostupných služieb. Servisná zmluva voľne spája systém SOA v tom zmysle, že rôzne komponenty SOA, služby, nemusia navzájom komunikovať. Servisná zmluva voľne spája systém SOA v tom zmysle, že rôzne komponenty SOA, služby, o sebe nemusia veľa vedieť. Register služieb potom používa spotrebiteľ služby na nájdenie služby alebo služieb od poskytovateľa, ktoré budú užitočné.[4]



Obrázok 1 - Systém SOA (Zdroj: vlastné spracovanie podľa Foxa a Hao)

Jedna webová služba pozostáva zo služby a popisu služby, kde služba je softvérový modul ponúkaný poskytovateľom služby, ktorý je dostupný cez web. Popis služby obsahuje podrobnosti o rozhraní a implementáciách služby vrátane typov údajov, metadát, informácií o kategorizácii a miesta, kde je služba vystavená. Popis služby je zverejnený v registri služieb, kde sú služby zobrazené a zoskupené podľa účelu. Webová služba je kľúčovým prvkom pri integrácii rôznych informačných systémov, keďže informačné systémy môžu byť založené na rôznych platformách, programovacích jazykoch a technológiách.[25]

1.1.1 Príklad jednoduchej webovej služby

Na lepšie predstavenie tematiky webových služieb sme na obrázku 2 uviedli príklad jednoduchej webovej služby na vytvorenie webovej služby *TestWS*, ktorá vykonáva jednoduchú metódu *HelloWorld()* a vracia stringový element s atribútom „*Hello World*“.

```
        using System;
        using System.Collections.Generic;
        using System.Linq;
        using System.Web;
        using System.Web.Services;


namespace Senioro_1
{
    [WebService(Namespace = "TestWS")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]

    public class WebService1 : System.Web.Services.WebService
    {

        [WebMethod]
        public string HelloWorld()
        {
            return "Hello World";
        }
    }
       }
```

```
<string xmlns="TestWS">Hello World</string>
```

Obrázok 2 - Príklad webovej služby (Zdroj: Vlastné spracovanie)

1.1.2 Rozdiel medzi webovou službou a webovou aplikáciou

Webová služba a webová aplikácia sú dva rôzne typy softvéru, ktoré slúžia na rôzne účely. Webová služba je softvérový systém, ktorý odhaľuje štandardizovaný spôsob interakcie s ňou cez sieť, často pomocou správ XML alebo JSON formátovaných podľa architektúry SOAP alebo REST. Primárnym cieľom webovej služby je umožniť interoperabilitu medzi rôznymi softvérovými systémami, čo im umožní navzájom komunikovať a zdieľať údaje.

Na druhej strane webová aplikácia je softvérový program, ku ktorému sa pristupuje prostredníctvom webového prehliadača alebo webového zariadenia. Webové aplikácie sú zvyčajne navrhnuté tak, aby vykonávali špecifické úlohy alebo funkcie, ako je elektronický obchod, sociálne siete alebo správa obsahu.

Stručne povedané, webová služba poskytuje štandardizovaný spôsob komunikácie a zdieľania údajov medzi rôznymi softvérovými systémami, zatiaľ čo webová aplikácia je samostatný softvérový program, ku ktorému sa pristupuje prostredníctvom webového prehliadača alebo zariadenia s podporou webu.

Webová služba je softvérový systém navrhnutý tak, aby podporoval interoperabilnú interakciu stroj-stroj cez sieť. Inými slovami, ide o štandardizovaný spôsob vzájomnej komunikácie softvérových systémov cez internet bez ľudského zásahu. Webové služby používajú XML na kódovanie údajov a SOAP (Simple Object Access Protocol) na ich prenos medzi systémami. Môžu byť použité na rôzne účely, ako je integrácia systémov, automatizácia obchodných procesov a poskytovanie prístupu k údajom a funkciám.

Webová aplikácia je softvérová aplikácia, ku ktorej sa pristupuje cez sieť, zvyčajne cez internet. Webové aplikácie sú navrhnuté tak, aby fungovali vo webovom prehliadači, čo znamená, že sú nezávislé od platformy a je možné k nim pristupovať z akéhokoľvek zariadenia s internetovým pripojením. Webové aplikácie možno použiť na širokú škálu účelov, ako je elektronický obchod, sociálne siete, online bankovníctvo a ďalšie.

Stručne povedané, webové služby predstavujú spôsob, akým softvérové systémy navzájom komunikujú, zatiaľ čo webové aplikácie sú typom softvérovej aplikácie, ku ktorej je možné pristupovať cez internet. Webové služby sa často používajú na podporu webových aplikácií, ale možno ich použiť aj na iné účely.[24, 27]

1.2 ASP .NET

ASP .NET je súbor nástrojov na vývoj webu ponúkaná spoločnosťou Microsoft. Programy ako Visual Studio .NET a Visual Web Developer umožňujú webovým vývojárom vytvárať dynamické webové stránky pomocou vizuálneho rozhrania. Samozrejme, programátori môžu napísať svoj vlastný kód a skripty a začleniť ich aj do webových stránok ASP. NET. Aj keď je ASP .NET často vnímaný ako nástupca programovacej technológie ASP od Microsoftu, podporuje aj Visual Basic .NET, JScript .NET a open-source jazyky ako Python a Perl.

ASP .NET je postavené na .NET frameworku, ktorý poskytuje aplikačné programové rozhranie (API) pre softvérových programátorov. Vývojové nástroje .NET možno použiť na vytváranie aplikácií pre operačný systém Windows aj pre web. Programy ako Visual Studio .NET poskytujú vývojárom vizuálne rozhranie na vytváranie ich aplikácií, vďaka čomu je .NET rozumnou voľbou aj na navrhovanie webových rozhraní.

Aby webová lokalita ASP .NET fungovala správne, musí byť zverejnená na webovom serveri, ktorý podporuje aplikácie ASP .NET. Webový server Internet Information Services (IIS) spoločnosti Microsoft je zďaleka najbežnejšou platformou pre webové lokality ASP .NET. Aj keď sú pre systémy založené na Linuxe k dispozícii niektoré možnosti open source, tieto alternatívy často poskytujú menej ako plnú podporu pre aplikácie ASP .NET.[4]

Od prvého vydania .NET Framework 1.0 na začiatku roku 2002, Microsoft vynaložil veľa úsilia a času na vývoj do ASP .NET, časti .NET frameworku, ktorá umožňuje vytvárať bohaté webové aplikácie. Toto prvé vydanie znamenalo radikálnu zmenu od staršej technológie spoločnosti Microsoft na vytváranie webových stránok nazývaných Active Server Pages (ASP), teraz označované ako klasické ASP. Uvedenie ASP .NET 1.0 a súvisiaceho Visual Studio .NET poskytlo vývojárom nasledujúce výhody oproti klasickému ASP:

- Čisté oddelenie medzi prezentáciou a kódom. Pri klasickom ASP bola programovacia logika často rozptýlená po celom HTML stránky, čo sťažovalo neskoršie vykonanie zmien na stránke.
- Vývojový model, ktorý bol oveľa bližší spôsobu, akým sa programujú desktopové

aplikácie. To uľahčilo mnohým programátorom Visual Basicu prechod na webové aplikácie.

- Vývojový nástroj bohatý na funkcie (nazývaný Visual Studio .NET), ktorý umožnil vývojárom vytvárať a kódovať svoje webové aplikácie vizuálne.
- Výber medzi množstvom objektovo orientovaných programovacích jazykov (OOP), z ktorých Visual Basic .NET a C# (vyslovuje sa ako C-Sharp) sú teraz najobľúbenejšie.
- Prístup k celému .NET Frameworku, čo po prvýkrát znamenalo, že weboví vývojári mali jednotný a jednoduchý spôsob prístupu k mnohým pokročilým funkciám pre prácu s databázami, súbormi, e-mailom, sieťovými nástrojmi a oveľa viac.

Napriek mnohým výhodám ASP .NET oproti staršiemu modelu, používanie ASP.NET znamenalo aj zvýšenie zložitosti a znalostí potrebných na vytváranie aplikácií s ním, čo sťažilo mnohým novým programátorom začať s ASP .NET.

Po prvom vydaní v roku 2002 Microsoft vydal ďalšiu verziu .NET Framework (nazývanú .NET 1.1) a vývojové IDE (nazývané Visual Studio .NET 2003). Mnoho ľudí to považovalo za servisný balík pre prvé vydanie, hoci priniesol aj veľa nových vylepšení v rámci a vo vývojových nástrojoch.

V novembri 2005 boli vydané Visual Studio 2005 a ASP.NET 2.0. K príjemnému prekvapeniu mnohých vývojárov na celom svete sa Microsoftu opäť podarilo výrazne zlepšiť a rozšíriť produkt pridaním mnohých funkcií a nástrojov, ktoré pomohli znížiť zložitosť, ktorá bola predstavená s ASP .NET 1.0. Noví sprievodcovia a inteligentné ovládacie prvky umožnili zredukovať kód potrebný na zostavenie aplikácie, čím sa skrátila krivka učenia nových vývojárov a zvýšila sa ich produktivita.

V novembri 2007 spoločnosť Microsoft vydala Visual Studio 2008 a framework ASP .NET 3.5, nasledovali Visual Studio 2010 a ASP .NET 4 v marci 2010 a Visual Studio 2012 a ASP .NET 4.5 v septembri 2012. Každá verzia pridala veľa nových funkčností, vrátane LINQ, integrácia AJAX Framework, ADO.NET Entity Framework, zahrnutie jQuery a ďalšie.[23]

1.2.1 ASMX

Súbor s príponami .asmx je súbor webovej služby ASP .NET, ktorý poskytuje komunikáciu medzi dvoma objektmi cez internet pomocou protokolu SOAP (Simple Object Access Protocol). Je nasadená ako služba na webovom serveri so systémom Windows na spracovanie prichádzajúcich požiadaviek a vrátenie odpovede. Na rozdiel od súborov ASPX, ktoré obsahujú kód na vizuálne zobrazenie webových stránok ASP .NET, súbory ASMX bežia na serveri na pozadí a vykonávajú rôzne úlohy, ako je pripojenie k databáze, získavanie údajov a ich vrátenie vo formáte, v akom bola požiadavka vykonaná. Používajú sa špeciálne pre webové služby XML.[3]

Súbory ASMX sú vo formáte obyčajného textu a možno ich otvárať alebo upravovať v aplikáciách, ako je Microsoft Visual Studio alebo textové editory. Je to proprietárny formát súborov spoločnosti Microsoft a má dobre definovanú syntax pre vytváranie webových služieb. Odpoveď zo súboru ASMX je vo forme SOAP XML a má nasledujúce prvky:

- **Obálka** – koreňový prvok, ktorý identifikuje dokument XML ako správu SOAP.
- **Hlavička** – Voliteľný prvok, ktorý obsahuje informácie špecifické pre aplikáciu, ako sú autentifikačné údaje. Ak je prítomný prvok Hlavička, musí to byť prvý podriadený prvok prvku Obálka.
- **Telo** – Obsahuje správu SOAP určenú pre príjemcu.
- **Porucha** – voliteľný prvok, ktorý sa používa na označenie chybových hlásení. Ak je prítomný prvok Porucha, musí to byť podriadený prvok prvku Telo.

Súbory ASMX je možné písať v jazykoch .NET, ako sú C#, Visual Basic (VBA) alebo JScript.[3]

Rozdiel medzi ASPX a ASCX:

- ASPX, Active Server Pages, súbory sú programovacie súbory, ktoré sú generované pomocou Microsoft ASP .NET framework bežiaceho na webových serveroch. Tie sa vykreslia vo webovom prehliadači klienta, keď používateľ požiada o prístup na takúto stránku.

- ASCX, Active Server User Control, definuje používateľské ovládacie prvky, ktoré sa používajú na definovanie opätovne použiteľných ovládacích prvkov na webových stránkach ASP .NET alebo celej webovej lokalite.[3]

1.3 XML

XML (eXtensible Markup Language) je metajazyk (jazyk používaný na opis iných jazykov) na definovanie slovnej zásoby (vlastné značkovacie jazyky), ktorý je kľúčom k dôležitosti a popularite XML. Slovníky založené na XML (napríklad XHTML) umožňujú popisovať dokumenty zmysluplným spôsobom.

Kľúčový rozdiel medzi XML a HTML je v tom, že XML umožňuje vytváranie vlastných slovníkov s vlastnými značkami a pravidlami, zatiaľ čo HTML poskytuje jeden vopred vytvorený slovník s vlastnou pevnou súpravou značiek a pravidiel. XHTML a ďalšie slovníky založené na XML sú aplikácie XML. XHTML bol vytvorený ako čistejšia implementácia HTML.[6]

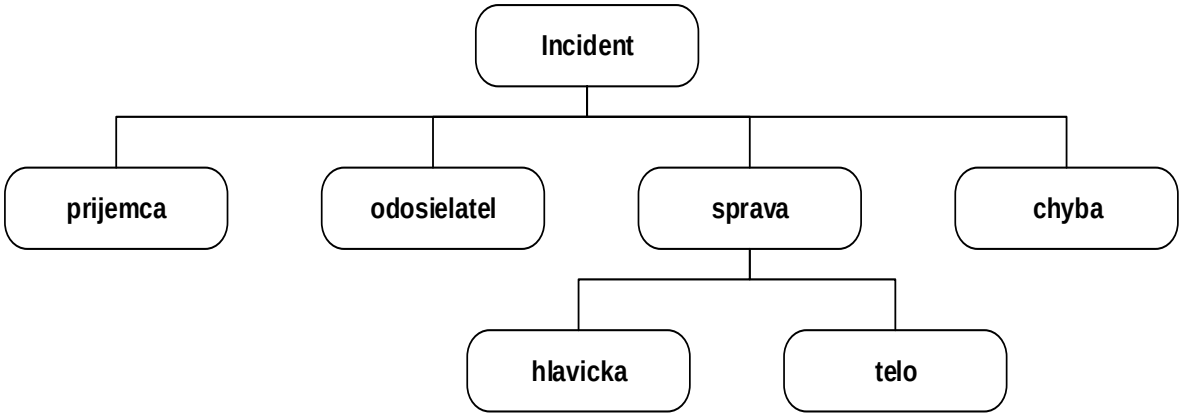
XML dokument obsahuje:

- **XML deklarácia** - Dokument XML zvyčajne začína nepovinnou deklaráciou XML, špeciálnou značkou, ktorá analyzátoru XML hovorí, že dokument je XML. Keď je prítomná deklarácia XML, nemôže sa pred ňou nachádzať nič iné. Deklarácia XML vyzerá ako <*?xml version="1.0"?*>, v ktorej atribút nepovinná verzia identifikuje verziu špecifikácie XML, ktorej dokument vyhovuje. Počiatočná verzia tejto špecifikácie (1.0) bola predstavená v roku 1998 a je široko implementovaná.

- **Prvky a atribúty** – za deklaráciou XML nasleduje hierarchická (stromová) štruktúra prvkov, kde prvok je časť dokumentu oddelená počiatočnou značkou (napríklad <*name*>) a koncovou značkou (napríklad </*name*>), alebo ide o značku s prázdnym prvkom (samostatná značka, ktorej názov končí lomkou [/], ako napríklad <*break/*>). Počiatočné značky a koncové značky obklopujú obsah a možno aj iné značky, zatiaľ čo značky s prázdnymi prvkami neobklopujú nič.[6]

1.3.1 *Príklad XML*

Ako príklad XML na obrázku 3 uvádzame schému hierarchie jednotlivých XML elementov. Vďaka tejto schéme sme vypracovali príklad na jednoduchý XML dokument, ktorý obsahuje element incident, ktorý je odoslaný príjemcovi Marcovi Smithovi od odosielateľa Johna Doea. Obsahuje hlavičku a telo správy incidentu a taktiež chybovú hlášku, ktorá v našom príklade neobsahuje žiadnu hodnotu a tento element je prázdny. Tento XML dokument sa nachádza na obrázku 4.



Obrázok 3 - Hierarchia XML elementov (Zdroj: vlastné spracovanie podľa Jeffa Friesena)

```
<?xml version="1.0" encoding="UTF-8"?>
  <Incident>
  <prijemca>Marco Smith</prijemca>
  <odosielatel>John Doe</odosielatel>
  <sprava>
  <hlavicka>Uspech</hlavicka>
  <telo>Incident uspesne odoslany!</telo>
  </sprava>
  <chyba/>
</Incident>
```

Obrázok 4 - Príklad XML dokumentu (Zdroj: vlastné spracovanie)

1.3.2 Výhody XML

1. **XML je štandard priemyslu** - XML je priemyselný štandard, ktorý nie je závislý od dodávateľa. Z histórie vyplýva, že štandardy, ktoré sú závislé od dodávateľov softvéru nie sú v softvérovom priemysle príliš akceptované. Táto neakceptácia ovplyvňuje celkové zdieľanie údajov naprieč platformami a integráciu. To, že je priemyselným štandardom, pomohlo XML získať obrovské uznanie.

