EKONOMICKÁ UNIVERZITA V BRATISLAVE

FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 103004/I/2019/36069387454791428

ASP .NET XML webová služba poskytujúca usporiadané informácie o realitách realitnej kancelárie

Diplomová práca

Bratislava 2019 **Šimon Krajniak, Bc.**

EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

ASP .NET XML webová služba poskytujúca usporiadané informácie o realitách realitnej kancelárie

Diplomová práca

Študijný program: Informačný manažment

Študijný odbor: Kvantitatívne metódy v ekonómii

Školiace pracovisko: Katedra aplikovanej informatiky FHI

Vedúci záverečnej práce: Ing. Igor Košťál, PhD.

Bratislava 2019 **Šimon Krajniak, Bc.**

Poďakovanie:

Ďakujem za rady a pomoc pri vypracovaní diplomovej práce vedúcemu záverečnej práce, Ing. Igorovi Košťálovi, PhD.,. Rád by som vyzdvihol jeho odborný prístup a ochotu.

ABSTRAKT

KRAJNIAK, Šimon: *ASP .NET XML webová služba poskytujúca usporiadané informácie o realitách realitnej kancelárie*. – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra aplikovanej informatiky FHI. – Vedúci záverečnej práce: Ing., Igor Košťál, PhD. – Bratislava: FHI EU, 2019, 76s

Cieľom záverečnej práce je vytvoriť ASP.NET XML webovú službu poskytujúcu usporiadané informácie o realitách realitnej kancelárie. Práca je rozdelená do 5 kapitol. Obsahuje 23 obrázkov, 4 tabuľky a 1 prílohu. Prvá kapitola je venovaná programovacím jazykom, technikám, protokolom a platformám používaných pri vytváraní webových služieb. V ďalšej časti sa charakterizujú ciele práce, metodika práce a metódy skúmania. Tretia kapitola hodnotí súčasný stav používania softvéru na evidenciu realitných položiek v oslovených realitných kanceláriách. Štvrtá kapitola prezentuje výsledky našej práce, s dôrazom na cieľ práce, ktorým je vytvorená ASP.NET XML Webová služba. Záverečná piata kapitola, ktorou je Diskusia, sa zaoberá zhodnotením súčasných možností pre rozšírenie vytvorenej aplikácie a jej možnom použití pre prax. Výsledkom riešenia danej problematiky je ASP.NET XML Webová služba používaná klientom, ktoré sú spoločne využiteľné realitnými sprostredkovateľmi a realitnými kanceláriami na evidenciu obchodných položiek.

Kľúčové slová:

ASP.NET, XML, Webová služba, Reality, Realitné kancelárie, Softvér, C#, Evidencia, Spracovanie údajov

ABSTRACT

KRAJNIAK, Šimon: *ASP .NET XML web service providing sorted information about estates of an estate agency*. – University of Economics in Bratislava. Faculty of Economic Informatics; Department of Applied Informatics. – Thesis supervisor: Ing., Igor Košťál, PhD. – Bratislava: FHI EU, 2019, 76p

The main goal of the diploma thesis is to design ASP.NET XML Web service providing sorted information about estates offered by real estate agencies. Thesis is divided to 5 chapters including 23 images, 4 tables and 1 attachment. The first chapter presents used programming languages, techniques, protocols, and platforms useful for creating web services. In the next chapter, goals of thesis, methodology and used examination methods are characterised. Results of our work are presented in fourth chapter mostly focusing on main goal of thesis, which is creating ASP.NET XML Web service. Final fifth chapter, which is Discussion contains consideration about use cases and extensibility of our application and its means for practise. The outcome of our examination and this thesis is designed ASP.NET XML Web Service providing sorted information about estates of estate agency used by client, which are together useful for real estate agents and agencies for keeping information about estates.

Keywords:

ASP.NET, XML, Web service, Estate, Estate agencies, Software, C#, Evidence, Data processing

OBSAH

Úvod

Ešte na počiatku 21. storočia, kedy penetrácia internetu nedosahovala zďaleka takých percentuálnych hodnôt ako je tomu dnes, si mohla väčšina z jeho vtedajších 300 miliónov používateľov[1] len ťažko predstaviť cestu, ktorou ľudstvo prejde vďaka jeho rozvoju v najbližších desaťročiach.

V dobe písania tejto diplomovej práce, v roku 2019, už vieme konštatovať absolútnu integráciu internetu do všetkých sfér života ľudí, aj do tej pracovnej, kde bol jeho rozvoj sprevádzaný mnohými novými technológiami, programovacími jazykmi, novými prístupmi k vývoju informačných systémov.
Jednou z technológií, ktoré prispeli k rozvoju internetu sú webové služby, ktorým sa budeme v práci venovať.

Zaujala nás možnosť naučiť sa viac o fungovaní webových služieb, využití ASP.NET, značkovacieho jazyka XML, s ktorým sme sa stretli pri práci v tlačových agentúrach ako so štandardom na distribúciu spravodajstva smerom ku klientom, médiám.

Cieľom tejto diplomovej je vytvoriť webovú službu použitím pomocou vývojárskej šablóny ASP.NET XML Web Service. Služba bude poskytovať prostredníctvom jej vystavených metód klientovi usporiadané informácie o inzerátoch realitnej spoločnosti.

Pri vytváraní webovej služby využijeme integrované vývojové prostredie od spoločnosti Microsoft, Visual Studio Enterprise 2017 s licenciou poskytnutou Ekonomickej univerzite v Bratislave.

Na prvotné zoznámenie sa s vývojovým prostredím Visual Studio sme použili zjednodušenú bezplatnú verziu s názvom Visual Studio Code, ktorá je dostupná aj na operačnom systéme MAC OS, s ktorým sme čiastočne pracovali.

[1] Internet World Stats [Online] Dostupné na internete:
<https://www.internetworldstats.com/emarketing.htm>. [16.2.2019].

Práca s webovými službami vyžaduje voľbu spoľahlivého webového prehliadača. Visual Studio používa pri testovaní webových služieb primárny prehliadač zvolený v operačnom systéme. My sme zvolili prehliadač od spoločnosti Google a preto sme webovú službu testovali vo webovom prehliadači Google Chrome.

Na úpravu obrázkov a tvorbu vizuálov sme použili aplikáciu Pixelmator, ktorá je dostupnejším substitútom k známejšiemu Adobe Photoshop.

Nemenej dôležitou súčasťou práce bude analýza možností použitia ASP.NET XML webových služieb v informačných systémoch realitných kancelárií na sledovanie ponuky realitných položiek. Pri analýze vykonáme porovnanie použitia webovej služby s aplikáciami, ktoré realitné kancelárie v súčasnosti reálne používajú. Tieto informácie získame od realitných kancelárií, ktoré nám ich budú ochotné, pre účely tejto práce, poskytnúť.

Práca je rozdelená do piatich kapitol. V prvej sa zoznámime s použitými jazykmi, technológiami, protokolmi a platformami na vývoj webových služieb. Zhromaždíme v nej teoretické informácie dostupné z domácich a zahraničných zdrojov, ktoré sú potrebné pre správne porozumenie fungovania architektúry webových služieb.