2. **XML je samopopisujúce** - Dokumenty XML sa popisujú samé. Vďaka využívaniu značiek na označenie elementov sú čitateľnejšie ako napríklad súbory CSV (comma-separated values).

3. **XML je rozšíriteľné** - Značkovacie jazyky ako HTML majú daný počet značiek a atribútov – nie je možné pridávať vlastné značky. XML, na druhej strane, umožňuje definovať svoje vlastné značky.

4. **XML možno ľahko spracovať** - Formát CSV býval tradičným spôsobom reprezentácie a prenosu údajov. Na spracovanie takýchto údajov je však potrebné poznať presné umiestnenie čiarky (,) alebo akéhokoľvek iného použitého oddeľovača. To sťažuje čítanie a písanie dokumentu. Problém sa stáva závažným, pri práci s množstvom úplne odlišných a neznámych súborov CSV. Dokumenty XML môžu byť spracované softvérom nazývaným syntaktický analyzátor. Pretože dokumenty XML používajú značky na označenie elementov, syntaktický analyzátor ich môže ľahko prečítať.

5. **XML možno použiť na jednoduchú výmenu údajov** - Integrácia aplikácií naprieč platformami a rôznymi dodávateľmi je vždy ťažká a náročná. Kľúčovým problémom takýchto aplikácií je výmena údajov v rôznorodých systémoch. XML je priemyselný štandard, takže má masívnu podporu u väčšiny predajcov a vývojárov softvérov.

6. **XML možno použiť na jednoduché zdieľanie údajov** - Skutočnosť, že XML nie je sú iné ako textové údaje, zabezpečuje, že ho možno zdieľať medzi rôznorodými systémami. Napríklad, použitím XML schém, môžu byť údaje generované aplikáciou Windows Forms, spustenou na počítači so systémom Windows, sprístupnené v aplikácii Java, spustenej na boxe Unix.

7. **Vytváranie špecializovaných XML slovníkov -** Použitím XML ako základu je možné vytvárať svoje vlastné slovníky. Wireless Application Protocol (WAP), Wireless Markup Language (WML) a Simple Object Access Protocol (SOAP) sú niektoré príklady špecializovaných slovníkov XML.[12]

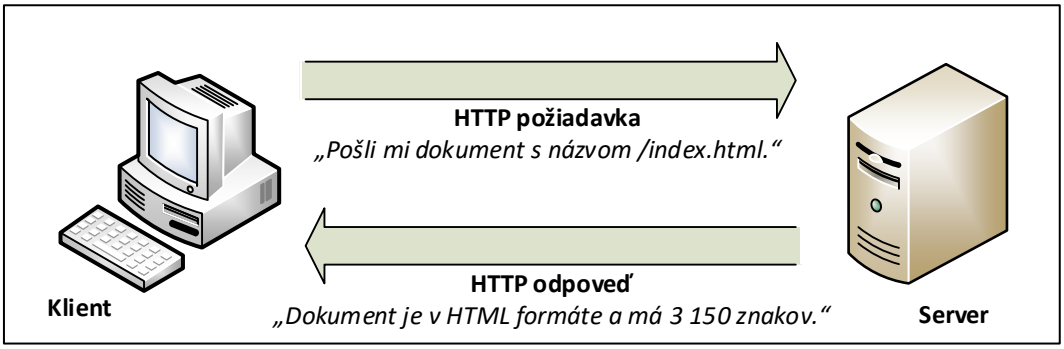
1.4 HTTP

HTTP presúva obrázky JPEG, stránky HTML, textové dokumenty, filmy MPEG, zvukové súbory WAV, Java aplety rýchlo, pohodlne a spoľahlivo z webových serverov po celom svete do webových prehliadačov na pracovných počítačoch ľudí. Pretože HTTP používa spoľahlivé protokoly na prenos údajov, zaručuje, že údaje sa pri prenose nepoškodia ani nezmenia, aj keď pochádzajú z opačnej strany zemegule. To je dobré pre používateľa, pretože má prístup k informáciám bez obáv o ich integritu. Spoľahlivý prenos je dobrý aj pre vývojára internetových aplikácií, pretože sa nemusí obávať zničenia, duplikácie alebo skreslenia komunikácie HTTP pri prenose. Môže sa sústrediť na programovanie charakteristických detailov aplikácie bez obáv z nedostatkov a slabín internetu.[7]

1.4.1 Webové servery a klienti

Webový obsah sa nachádza na webových serveroch. Webové servery používajú protokol HTTP, preto sa často nazývajú servery HTTP. Tieto servery HTTP ukladajú údaje z internetu a poskytujú údaje, keď si ich vyžiadajú klienti HTTP. Klienti posielajú HTTP požiadavky na servery a servery vracajú požadované dáta v HTTP odpovediach, ako je načrtnuté na obrázku 5. HTTP klienti a HTTP servery spolu tvoria základné komponenty World Wide Web.

Najbežnejším klientom je webový prehliadač, napríklad Microsoft Edge alebo Google Chrome. Webové prehliadače požadujú objekty HTTP zo serverov a zobrazujú ich na obrazovke. Keď používateľ prejde na stránku, akou je napríklad vzorová stránka „http://www.prikladwebstranky.com/index.html“, prehliadač odošle požiadavku HTTP na server www.prikladwebstranky.com (pozri obrázok 5). Server sa pokúsi nájsť požadovaný objekt (v tomto prípade „/index.html“) a v prípade úspechu odošle objekt klientovi v odpovedi HTTP spolu s typom objektu, dĺžkou objektu, a ďalšie informácie.[7]



Obrázok 5 – HTTP Klient/Server (Zdroj: vlastné spracovanie podľa Gourtley, et. al.)

1.4.2 Aplikačné rozhrania a webové služby

Pri webových aplikáciách, ktoré poskytujú stále viac typov služieb, je jasné, že HTTP môže byť súčasťou základu pre prepojenie aplikácií. Jedným zo zložitejších problémov pri zapájaní aplikácií je určenie protokolového rozhrania medzi týmito dvoma aplikáciami, aby si mohli vymieňať údaje.

Aby aplikácie mohli spolupracovať, zvyčajne si potrebujú navzájom vymieňať komplexnejšie informácie, než aké je možné vyjadriť v hlavičkách HTTP. Internetová komunita vyvinula súbor štandardov a protokolov, ktoré umožňujú webovým aplikáciám komunikovať medzi sebou. Tieto štandardy sa voľne označujú ako webové služby. Predpoklad webových služieb nie je nový, ale ide o nový mechanizmus pre aplikácie na zdieľanie informácií. Webové služby sú postavené na štandardných webových technológiách, ako je HTTP.

Webové služby si vymieňajú informácie pomocou XML cez SOAP. Jazyk XML (Extensible Markup Language) poskytuje spôsob, ako vytvoriť a interpretovať prispôsobené informácie o dátovom objekte. Simple Object Access Protocol (SOAP) je štandard na pridávanie informácií XML do správ HTTP.[7]

1.5 SOAP

SOAP oproti REST je starší protokol, ktorý bol vyvinutý ako alternatíva k štandardu CORBA (Common Object Request Broker Architecture). Na zabezpečenie prenosu dát v SOAP sa používajú protokoly ako HTTP, SMTP a pod., pričom dáta sú odosielané vo formáte XML. Hlavný princíp tohto prístupu je nasledovný: poskytovateľ služby zverejní popis služby alebo rozhranie do registra služieb, takže žiadateľ o službu môže nájsť správnu inštanciu služby a použiť ju.

Množstvo údajov odoslaných protokolom SOAP môže spôsobiť určité problémy s výkonom, pretože pri vytváraní správy SOAP pridáva do správy ďalšie časti hlavičky a tela. Webové služby založené na SOAP zahŕňajú rôzne štandardy, ako napríklad WSDL, WSBPEL, WS-Security, WS-Addressing (zodpovedné za webovú službu a adresovanie správ). Tieto štandardy boli vyvinuté normalizačnými organizáciami, ako sú W3C a OASIS.[25]

SOAP je komunikačný protokol pre webové služby. Definuje formát požiadaviek, odpovedí a chybových správ webových služieb. Služba pre integrovanie dát môže spracovávať správy SOAP 1.1 a SOAP 1.2 s dokumentovým/konkrétnym kódovaním.

SOAP správa obsahuje tieto sekcie:

1. **SOAP obálka (envelope)** - Obálka definuje rámec správy, obsah správy a to, kto by mal správu spracovať.
2. **SOAP hlavička (header)** - Hlavička identifikuje entitu, ktorá odoslala SOAP správu. Zahŕňa autentifikačné informácie a informácie o tom, ako spracovať SOAP správu.
3. **SOAP telo (body)** - Telo je kontajner pre údaje, ktoré klient a poskytovateľ webovej služby si navzájom posielajú.

SOAP správy sú v XML formáte. Ak obsahuje SOAP správa viacnásobne sa vyskytujúce prvky, skupiny prvkov tvoria úrovne v XML hierarchii. Skupiny sú vzťahované, keď sa jedna úroveň vkladá do druhej. SOAP požiadavková správa môže obsahovať hierarchické údaje. Napríklad, klient posiela požiadavku na pridanie objednávok zákazníkov do predajnej databázy. Klient v SOAP požiadavke prenáša dve skupiny údajov. Jedna skupina obsahuje

ID a meno zákazníka, druhá skupina obsahuje informácie o objednávke, ktoré sa vyskytujú viackrát.

SOAP odpovedajúca správa môže obsahovať hierarchické údaje. Napríklad, klient webovej služby generuje SOAP požiadavku na získanie objednávok zákazníkov. Webová služba vráti hlavičku objednávky a viackrát sa vyskytujúce prvky detailov objednávok v SOAP odpovedi.[10]

1.5.1 Vývoj webových služieb s protokolom SOAP

Webové služby s protokolom SOAP umožňujú vykonávať operácie, ktoré môže použiť webový klient. Tento klient môže byť externým používateľom alebo spotrebiteľom tejto služby. Napríklad používateľ môže použiť túto službu na zobrazenie podrobností o zákazníkovi na základe mena alebo identifikačného čísla zákazníka.[10]

Postup pre vytvorenie webovej služby:

1. Vytvorenie objektu dát WSDL z definície súboru WSDL, ktorý obsahuje definíciu vstupu, výstupu a chýb operácie.
2. Manuálne nastavenie vstupu, výstupu a chýb operácie. Použitie schémy na definovanie komponentov operácie. Použitie znovu použiteľných mappletov, transformácií a logických objektov dát na definovanie prvkov vstupu a výstupu operácie.
3. Nastavenie mapovania operácií medzi SOAP správami a portami vstupnej a výstupnej transformácie. Otestovanie každého mapovania operácie.
4. Nasadenie webovej služby do Data Integration Service a pridanie do aplikácie. V prípade nasadenia aplikácie, ktorá obsahuje webovú službu, ktorá už beží na Data Integration Service, Data Integration Service pripojí číslo k názvu služby.
5. Nastavenie vlastností a zabezpečenia webovej služby v nástroji Administrátor.

Webový klient používa obsah WSDL na pripojenie k webovej službe a môže byť externým používateľom alebo spotrebiteľom služby. Transformácia spotrebiteľa webovej služby môže byť použitá na pripojenie k službe pomocou objektu pripojenia webovej

služby.[10]

1.5.2 Príklad SOAP správy

Na obrázku 6 sa nachádza príklad schémy správy SOAP na lepšie predstavenie tejto tematiky. Ako sme už v kapitole 1.3 spomínali, SOAP správa musí obsahovať obálku, hlavičku a telo správy.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
 ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

Obrázok 6 - Príklad SOAP správy (Zdroj: https://www.w3schools.com/xml/xml_soap.asp)

1.6 REST

REST je architektúra založená na zdrojoch. K prostriedku sa pristupuje cez spoločné rozhranie založené na štandardných metódach HTTP. REST žiada vývojárov, aby používali metódy HTTP explicitne a spôsobom, ktorý je v súlade s definíciou protokolu. Každý zdroj je identifikovaný adresou URL. Každý zdroj by mal podporovať bežné operácie HTTP a REST umožňuje, aby tento zdroj mal rôzne reprezentácie, napr. text, xml, json atď. Zvyšný

klient môže požiadať o špecifické zastúpenie prostredníctvom protokolu HTTP (Content Negotiation).[22]

REST je novší prístup, ktorý využíva na prenos dát HTTP protokol, pričom dáta sú tvorené formátmi XML, JSON a pod. Zjednodušuje prístup k webovým službám využívaním existujúcich a dobre známych štandardov namiesto pridávania nových vrstiev spracovania dát na prenosový a komunikačný zásobník. REST má teda tendenciu byť ľahšou alternatívou k ťažkému protokolu SOAP. Webové služby REST sú založené na samo definujúcich zdrojoch, kde sa na ich dosiahnutie používa protokol HTTP. Služba je poskytovaná ako zdroj, ktorý možno identifikovať pomocou URI (Uniform Resource Identifier). Napríklad, ak ide o prehliadač URI, webová služba vráti informácie o používateľovi, ktorého ID je „1“. Na vykonávanie dátových operácií so službou sa používajú štandardné HTTP metódy ako **GET**, **PUT**, **POST**, **DELETE** atď.[25]

- **GET** = „dajte mi nejaké informácie“ (Získať)
- **POST** = „tu sú nejaké informácie o aktualizácii“ (Aktualizovať)
- **PUT** = „tu sú nejaké nové informácie“ (Vytvoriť)
- **DELETE** = „vymazať nejaké informácie“ (Odstrániť)
- **PATCH** = HTTP metódu PATCH možno použiť na aktualizáciu čiastočných zdrojov. Napríklad, keď sa potrebuje aktualizovať iba jedno pole zdroja, použitie PUT celého zdroja môže byť ťažkopádne a využíva väčšiu šírku pásma.
- **HEAD** = Metóda HEAD je identická s metódou GET, okrem toho, že server nesmie v odpovedi vrátiť telo správy. Táto metóda sa často používa na testovanie platnosti, dostupnosti a nedávnych úprav hypertextových odkazov.
- **OPTIONS** = Táto metóda umožňuje klientovi určiť možnosti a/alebo požiadavky spojené so zdrojom alebo schopnosťami servera bez toho, aby to znamenalo akciu zdroja alebo iniciovanie získavania prostriedkov.

Pojem „Idempotencia“ – myšlienka, že pri odoslaní príkazu **GET**, **DELETE** alebo **PUT** do systému by mal byť efekt rovnaký, či už je príkaz odoslaný raz alebo viackrát, ale **POST** vytvorí entitu v kolekcii, a preto nie je idempotentný.[22]

1.6.1 Príklad REST

Ako príklad webovej služby REST uvádzame odpoveď pre REST webovú službu vo formáte XML. **GET** (Získaj) ID určitého zákazníka.

```
<Customer>
        <id>123</id>
        <name>John</name>
</Customer>
```

Obrázok 7 - Príklad REST odpovede v XML
(Zdroj: vlastné spracovanie)

Taktiež uvádzame príklad JSON odpovede pre REST. **GET** (Získaj) ID určitého zákazníka: *{"Customer":{"id":"123","name":"John"}}*[22]

1.7 WSDL

WSDL je formát XML, ktorý poskytuje rozšíriteľný model na popis rozhraní webových služieb a spôsobu prístupu k nim. Spočiatku bol WSDL vyvinutý spoločnosťami Microsoft a IBM. Strojovo čitateľný súbor WSDL v podstate popisuje, aké služby sú dostupné, kde sa nachádzajú a ako ich možno vyvolať.[26]

Najnovšia špecifikácia WSDL je verzia 2.0. V čase písania tohto článku však verzia 2.0 nezískala široké uplatnenie v priemysle. Väčšina podpory nástrojov je navyše založená na verzii 1.1. WSDL 1.1 nebol schválený konzorciom World Wide Web Consortium (W3C), avšak W3C vydal návrh verzie 2.0, ktorý bude oficiálnym štandardom, a teda bude schválený W3C.[26, 2]

Nová verzia WSDL nezmenila základný koncept, pretože sa vykonali len drobné opravy, napríklad premenovanie niektorých výrazov (napr. <*portType*> premenované na <*interface*>, <*port*> premenované na <*endpoint*>) alebo odstránenie definícií <*message*> ktoré sú teraz definované v elemente <*types*>. Obrázok 7 zobrazuje štruktúru popisu WSDL.[2]

```
<definitions>
<types>
      definícia typov……
</types>

<message>
     definícia správy……
</message>

<portType>
    <operations>
         definícia operácií……
    </operations>
</portType

<binding>
    definícia viazania……
</binding>

<service>
     definícia služby……
</service>
</definitions>
```

Obrázok 8 - Štruktúra popisu WSDL (Zdroj: vlastné spracovanie podľa Bravo, Pascual a Rodríguez)

Koreňom každého súboru WSDL je prvok <*definitions*>, ktorý zapuzdruje celý dokument a poskytuje niekoľko definícií menného priestoru. Ako už bolo uvedené, je možné oddeliť abstraktné a konkrétne opisy. Prvý element abstraktného popisu je <*types*>, kde je možné definovať dátové typy vymieňaných správ medzi klientom a serverom. Ako už bolo navrhnuté, preferovanou voľbou by mala byť schéma XML.