V kapitole Cieľ práce, metodika práce a metódy skúmania si určíme hlavný cieľ práce a popíšeme postup, akým sa k nemu dopracujeme.

V tretej kapitole zanalyzujeme súčasný stav vo vybraných firmách z realitného segmentu vo vzťahu k využívaniu softvéru na evidenciu obchodných položiek.

V najdôležitejšej, štvrtej kapitole s názvom Výsledky práce navrhneme, naprogramujeme a otestujeme webovú službu. Hotovú webovú službu umiestnime na webový HTTP server IIS. Zároveň pripravíme klienta, ktorý bude webovú službu používať a jej výstupy zobrazovať v prehľadnej tabuľke.

V kapitole diskusia zanalyzujeme vhodné rozšírenia a úpravy webovej služby a zamyslíme sa nad jej komerčným využitím. Tiež zhodnotíme splnenie cieľa.

1. Použité programovacie jazyky, techniky, protokoly a platformy pri vytváraní webovej služby

V prvých rokoch 21. storočia sa programovanie na báze komponentov stalo veľmi populárnym. Príkladom aplikácie fungujúcej na báze komponentov je akékoľvek "end to end" aplikácia vo sfére e-obchodu. Aplikácia sídliaca na webovej farme musela posielať objednávky do aplikácie na plánovanie podnikových zdrojov, tzv. Enterprise Resource Planning aplikácie alebo skrátene ERP. V mnohých prípadoch ERP aplikácia sídlila na odlišnom stroji s úplne odlišným operačným systémom.

Táto situácia spôsobovala klientom mnohé vrásky na čele pri pokusoch o komunikáciu so serverom. Proti nim hralo aj zavádzanie firewallov zo strany správcov sietí.

Funkčnosť takýchto aplikácií preto do veľkej miery závisela na komunikácii so správcami siete a žiadostiach o otváranie správnych portov pre podporu požadovanej aplikácie. Spoločnosť Microsoft sa pôvodne pokúsila riešiť tieto scenáre rozširovaním existujúcich technológií prezentovaním CIS - COM Internet Services, no tie neboli akceptované širokou verejnosťou.

Microsoft musel zvoliť nový prístup v hľadaní riešenia, ktoré muselo spĺňať nasledovné podmienky:

- **interoperabilita** - vzdialená služba nemôže byť naviazaná na konkrétnu platformu, ale musí umožňovať svoje používanie naprieč platformami,

- **priateľskosť k sieti internet** - pre klientov, ktorí pristupujú k službe cez sieť internet musí vzdialená služba fungovať správne,

- **silne typizované rozhranie** - keď hovoríme o typoch dát odoslaných a prijímaných zo vzdialenej služby, nemalo by dochádzať k nejednoznačnostiam,

- **schopnosť využiť existujúce internetové štandardy** - nasadenie vzdialenej služby by malo využiť internetové štandardy do takej miery, ako je to možné a vyhnúť sa znovuobjavovaniu riešení problémov, ktoré už boli vyriešené,

- **podpora pre akýkoľvek jazyk** – úzka prepojenosť s konkrétnym programovacím jazykom nie je vhodná. Klient by mal byť schopný použiť službu bez ohľadu na jazyk, v ktorom bol napísaný,

- **podpora pre akúkoľvek distribuovanú infraštruktúru komponentov** - nie je vhodná ani úzka prepojenosť na jednotlivé komponenty infraštruktúry.

Odpoveďou spĺňajúcou všetky tieto podmienky sa stalo riešenie Microsoftu známe ako Webová služba vystaví rozhranie na vyvolanie aktivity v mene klienta. Prístup k webovej služby je dostupný prostredníctvom internetových štandardov.[2]

Termín "Webová služba" bol oficiálne použitý softvérovou spoločnosťou Microsoft pri prezentácii ich .NET iniciatívy, ako jej komponent, v júni roku 2000. Tá mala priniesť novú víziu využitia internetu vo vývoji softvéru a v inžinierstve[3].

Ako webové služby oslovovali čoraz viac vývojárov, bolo jasné, že spôsobia revolúciu v distribuovaných výpočtových systémoch.

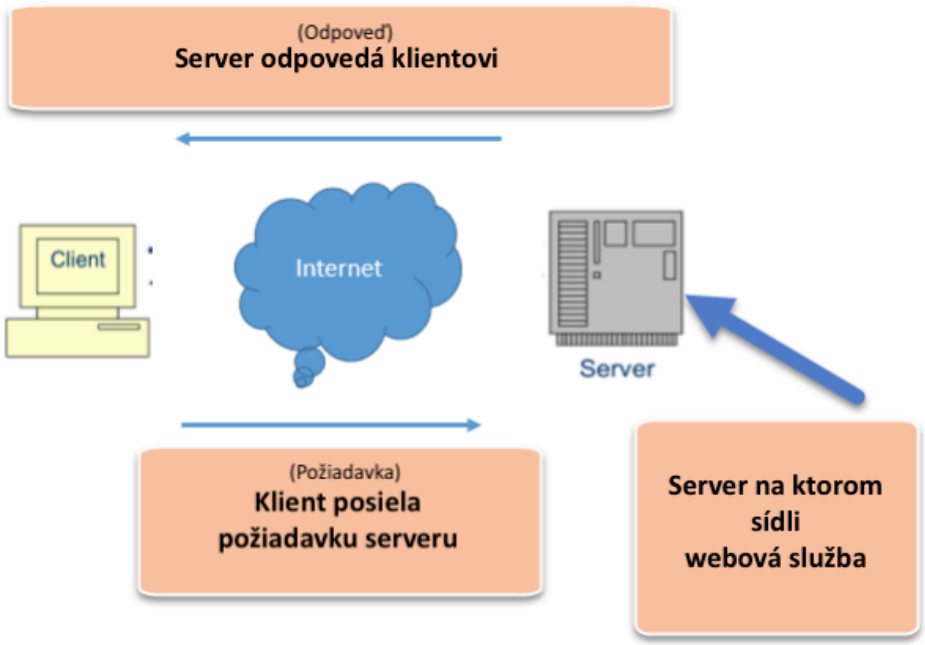
[2] SHORT, Scott. *Building XML Web Services for the Microsoft .NET Platform.* Redmont: Microsoft Press, 2002. s. 1-3. ISBN 0-7356-1406-7

[3] C# Corner [Online] Dostupné na internete: <https://www.c-sharpcorner.com/UploadFile/1d42da/web-service-basics/>. [16.2.2019].

1.1. Definícia webových služieb a ich fungovanie

Webová služba je štandardizované médium, ktoré slúži na sprostredkovanie komunikácie medzi klientom a aplikáciami na serveri cez web. Ide o časť softvéru navrhnutú na splnenie definovaných úloh.
Môžeme ich vyhľadať na sieti a rovnako ich aj zavolať. Po zavolaní webová služba ponúkne klientovi potrebnú funkcionalitu, ktorá spustí zavolanú metódu webovej služby.