Ďalším prvkom je <*message*> poskytujúca abstraktnú informáciu o vstupe, výstupe alebo chybe operácie vo forme jednej alebo viacerých logických častí. Posledným abstraktným prvkom je <*portType*>, ktorý definuje, čo webová služba robí. Popisuje vykonateľné operácie a príslušné vstupné a výstupné správy.

Prvým konkrétnym prvkom, o ktorom sa bude diskutovať, je <*binding*>. Spája definície <*portType*> so špecifikáciou formátu údajov a konkrétnym transportným protokolom. Ďalej obsahuje prvok <*operácie*> pre každú operáciu v <*portType*>, ktorý popisuje. Väzby môžu byť ponúkané prostredníctvom viacerých protokolov, napríklad cez

Hypertext Transfer Protocol (HTTP) (s metódou GET alebo POST) alebo SOAP. Nakoniec prvok <*service*> definuje, kde sa má k službe pristupovať. Obsahuje jeden alebo viac podporovaných portov, pričom každý port predstavuje prístupový bod. Port špecifikuje jedinečnú adresu a je spojená s jeho väzbou. V skutočnosti je kombinácia WSDL s už spomínaným protokolom SOAP v praxi veľmi obľúbeným spôsobom budovania webových služieb. Treba však poznamenať, že WSDL je nezávislý od protokolu.[2]

WSDL znamená jazyk popisu webových služieb. Je to štandardný formát na popis webovej služby. WSDL bol vyvinutý spoločne spoločnosťami Microsoft a IBM. WSDL sa často používa v kombinácii so SOAP a XML Schema na poskytovanie webových služieb cez internet. Klientsky program, ktorý sa pripája k webovej službe, dokáže prečítať WSDL, aby určil, aké funkcie sú dostupné na serveri. Akékoľvek použité špeciálne dátové typy sú vložené do súboru WSDL vo forme XML Schema. Klient potom môže použiť SOAP na skutočné volanie jednej z funkcií uvedených vo WSDL.[26]

Vlastnosti WSDL:

- WSDL je protokol založený na XML na výmenu informácií v decentralizovaných a distribuovaných prostrediach.
- Definície WSDL popisujú, ako pristupovať k webovej službe a aké operácie bude vykonávať.
- WSDL je jazyk na popis toho, ako vytvoriť rozhranie so službami založenými na XML.
- WSDL je integrálnou súčasťou Universal Description, Discovery, and Integration (UDDI), celosvetového obchodného registra založeného na XML.
- WSDL je jazyk, ktorý používa UDDI.[26]

1.8 Pojmy spojené s tvorbou webového klienta

V kapitole 1.8 sme sa zamerali na vysvetlenie pojmov, ktoré sa využívajú pri tvorbe webového klienta. Dané pojmy uľahčia pochopenie tejto tematiky a poskytnú náhľad do jednotlivých častí spojených s tvorbou webového klienta. Bližšie si vysvetlíme pojmy ako HTML, CSS a AJAX.

1.8.1 HTML

HTML sa ako technologický štandard neustále vyvíja už od svojho uvedenia v roku 1993 Timom Berners-Lee, pôvodným „webovým vývojárom“. konzorcium World Wide Web (W3C). Spoločnosti, ktoré vytvárajú webové prehliadače, preberajú špecifikácie z W3C a implementujú správanie, ktoré sa očakáva, keď prehliadač narazí na akékoľvek povolené formátovanie, ako je napríklad zvýraznenie textu alebo zmena jeho farby (alebo dokonca oboje súčasne).

Ako naznačuje názov HyperText Markup Language, HTML je značkovací jazyk, nie programovací jazyk. HTML umožňuje autorovi webu organizovať a definovať, ako by sa mal obsah zobrazovať, čo znamená, že môže robiť veci, ako je pridávanie formátovania textu; vytvárať nadpisy, zoznamy a tabuľky; a zahŕňať obrázky a odkazy. Súbor HTML je možné predstaviť si ako obyčajný písaný dokument, ktorý autor starostlivo anotoval. Niektoré zo značiek môžu zvýrazniť časti textu, niektoré môžu obsahovať obrázok, ktorý bol pripojený k dokumentu, a iné môžu povedať, kde je možné nájsť ďalšie informácie.

Časť skratky HTML „Hypertext“ označuje spôsob, akým odkazy na webe umožňujú nelineárnym spôsobom prechádzať z jedného dokumentu do druhého. Ak napríklad používateľ číta článok na Wikipédii o HTML a vidí zvýraznený odkaz na súvisiacu tému, ako je CSS, môže kliknúť na tento odkaz a okamžite prejsť na ďalší článok.

Technologicky je *hypertext* veľkým vylepšením oproti neprepojeným dokumentom, pretože eliminuje potrebu listovať alebo posúvať stránky obsahu, aby ste používateľ našiel to, čo hľadá. V súčasnosti je schopnosť spájať dokumenty niečo, čo všetci považujú za samozrejmosť, no keď bola vytvorená špecifikácia HTML, bola to inovácia dostatočne dôležitá na to, aby bola zahrnutá do názvu technológie.

Zdroj HTML je obyčajný text, vďaka čomu je ideálny na úpravy pomocou textového editora. Namiesto použitia pohodlného, ale neflexibilného prístupu textových procesorov WYSIWYG (What You See Is What You Get), HTML označuje formátovanie pomocou špeciálnych značiek, inak povedané textových anotácií. Na obrázku 9 je zobrazený príklad HTML dokumentu.[8]

```html
<!DOCTYPE html>
<html>
<head>
    <title>Názov stránky</title>
    <meta charset="utf-8">
</head>
<body>
</body>
</html>
```

Obrázok 9 - Príklad HTML dokumentu (Zdroj: vlastné spracovanie)

1.8.2 CSS

Cascading Style Sheets (CSS) je jazykom, ktorý sa používa na definovanie vzhľadu a formátovania dokumentu napísaného v jazyku HTML alebo XML. CSS umožňuje oddeliť vizuálne aspekty stránky od jej štruktúry a obsahu. Týmto spôsobom umožňuje tvorcom webu určovať vizuálne vlastnosti, ako sú farby, fonty, veľkosti písma, rozloženie, okraje a pozadia, bez toho, aby museli zasahovať do kódu HTML. Umožňuje vytvárať zložité dizajny súčasne s dodržaním správnej štruktúry a sémantiky dokumentu. CSS definuje súbor pravidiel, ktoré sa aplikujú na rôzne časti dokumentu. Tieto pravidlá sa zvyčajne zapisujú v samostatnom súbore s príponou *.css*, ktorý sa potom odkazuje v kóde HTML pomocou značky <*link*>. Tento spôsob umožňuje jednoduché zmeny vzhľadu stránky, pretože stačí upraviť len jeden súbor CSS namiesto mnohých stránok HTML.

CSS umožňuje tvorcom webu používať selektory, ktoré sa vzťahujú na konkrétne časti HTML kódu a určujú, ktoré pravidlá CSS sa na ne majú aplikovať. Selektory môžu byť založené na názve elementu, triede, identifikátore, stave alebo pozícii v strome dokumentu. Kombinovaním rôznych selektorov a pravidiel CSS môžu tvorcovia webu vytvárať rôzne efekty, ako napríklad *hover* efekty, animácie, gradienty, pruhy, tieňovanie, zaoblené hrany, mriežkové rozloženie a mnoho ďalších. S pomocou CSS je možné vytvárať aj responzívne dizajny, ktoré sa prispôsobujú rôznym zariadeniam a rozlíšeniam obrazovky.

V súčasnosti existuje niekoľko verzií CSS, pričom najnovšou je CSS3. Táto verzia prináša množstvo nových funkcií a vylepšení, ktoré umožňujú tvorcom webu ešte viac prispôsobovať vzhľad a správanie stránok.[13]

```css
.table {
    margin-bottom: 20px;
    background-color: #FFF;
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.15);
}
```

Obrázok 10 - Príklad CSS dokumentu (Zdroj: vlastné spracovanie)

1.8.3 AJAX

Asynchronous JavaScript and XML (Ajax) označuje skupinu technológií, ktoré sa používajú na vývoj webových aplikácií. Vďaka kombinácii týchto technológií sa webové stránky javia ako pohotovejšie, pretože so serverom sa vymieňajú malé balíky údajov a webové stránky sa nenačítavajú zakaždým, keď používateľ vykoná zmenu vstupu. Štruktúra aplikácie Ajax je postavená na základnej štruktúre XHTML, ktorá bola pôvodne iba rozšírením HTML.

Nástroje používané na vytváranie webových aplikácií Ajax:

- Extensible HyperText Markup Language (**XHTML**)
- Document Object Model (**DOM**)
- JavaScript
- Cascading Style Sheets (**CSS**)
- Extensible Markup Language (**XML**)
- JavaScript Object Notation (**JSON**)

Do vytvárania aplikácie Ajax vchádzajú aj iné veci, ako napríklad transformácia jazyka Extensible Stylesheet Language Transformation (XSLT), informačné kanály s RSS a Atom, nejaký druh skriptovania na strane servera a prípadne databázu alebo XML dokument. XHTML obsahuje všetko, čo sa zobrazí v prehliadači klienta, a všetko ostatné funguje priamo z neho. DOM sa používa na navigáciu v celom XHTML dokumente na stránke. JavaScript má najdôležitejšiu úlohu v aplikácii Ajax. Používa sa na manipuláciu s DOM pre stránku, ale čo je dôležitejšie, JavaScript vytvára všetku komunikáciu medzi klientom a serverom. CSS sa používa na ovplyvnenie vzhľadu stránky a dynamicky sa s ním manipuluje prostredníctvom modelu DOM. Nakoniec, XML je protokol, ktorý sa používa na prenos údajov tam a späť medzi klientmi a servermi. Aplikácie AJAX môžu na prenos údajov používať XML, ale rovnako bežné je prenášať údaje ako obyčajný text alebo text JSON. AJAX umožňuje asynchrónnu aktualizáciu webových stránok výmenou údajov s webovým serverom v pozadí. To znamená, že je možné aktualizovať časti webovej stránky bez opätovného načítania celej stránky.[9]

2 Cieľ, metodika práce a metódy skúmania

2.1 Cieľ práce

Pred spracovaním teoretickej časti bolo potrebné stanoviť si ciele, ktoré nám pomohli so spracovaním našej diplomovej práce. Následne sme sa naše predom určené ciele snažili naplniť. Ciele teoretickej časti našej diplomovej práce boli nasledovné:

- Definícia webových služieb,
- vysvetlenie použitých pojmov,
- uvedenie príkladov použitých technológií,
- analyzovanie technológií použitých pri tvorbe klienta.

Stanovené ciele nám pomohli s pochopením jednotlivých pojmov spojených s webovými službami a samotným vývojom webovej služby a jej webového klienta.

Hlavným cieľom praktickej časti našej diplomovej práce bolo analyzovať možnosti použitia XML webových služieb na sledovanie parametrov klientov domova sociálnych služieb vo vybratom období v elektronickom informačnom systéme a porovnať ich použitie s doterajším spôsobom sledovania týchto parametrov vo vybraných domovoch sociálnych služieb. Najdôležitejším cieľom bolo vyvinutie ASP .NET XML webovej služby v riadenom programovacom jazyku, ktorá poskytovala klientom usporiadané informácie o parametroch klientov domova sociálnych služieb v sledovanom období, podľa vybraných kritérií. Medzi sledované parametre klientov mali patriť:

- Celkový počet klientov v danom domove na začiatku a konci sledovaného obdobia,
- počty a zoznamy klientov s ich základnými dátami v jednotlivých kategóriách imobility,
- jednotlivé stanovené diagnózy a vek,
- a ďalšie parametre.

Posledným cieľom bola analýza a porovnanie použitia nami vyvinutej webovej služby a jej klienta s existujúcim riešením, ktoré sa používa v domovoch sociálnych služieb. Všetky stanovené ciele teoretickej, no aj praktickej časti diplomovej práce sme naplnili.

2.2 Metodika práce a metódy skúmania

Pred spracovaním teoretickej časti diplomovej práce sme využili literatúru prevažne od zahraničných autorov a to najmä v knižnej, no taktiež elektronickej forme. Čerpali sme z rôznych odborných kníh a akademických žurnálov, no v niektorých prípadoch z overených technických web stránok.

Pri vývoji webovej služby a jej klienta sme využívali Visual Studio 2019. Visual Studio je integrované vývojové prostredie (IDE) vyvinuté spoločnosťou Microsoft, ktoré sa používa na vytváranie aplikácií pre Windows, webových aplikácií a mobilných aplikácií. Poskytuje širokú škálu funkcií a nástrojov na uľahčenie a zefektívnenie vývoja .

Podľa „Professional Visual Studio 2019“ od Bruce Johnsona Visual Studio obsahuje množstvo funkcií, ako je *IntelliSense*, ktorá pomáha vývojárom písať kód rýchlejšie a s menším počtom chýb tým, že poskytuje návrhy na dokončenie kódových príkazov; nástroje na ladenie, ktoré umožňujú vývojárom rýchlo identifikovať a opraviť chyby v kóde; a celý rad projektových šablón, ktoré poskytujú východiskový bod pre vývoj aplikácií. Okrem toho Visual Studio podporuje viacero programovacích jazykov vrátane C++, C# a Visual Basic a umožňuje vývojárom písať kód pre desktopové, webové a mobilné aplikácie.[11]

Na vývoj našej ASP .NET webovej služby sme používali programovací jazyk C#. Podľa „C# 9.0 in a Nutshell: The Definitive Reference“ od Joseph Albahariho a Eric Johannsena je C# moderný, objektovo orientovaný programovací jazyk určený na vytváranie rôznych aplikácií, ktoré bežia na .NET Frameworku. Bol vyvinutý spoločnosťou Microsoft a prvýkrát vydaný v roku 2002 ako súčasť .NET Framework. C# je silne typizovaný jazyk, ktorý je skompilovaný do jazyka CIL, ktorý môže spustiť .NET runtime. Má funkcie, ako je automatická správa pamäte a schopnosť vytvárať a využívať knižnice kódov.[1]

Pri dizajnovaní webového klienta našej webovej služby sme si dopomohli nástrojom Bootstrap 4, ktorý poskytuje predpripravené CSS triedy a JavaScript komponenty, umožňujúce rýchle a efektívne vytváranie responzívneho a moderného dizajnu webovej stránky. Bootstrap 4 je bezplatný a otvorený zdroj CSS, ktorý sa používa na vývoj responzívnych a mobilných webových stránok. Zahŕňa komponenty HTML, CSS a JavaScript na vytváranie navigačných panelov, formulárov, tlačidiel, typografie a ďalších prvkov rozhrania. Bootstrap 4 tiež poskytuje mriežkový systém, ktorý umožňuje jednoduché rozloženie a zarovnanie obsahu.[5]

3 Výsledky práce

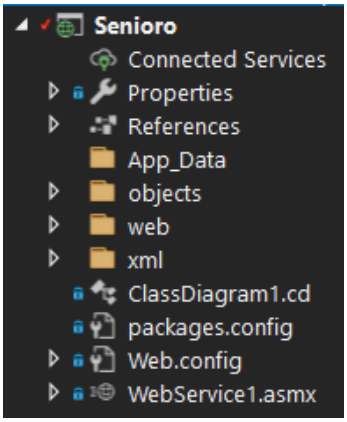
Výsledkom našej diplomovej práce bolo vytvorenie funkčnej ASP .NET XML webovej služby, ktorá poskytuje usporiadané informácie o klientoch domova sociálnych služieb a taktiež poskytované usporiadané informácie ukladá do XML súborov. Daná webová služba obsahuje viacero metód na zbieranie, vyhodnocovanie, usporiadanie a filtrovanie údajov a následne sú tieto webové služby implementované do webového klienta.

3.1 Štruktúra projektu

Štruktúru nami vyvinutého projektu, ktorý nazývame Senioro môžeme rozdeliť na štruktúru projektového adresára, adresára vstupných a výstupných XML adresárov a na štruktúru webového klienta.

3.1.1 Štruktúra projektového adresára

Projektový adresár sa skladá z viacerých dôležitých častí. Adresár „App_Data“ obsahuje výstupné údaje, v priečinku „objects“ sa nachádza trieda na filtrovanie vyhľadávaných klientov, adresár „web“ obsahuje údaje, ktoré slúžia na vytvorenie webového klienta a v priečinku „xml“ sa nachádzajú vstupné údaje o klientoch. Okrem adresárov sa v koreni projektového adresára nachádza dokument diagramu tried „ClassDiagram1.cd“ a webová služba ASP .NET „WebService1.asmx“. Štruktúra projektového adresára je zobrazená na obrázku 11.