Obrázok č. 1: Architektúra webovej služby



Zdroj: (https://www.guru99.com/web-service-architecture.html, preklad: Vlastné spracovanie)

Vykonané požiadavky poznáme ako procedurálne volania, RPC (Remote Procedure Call). Ide o volania metód webovej služby.

Keďže webové služby predstavujú súhrn súvisiacich štandardov, ktoré umožňujú počítačom navzájom komunikovať a vymieňať si informácie cez sieť, bolo nutné zvoliť primárny štandard. Primárnym štandardom používaným vo webových službách sa stal "rozšíriteľný značkovací jazyk" - Extensible Markup Language, známy pod skratkou XML.[4]

[4] Guru99 Elearning [Online]. Dostupné na internete: < https://www.guru99.com/web-service-architecture.html>. [12.2.2019].

Na výmenu údajov vo formáte XML medzi aplikáciami používajú webové služby protokol SOAP (Simple Object Access Protocol). Takéto údaje voláme SOAP správy a sú prenášané prostredníctvom bežného prenosového protokolu HTTP. Keďže správy sú vo formáte XML, aplikácia na strane klienta môže byť napísaná v akomkoľvek programovacom jazyku.

1.2.HTTP

HTTP je protokol, ktorý používame na prenos dát. Definuje komunikáciu medzi klientom a serverom špecifikovaním spôsobu, ako majú klient a server navzájom interagovať. Definuje požiadavku (request) vykonanú klientom, ktorá obsahuje URL a príkaz (command) pre server na interpretovanie a poslanie odpovede.

Webové aplikácie, ktoré sú založené na protokole HTTP sú bezstavové a nie je teda nutné neustále pripojenie medzi klientom a serverom. Práve kvôli tejto vlastnosti je HTTP vhodným protokolom pre konfigurácie s firewallom. Pri nedostupnosti servera spracúvajúceho prvotnú požiadavku klienta, môžu byť nasledujúce požiadavky presunuté na iný server bez toho, aby sa o ne klient staral.[5]

Z pohľadu našej práce je dôležité pochopiť HTTP požiadavkám. HTTP požiadavka vždy obsahuje sloveso popisujúce serverovej strane činnosť, ktorú má vykonať, s údajmi mu poskytnutému.

Požiadavka na získanie zdroja GET je používaná v prípade záujmu klienta tento zdroj čítať, bez potreby jeho úpravy. Ako príklad môžeme uviesť načítanie stránky s kontaktnými údajmi na webovom sídle spoločnosti.

Sloveso POST v HTTP požiadavke identifikuje potrebuje vloženia, respektíve odoslania údajov zdroju a ich uloženie. Ako príklad uvedieme vyplnenie kontaktných údajov na registračnej stránke.[6]

---

[5] SHORT, Scott. *Building XML Web Services for the Microsoft .NET Platform.* Redmont: Microsoft Press, 2002. s. 6-7 ISBN 0-7356-1406-7

[6] HUME, Dean Allan. *Fast ASP.NET websites*. New York: Manning Publications Co. 2013, s. 12. ISBN 9781617291258

1.3.XML

Prenosové protokoly HTTP a SMTP nám umožňujú odosielať údaje medzi klientom a serverom, nehovoria nám však, v akom tvare by tieto údaje mali byť. Pri návrhu webových služieb potreboval Microsoft zvoliť platformovo neutrálny spôsob kódovania údajov prenášaných medzi klientom a serverom. Keďže jednou z podmienok pri návrhu fungovania webových služieb bolo využitie existujúcich internetových štandardov, optimálnou voľbou bol jazyk XML.

XML je skratka pre rozšíriteľný značkovací jazyk (Extensible Markup Language). Jazyk bol definovaný World Wide Web Konzorciuom XML môžeme použiť na definovanie vlastných prvkov, značiek a tak si vytvoriť značkovací jazyk prispôsobený presne pre naše potreby. Pri iných značkovacích jazykoch ako je napríklad HTML (Hypertext Markup Language) sú všetky značky preddefinované. XML voláme aj meta-značkovacím jazykom, keďže nám umožňuje vytvárať si vlastné značky.

XML je veľmi silný pri použití s webom, ktorý poskytuje protokoly na pohyb údajov. XML definuje, ako majú tieto údaje vyzerať. Stal sa nespochybniteľným štandardom pre štrukturovanie údajov, ktoré chceme zdieľať. Môže byť použitý mnohými spôsobmi, od organizovania údajov v jednoduchých súboroch, až po komplexnejšie úlohy, ako je posielanie informácii z jedného programu alebo procesu ďalšiemu.

Stal sa bežným dátovým formátom pre spoločností, ktoré si potrebujú vymieňať dáta. Webové služby ho používajú na zakódovanie správ a údajov spôsobom nezávislým od platformy, čoho podstatou je jeho multiplatformový charakter.[7]

Jednou z výhod XML je vyhnutie sa problémom s binárnym kódovaním, pretože používa kódovanie na báze textu.[8]

---

[7] THANGARATHINAM, Thiru. *Professional ASP.NET 2.0 XML.* Indianapolis: Wiley Publishing, 2006. s. 1 ISBN 978-0-7645-9677-3

[8] SHORT, Scott. *Building XML Web Services for the Microsoft .NET Platform.* Redmont: Microsoft Press, 2002. s. 6-7 ISBN 0-7356-1406-7

Súbory typu XML môžu byť prenášané medzi dvoma aplikáciami v sieti. Vyzerajú ako čisté, textové dokumenty a obsahujú špeciálne značky, ktoré slúžia na rozlišovanie rôznych častí dokumentu, respektíve rôznych položiek a údajov.[9]

Obrázok č. 2: Príklad štruktúry údajov v XML súbore

```
<bookstore>
 <book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
 </book>
 <book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
 </book>
 <book category="WEB">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
 </book>
</bookstore>
```

Zdroj: https://www.javatpoint.com/xml-example

Na vyššie uvedenej ukážke vidíme, ohraničenie párom značiek pre každý údaj, napríklad <author></author>. Vďaka takejto štruktúre dokumentu môžeme jednoducho identifikovať oblasti údajov, ich začiatky aj konce.

1.4. Vznik XML

Vývoj XML sa datuje na rok 1996. Požiadavka na nový značkovací jazyk vznikla kvôli nutnosti zjednodušiť SGML (Standard Generalized Markup Language), štandardný všeobecný značkovací jazyk. Inšpiráciou na vznik nového jazyka bol aj úspech HTML.

[9] MONSON-HAEFEL, Richard. *J2EE™ Web Services.* Boston: Adison-Wesley. 2003. s. 27-28 ISBN 0-321-14618-2

XML vytvorila skupina expertov na značkovacie jazyky pod hlavičkou spoločnosti Sun Microsystems. Štandardizačný proces malo na starosti konzorcium W3C. XML sa stal základom aj pre ostatné technológie webových služieb.[10]

1.5. Výhody XML

Značkovací jazyk XML je použiteľný v mnohých oblastiach. Tu je sumár niektorých výhod, vďaka ktorým sa XML stal akceptovateľným a dostal sa do organizácii.