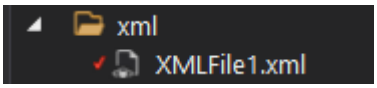


Obrázok 11 - Štruktúra projektového adresára (Zdroj: vlastné spracovanie)

3.1.2 Štruktúra adresárov vstupných a výstupných údajov

Vstupné údaje

Adresár vstupných údajov pozostáva z dokumentu „XMLFile1.xml“, ktorý, ako je možné vidieť na obrázku 12, obsahuje údaje o klientoch a to presnejšie, identifikačné číslo klienta, jeho meno a priezvisko, dátum narodenia, telefónne číslo, pohlavie, výšku platby za mesiac, výšku dôchodku, číslo izby, v ktorej sa daný klient nachádza, jednotlivé diagnózy klienta, či je klient imobilný alebo nie, výšku odkázanosti klienta, dátum príchodu, dátum prípadného odchodu zo zariadenia, údaje o kontaktnej osobe a vek klienta, ktorý sa automaticky vypočítava a prepisuje pri spúšťaní jednotlivých metód.

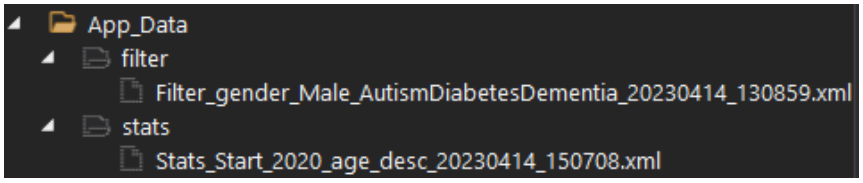



```
<Client>
<record>
    <id>6</id>
    <firstName>Geoffrey</firstName>
    <lastName>Featherstonhalgh</lastName>
    <dateOfBirth>13.02.1955</dateOfBirth>
    <phoneNumber>+86 149 325 8393</phoneNumber>
    <gender>Female</gender>
    <monthPayment>275</monthPayment>
    <pension>440</pension>
    <roomNo>106</roomNo>
    <diagnose1>Alzheimer's Disease</diagnose1>
    <diagnose2>Parkinson's Disease</diagnose2>
    <diagnose3>Dementia</diagnose3>
    <immobility>Yes</immobility>
    <dependency>Medium</dependency>
    <dateOfArrival>07.04.2006</dateOfArrival>
    <dateOfDeparture>27.11.2016</dateOfDeparture>
    <contactPerson>
      <name>Gordan</name>
      <lastName>Callaghan</lastName>
      <phoneNumber>+86 967 335 7745</phoneNumber>
    </contactPerson>
    <age>68</age>
  </record>

...

</Client>
```

Obrázok 12 – Štruktúra adresára vstupných údajov (Zdroj: Vlastné spracovanie)

Výstupné údaje

Výstupné údaje sa delia do dvoch adresárov a to do adresára „filter“ a adresára „stats“. Túto skutočnosť je možné vidieť na obrázku 13. V priečinku „filter“ sa nachádzajú vyfiltrované údaje o klientoch spolu s časovou pečiatkou, ktorá prislúcha dátumu a času, kedy bolo toto vyhľadávanie klientov vykonané. Priečinok „stats“ obsahuje uložené údaje o ročných štatistikách na začiatku alebo konci používateľom zadaného roka.



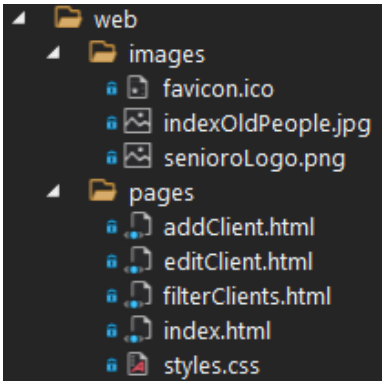
Obrázok 13 – Štruktúra adresárov výstupných údajov (Zdroj: Vlastné spracovanie)

Na rozdiel od štruktúry predošlého XML dokumentu, vyfiltrované XML dokumenty obsahujú len záznamy, ktoré spĺňajú používateľom zadané požiadavky a sú zoradené podľa používateľsky daného poradia. Taktiež sa lýšia pridaním koreňového elementu *<Result>* pred element *<Client>*.

3.1.3 Štruktúra projektu webového klienta

.

Štruktúra HTML klienta, ktorá je zobrazená na obrázku 14 pozostáva z dvoch adresárov, ktoré obsahujú dokumenty súvisiace s funkčnosťou klienta. Adresár „images“ obsahuje obrazové stopy, ktoré sú využívané najmä na indexovej html stránke. Ďalej pozostáva z priečinku „pages“, ktorý obsahuje zdrojové html súbory webového klienta a súbor na určovanie vizuálnych prvkov „styles.css“.

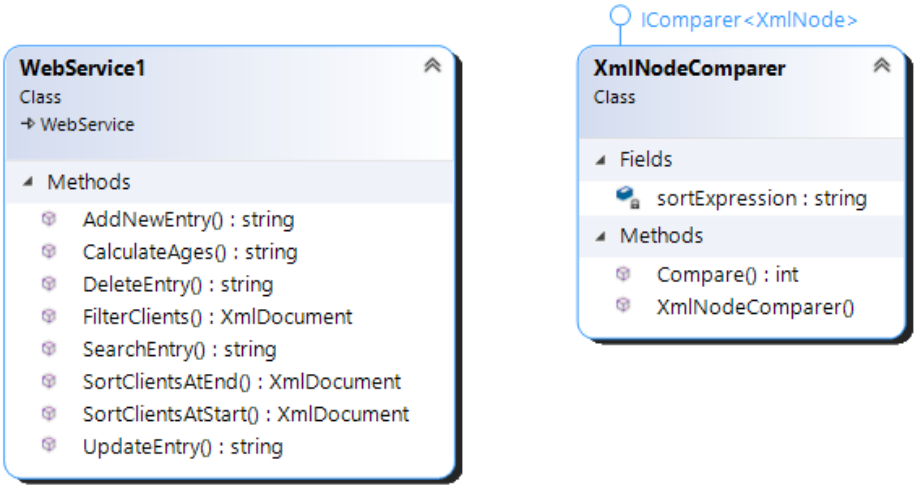


Obrázok 14 – Štruktúra projektu webového klienta (Zdroj: vlastné spracovanie)

3.1.4 Diagram tried

V diagrame tried na obrázku 15 môžeme vidieť, že sme využili 2 triedy, z ktorých jedna je trieda Webovej služby *WebService1* a druhá je trieda *XmlNodeComparer*, ktorá implementuje rozhranie *IComparer* a používa sa na porovnávanie dvoch *XmlNode* objektov v rámci zoradenia zoznamu uzlov XML dokumentu podľa určitého používateľom zadaného výrazu. Trieda *IComparer* má konštruktor, ktorý prijíma reťazec *sortExpression* a ten určuje vlastnosť, podľa ktorej sa budú zoraďovať dokumenty a ich poradie (vzostupne alebo zostupne). Následne sa v metóde *Compare* porovnávajú hodnoty vlastností uzlov *XmlNode* objektov.

Ak má parameter *sortExpression* hodnotu *asc*, uzly s menšou hodnotou vybraného atribútu budú umiestnené pred uzly s väčšou hodnotou vybraného atribútu. Ak má parameter *sortExpression* hodnotu *desc*, uzly s väčšou hodnotou daného atribútu budú umiestnené pred uzly s menšou hodnotou atribútu. Triede *WebService1* a jej metódam sa budeme podrobnejšie venovať v nasledujúcich kapitolách.



Obrázok 15 - Diagram tried (Zdroj: vlastné spracovanie)

3.2 Použité knižnice

```
using System;
using System.Collections.Generic;
using System.Globalization;
using System.IO;
using System.Web;
using System.Web.Services;
using System.Xml;
using System.Xml.Linq;
```

Obrázok 16 - Použité knižnice (Zdroj: vlastné spracovanie)

- **System** - základná knižnica, ktorá obsahuje triedy a funkcie potrebné pre vývoj aplikácií v jazyku C#.[14]
- **System.Collections.Generic** - obsahuje triedy a rozhrania pre prácu s generickými kolekciami, napr. List, Dictionary a HashSet.[15]
- **System.Globalization** - obsahuje triedy a funkcie pre prácu s miestnymi zvyklosťami, napr. formátovanie dátumu a času, konverziu reťazcov a čísel.[16]
- **System.IO** - obsahuje triedy a funkcie pre prácu s vstupno-výstupnými operáciami, napr. čítanie a zapisovanie súborov, priečinkov a prác s dátovými prúdmi.[17]
- **System.Web** - obsahuje triedy a funkcie pre vývoj webových aplikácií, napr. triedy pre spracovanie požiadaviek a odpovedí, správu stavu relácie a spracovanie HTTP hlavičiek.[18]
- **System.Web.Services** - obsahuje triedy a funkcie pre vývoj webových služieb, napr. triedy pre vytvorenie a publikovanie webových služieb, spracovanie požiadaviek a odpovedí na webové služby.[19]
- **System.Xml** - obsahuje triedy a funkcie pre prácu s XML dokumentami, napr. triedy pre načítanie, spracovanie a vytvorenie XML dokumentov a uzlov.[20]
- **System.Xml.Linq** - obsahuje triedy a funkcie pre prácu s XML dokumentami pomocou LINQ, napr. triedy pre prácu s XDocument a XElement.[21]

3.3 Kľúčové členské metódy webovej služby

Kľúčovými metódami našej webovej služby *WebService1*, ktoré slúžili na splnenie cieľu našej diplomovej práce, boli metódy *FilterClients, SortClientsAtEnd* a *SortClientsAtStart*. Metóda *FilterClients* slúžila na vyfiltrovanie a zoradenie jednotlivých údajov z načítaného XML dokumentu a následné uloženie vyhľadaných a zoradených klientov.

3.3.1 Webová metóda FilterClients

Metóda *FilterClients* dostáva od používateľa viacero vstupov. Všetky z nich sú nepovinné parametre, keďže používateľ by mal mať možnosť zobraziť všetkých klientov bez použitia filtrovania. Parametre, ktoré metóda dostáva sú nasledovné:

- **filterBy** - reťazec, ktorý udáva podľa akého atribútu majú byť filtrované záznamy v XML súbore. Ak je táto hodnota *null*, záznamy sa nebudú filtrovať.
- **filterValue** - reťazec, ktorý udáva hodnotu, podľa ktorej sa majú záznamy filtrovať. Tento parameter sa použije iba vtedy, ak parameter *filterBy* má nejakú hodnotu.
- **dateOfArrival** - reťazec v tvare "dd.MM.yyyy", ktorý udáva dátum príchodu, podľa ktorého sa majú záznamy filtrovať. Ak je táto hodnota *null*, záznamy sa nebudú filtrovať podľa dátumu príchodu.
- **dateOfDeparture** - reťazec v tvare "dd.MM.yyyy", ktorý udáva dátum odchodu, podľa ktorého sa majú záznamy filtrovať. Ak je táto hodnota *null*, záznamy sa nebudú filtrovať podľa dátumu odchodu.
- **sortBy** - reťazec, ktorý udáva podľa akého atribútu majú byť záznamy výsledku zoradené. Ak je táto hodnota *null*, záznamy sa nebudú triediť.
- **sortOrder** - reťazec, ktorý udáva poradie triedenia (vzostupné alebo zostupné). Ak je táto hodnota *null*, záznamy sa budú triediť vzostupne. Ak má parameter *sortBy* hodnotu *null*, musí byť zadaný aj tento parameter.
- **diagnose1** - reťazec, ktorý udáva prvé diagnózu, podľa ktorej sa majú záznamy filtrovať. Ak je táto hodnota *null*, záznamy sa nebudú filtrovať podľa 1. diagnózy.
- **diagnose2** - reťazec, ktorý udáva druhú diagnózu, podľa ktorej sa majú záznamy filtrovať. Ak je táto hodnota *null*, záznamy sa nebudú filtrovať podľa 2. diagnózy.
- **diagnose3** - reťazec, ktorý udáva tretiu diagnózu, podľa ktorej sa majú záznamy filtrovať. Ak je táto hodnota *null*, záznamy sa nebudú filtrovať podľa 3. diagnózy.

```
[WebMethod]
public XmlDocument FilterClients(string filterBy = null, string filterValue =
null, string dateOfArrival = null, string dateOfDeparture = null, string sortBy
= null, string sortOrder = null, string diagnose1 = null, string diagnose2 =
null, string diagnose3 = null)
{
CalculateAges();
string path = Server.MapPath("~/xml/XMLFile1.xml");
XmlDocument xmlDoc = new XmlDocument();
xmlDoc.Load(path);

XmlElement root = xmlDoc.DocumentElement;
```

Obrázok 17 – Časť kódu webovej metódy FilterClients určujúca vstupné údaje (Zdroj: vlastné spracovanie)

Prvá časť kódu, zobrazená na obrázku 17 má za úlohu načítať XML súbor a získať jeho koreňový element, ktorý reprezentuje celý dokument. Najprv sa zavolá metóda *CalculateAges()*, ktorá sa stará o výpočet veku klientov podľa dátumu narodenia a

aktualizuje ich vekové informácie v XML súbore. Potom sa načíta XML súbor zo servera pomocou metódy *Server.MapPath()*, ktorá určuje fyzickú cestu k súboru na serveri. Následne sa pomocou metódy *XmlDocument.Load()* načíta celý obsah XML súboru do objektu *XmlDocument*. Tento objekt reprezentuje celý dokument, ktorý je možné prehľadávať a upravovať pomocou rôznych metód a vlastností.

Nakoniec sa pomocou vlastnosti *XmlDocument.DocumentElement* získa koreňový element dokumentu, ktorý sa uloží do premennej *root*. Tento koreňový element slúži ako východisko pre ďalšie spracovanie a vyhľadávanie informácií v dokumente.

```
...

XmlNodeList nodes = root.SelectNodes("record");
List<XmlNode> filteredList = new List<XmlNode>();
foreach (XmlNode node in nodes)
{
bool shouldInclude = true;
if (!string.IsNullOrEmpty(filterBy) && !string.IsNullOrEmpty(filterValue))
{
if (node.SelectSingleNode(filterBy).InnerText.Contains(filterValue))
{
shouldInclude = true;
}
        else
        {
        shouldInclude = false;
        }
}
if (!string.IsNullOrEmpty(dateOfArrival))
{
DateTime arrival;
if (DateTime.TryParseExact(node.SelectSingleNode("dateOfArrival").InnerText,
"dd.MM.yyyy", CultureInfo.InvariantCulture, DateTimeStyles.None, out
arrival))
{
shouldInclude &= arrival <= DateTime.ParseExact(dateOfArrival, "dd.MM.yyyy",
CultureInfo.InvariantCulture);
}
       else
       {
       shouldInclude = false;
       }
}
if (!string.IsNullOrEmpty(node.SelectSingleNode("dateOfDeparture").InnerText)
&& !string.IsNullOrEmpty(dateOfDeparture))
{
DateTime departure;
if
(DateTime.TryParseExact(node.SelectSingleNode("dateOfDeparture").InnerText,
"dd.MM.yyyy", CultureInfo.InvariantCulture, DateTimeStyles.None, out
departure))
{
shouldInclude &= departure >= DateTime.ParseExact(dateOfDeparture,
"dd.MM.yyyy", CultureInfo.InvariantCulture);
}
```

```
            else
            {
            shouldInclude = false;
            }
        }
        if (!string.IsNullOrEmpty(diagnose1))
        {
        shouldInclude &=
        node.SelectSingleNode("diagnose1").InnerText.Equals(diagnose1) ||
        node.SelectSingleNode("diagnose2").InnerText.Equals(diagnose1) ||
        node.SelectSingleNode("diagnose3").InnerText.Equals(diagnose1);
        }

        if (!string.IsNullOrEmpty(diagnose2))
        {
        shouldInclude &=
        node.SelectSingleNode("diagnose1").InnerText.Equals(diagnose2) ||
        node.SelectSingleNode("diagnose2").InnerText.Equals(diagnose2) ||
        node.SelectSingleNode("diagnose3").InnerText.Equals(diagnose2);
        }

        if (!string.IsNullOrEmpty(diagnose3))
        {
        shouldInclude &=
        node.SelectSingleNode("diagnose1").InnerText.Equals(diagnose3) ||
        node.SelectSingleNode("diagnose2").InnerText.Equals(diagnose3) ||
        node.SelectSingleNode("diagnose3").InnerText.Equals(diagnose3);
        }

        if (shouldInclude)
        {
        filteredList.Add(node);
        }
        }
```

Obrázok 18 – Časť kódu webovej metódy FilterClients určujúca filtrovacie kritériá (Zdroj: vlastné spracovanie)

V časti kódu na obrázku 18 sa filtruje zoznam záznamov klientov na základe určitých podmienok. Najskôr sa získa zoznam všetkých elementov *record* zo súboru XML. Potom sa pre každý záznam skontroluje, či spĺňa požadované podmienky:

- **Filtrovanie podľa názvu vlastnosti a hodnoty:** Ak sú zadané parametre *filterBy* a *filterValue*, pre každý záznam sa skontroluje, či element s názvom *filterBy* obsahuje hodnotu *filterValue*. Ak áno, tak by mal byť záznam zaradený do výsledného zoznamu. Ak nie, tak sa nezaradí.
- **Filtrovanie podľa dátumu príchodu:** Ak je zadaný parameter *dateOfArrival*, pre každý záznam sa skontroluje, či hodnota elementu *dateOfArrival* je menšia alebo rovná zadanému dátumu. Ak áno, tak by mal byť záznam zaradený do výsledného zoznamu. Ak nie, tak sa nezaradí.
- **Filtrovanie podľa dátumu odchodu:** Ak je zadaný parameter *dateOfDeparture*, pre každý záznam sa skontroluje, či hodnota elementu *dateOfDeparture* je väčšia alebo

rovná zadanému dátumu. Ak áno, tak by mal byť záznam zaradený do výsledného zoznamu. Ak nie, tak sa nezaradí.