- Súbory XML môže čítať človek. Boli navrhnuté ako text, preto ich v najhoršom prípade môže prečítať človek a získať ich obsah, čo nie je možné pri binárnych dátových formátoch.

- Má širokú podporu medzi organizáciami a odvetviami. Existuje množstvo nástrojov pre webové prehliadače, databázy, operačné systémy, ktoré umožňujú exportovať údaje vo formáte XML. XML je podporované naprieč celým .NET Frameworkom.

- Väčšina relačných databázových serverov, ako napríklad SQL Server majú prirodzenú schopnosť ukladať, čítať a generovať XML údaje.

- Existuje veľká skupina technológii na interpretáciu a transformáciu XML údajov.

1.6. JSON

Formát JSON (JavaScript Object Notation) je jednoduchým formátom pre výmenu údajov. Rovnako ako XML ho môže človek jednoducho čítať a stroj jednoducho generovať a parsovať .[11]

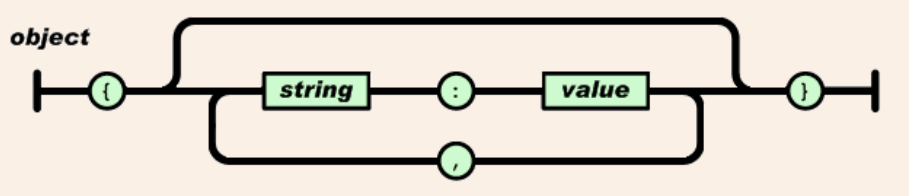
JSON je formátom nezávislým na programovacom jazyku, ale používa konvencie dobre známe z jazykovej rodiny C, aj preto je ideálnym na dátovú výmenu.

[10] BERKLEY [Online]. Dostupné na internete: <http://courses.ischool.berkeley.edu/i290-14/s05/lecture-2/slide2.xhtml>. [20.3.2019].
[11] JSON [Online]. Dostupné na internete: <https://www.json.org>. [20.3.2019].

JSON pracuje s dvoma štruktúrami:

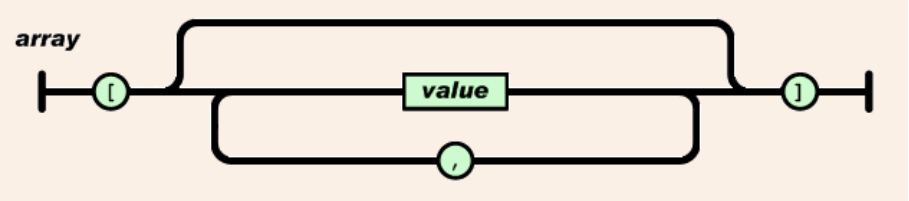
- kolekcie párov mien a hodnôt, ktoré sú v programovacích jazykoch realizované napríklad ako objekty,
- zoradené zoznamy hodnôt, v programovacích jazykoch realizované napríklad ako listy a polia.

Obrázok č. 3: Schéma objektu v JSON



Zdroj: https://www.json.org

Obrázok č. 4: Schéma poľa v JSON



Zdroj: https://www.json.org

Ukážka údajov vo formáte JSON:

[{"RealId":0,"Name":"Krasna rodinna vila s dvorom a fontanou","Size":555,"Category":"dom","Price":120000,"Location":"Levice","Age":20,"Rooms":7,"Isolation":"ano","TimeOfCreation":"2019-05-01T00:00:00+02:00"}]

1.7.Porovnanie JSON a XML

JSON a XML sú formáty slúžiace podobnému účelu. Umožňujú nám získavať dáta z webového servera. Oba jazyky:

- sami seba popisujú a preto ich môže človek jednoducho čítať,
- dáta v v nich zapísané majú svoju hierarchickú štruktúru,
- sú vhodné pri práci s rôznymi programovacími jazykmi,

- môžu byť prevzaté cez požiadavku XMLHttpRequest.

Na rozdiel od XML, JSON:

- nepoužíva značky,
- je kratší,
- rýchlejšie sa číta aj píše,
- umožňuje prácu s poľami,
- môže byť parsovaný štandardnou JavaScriptovou funkciou, na rozdiel od XML, ktorý môže byť parsovaný len XML parserom.

Samotná iniciatíva, ktorá stojí za formátom JSON nazýva svoj formát "nízkotučným XML". V prospech JSON argumentuje jednoduchosťou, nemožnosťou rozšírenia, ktoré nie je potrebné, keďže nejde o značkovací jazyk, rovnakou interoperabilitou a otvorenosťou na úrovni XML, ak nie lepšou, keďže nie je v centre korporátnych sporov.[12]

1.8.SOAP

SOAP je v zásade protokol pre volanie a definovanie jednotlivých služieb. SOAP (skrátene Simple Object Access Protocol) definuje štandardný formát balíka na prenášanie XML dát medzi aplikáciami v sieti. Nie je špecifikovaný pre žiadny konkrétny programovací jazyk, produkt ani hardvérovú platformu, tak môže byť použitý pre akýkoľvek druh aplikácie (C++, .NET, Java aplikácie a iné).

SOAP je XML dokument. Je špeciálne navrhnutý aby mohol obsahovať a prenášať iné XML dokumenty ako aj informácie vo vzťahu k smerovaniu, spracovaniu, bezpečnosti, transakciám a k iným kvalitám služby.

Dôležitá skutočnosť, ktorej musíme porozumieť je, že SOAP správy slúžia ako obálka (network envelope) na výmenu XML dokumentov a dát. Jednotný štandard na balenie a výmenu XML dát robí webové služby viac interoperabilné a zrozumiteľnejšie, jednoduchšie implementovateľné.

[12] JSON [Online]. Dostupné na internete: <https://www.json.org/xml.html>. [20.3.2019].

Pred existenciou SOAP protokolu by sme boli odkázaný na individuálne (custom) riešenia, aby sme prinútili dva XML systémy komunikovať. Neboli totiž používané žiadne spoločné protokoly pre zosieťovanie, adresovanie alebo smerovanie.[13]

1.9. WSDL

Aby sme mohli pracovať so SOAP správami konkrétnej webovej služby, musíme vopred vedieť, ako sú správy štrukturované a aká je adresa umiestnenia webovej služby. Pre tento účel si komunita vývojárov webových služieb osvojila WSDL (Web Services Description Language), popisný jazyk webových služieb.

WSLD exaktne definuje:

- formát správy,
- internetový protokol,
- adresu, cez ktorú klient môže komunikovať s webovou službou.

WSDL má rovnako ako SOAP širokú podporu v organizáciách, osvojili si ho najuznávanejšie štandardizačné organizácie ako W3C, WS-I, OASIS.

Ďalšou z výhod WSDL je jeho použiteľnosť v generátoroch kódu, ktoré môžu prečítať WSDL dokument a na základe neho pripraviť programovacie rozhranie na prístup k webovej službe. Obľúbenými generátormi kódu sú napríklad IBM WebSphere, Apache Axis, Microsoft .NET.

Nevýhodou WSDL je jeho pomerná komplikovanosť, je veľmi komplexný, nakoľko ho jeho návrhári chceli využiť ako jazyk definujúci rozhrania (Interface Definition Language - IDL) pre webové služby nezávislé od rozhrania, jazyka ani operačného systému.

[13] MONSON-HAEFEL, Richard. *J2EE™ Web Services.* Boston: Adison-Wesley. 2003. s. 28-29 ISBN 0-321-14618-2.

Je dôležité poznamenať, že WSDL nie je naviazaný na prácu so SOAP správami a môže byť preto použitý aj pre webové služby, ktoré nepoužívajú SOAP. Alternatívou sú webové služby používajúce REST (Representational State Transfer).[14]

1.10. REST webové služby

RESTové webové služby sú alternatívou k webovým službám používajúcim protokol SOAP, ktorý bol do nedávna dominantným prístupovým protokolom. REST (Representational State Transfer) sa rýchlo dostal do popredia a podľa článku publikovaného na server Stormpath.com koncom roka 2016 reprezentoval až 70% verejných aplikačných programových rozhraní (API).

Dôvodom obľúbenosti REST na úkor SOAP je jeho veľmi dobrá pochopiteľnosť. Používa základné HTTP CRUD operácie ako GET, PUT, POST, DELETE a veľmi dobrá kooperácia s jazykom JSON.

Na rozdiel od SOAP je REST bezstavový a umožňuje cacheovanie pre lepší výkon a škálovateľnosť.[15]

1.11. Vývojová platforma .NET Framework

Platforma na vývoj .NET aplikácií, pozostáva z nástrojov, programovacích jazykov a knižníc dôležitých pre vývoj webových, desktopových, mobilných, herných aplikácií alebo aplikácií pre internet vecí.

Platforma .NET Framework oficiálne podporuje nasledovné jazyky:

- C#
- F#
- Visual Basic
- C++.NET

[14] MONSON-HAEFEL, Richard. *J2EE™ Web Services.* Boston: Adison-Wesley. 2003. s. 141 ISBN 0-321-14618-2.
[15] Stormath [Online]. Dostupné na internete: <https://stormpath.com/blog/rest-vs-soap>. [2.4.2019].

Existujú však aj stovky ďalších jazykov, ktoré vytvorili vývojári tretích strán.

Nezávisle na jazyku funguje kód natívne na akomkoľvek z podporovaných operačných systémov. Zabezpečujú to .NET implementácie:

- .NET Core
- .NET Framework
- Xamarin/Mono[16]

1.11.1. Common Language Runtime (CLR)

Pre exekúciu .NET aplikácií poskytuje .NET Framework vykonávacie prostredie zvané CLR (Common Language Runtime).

.NET CLR je vykonávacie prostredie, ktoré riadi a spúšťa kód, pri čom ho prekladá do stroju prirodzeného jazyka tak aby ho mohol procesor spracovať.[17]

1.11.2. ASP.NET

Prvé platformy pre webový vývoj mali dva zásadné problémy. Prvým bola slabá škálovateľnosť. Stránky mali problémy s rýchlosťou pri nápore používateľov havarovali. Druhým problémom bolo, že poskytovali len "holú kostru" programovacieho prostredia, takže pri potrebe zložitejšej funkcionality, ako je autentifikácia alebo práca s databázami bolo potrebné písať dlhý a komplikovaný kód od začiatku.

Ako riešenie týchto problémov predstavila spoločnosť Microsoft vývojovú platformu vyššej úrovne, ktorú nazvali ASP (Active Server Pages), neskôr ASP.NET. Tieto technológie umožnili vývojárom tvoriť dynamické webové stránky bez starostí s implementovaním nízkoúrovňových funkcionalít. ASP.NET priniesol množstvo sofistikovaných možností ako sú nástroje na implementáciu bezpečnosti, manažmentu údajov, uchovávania špecifických informácií o používateľoch a mnoho iného.

---

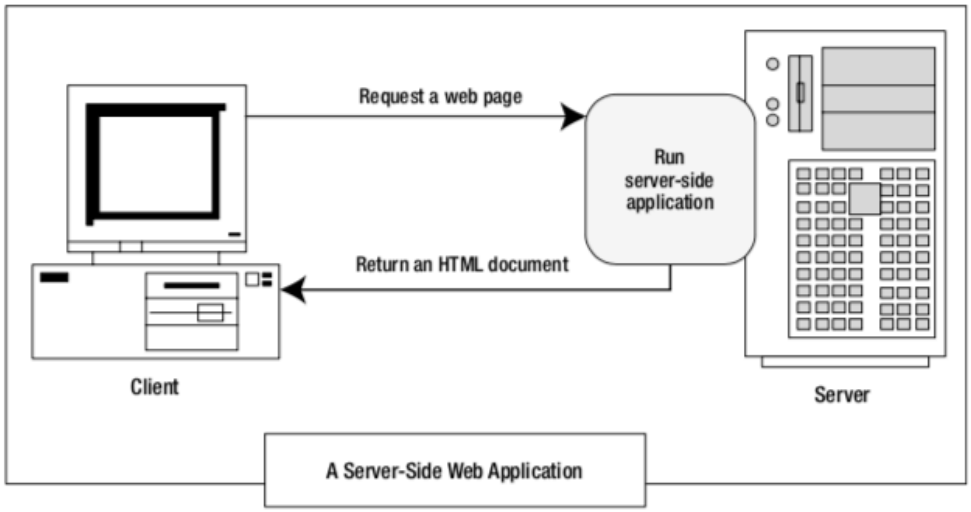
[16] Microsoft [Online]. Dostupné na internete: <https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet>. [2.4.2019].

[17] Javapoint. [Online]. Dostupné na internete: <https://www.javatpoint.com/net-common-language-runtime>. [2.4.2019].

ASP.NET je navrhnutá najmä ako platforma na strane servera. To znamená, že kód sa vykonáva na webovom serveri a keď skončí, webový server pošle klientovi výsledok, zväčša HTML stránku zobrazenú webovým prehliadačom.

Nezáleží na tom, aké sú možnosti prehliadača, kód bude vždy vykonaný na strane servera.[18]

Obrázok č. 5: Aplikácia fungujúca na strane servera



Zdroj: Begining ASP.NET 4.5 in C#

ASP.NET je súčasťou .NET Frameworku a slúži na vývoj webových aplikácií používajúcich HTML, CSS a JavaScript. Je podporovaný na operačných systémoch macOS, Microsoft Windows aj Linux. Ponúka tri frameworky:

- Web Forms
- ASP.NET MVC
- ASP.NET Web Pages

Aplikácie ASP.NET majú plný, neobmedzený prístup ku knižniciam tried .NET Frameworku.[19]

[18] MACDONALD, Matthew. *Begining ASP.NET 4.5 in C#.* New York: Apress, 2012. s. 6-7 ISBN 978-1-4302-4251-2

[19] Microsoft [Online]. Dostupné na internete: <https://dotnet.microsoft.com/learn/web/what-is-aspnet>. [2.4.2019].