- **Filtrovanie podľa diagnózy:** Ak sú zadané parametre *diagnose1*, *diagnose2* alebo *diagnose3*, pre každý záznam sa skontroluje, či aspoň jeden z elementov *diagnose1*, *diagnose2* alebo *diagnose3* obsahuje zadanú hodnotu. Ak áno, tak by mal byť záznam zaradený do výsledného zoznamu. Ak nie, tak sa nezaradí.

Záznamy, ktoré spĺňajú všetky podmienky, sú zaradené do zoznamu *filteredList*, ktorý sa následne vráti ako výsledok metódy.

```
                                ...

        if (!string.IsNullOrEmpty(sortBy) && !string.IsNullOrEmpty(sortOrder))
        {
        string sortExpression = string.Format("{0} {1}", sortBy, sortOrder);
        filteredList.Sort(new XmlNodeComparer(sortExpression));
        }

        XmlDocument sortedXmlDoc = new XmlDocument();
        sortedXmlDoc.LoadXml("<Client></Client>");
        XmlElement sortedRoot = sortedXmlDoc.DocumentElement;

        foreach (XmlNode node in filteredList)
        {
        XmlNode importedNode = sortedXmlDoc.ImportNode(node, true);
        sortedRoot.AppendChild(importedNode);
        }

        XmlDocument resultXmlDoc = new XmlDocument();
        resultXmlDoc.LoadXml("<Result></Result>");
        XmlElement resultRoot = resultXmlDoc.DocumentElement;

        XmlElement countElement = resultXmlDoc.CreateElement("Count");
        countElement.InnerText = string.Format("{0} of the entries fulfil the given
        conditions", filteredList.Count);
        resultRoot.AppendChild(countElement);

        XmlNode importedSortedNode = resultXmlDoc.ImportNode(sortedRoot, true);
        resultRoot.AppendChild(importedSortedNode);
```

Obrázok 19 – Časť kódu webovej metódy FilterClients zoraďujúca záznamy
(Zdroj: vlastné spracovanie)

Časť kódu zobrazená na obrázku 19 slúži na zoradenie zoznamu XML uzlov (*nodes*), ktoré boli predtým filtrované na základe zadaných podmienok. Konkrétne, ak sú zadané hodnoty pre premenné *sortBy* a *sortOrder*, zavolá sa metóda *Sort()* nad zoznamom *filteredList*, ktorá triedi uzly podľa zadaných kritérií.

Triediace parametre sa vytvoria pomocou formátovania reťazca, ktorý obsahuje hodnotu *sortBy* a *sortOrder*. Potom sa vytvorí nový objekt triedy *XmlNodeComparer* a

prenáša sa mu parametrom hodnota reťazca *sortExpression*. Trieda *XmlNodeComparer* implementuje rozhranie *IComparer<XmlNode>*, ktoré určuje, ako sa budú uzly zoradzovať. Pri porovnávaní hodnôt uzlov sa použije hodnota *sortExpression* na určenie poradia a smeru triedenia.

Následne sa vytvára nový XML dokument s názvom *sortedXmlDoc*, ktorý bude obsahovať zoradené uzly. Tento dokument má koreňový element s názvom *Client*. V cykle *foreach* sa každý uzol z *filteredList* postupne prechádza a pridáva sa do nového dokumentu *sortedXmlDoc* pomocou metódy *AppendChild*.

Ako ďalší sa vytvára nový XML dokument s názvom *resultXmlDoc*, ktorý bude obsahovať výsledok zoradenia a filtrovania. Tento dokument má koreňový element s názvom *Result*. Ďalším krokom je pridanie prvku *Count* do dokumentu *resultXmlDoc*. Do tohto prvku sa vloží počet uzlov v zozname *filteredList*, ktoré vyhovujú zadaným podmienkam. Nakoniec sa pridajú uzly, ktoré boli zoradené a filtrované, do dokumentu *resultXmlDoc*. Na to sa najskôr použije metóda *ImportNode*, ktorá zabezpečí import uzlov z dokumentu *sortedXmlDoc*. Potom sa pridajú uzly do dokumentu *resultXmlDoc* pomocou metódy *AppendChild*.

```
                                        ...
    string filePath =
    Path.Combine(HttpContext.Current.Server.MapPath("~/App_Data/filter"),
    "Filter" + "_" + filterBy + "_" + filterValue + "_" + diagnose1 + diagnose2 +
    diagnose3 + "_" + DateTime.Now.ToString("yyyyMMdd_HHmmss") + ".xml");
    resultXmlDoc.Save(filePath);


    return resultXmlDoc;
    }
```

Obrázok 20 - Časť kódu webovej metódy FilterClients vracajúca XML dokument (Zdroj: vlastné spracovanie)

Posledná časť kódu, zobrazená na obrázku 20 vytvára názov súboru, v ktorom budú uložené filtrované a zoradené výsledky. Názov súboru je zostavený pomocou kombinácie rôznych parametrov, ako sú *filterBy*, *filterValue*, *diagnose1*, *diagnose2*, *diagnose3* a aktuálny dátum a čas v špecifickom formáte.

Následne sa vytvorený dokument s výsledkami uloží na zadanú cestu. Táto cesta sa zostavuje pomocou funkcie *Path.Combine()*, ktorá zlúči zadané reťazce v jednu cestu. Na začiatku cesty sa používa funkcia *HttpContext.Current.Server.MapPath()*, ktorá vráti fyzickú cestu na serveri pre virtuálnu cestu *~/App_Data/filter*. To znamená, že výsledný

súbor sa uloží do priečinku *App_Data/filter* na fyzickej adrese.

Nakoniec sa výsledný XML dokument vráti ako výstup z funkcie. Týmto krokom sa ukončí funkcia pre filtrovanie a zoradenie záznamov podľa požadovaných parametrov a uloženie výsledkov na disk.

3.3.2 Webová metóda SortClientsAtStart

Webová metóda *SortClientsAtStart* na rozdiel od metódy *FilterClients* neobsahuje parameter *dateOfArrival* a parameter *dateOfDeparture.* Namiesto spomenutých parametrov obsahuje parameter:

- **year** - celé číslo reprezentujúce rok, podľa ktorého sa majú záznamy triediť a filtrovať.

```
public XmlDocument SortClientsAtStart(int year, string filterBy = null, string
filterValue = null, string diagnose1 = null, string diagnose2 = null, string
diagnose3 = null, string sortBy = null, string sortOrder = null)
{
CalculateAges();
int count = 0;

string path = Server.MapPath("~/xml/XMLFile1.xml");
XmlDocument xmlDoc = new XmlDocument();
xmlDoc.Load(path);

XmlNode root = xmlDoc.DocumentElement;
```

Obrázok 21 - Časť kódu webovej metódy SortClientsAtStart určujúca vstupné údaje (Zdroj: vlastné spracovanie)

1. Časť kódu na obrázku 21 ukazuje, že podobne ako pri webovej metóde v kapitole 3.3.1 sa najskôr zavolá metóda *CalculateAges()* a vypočíta vek klientov podľa dátumu narodenia a aktualizuje ich vekové informácie v XML súbore.
2. Nastaví premennú *count* na hodnotu 0. Táto premenná slúži na počítanie záznamov klientov, ktorí splnili kritériá vyhľadávania a budú zahrnutí v návrate metódy.
3. Načíta obsah XML súboru, ktorý obsahuje zoznam všetkých klientov a ich informácií. Cesta k tomuto súboru je určená pomocou *Server.MapPath()* funkcie, ktorá získava fyzickú cestu na serveri pre určený virtuálny adresár alebo súbor. Potom je vytvorený nový *XmlDocument* objekt a načíta sa obsah XML súboru do tohto objektu. Nakoniec sa získa koreňový uzol dokumentu, ktorý bude použitý na vyhľadávanie konkrétnych uzlov a pridávanie nových uzlov.

```
...

XmlNodeList records = root.SelectNodes("record");
List<XmlNode> filteredRecords = new List<XmlNode>();
foreach (XmlNode record in records)
{
bool shouldInclude = true;
XmlNode dateOfArrivalNode = record.SelectSingleNode("dateOfArrival");
DateTime dateOfArrival = DateTime.Parse(dateOfArrivalNode.InnerText);
if (dateOfArrival <= new DateTime(year, 1, 1))
{
XmlNode dateOfDepartureNode = record.SelectSingleNode("dateOfDeparture");
if (dateOfDepartureNode != null &&
!string.IsNullOrEmpty(dateOfDepartureNode.InnerText))
{
DateTime dateOfDeparture;
if (DateTime.TryParse(dateOfDepartureNode.InnerText, out dateOfDeparture))
shouldInclude &= dateOfDeparture >= new DateTime(year, 1, 1);
}
        else
        shouldInclude = true;
}
        else
        shouldInclude = false;

if (!string.IsNullOrEmpty(filterBy) && !string.IsNullOrEmpty(filterValue))
{
shouldInclude &=
record.SelectSingleNode(filterBy).InnerText.Contains(filterValue);
}
if (!string.IsNullOrEmpty(diagnose1))
{
shouldInclude &=
record.SelectSingleNode("diagnose1").InnerText.Equals(diagnose1) ||
record.SelectSingleNode("diagnose2").InnerText.Equals(diagnose1) ||
record.SelectSingleNode("diagnose3").InnerText.Equals(diagnose1);
}
if (!string.IsNullOrEmpty(diagnose2))
{
shouldInclude &=
record.SelectSingleNode("diagnose1").InnerText.Equals(diagnose2) ||
record.SelectSingleNode("diagnose2").InnerText.Equals(diagnose2) ||
record.SelectSingleNode("diagnose3").InnerText.Equals(diagnose2);
}
if (!string.IsNullOrEmpty(diagnose3))
{
shouldInclude &=
record.SelectSingleNode("diagnose1").InnerText.Equals(diagnose3) ||
record.SelectSingleNode("diagnose2").InnerText.Equals(diagnose3) ||
record.SelectSingleNode("diagnose3").InnerText.Equals(diagnose3);
}
if (shouldInclude)
{
filteredRecords.Add(record);
count++;
}
}
}
```

Obrázok 22 - Časť kódu webovej metódy SortClientsAtStart určujúca filtrovacie kritériá (Zdroj: vlastné spracovanie)

Časť kódu na obrázku 22 sa zaoberá filtrovaním záznamov klientov podľa rôznych kritérií. Na začiatku sa vytvorí zoznam všetkých záznamov klientov zo súboru XML. Následne sa vytvára nový zoznam *filteredRecords*, do ktorého sa budú ukladať vyfiltrované záznamy. Potom sa iteruje cez každý element v XML dokumente nazvaný *record*.

V cykle sa kontroluje, či záznam spĺňa určité podmienky a mal by byť zahrnutý do filtrovaného zoznamu. Ak áno, pridá sa do zoznamu *filteredRecords*. Ak nie, cyklus pokračuje. V prvom rade sa zistuje, či dátum príchodu je starší ako prvý deň v roku, ak áno, premenná *shouldInclude* sa nastaví na hodnotu *false*, t. j. tento záznam sa nepridá do filtrovaného zoznamu. Ak dátum príchodu nie je starší ako prvý deň v roku, pokračuje sa v kontrole dátumu odchodu. Ak záznam obsahuje dátum odchodu, kontroluje sa, či je starší alebo rovnaký ako prvý deň v roku. Ak áno, premenná *shouldInclude* sa nastaví na hodnotu *true*. Ak dátum odchodu nie je uvedený, záznam sa pridá do zoznamu a premenná *shouldInclude* sa nastaví na hodnotu *true*.

Následne sa kontroluje, či sa používajú filtrovacie parametre *filterBy* a *filterValue*. Ak áno, skontroluje sa, či hodnota v uzle *filterBy* obsahuje reťazec *filterValue*. Ak áno, premenná *shouldInclude* sa nastaví na hodnotu *true*, inak sa nastaví na hodnotu *false*. Potom sa kontroluje, či sa používajú diagnostické kódy *diagnose1*, *diagnose2* a *diagnose3*. Ak je niektorý z nich zadaný, kontroluje sa, či záznam obsahuje aspoň jeden z týchto prvkov. Ak áno, premenná *shouldInclude* sa nastaví na hodnotu *true*, v opačnom prípade sa nastaví na hodnotu *false*. Ak premenná *shouldInclude* na konci zostáva nastavená na hodnotu *true*, záznam sa pridá do zoznamu *filteredRecords* a hodnota premennej *count* sa zvýši o jednotku. Kód sa opakuje pre každý záznam v zozname nazvaný *records*, a na konci sa vráti filtrovaný zoznam záznamov.

```
...

if (!string.IsNullOrEmpty(sortBy) && !string.IsNullOrEmpty(sortOrder))
{
string sortExpression = string.Format("{0} {1}", sortBy, sortOrder);
filteredRecords.Sort(new XmlNodeComparer(sortExpression));
}
XmlDocument sortedXmlDoc = new XmlDocument();
sortedXmlDoc.LoadXml("<Result></Result>");
XmlElement sortedRoot = sortedXmlDoc.DocumentElement;
foreach (XmlNode node in filteredRecords)
{
XmlNode importedNode = sortedXmlDoc.ImportNode(node, true);
sortedRoot.AppendChild(importedNode);
}
```

Obrázok 23 - Časť kódu webovej metódy SortClientsAtStart zoraďujúca záznamy (Zdroj: vlastné spracovanie)

Rovnako ako v predošlej kapitole 3.3.1, časť kódu na obrázku 23 vytvára triedený a filtrovaný zoznam klientov, ak boli zadané parametre triedenia *sortBy* a *sortOrder*. Ak boli tieto parametre zadané, vytvorí sa z týchto parametrov výraz *sortExpression* z a tento výraz sa používa na triedenie záznamov v metóde *filteredRecords* pomocou triedy *XmlNodeComparer*.

Potom sa vytvára nový dokument XML, ktorý bude obsahovať zotriedené a filtrované záznamy klientov. Vytvára sa prázdny element *Result* ako koreňový element v novom dokumente. Nakoniec sa prechádza zoznam *filteredRecords* a každý záznam sa importuje do nového dokumentu *sortedXmlDoc* pomocou metódy *ImportNode* a pridá sa ako potomok koreňového elementu *sortedRoot* pomocou metódy *AppendChild*.

```
                                    ...

    XmlNode countNode = sortedXmlDoc.CreateElement("Count");
    countNode.InnerText = count.ToString();
    sortedRoot.InsertBefore(countNode, sortedRoot.FirstChild);
    string filePath =
    Path.Combine(HttpContext.Current.Server.MapPath("~/App_Data/stats"),
    "Stats_Start_" + year + "_" + sortBy + "_" + sortOrder + "_" +
    DateTime.Now.ToString("yyyyMMdd_HHmmss") + ".xml");
    sortedXmlDoc.Save(filePath);
    return sortedXmlDoc;
    }
```

Obrázok 24 - Časť kódu webovej metódy SortClientsAtStart vracajúca XML dokument (Zdroj: vlastné spracovanie)

V tejto časti kódu zobrazenej na obrázku 24 sa vytvára uzol *Count* v novom XML dokumente *sortedXmlDoc* s počtom záznamov, ktoré prešli filtrovaním a triedením v predchádzajúcich krokoch. Tento uzol sa vloží na začiatok dokumentu.

Následne sa vytvorí reťazec *filePath*, ktorý obsahuje cestu k súboru, v ktorom bude uložený výsledok triedenia a filtrovania. Tento súbor sa uloží pomocou metódy *Save()* do adresára, ktorý sa nachádza v adresáre projektu, „App_Data/stats“ a bude mať názov „*Stats_Start_{year}_{sortBy}_{sortOrder}_{DateTime.Now.ToString("yyyyMMdd_HHmmss")}.xml*“, kde *{year}* predstavuje rok, *{sortBy}* a *{sortOrder}* sú parametre triedenia a *{DateTime.Now.ToString("yyyyMMdd_HHmmss")}* predstavuje aktuálny dátum a čas v špecifickom formáte. Nakoniec sa vráti nový dokument *sortedXmlDoc* s filtrovanými a zoradenými záznamami ako výstup metódy *SortClientsAtStart()*.

3.3.3 *Webová metóda SortClientsAtEnd*

Webová metóda *SortClientsAtEnd* prijíma rovnaké parametre a funguje na podobnom princípe ako metóda *SortClientsAtStart*, s tým rozdielom, že metóda *SortClientsAtEnd* poskytuje ako výsledok klientov, ktorí sa v zariadení sociálnych služieb nachádzali na konci zadaného roku.

```
...

foreach (XmlNode record in records)
{
bool shouldInclude = true;
XmlNode dateOfArrivalNode = record.SelectSingleNode("dateOfArrival");
DateTime dateOfArrival = DateTime.Parse(dateOfArrivalNode.InnerText);
if (dateOfArrival <= new DateTime(year, 12, 31))
{
XmlNode dateOfDepartureNode = record.SelectSingleNode("dateOfDeparture");
if (dateOfDepartureNode != null &&
!string.IsNullOrEmpty(dateOfDepartureNode.InnerText))
{
DateTime dateOfDeparture;
if (DateTime.TryParse(dateOfDepartureNode.InnerText, out dateOfDeparture))
shouldInclude &= dateOfDeparture >= new DateTime(year, 12, 31);
}
else
shouldInclude = true;
}
else
shouldInclude = false;

...
```

Obrázok 25 - Časť kódu webovej metódy SortClientsAtEnd určujúca filtrovacie kritériá (Zdroj: vlastné spracovanie)

Ako je možné vidieť na obrázku 25, jediná zmena v kóde sa týka cyklu *foreach*, no obsahuje podobnú logiku ako cyklus *foreach* metódy *SortClientsAtStart*, no s niekoľkými drobnými rozdielmi:

1. V cykle *foreach* je použitý dátum 31. decembra, zatiaľ čo v kóde v predchádzajúcej kapitole 3.3.2 je použitý dátum 1. januára. To znamená, že cyklus *foreach* zahrnie klientov, ktorí prichádzajú do konca roka, zatiaľ čo metóda *SortClientsAtStart* zahrnie klientov, ktorí prichádzajú do začiatku daného roka.
2. V cykle *foreach* je filtrovanie klientov na základe dátumu odchodu vykonané pomocou podmienky *dateOfDeparture >= new DateTime(year,12,31).* V kóde v predchádzajúcej kapitole 3.3.2 je táto podmienka upravená na *dateOfDeparture >= new DateTime(year,1,1),* takže sú zahrnutí klienti, ktorí odchádzajú počas celého roka.

3.4 Vedľajšie členské metódy webovej služby

Webová metóda SearchEntry

Webová metóda má jeden parameter *id*, ktorý predstavuje identifikátor záznamu, ktorý chce používateľ vyhľadať v XML dokumente. V prvom rade sa načíta XML dokument z disku na miesto určené pomocou *Server.MapPath* metódy. Následne sa vytvorí nová inštancia triedy *XmlDocument* a načíta sa do nej obsah XML súboru.

Webová metóda vyhľadáva záznamy v XML dokumente na základe ID pomocou *XPath* výrazu, ktorý vyhľadáva všetky uzly s názvom *record*, ktoré majú atribút *id* rovný zadanému identifikátoru. Ak sa v XML dokumente nachádza záznam s daným ID, uloží sa prvý uzol s týmto ID do premennej *recordNode* a vráti sa jeho obsah vo forme XML reťazca. Ak sa záznam s daným ID v XML súbore nenašiel, metóda vráti reťazec *"Entry not found"*. Celkovo táto webová metóda slúži na vyhľadávanie záznamov v XML súbore podľa ich identifikátoru a vracia ich obsah vo forme XML reťazca.

Metóda CalculateAges

Metódu *CalculateAges* sme už spomínali v predchádzajúcej kapitole 3.3, no v kapitole 3.4 sa jej budeme venovať podrobnejšie. Metóda neprijíma žiaden parameter. Metóda *CalculateAges* slúži na výpočet veku pre každý záznam *record* v XML súbore. Najprv sa načíta XML súbor pomocou metódy *XDocument.Load()* a uloží sa do premennej *xmlDoc*.

Následne sa prechádzajú všetky elementy *record* pomocou metódy *xmlDoc.Descendants("record")*. Pre každý záznam sa skontroluje, či už existuje element *age*. Ak áno, tak sa jeho hodnota aktualizuje na základe dátumu narodenia *dateOfBirth*, ktorý sa načíta z XML súboru a prevedie sa na vek. Ak element *age* ešte neexistuje, tak sa vypočíta vek a vytvorí sa nový element *age*, ktorý sa pridá k záznamu. Nakoniec sa upravený XML súbor uloží späť do pôvodného súboru pomocou metódy *xmlDoc.Save()* a metóda vráti hodnotu "*true*".

Webová metóda AddNewEntry

Webová metóda *AddNewEntry* slúži na pridanie nového záznamu do XML súboru. Daná metóda má nasledovné povinné a nepovinné parametre:

- **firstName**: meno pacienta
- **lastName**: priezvisko pacienta
- **dateOfBirth**: dátum narodenia pacienta
- **phoneNumber**: telefónne číslo pacienta
- **gender**: pohlavie pacienta
- **monthPayment**: mesačná platba pacienta
- **pension**: dôchodok pacienta
- **roomNo**: číslo izby, v ktorej pacient býva
- **diagnose1**: hlavná diagnóza pacienta
- **diagnose2**: druhá diagnóza pacienta (nepovinný parameter)
- **diagnose3**: tretia diagnóza pacienta (nepovinný parameter)
- **immobility**: informácie o imobilite pacienta
- **dependency**: informácie o závislosti pacienta
- **dateOfArrival**: dátum príchodu pacienta do zariadenia
- **dateOfDeparture**: dátum odchodu pacienta zo zariadenia (nepovinný parameter)
- **contactName**: meno kontaktného človeka
- **contactLastName**: priezvisko kontaktného človeka
- **contactPhoneNumber**: telefónne číslo kontaktného človeka

Webová metóda má jednu hlavnú funkciu, ktorou je pridať nový záznam do XML dokumentu. Na začiatku metódy sa načíta XML súbor z určenej cesty pomocou triedy *XmlDocument*. Následne sa vytvorí nový prvok *record* pre nový záznam a vyplnia sa mu všetky potrebné údaje ako *firstName, lastName, dateOfBirth* a tak ďalej. Pri vytváraní nového záznamu sa zároveň kontroluje posledné použité ID v XML súbore, aby bolo možné pridať nový záznam s unikátnym ID. V prípade, že nebola uvedená diagnóza 2 alebo diagnóza 3, záznam sa pridá bez týchto položiek.

Po vytvorení a naplnení nového prvku *record* sa skontroluje, či boli vyplnené všetky potrebné údaje. Ak niektorý z nich chýba, metóda vráti hodnotu *false*. V opačnom prípade sa nový záznam pridá do XML súboru pomocou metódy *AppendChild*, ktorá pridá nový

prvok na koniec XML súboru. Nakoniec sa XML súbor uloží a metóda vráti správu "*New entry added*".

Webová metóda UpdateEntry

Táto webová metóda slúži na aktualizáciu záznamu v XML súbore. Metóda má názov *UpdateEntry* a má jeden povinný parameter *id*, ktorý slúži na identifikáciu záznamu, ktorý sa má aktualizovať.

Okrem toho, metóda má 18 voliteľných parametrov, ktoré predstavujú rôzne polia v zázname a môžu byť aktualizované pomocou tejto metódy. Tieto parametre zahŕňajú polia *firstName*, *lastName*, *dateOfBirth*, *phoneNumber*, *gender*, *monthPayment*, *pension*, *roomNo*, *diagnose1*, *diagnose2*, *diagnose3*, *immobility*, *dependency*, *dateOfArrival*, *dateOfDeparture*, *contactName*, *contactLastName* a *contactPhoneNumber*.

Metóda začína načítaním XML súboru do pamäti pomocou metódy *Load* z triedy *XmlDocument*. Potom sa vyberú všetky uzly v súbore, ktoré majú atribút *id* zhodný s hodnotou parametra *id*. Ak sa nájde aspoň jeden taký uzol, prejdú sa všetky jeho poduzly a aktualizujú sa hodnoty podľa zadaných parametrov. Ak sa podarí úspešne aktualizovať záznam, XML súbor sa uloží a webová metóda vráti správu "*Entry updated*". V prípade neúspechu sa vráti chybová správa obsahujúca ID záznamu, ktorý sa nepodarilo aktualizovať.

Webová metóda DeleteEntry

Webová metóda *DeleteEntry* slúži na odstránenie existujúcej položky z XML súboru. Akékoľvek záznamy v súbore, ktoré majú zhodné ID ako zadané ID, budú odstránené. Metóda začína načítaním XML súboru pomocou metódy *Server.MapPath*, ktorá získa fyzickú cestu k súboru na serveri. Potom vyhľadá záznam v súbore s ID rovnakým ako zadané ID a priradí ho do premennej *recordToDelete*. Ak bol záznam nájdený, odstráni ho z XML súboru použitím metódy *RemoveChild*. Nakoniec sa XML súbor uloží s aktualizovanými dátami a vráti sa správa o úspešnom odstránení záznamu. Ak zadané ID nie je platné alebo sa nenašiel žiaden záznam, metóda vráti chybovú správu.

3.5 Používateľské rozhranie webového klienta

V kapitole 3.5 sme sa zamerali na predstavenie používateľského rozhrania, ktoré je reprezentované v podobe webového klienta. Naše UI klienta, alebo inak povedané user interface klienta pozostáva zo zdrojových súborov webového klienta *filterClients.html, addClient.html, editClient.html* a indexovej stránky. V nasledovných podkapitolách si ich podrobnejšie vysvetlíme a ukážeme ako vyzerajú vstupy používateľa a aké výstupy weboví klienti vrátia používateľovi.

3.5.1 Kľúčový zdrojový súbor filterClients.html webového klienta

V tejto kapitole sa zameriame na popis zdrojového súboru *filterClients.html* webového klienta, ktorý sme vytvorili ako súčasť našej diplomovej práce. V nasledujúcich častiach si ukážeme, ako sme tento zdrojový súbor navrhli, a popíšeme hlavné skripty, ktoré slúžili na implementáciu webových metód *FilterClients, SortClientsAtStart a SortClientsAtEnd* do daného zdrojového súboru. Tento zdrojový súbor webového klienta bol navrhnutý a implementovaný v jazyku HTML, CSS a JavaScript s využitím knižnice jQuery.

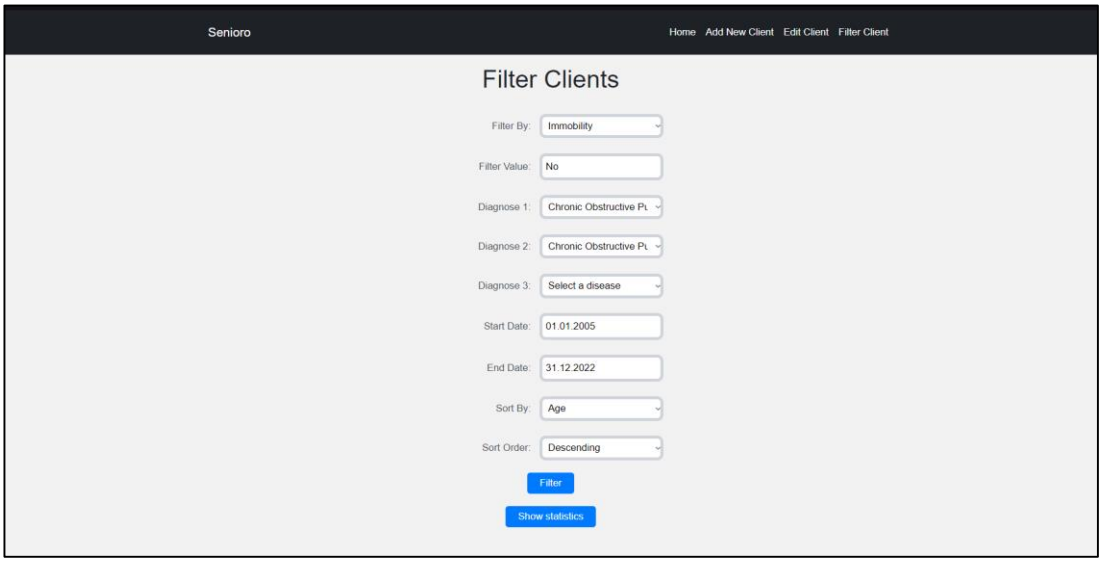
FilterClients je webová metóda, ktorá umožňuje filtrovať XML údaje o klientoch pomocou rôznych kritérií a následne tieto informácie zobrazovať. Na obrázku číslo 26 sa nachádza skript daného zdrojového súboru webového klienta. Pri spustení stránky sa pridá funkcia, ktorá sa vykoná po kliknutí na tlačidlo s id *filterBtn*. V tejto funkcii sa získajú hodnoty z *input* polí s príslušnými id a uložia sa do premenných. Potom sa pomocou funkcie *$.ajax()* zavolá webová metóda s názvom *FilterClients* na adrese *"https://localhost:" + port + "/WebService1.asmx/FilterClients"* a súčasne sa odosielajú hodnoty filtru ako parametre vo forme objektu s kľúčmi a hodnotami. Parameter *port* určuje adresu aktuálneho portu, ktorý sa používa na zobrazenie webového klienta. Parameter *dataType* určuje, že očakávaný dátový typ odpovede bude XML.

V prípade úspešnej odpovede sa výsledok spracuje pomocou funkcie *success()*, ktorá extrahuje počet nájdených záznamov a samotné záznamy z odpovede v XML formáte. Následne sa tieto údaje zobrazia v tabuľke a pridá sa riadok s počtom nájdených záznamov pred hlavičku tabuľky. Nakoniec sa pomocou funkcie *window.scrollTo()* posunie okno na začiatok zobrazenej tabuľky. V prípade chyby sa vypíše upozornenie s chybovou hláškou. Na obrázkoch 27 a 28 je možné vidieť vstup a výstup webovej metódy *FilterClients*.

```
<script>
$(document).ready(function () {
	$("#filterBtn").click(function () {
	var filterBy = $("#filterBy").val();
	var filterValue = $("#filterValue").val();
	var dateOfArrival = $("#dateOfArrival").val();
	var dateOfDeparture = $("#dateOfDeparture").val();
	var diagnose1 = $("#diagnose1").val();
	var diagnose2 = $("#diagnose2").val();
	var diagnose3 = $("#diagnose3").val();
	var sortBy = $("#sortBy").val();
	var sortOrder = $("#sortOrder").val();
	let port = location.port;
	$.ajax({
	type: "POST",
	url: "https://localhost:" + port + "/WebService1.asmx/FilterClients",
	data: {
		filterBy: filterBy, filterValue: filterValue,
		dateOfArrival: dateOfArrival, dateOfDeparture: dateOfDeparture,
		diagnose1: diagnose1, diagnose2: diagnose2, diagnose3: diagnose3,
		sortBy: sortBy, sortOrder: sortOrder
		},
	dataType: "xml",
	success: function (xml) {
	var Count = $(xml).find("Count").text();
	var clients = $(xml).find("record");
	$("#resultTable thead tr.count-row").remove();
	document.getElementById("tableHead").style.display = "table-header-group";
	$("<tr class='count-row'><td colspan='14'><b>" + Count +
	"</b></td></tr>").insertBefore($("#resultTable thead").children().first());
	$("#resultTable tbody").empty();
	clients.each(function () {
	var firstname = $(this).find("firstName").text();
	var lastname = $(this).find("lastName:not(contactPerson > lastName)").text();
	var age = $(this).find("age").text();
	var diagnose1 = $(this).find("diagnose1").text();
	var diagnose2 = $(this).find("diagnose2").text();
	var diagnose3 = $(this).find("diagnose3").text();
	var diagnoses = diagnose1 + (diagnose2 ? ", " + diagnose2 : "") + (diagnose3
	? ", " + diagnose3 : "");
	var roomno = $(this).find("roomNo").text();
	var immobility = $(this).find("immobility").text();
	var dependency = $(this).find("dependency").text();
	var monthpayment = $(this).find("monthPayment").text();
	var pension = $(this).find("pension").text();
	var contactpersonname = $(this).find("contactPerson name").text();
	var contactpersonlastname = $(this).find("contactPerson lastName").text();
	var contactpersonphonenumber = $(this).find("contactPerson
	phoneNumber").text();
	var dateofarrival = $(this).find("dateOfArrival").text();
	var dateofdeparture = $(this).find("dateOfDeparture").text();
	$("#resultTable tbody").append("<tr><td><b>" + firstname + "</b></td><td><b>"
	+ lastname + "</b></td><td>" + age + "</td><td>" + diagnoses + "</td><td>" +
	roomno + "</td><td>" + immobility + "</td><td>" + dependency + "</td><td>" +
	monthpayment + ' €' + "</td><td>" + pension + ' €' + "</td><td>" +
	contactpersonname + "</td><td>" + contactpersonlastname + "</td><td>" +
	contactpersonphonenumber + "</td><td>" + dateofarrival + "</td><td>" +
	dateofdeparture + "</td></tr>"); });
	var table = document.getElementById("tableHead");
	var offset = table.getBoundingClientRect().top - 70;
	wind ow.scrollTo({ top: offset, behavior: "smooth" });},
	error: function (xhr, textStatus, errorThrown) {
```

```
        alert("Error calling web service: " + errorThrown);}
                                    });
        });
    });
</script>
```

Obrázok 26 – Filtrovací skript zdrojového súboru filterClients.html webového klienta (Zdroj: vlastné spracovanie)



Obrázok 27 – Filtrovací formulár (Zdroj: vlastné spracovanie)

3 of the entries fulfil the given conditions

First Name	Last Name	Age	Diagnoses	Room Number	Immobility	Dependency	Month Payment	Pension	Contact Person Name	Contact Person Surname	Contact Person Phone Number	Date Of Arrival	Date of Departure
Hagan	Klulicek	80	Osteoporosis, Chronic Obstructive Pulmonary Disease, Depression	201	No	Low	275 €	460 €	Casper	Farthin	+86 591 844 3823	18.02.2002	
Blancha	Grimsdith	80	Osteoporosis, Chronic Obstructive Pulmonary Disease, Arthritis	501	No	Low	300 €	400 €	Maxie	Owenson	+33 211 805 7251	12.01.2002	
Nicolle	Fallis	54	Chronic Obstructive Pulmonary Disease, Osteoporosis	220	No	High	275 €	440 €	Corabel	Mickleborough	+880 485 531 8836	10.11.2004	

Obrázok 28 – Výstup metódy FilterClients zobrazený v tabuľke (Zdroj: vlastné spracovanie)

Na obrázku číslo 29 sa nachádza skript, ktorý sa využíva na implementáciu webovej metódy *SortClientsAtStart* jazyk JavaScript a je vykonávaný na strane klienta v prehliadači. Je závislý na knižnici jQuery a načítava sa po načítaní celého dokumentu. Pri kliknutí na tlačidlo s id *StartYearBtn* sa získa rok z textového poľa s id *yearInput* a hodnoty sortovacieho kritéria s id *sortOrderStat* a *sortByStat*. Následne sa pomocou metódy AJAX odosiela POST požiadavka na URL adresu *"https://localhost:"* + *port* + *"/WebService1.asmx/SortClientsAtStart"* a súčasne sa odosielajú hodnoty filtru ako parametre vo forme objektu s kľúčmi a hodnotami.