1.11.3. Rozdiely medzi .NET a ASP.NET

Tabuľka č. 1: Porovnanie .NET a ASP.NET

.NET	ASP.NET
Je vývojový a exekučný framework	Je hlavnou súčasťou .NET Frameworku, ktorá zjednodušuje vytváranie, ladenie a nasadzovanie webových aplikácií.
Podporuje vývoj na strane servera aj na strane klienta.	Je plne integrovaný pre vývoj webových aplikácií na strane servera.
Kód môže byť napísaný v akomkoľvek jazyku, ktorý má CIL (Common Intermediate Language) kompilátor.	Používa sa na vývoj dynamických webových stránok použitím podporovaných .NET jazykov.[20]

Zdroj: http://www.differencebetween.net/technology/difference-between-net-and-asp-net/

1.11.4. ASP.NET Core

Microsoft v roku 2015 aktualizoval a nastavil nové smerovanie pre ASP.NET. ASP.NET Core je postavený na multiplatformovej verzii .NET Frameworku bez aplikačných programovacích rozhraní (API) charakteristických pre Windows.

Aj keď je operačný systém Windows stále najpoužívanejším operačným systémom, webové aplikácie sú stále častejšie umiestňované na cloudových platformách, čo vyžaduje multiplatformový prístup. Microsoft tak rozšíril možnosti ASP.NET a umožnil tak umiestnenie ASP.NET Core aplikácií v početnejšom množstve prostredí. Umožnil aj vývoj aplikácií pod operačným systémom Linux a MacOS.

[20] Diferencebetween [Online]. Dostupné na internete: <http://www.differencebetween.net/technology/difference-between-net-and-asp-net/>. [4.4.2019].

V porovnaní s ASP.NET je ASP.NET Core úplne nový framework, ktorý je jednoduchší a ľahšie sa s ním pracuje. [21]

1.11.5. Typy súborov v ASP.NET

V tabuľke sa nachádza niekoľko súborových typov, s ktorými sa pri vývoji v ASP.NET môžeme stretnúť najčastejšie.

Tabuľka č. 2: Typy súborov v ASP.NET

Názov súboru alebo koncovka	Popis
.aspx	ASPX sú webstránky v ASP.NET. Používatelia posielajú požiadavku na zobrazenie týchto stránok aby spustili webovú aplikáciu. Obsahujú užívateľské rozhranie a základný kód aplikácie.
.ascx	Ide o užívateľské ovládacie prvky, podobné webovým stránkam. Rozdielom je, že užívateľ nemôže pristupovať k súborom priamo. Namiesto toho musia byť súbory umiestnené vo vnútri ASP.NET webstránky. Umožňujú nám naprogramovať užívateľské rozhrania a použiť ich v množstve webových formulárov, čo zabraňuje opakovaniu sa kódu.
web.config	Konfiguračný súbor pre ASP.NET aplikáciu, obsahuje mnohé nastavenia, napríklad pre bezpečnosť, manažment stavov alebo pamäte.
global.asax	Globálny aplikačný súbor môže slúžiť na definovanie globálnych premenných alebo na interakciu s globálnymi udalosťami.

[21] FREEMAN, Adam. *Pro ASP.NET Core MVC.* Sixth Edition. New York: Apress, 2016. s. 5 ISBN 978-1-4842-0398-9.

.cs	Súbory, ktoré obsahujú kód v jazyku C#. Umožňujú nám oddeliť aplikačnú logiku od používateľského rozhrania.[22]

Zdroj: https://www.techrepublic.com/article/make-sense-out-of-the-confusing-world-of-net-file-types/

1.12. Programovacie jazyky a techniky pre stranu klienta

Cieľom webového klienta je používať webovú službu. Pod pojmom používanie webovej služby rozumieme naplnenie požiadavky klienta webovou službou.

Spúšťačom používania webovej služby je koncový používateľ webovej stránky, ktorý použil webovú aplikáciu, ktorá môže pristupovať k webovej službe. Webový klient vyšle požiadavku na koncové body webovej služby.

Ako príklad môžeme uviesť načítanie stránky o počasí webovým prehliadačom (koncovým užívateľom). Po načítaní webstránky je vykonaná požiadavka smerom k webovej službe na získanie údajov o aktuálnom počasí. Koncový používateľ o tejto aktivite ním načítanej webstránky nemusí nutne vedieť.[23]

Mohamed Azam z Codeproject.com hovorí: *Na používanie webovej služby môžeme použiť akýkoľvek programovací jazyk alebo platformu a práve to je podstata XML webových služieb.*[24]

[22] Techrepublic [Online]. Dostupné na internete: < https://www.techrepublic.com/article/make-sense-out-of-the-confusing-world-of-net-file-types/>. [15.4.2019].
[23] Microsoft [Online]. Dostupné na internete: <https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-netod/4950065f-7a79-4021-85e6-33ee9b5e91ba>. [15.4.2019].
[24] MOHAMED, Azam [Online]. Dostupné na internete: < https://www.codeproject.com/Articles/8257/How-to-Make-a-Simple-WebService-and-Consume-It >. [15.4.2019].

1.12.1. HTML

HTML (Hypertext Markup Language) je hypertextový značkovací jazyk, ktorý používame na tvorbu webových stránok. Pomocou značiek popisuje ich štruktúru a vytvára stavebné bloky HTML stránok.

Webový prehliadač HTML značky nezobrazuje ale používa ich na zobrazovanie obsahu stránky. Najnovšou verziou HTML je verzia HTML 5 predstavená v roku 2014.

1.12.2. JavaScript

Koncom minulého storočia predstavila spoločnosť Netscape nový jazyk s názvom LiveScript. Po popularizácii programovacieho jazyka Java sa LiveScript z marketingového hľadiska rozhodla premenovať na JavaScript aj napriek tomu, že s jazykom Java veľa spoločného nemá.[25]

JavaScript je najrozšírenejším scriptovacím jazykom na svete s najväčším ekosystémom knižníc. Zároveň je jediným programovacím jazykom, ktoreho aplikácie môžu byť vykonávané vo všetkých webových prehliadačoch. Medzi najrozšírenejšie JavaScriptové frameworky patria jQuery, nodeJS, Angular JS, React JS a iné.

Najväčšou výhodou JavaScriptového kódu je fakt, že o jeho vykonávanie sa stará webový prehliadač na počítači používateľa stránky, teda jej návštevníka. JavaScriptový kód funguje mimo servera, na ktorom je stránka uložená.[26]

[25] ŠKULTÉTY, Rastislav. *JavaScript: Programujeme internetové aplikace.* 2. Aktualizované vydání. Brno: Computer Press, 2004. s. 5 ISBN 80-251-0144-4.

[26] Freecodecamp [Online]. Dostupné na internete: <https://guide.freecodecamp.org/javascript>. [16.4.2019].

1.12.3. jQuery

John Resig v roku 2006 predstavil knižnicu pre jazyk JavaScript s názvom jQuery. Knižnica je kompatibilná naprieč prehliadačmi a bola navrhnutá aby zjednodušila život vývojárom HTML aplikácií.