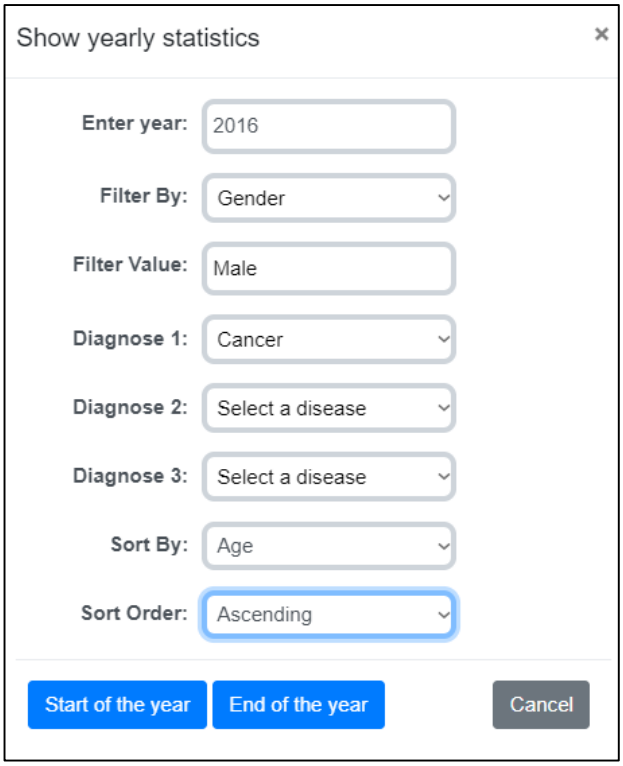
V prípade úspešnej odpovede sa vykonajú určité akcie. Modálne okno sa zatvorí, a pomocou metódy *find* sa z XML dokumentu získajú záznamy klientov a počet klientov. Záznamy klientov sa postupne prechádzajú a vyplňujú do HTML tabuľky. Nakoniec sa pomocou JavaScriptu posunie na začiatok tabuľky a celý proces končí. V prípade chyby sa

zobrazí hlásenie s informáciami o chybe. Na obrázkoch 30 a 31 sa nachádza vstup a výstup webovej metódy *SortClientsAtStart*.

```
<script>
$(document).ready(function () {
	$('#StartYearBtn').click(function () {
	var year = $('#yearInput').val();
	var filterBy = $('#filterByStat').val();
	var filterValue = $('#filterValueStat').val();
	var diagnose1 = $('#diagnose1Stat').val();
	var diagnose2 = $('#diagnose2Stat').val();
	var diagnose3 = $('#diagnose3Stat').val();
	var sortOrder = $('#sortOrderStat').val();
	var sortBy = $('#sortByStat').val();
	let port = location.port;
	$.ajax({
	type: "POST",
	url: "https://localhost:" + port + "/WebService1.asmx/SortClientsAtStart",
		data:{
		year: year, filterBy: filterBy, filterValue: filterValue,
		diagnose1: diagnose1, diagnose2: diagnose2, diagnose3: diagnose3,
		sortOrder: sortOrder, sortBy: sortBy
		},
	dataType: "xml",
	success: function (xml) {
	$('#yearModal .close').trigger('click');
	var clients = $(xml).find("record");
	var Count = $(xml).find("Count").text();
	$("#resultTable thead tr.count-row").remove();
	document.getElementById("tableHead").style.display = "table-header-group";
	$("<tr class='count-row'><td colspan='14'><b>" + Count + " clients were
	present at the start of the year " + year + "." +
	"</b></td></tr>").insertBefore($("#resultTable thead").children().first());
	$("#resultTable tbody").empty();
	clients.each(function () {
	var firstname = $(this).find("firstName").text();
	var lastname = $(this).find("lastName:not(contactPerson > lastName)").text();
	var age = $(this).find("age").text();
	var diagnose1 = $(this).find("diagnose1").text();
	var diagnose2 = $(this).find("diagnose2").text();
	var diagnose3 = $(this).find("diagnose3").text();
	var diagnoses = diagnose1 + (diagnose2 ? ", " + diagnose2 : "") + (diagnose3
	? ", " + diagnose3 : "");
	var roomno = $(this).find("roomNo").text();
	var immobility = $(this).find("immobility").text();
	var dependency = $(this).find("dependency").text();
	var monthpayment = $(this).find("monthPayment").text();
	var pension = $(this).find("pension").text();
	var contactpersonname = $(this).find("contactPerson name").text();
	var contactpersonlastname = $(this).find("contactPerson lastName").text();
	var contactpersonphonenumber = $(this).find("contactPerson
	phoneNumber").text();
	var dateofarrival = $(this).find("dateOfArrival").text();
	var dateofdeparture = $(this).find("dateOfDeparture").text();
	$("#resultTable tbody").append("<tr><td>" + firstname + "</td><td>" +
	lastname + "</td><td>" + age + "</td><td>" + diagnoses + "</td><td>" + roomno
	+ "</td><td>" + immobility + "</td><td>" + dependency + "</td><td>" +
	monthpayment + ' €' + "</td><td>" + pension + ' €' + "</td><td>" +
	contactpersonname + "</td><td>" + contactpersonlastname + "</td><td>" +
	contactpersonphonenumber + "</td><td>" + dateofarrival + "</td><td>" +
	dateofdeparture + "</td></tr>");
	});
```

```
        var table = document.getElementById("tableHead");
        var offset = table.getBoundingClientRect().top - 70;
        window.scrollTo({ top: offset, behavior: "smooth" });
        },
        error: function (xhr, textStatus, errorThrown) {
        alert("Error calling web service: " + errorThrown);
        }
                                        });
        });
});
</script>
```

Obrázok 29 – Zoraďujúci skript zdrojového súboru filterClients.html webového klienta (Zdroj: vlastné spracovanie)



Obrázok 30 - Modálne okno štatistík (Zdroj: vlastné spracovanie)

2 clients were present at the start of the year 2016.

First Name	Last Name	Age	Diagnoses	Room Number	Immobility	Dependency	Month Payment	Pension	Contact Person Name	Contact Person Surname	Contact Person Phone Number	Date Of Arrival	Date of Departure
Urbain	Ditchburn	65	Autism, Cancer	111	Yes	High	250 €	440 €	Darell	Pass	+62 362 486 9094	20.12.2014	01.01.2022
Magda	Quig	79	Cancer, Autism, Depression	518	No	Low	275 €	400 €	Poppy	Brandi	+55 676 250 2447	16.03.2006	03.10.2020

Obrázok 31 - Výstup metódy SortClientsAtStart zobrazený v tabuľke (Zdroj: vlastné spracovanie)

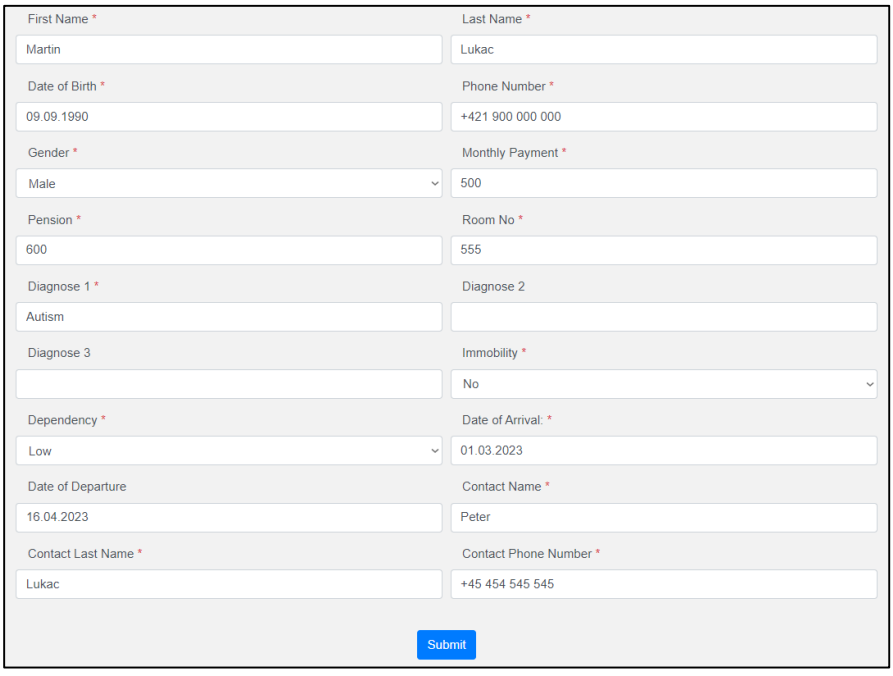
Posledný skript, ktorý sa vykoná je skript naviazaný na tlačidlo *EndYearBtn*, ktorý principiálne funguje podobne ako skript naviazaný na tlačidlo *StartYearBtn*. Rozdielom medzi týmito skriptami je, že skript naviazaný na tlačidlo *EndYearBtn* volá webovú metódu *SortClientsAtEnd*, ktorá zobrazí a zoradí klientov, ktorí sa nachádzali v zariadení na konci roka a skript naviazaný na tlačidlo *StartYearBtn* volá metódu *SortClientsAtStart*, ktorá zobrazí a zoradí klientov, ktorí sa nachádzali v zariadení na začiatku roka.

3.5.2 Webový klient vedľajších metód

Zdrojový súbor addClient.html webového klienta

Skript v zdrojovom súbore "*addClient.html*" webového klienta pri kliknutí na tlačidlo s id *submitBtn* spúšťa funkciu, ktorá získava hodnoty z rôznych polí na stránke využitím jQuery a posiela ich na server pomocou AJAX volania.

Konkrétne, funkcia získava hodnoty z polí s id *firstName*, *lastName*, *dateOfBirth*, *phoneNumber*, *gender*, *monthPayment*, *pension*, *roomNo*, *diagnose1*, *diagnose2*, *immobility*, *dependency*, *dateOfArrival*, *contactName*, *contactLastName*, *contactPhoneNumber*, *diagnose3* a *dateOfDeparture*. Tieto hodnoty sa posielajú ako dáta v tele HTTP požiadavky URL adresy „*https://localhost:" + port + "/WebService1.asmx/SortClientsAtEnd*“ pomocou HTTP POST metódy. Parameter *port* určuje adresu aktuálneho portu, ktorý sa používa na zobrazenie webového klienta. Na obrázku 32 je možné vidieť volanie webovej metódy *AddNewClient* pomocou formulára.



Obrázok 32 – Volanie metódy AddNewClient pomocou formulára (Zdroj: vlastné spracovanie)

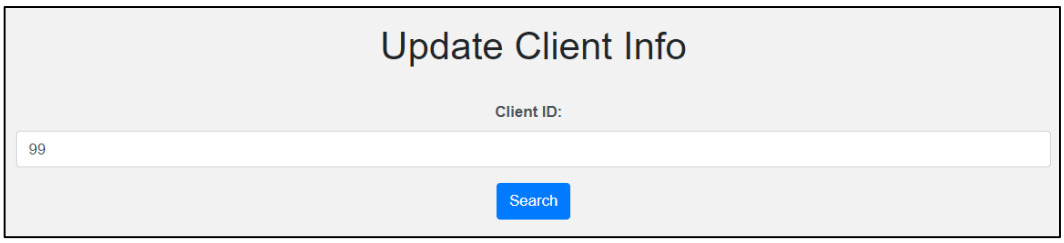
Ako je možné vidieť na obrázku 33, v prípade úspešného odoslania požiadavky serveru, funkcia spracúva odpoveď a vypíše hlásenie o úspešnom pridaní záznamu v konzole. Ak sa nepodarí pridať nový záznam, funkcia zobrazí upozornenie s chybovou

hláškou hlásiacou, že záznam nebolo možné pridať.

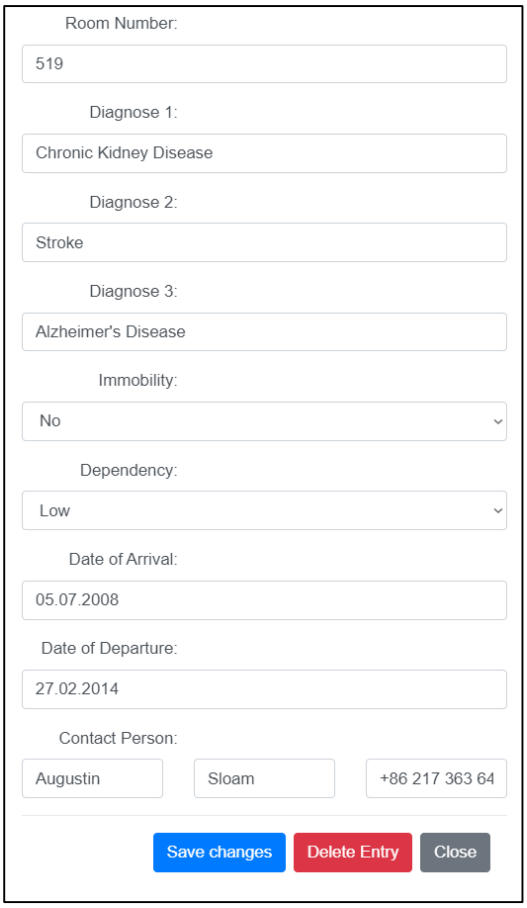
Obrázok 33 – Modálne okno pri úspešnom pridaní klienta (Zdroj: vlastné spracovanie)

Zdrojový súbor editClient.html webového klienta

„*editClient.html*“ je zdrojový súbor webového klienta, ktorý obsahuje formulár pre úpravu záznamu a sú k nemu pripojené tri skripty, ktoré volajú jednotlivé webové metódy. Prvý skript vyhľadáva záznamy podľa id a vyplní údaje do modálneho formulára. Pretože ide o asynchrónne volanie (ajax), môže sa vykonať bez obnovenia stránky. Používa sa po stlačení tlačidla *Search* a vracia vyhľadaný záznam z XML dokumentu pomocou metódy *SearchEntry* zo serverovej strany. Na obrázku 34 je zobrazené textové pole na zadanie ID klienta a tlačidlo na zavolanie webovej metódy *SearchEntry*. Na obrázku 35 je následne zobrazený formulár s načítanými údajmi z XML dokumentu, ktorý slúži na úpravu alebo prípadné vymazanie údaju v XML dokumente.