JQuery je najpopulárnejšou JavaScriptovou knižnicou na svete, vďaka jednoduchému spracovávaniu udalostí, manipuláciou s DOM (dokumentovým objektovým modelom) a podporou pre AJAX a animácie.[27]

1.12.4. AJAX

Moderné webové aplikácie používajú súbor návrhových prístupov a technológií známy ako Ajax. Ajax je sktratka reprezentujúca súbor techník pre vývoj responzívnejších, dynamických stránok. Doslovne skratka reprezentuje Asynchrónny JavaScript a XML.
Najznámejšou výhodou Ajaxu je schopnosť aktualizovať určitú časť stránky pri zachovaní ostatných častí bez zmeny, respektíve bez nutnosti aktualizovať celú stránku.

AJAX nám umožňuje:

- aktualizovať webovú stránku bez nutnosti ju opätovne načítať,
- vyžiadať si údaje zo servera potom, ako bola stránka načítaná,
- prevziať údaje zo servera po načítaní stránky,
- na pozadí odoslať údaje serveru.

Aj keď Ajaxové stránky používajú výrazne viac JavaScriptového kódu ako je bežné, často je vyžadovaná interakcia medzi ovládačmi, ktoré si často vyžadujú dodatočné informácie z webového servera použitím špeciálneho objektu nazvaného XMLHttpRequest dostupného na klientskej strana JavaScriptového kódu.

[27] ERASMUS, Michael. *CoffeeScript Programming with jQuery, Rails, and Node.js*. Birmingham: Packt Publishing, 2012. s. 55 ISBN 978-1-84951-958-8.

1.12.5. AJAX v ASP.NET

Aby sme mohli implementovať AJAX v ASP.NET aplikáciách, musíme mať dôkladnú znalosť JavaScriptového kódu. Práve JavaScriptový kód sa vykonáva vo webovom prehliadači a posiela požiadavku na novú informáciu z webového servera a podľa nej aktualizuje stránku.

Správne programovanie v jazyku JavaScript je náročné, pretože správna implementácia kódu sa môže líšiť v závislosti od webového prehliadača. Na správne fungovanie stránky na každom prehliadači preto potrebuje vývojár mnoho skúseností alebo musí použiť knižnicu vyššej úrovne, ako je jQuery.

Ďalším problémom JavaScriptu je jeho prílišná voľnosť. Ignoruje menšie preklepy a chyby v písaní kódu, preto je ich hľadanie a odstraňovanie nepríjemný proces, navyše sa ladenie kódu môže líšiť v závislosti od prehliadača.

V ASP.NET existuje model vyššej úrovne s názvom ASP.NET AJAX. Model poskytuje vývojárom súpravu komponentov pre stranu servera, a ovládacích prvkov, ktoré môžeme použiť pri návrhu webovej stránky. JavaScript je za pomoci týchto komponentov zobrazený pre dosiahnutie požadovaného efektu. Výsledkom je možnosť použiť Ajaxové efekty pri programovaní modelu objektov na strane servera.

Pri použití ASP.NET AJAX máme k dispozícii dôležitú funkcionalitu s minimálnym úsilím.[28]

[28] MACDONALD, Matthew. *Begining ASP.NET 4.5 in C#.* New York: Apress, 2012. s. 793 ISBN 978-1-4302-4251-2.

2. Cieľ práce, metodika práce a metódy skúmania

Cieľom diplomovej práce je vytvoriť aplikáciu vhodnú pre použitie v segmente realitných kancelárií, ktorá bude spĺňať požiadavky na evidovanie obchodných položiek, respektíve realít v portfóliu realitnej kancelárie. Pri vývoji aplikácie zohľadníme vlastné poznatky z daného odvetvia ale aj skutočné informácie od realitných agentúr a realitných sprostredkovateľov, ktoré získame osobnou konzultáciou. Takto získané informácie popíšeme a zhodnotíme. V závere práce porovnáme fungovanie nášho produktu s produktami alebo zaužívanými postupmi na trhu.

Softvér, ktorý vytvoríme by mal fungovať ako ASP.NET XML webová služba, ktorá bude pracovať s údajmi vo formáte XML. Webová služba musí vedieť údaje prijať, uchovať a vrátiť ich vo vhodnom formáte tak, aby ich webový klient vedel ďalej spracovať a usporiadane ich zobraziť v logických súvislostiach. Výstupným formátom údajov z webovej služby bude formát JSON (JavaScript Object Notation).

Webová služba je najdôležitejšou časťou našej práce, preto je dôležité, aby fungovala nezávisle na vývojovom prostredí Microsoft Visual Studio 2017 Enterprise a jej aplikačná logika bola dostupná nepretržite. Kompilovanú webovú službu preto umiestnime na vhodný server.

Vstupné údaje bude môcť používateľ do systému pridať manuálne, alebo ich bude možné nahrať zo zdrojového súboru. Zdrojový súbor s údajmi vo formáte XML musí mať vhodne zvolenú štruktúru. Webová služba bude vedieť z tejto štruktúry údaje načítať a ďalej s nimi pracovať. Okrem iného webová služba umožní výsledky práce uložiť do súboru identickej štruktúry ako je súbor používaný na nahratie údajov do systému. To zabezpečí používateľovi softvéru možnosť opätovného nahratia spracovaných údajov do systému.

2.1. Požiadavky na webovú službu

Webová služba musí byť navrhnutá tak, aby spĺňala ciele našej práce a dokázala poskytnúť jej budúcim používateľom, realitným maklérom alebo administratívnym pracovníkom realitných kancelárií, potrebnú funkcionalitu. Za týmto účelom musí byť schopná spracovať minimálne nasledovné úlohy:

- pridávanie položiek do evidencie,
- zobrazenie položiek v evidencii,
- zobrazenie špecifických položiek z evidencie,
- zobrazenie položiek zaevidovaných v konkrétnom rozsahu dátumov,
- zobrazenie položky alebo položiek pre uspokojenie najčastejších potrieb zákazníka. Predpokladáme, že najdôležitejšími faktormi pri výbere nehnuteľnosti sú typ nehnuteľnosti, lokalita, počet izieb a rozpočet zákazníkovej investície.

2.2. Metodika práce

Na naprogramovanie našej ASP.NET XML webovej služby poskytujúcej usporiadané informácie o realitách realitnej kancelárie sme použili integrované vývojové prostredie (IDE) od spoločnosti Microsoft, Visual Studio Enterprise 2017 s licenciou poskytnutou Ekonomickej univerzite v Bratislave, ktoré poskytuje sadu nástrojov potrebnú pre vývoj ASP.NET XML webových služieb, ktorú Microsoft na svojich webstránkach veľmi dobre popisuje.

Na zoznámenie sa s vývojovým prostredím Visual Studio sme použili zjednodušenú bezplatnú verziu Visual Studio Code, ktorá je dostupná aj na MAC OS, s ktorým sme čiastočne pracovali. Samotná webová služba bola vyvíjaná na operačnom systéme Windows 10.

Aplikačná logika našej webovej služby je napísaná v objektovo orientovanom jazyku C#.