Obrázok 34 – Tlačidlo na volanie metódy SearchEntry (Zdroj: vlastné spracovanie)

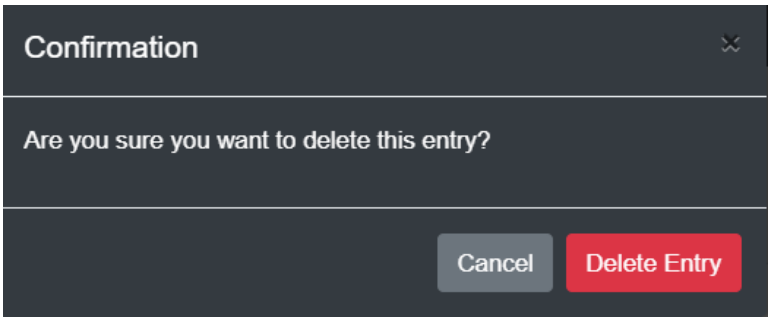


Obrázok 35 – Formulár na úpravu a vymazanie údaju (Zdroj: vlastné spracovanie)

Druhý skript sa spustí po stlačení tlačidla *Save Changes* a odosiela zmeny z formulára na serverovú stranu pomocou metódy *UpdateEntry*. Tento skript používa rovnaké tlačidlo ako prvý skript a využíva AJAX požiadavku na odoslanie údajov z formulára na serverovú stranu a aktualizuje záznam v XML dokumente. Ak bol záznam úspešne aktualizovaný, otvorí sa modálne dialógové okno *editSuccessModal* s oznámením o úspešnej úprave záznamu, v opačnom prípade sa zobrazí chybová správa. Tento skript tiež používa potvrdenie modálneho okna *updateModal* pred zobrazením modálneho okna *editSuccessModal*, aby bolo zabezpečené, že sa úpravy na stránke nebudú môcť robiť počas zobrazovania modálneho okna. Ako je možné vidieť na obrázku 36, po úspešnom upravení záznamu sa zobrazí modálne okno s nápisom „*Entry updated*“.

Obrázok 36 – Modálne okno pri úspešnej úprave dokumentu (Zdroj: vlastné spracovanie)

Tretí skript *DeleteEntry* odstraňuje záznam z XML dokumentu. Používa AJAX požiadavku a volá metódu *DeleteEntry* na serverovej strane. Po potvrdení mazania záznamu sa záznam odstráni z XML dokumentu a zobrazí sa modálne okno s oznámením o úspešnom mazaní. Tieto úkony sú zobrazené na obrázku 37 a obrázku 38.



Obrázok 37 – Modálne okno potvrdenia vymazania údaju (Zdroj: vlastné spracovanie)

Obrázok 38 – Modálne okno úspešného vymazania údaju (Zdroj: vlastné spracovanie)

4 Analýza súčasného stavu evidencie klientov

V tejto kapitole sme sa zamerali na porovnanie existujúceho riešenia na evidenciu klientov domova sociálnych služieb XYZ, nachádzajúceho sa v Košickom kraji, s nami vyvinutým riešením. Nami vytvorené riešenie sme podrobnejšie popísali v kapitolách 3.1, 3.2, 3.3, 3.4 a 3.5. Keďže išlo o domov sociálnych služieb, ktorý bol založený pred rokom 2000 a do informačného systému dlhé roky neinvestovali, využívajú neaktuálne metódy ukladania údajov o klientoch. Domov sociálnych služieb XYZ používa na ukladanie údajov o klientoch softvér z balíka Microsoft Office, presnejšie Microsoft Excel 2016. Keďže domov sociálnych služieb nám neposkytol prístup ku konkrétnym atribútom, ktoré o klientoch ukladajú, nemôžeme aktuálnosť nami uložených parametrov potvrdiť, ale ani vyvrátiť.

Na rozdiel od Excelovskej tabuľky, nami navrhnutá webová služba ponúka prehľadné a intuitívne používateľské rozhranie, v ktorom sa môže opatrovateľ oveľa

jednoduchšie orientovať a pracovať s údajmi. Navyše, dobre spravené používateľské rozhranie môže zlepšiť celkovú efektivitu a produktivitu opatrovateľa pri práci s údajmi o klientoch. Webová služba poskytuje možnosť jednoduchého a intuitívneho používania cez webový prehliadač alebo iný program na strane klienta.

Na druhej strane, ak sa údaje o klientoch ukladajú v Excel tabuľkách, môže byť orientácia v údajoch trochu zložitejšia, pretože Excel tabuľky sú zvyčajne zložitejšie na používanie ako webové stránky alebo aplikácie. Navyše, webová služba je prístupnejšia a flexibilnejšia ako Excel tabuľky, pretože umožňuje prístup k údajom z rôznych zariadení a umožňuje viacerým používateľom pracovať s údajmi súčasne.

Taktiež vyhľadávanie a filtrovanie údajov môže byť v Excel tabuľkách niekedy zložité a môže vyžadovať určitú technickú zručnosť. Avšak, nami vytvorená webová služba a html klient sú navrhnuté tak, aby bolo filtrovanie a vyhľadávanie údajov jednoduché aj pre používateľov bez technických zručností. Napríklad, nami vytvorená webová služba a html klient obsahujú prehľadné filtre, ktoré umožnia vyhľadávanie klientov podľa rôznych kritérií, ako napríklad vek, diagnóza, priezvisko alebo obdobie pobytu v centre a následné zoradenie zobrazených údajov. Bolo pre nás dôležité, aby bolo používateľské rozhranie navrhnuté s ohľadom na potreby používateľa a aby obsahovalo všetky potrebné funkcie na rýchle a efektívne vyhľadávanie a filtrovanie údajov.

Taktiež pri používaní Excelu môžu vzniknúť bezpečnostné riziká, ako sú chyby v údajoch, nesprávne nastavenie prístupu k súborom alebo nedostatočné zabezpečenie súborov a prenosu údajov. Okrem toho, ak nie sú súbory Excelu pravidelne zálohované, môže dôjsť k strate údajov v prípade zlyhania alebo poškodenia súboru.

Na druhej strane, ASP .NET webová služba umožňuje pracovníkom domova sociálnych služieb pristupovať k dátam a funkciám servera pomocou štandardizovaného protokolu HTTP/S. Tento protokol umožňuje šifrovanie prenosu údajov a autentifikáciu klienta na serveri. Webová služba môže byť tiež ľahko aktualizovaná alebo opravená centrálne na serveri, čím sa minimalizuje riziko chýb klientov.

5 Diskusia

V kapitole diskusia zhrnieme prínos nášho riešenia, no taktiež poukážeme na možné zlepšenia webového klienta a taktiež aj metód webovej služby. Nami vyvinuté riešenie, pozostávajúce z webovej služby a jej webového klienta poskytuje prehľadné riešenie na sledovanie parametrov klientov domova sociálnych služieb a ich následné analyzovanie. Zároveň poskytuje uchovanie zanalyzovaných údajov na fyzickej adrese v XML dokumentoch. Webový klient webovej služby je navrhnutý, aby používateľovi poskytol jednoduché a prehľadné vyhľadávanie klientov.

Ako sme spomenuli v kapitole 4, rozdiel medzi ukladaním informácií o klientoch do Excelovských tabuliek a použitím webovej služby môže byť výrazný. Pri ukladaní informácií do Excelovských tabuliek môže byť orientácia v dátach náročnejšia, pretože tabuľky Excel sú často zložitejšie na používanie ako moderné webové stránky alebo aplikácie. S použitím webovej služby je však prístup k informáciám oveľa jednoduchší a prístupnejší. Webová služba umožňuje prístup k informáciám z rôznych zariadení, čo znamená, že používatelia môžu ľahko pracovať s informáciami, bez ohľadu na to, kde sa práve nachádzajú. Okrem toho, webová služba umožňuje zamestnancom pracovať s informáciami o klientoch súčasne, čo umožňuje efektívnejšie a pohodlnejšie spravovanie a zabezpečenie kvality poskytovaných sociálnych služieb.

Nami vytvorené riešenie by bolo možné rozšíriť o webovú metódu webovej služby, ktorá by z odlišného XML súboru načítala údaje o zamestnancoch klientskeho centra, a to konkrétne prvky slúžiace na prihlásenie *username* a *password*, a zároveň prvky slúžiace na vytvorenie a prezretie profilu prihláseného zamestnanca *empFirstName, empLastName, empJobTitle, empSalary* a podobne, čím by sa zvýšila bezpečnosť pri manipulácii s citlivými klientskymi údajmi. Taktiež by sa zamedzilo neoprávnenému prístupu ku klientskym údajom. Ďalším možným zlepšením riešenia je prehľadnejšie zobrazenie štatistík pre zistenie ziskovosti, napríklad zobrazenie klientov s najvyššou ziskovosťou pomocou vynásobenia platby za mesiac počtom mesiacov, počas ktorých je klient ubytovaný v klientskom centre. Následne by týmto klientom mohli byť poskytnuté nadštandardné služby ako napríklad možnosť využitia doplnkových služieb za zvýhodnené ceny, špeciálne akcie a podujatia pre dlhodobých klientov, osobný asistent, ktorý by bol k dispozícii pre potreby klientov a špeciálne odmeny a dary pre dlhodobých klientov. Tento prístup by mohol zvýšiť spokojnosť zákazníkov a viesť k ďalšiemu rozvoju klientskeho centra.

Záver

Evidencia klientov sa v súčasnosti považuje za prioritu každého ubytovacieho zariadenia, či už ide o hotel, penzión, turistickú ubytovňu alebo domov sociálnych služieb. Domovy sociálnych služieb sa zameriavajú nie len na ubytovanie svojich klientov, no taktiež na rôzne liečebné procedúry, ktoré svojim klientom poskytujú. Z tohto dôvodu je okrem evidencie údajov pre zamestnancov dôležitý aj prístup k možnosti sledovania stavu klientov pomocou rôznych parametrov. V diplomovej práci sme sa zaoberali analýzou možností použitia XML webových služieb na sledovanie parametrov klientov domova sociálnych služieb v elektronickom informačnom systéme a ich porovnanie s doterajším spôsobom sledovania parametrov klientov. Vytvorili sme ASP .NET XML webovú službu, ktorá svojmu klientovi poskytuje údaje podľa používateľom zadaných požiadaviek.

V prvých dvoch kapitolách sme predstavili teoretický základ týkajúci sa webových služieb a webového klienta, ako aj metodiku a metódy použité v našej práci. V tretej kapitole sme následne bližšie predstavili nami vyvinuté filtrovanie klientov na základe používateľom daných požiadaviek, použitím ASP .NET webových služieb.

Výsledkom našej práce je funkčná ASP .NET XML webová služba, ktorá umožňuje používateľovi štandardné CRUD operácie, to znamená pridanie nového klienta, čítanie už existujúceho XML dokumentu, aktualizáciu údajov klienta a vymazanie klientskeho záznamu z XML dokumentu. Zároveň poskytuje zamestnancovi domova sociálnych služieb možnosť zobrazenia všetkých klientov, ktorí sa v zariadení nachádzajú, no taktiež filtrovanie klientov podľa zadaných požiadaviek a zobrazenie usporiadaného výstupu tohto filtrovania v tabuľke. Okrem toho umožňuje zobrazenie štatistík o počte klientov, ktorí sa v zariadení nachádzali na začiatku a konci používateľom zadaného roka. Tieto štatistiky je taktiež možné vyfiltrovať podľa používateľom zadaných parametrov a výstupy, ktoré vráti webová metóda webovému klientovi je taktiež možné v tabuľke usporiadať vzostupne alebo zostupne podľa vybraných prvkov z XML dokumentu. Každé filtrovanie záznamov, či už sa týka bežného vyfiltrovania klientov alebo zobrazenia štatistík o stave klientov vytvára na fyzickej adrese XML dokument, ktorý v názve obsahuje časovú pečiatku a obsahuje všetky vyfiltrované záznamy a taktiež ich počet.

Využiteľnosť našej diplomovej práce vyplýva z kapitoly 4, v ktorej sme naše riešenie porovnali s už existujúcim riešením, slúžiacim na evidenciu klientov, využívaným

domovom sociálnych služieb, ktoré sa nachádza v Košickom kraji. Pre dané seniorské centrum by po implementácii rozšírení spomenutých v kapitole 5 Diskusia, mohlo naše riešenie poskytnúť zefektívnenie evidencie a sledovania zdravotného stavu ich klientov. Naša práca by mohla taktiež slúžiť ako podnet pre ďalšie skúmanie možností použitia webových služieb na evidenciu klientov v oblasti sociálnych služieb.

Bibliografické zdroje

[1] ALBAHARI, J. – JOHANNSEN, E. C# 9.0 in a Nutshell: The Definitive Reference. Sebastopol: O'Reilly Media, 2021. 1088 s. ISBN: 978-1098101603

[2] BRAVO, M. – PASCUAL, J. – RODRÍGUEZ, J. Semantic Representation of Public Web Service Descriptions. In *ICCSA*. 2013. 636 – 651 s. ISBN 978-3-642-39639-7

[3] Fileformat.com. What is an ASMX file? [online] Dostupné na internete: https://docs.fileformat.com/web/asmx/

[4] FOX, R. – HAO, W. Internet infrastructure: Networking, Web Services and Cloud Computing. Boca Raton: CRC Press, 2018. 633 s. ISBN 978-1-1380-3991-9.

[5] FREEMAN, A. Pro Bootstrap 4, New York: Apress Media LLC, 2017. ISBN: 978-1-4842-2919-9

[6] FRIESEN, J. Java XML and JSON: Document Processing for Java SE. Dauphin: Apress Media LLC, 2018. 6-8 s. ISBN 978-1-4842-4329-9

[7] GOURLEY, D. et. al. HTTP: The definitive Guide. Sebastobol: O'Reilly Media, Inc, 2002, 635 s. ISBN 978-1-56592-509-0

[8] HARL, M. – DONAHOE, L. Learn Enough HTML, CSS and Layout to be Dangerous: An Introduction to Modern Website Creation and Templating Systems. Boston: Addison-Wesley Professional, 2022, 688 s. ISBN 978-0-13784-318-3

[9] HOLDENER III, A. AJAX: The definitive guide. Sebastopol: O'Reilly Media, Inc., 2008, 982 s. ISBN 978-0-596-52838-8

[10] Informatica. Web Services Guide. [elektronický dokument] Dostupné na internete: https://docs.informatica.com/content/dam/source/GUID-5/GUID-51941C56-55D7-4195-A78C-29263FBDCB12/31/en/IN_1021_NewFeaturesGuide_en.pdf

[11] JOHNSON, B. Professional Visual Studio 2019. Hoboken: Wiley, 2019. 912 s. ISBN 978-1-119-56563-2

[12] JOSHI, B. Beginning XML with C# 7: XML Processing and Data Access for C# Developers. Thane: Apress Media LLC, 2017. 21 – 22 s. ISBN 978-1-4842-3104-3

[13] MEYER, E. A. – WEYL, E. CSS: The definitive guide. Sebastopol: O'Reilly Media, Inc., 2018, 1088 s. ISBN 978-1-449-39319-9

[14] Microsoft Docs. System Namespace. [online]. Dostupné na internete: https://docs.microsoft.com/en-us/dotnet/api/system?view=net-6.0

[15] Microsoft Docs. System.Collections.Generic Namespace. [online]. Dostupné na

internete: https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic?view=net-6.0
[16] Microsoft Docs. System.Globalization Namespace. [online]. Dostupné na internete: https://docs.microsoft.com/en-us/dotnet/api/system.globalization?view=net-6.0
[17] Microsoft Docs. System.IO Namespace. [online]. Dostupné na internete: https://docs.microsoft.com/en-us/dotnet/api/system.io?view=net-6.0
[18] Microsoft Docs. System.Web Namespace. [online]. Dostupné na internete: https://docs.microsoft.com/en-us/dotnet/api/system.web?view=net-6.0
[19] Microsoft Docs. System.Web.Services Namespace. [online]. Dostupné na internete: https://docs.microsoft.com/en-us/dotnet/api/system.web.services?view=netframework-4.8.1&viewFallbackFrom=net-6.0
[20] Microsoft Docs. System.Xml Namespace. [online]. Dostupné na internete: https://docs.microsoft.com/en-us/dotnet/api/system.xml?view=net-6.0
[21] Microsoft Docs. System.Xml.Linq Namespace. [online]. Dostupné na internete: https://docs.microsoft.com/en-us/dotnet/api/system.xml.linq?view=net-6.0
[22] PATNI, S. Pro RESTful APIs: Design, Build and Integrate with REST, JSON, XML and JAX-RS. Santa Clara: Apress Media LLC, 2017. 7 - 11 s. ISBN 978-1-4842-2664-3
[23] SPAANJAARS, I. Beginning ASP.NET 4.5.1 in C# and VB, Wrox, 2014, ISBN 978-1-1188-4696-4
[24] Stackpath.com. WHAT IS A WEB APPLICATION?. [online] Dostupné na internete: https://www.stackpath.com/edge-academy/what-is-a-web-application/
[25] TIHIMIROVS, J. – GRABIS, J. Comparison of SOAP and REST Based Web Services Using Software Evaluation Metrics. *In: Information Technology and Management Science.* Riga: Riga Technical University, 2016, s. 92-97. ISSN 2255-9094
[26] Tutorialspoint.com. WSDL: Web services description language. [elektronický zdroj] Dostupné na internete: http://www.tutorialspoint.com/wsdl/wsdl_tutorial.pdf
[27] W3.org. Web Services Architecture. [online] Dostupné na internete: https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest

Prílohy

Príloha č. 1 – Projektový adresár Senioro

Príloha č. 2 – Manuál na spustenie webového klienta webovej služby