Visual Studio používa pri zostavovaní webových služieb zjednodušenú verziu internetového informačného HTTP servera optimalizovaného pre vývojárov, IIS Express. Testovacie rozhranie webovej služby sa spúšťa v primárnom prehliadači zvolenom v operačnom systéme. Pri vývoji sme pracovali s prehliadačom od spoločnosti Google a preto webová služba testovala v prehliadači Google Chrome.

Na úpravu obrázkov a tvorbu vizuálov bola použitá aplikácia Pixelmator, ktorá je cenovo výrazne dostupnejším substitútom k známejšiemu Adobe Photoshop.

Táto práca bola napísaná pomocou Dokumentov Google a Microsoft Word ako súčasť Office 365.

Nami vytvorená webová služba pozostáva z troch tried, obsahuje 9 vystavených (webových) metód a 12 servisných (newebových) metód, ktorých fungovanie zhodnotíme vo výsledkoch práce.

2.3. Metódy skúmania

2.3.1. Metóda abstrakcie

Pri písaní práce sme abstrahovali od zložitejších technických podrobností a snažili sme sa dôkladne vysvetliť to najpodstatnejšie, a to logické fungovanie nami navrhnutého softvéru. Problematika webových služieb a vývoja softvéru ako takého je veľmi komplexná a rýchlo sa meniaca, preto sme sa sústredili na náš produkt a jeho využitie.

2.3.2. Metóda syntézy

V kapitolách výsledky práce a diskusia sme použili metódy syntézy na utvorenie záverov z prieniku teórie spracovanej v teoretickej časti práce a praxe spracovanej v praktickej časti práce. Na ich základe sme zhodnotili možnosti komerčného použitia nami vytvorenej webovej služby.

3. Súčasný stav evidencie realít vo vybraných firmách

Za účelom zistenia aktuálneho stavu na trhu sme oslovili niekoľko menších realitných kancelárií. Popísali nám akým spôsobom evidujú inzeráty a s akými rôznymi systémami a komerčnými produktami pri práci s obchodnými položkami pracujú.

Informácie o správaní sa konkurenčných firiem sú v trhovej ekonomike veľmi dôležité a poznanie aktuálneho stavu nám poskytuje konkurenčnú výhodu pri tvorbe alternatívneho produktu, ktorý bude používaný vo vlastnej realitnej kancelárii, no potenciálne môže byť použitý aj pri transformácii softvéru na komerčný produkt.

Firmy o ktorých správaní informujeme v tejto práci reálne podnikajú ako realitné kancelárie alebo realitní sprostredkovatelia. Ich názvy však z pochopiteľných dôvodov musíme anonymizovať.

Realitná kancelária Alfa1 je malá realitná kancelária pôsobiaca v nemenovanom meste. Sústreďuje sa hlavne na predaj bytov. Každý inzerát pridáva najskôr na svoju webstránku a následne na inzertné portály. Na webstránke vykonáva evidenciu inzerátov prostredníctvom CMS (Content Management System) Wordpress. Na CMS Wordpress je k dispozícii veľké množstvo šablón, jednou z nich je šablóna pre realitné kancelárie s názvom *Real Homes*, ktorú spoločnosť Alfa1 zakúpila na serveri *themeforest.net*. Výhodou je aj cena za licenciu, ktorá sa v dobe písania tejto práce pohybovala na úrovni 59 dolárov. Ponúka široké možnosti filtrácie pridaných položiek podľa rôznych parametrov a modifikácie realitných položiek. Realitná kancelária Alfa1 je so súčasným stavom evidencie inzertných položiek spokojná.

Realitná kancelária Alfa2 používa na evidenciu jednotlivých kontraktov a základných informácií o realitách bezplatný online softvér *hubspot.com*, ktorý je primárne určený ako CRM. Alfa2 ho využíva najmä na evidenciu klientov vo vytvorenej tabuľke kontaktov, ku ktorým v na mieru nadefinovaných stĺpcoch priraďujú základné informácie o nehnuteľnosti patriacej klientovi. Samotné inzeráty publikuje na vlastnej webstránke podobne ako kancelária Alfa1. Alfa2 však nepoužíva CMS Wordpress, ale na mieru navrhnuté, vlastné

CMS naprogramované v jazyku PHP. Webstránka a softvér *hubspot* nie sú navzájom prepojené.

Realitný sprostredkovateľ Beta1 je individuálny realitný agent pôsobiaci v menšom okresnom meste Levice. Jeho podnikanie je založené na širokom okruhu priateľov a dobrom mene. Keďže v meste nepôsobí priveľký počet maklérov a počet ponúkaných nehnuteľností najmä na trhu s prenájmom sa po rozšírení priemyselného parku zvýšil, svoju prácu definuje ako „plynulú obsluhu klientov s efektami sezónnosti".
Beta1 uzatvorí počas jedného mesiaca do 10 obchodov. Na evidenciu realitných inzerátov používa výhradne užívateľské kontá na inzertných portáloch. Vlastnú webstránku používa iba na sebaprezentáciu. Nepublikuje na nej inzeráty.
Beta1 zdôrazňuje potrebu spolupráce realitného agenta s inzertnými portálmi. Dobrá práca s inzerátmi, vďaka ktorým ho kontaktuje vyše 90% záujemcov je podľa Beta1 nevyhnutnosťou. V najbližšej dobe neplánuje zmenu systému práce ani zakúpenie akejkoľvek licencie na softvér. Odraďuje ho najmä práca navyše pri duplicitnej evidencii v rôznych systémoch.

Realitný sprostredkovateľ Beta2 používa na evidenciu klientov a stráženie termínov aktualizácie inzerátov Google tabuľky, bezplatnú alternatívu k tabuľkovému procesoru Excel od spoločnosti Microsoft. Rovnako ako iné oslovené firmy pracuje s realitnými inzertnými servermi. Za najpoužívanejšie inzertné webstránky pre publikáciu realitných ponúk označil:

- topreality.sk,
- nehnutelnosti.sk,
- reality.bazos.sk,
- reality.sk.

Beta2 má vlastnú webstránku, na ktorej publikuje iba inzeráty, ktoré podporuje reklamou na sociálnych sieťach alebo kontextovou reklamu vo vyhľadávačoch.

4. Výsledky práce

V kapitole Výsledky práce predstavíme vyvinutý softvér. Popíšeme jeho štruktúru, všetky dôležité časti, ako sú triedy a metódy. Dôraz bude kladený na podrobný opis webovej služby, ktorej vývoj je hlavným cieľom tejto práce.

4.1. Základná štruktúra

4.1.1. Štruktúra XML súboru

Používateľ nášho softvéru sa nemusí rozhodnúť pridávať všetky položky do systému len manuálne. Môže ich nahrať z XML súboru. Pre účely vývoja aplikácie sme navrhli XML dokument s pracovnými položkami.

Naše pracovné položky sa nachádzajú v elemente <ArrayOfItem>. Ich koreňovým elementom je element <item>, ktorý je rodičovským elementom pre ostatné elementy s parametrami nehnuteľnosti. Tieto parametre budú vysvetlené pri opise triedy Item. Celý zdrojový XML súbor je prílohou tejto práce.

Obrázok č. 6: Ukážka štruktúry zdrojového XML

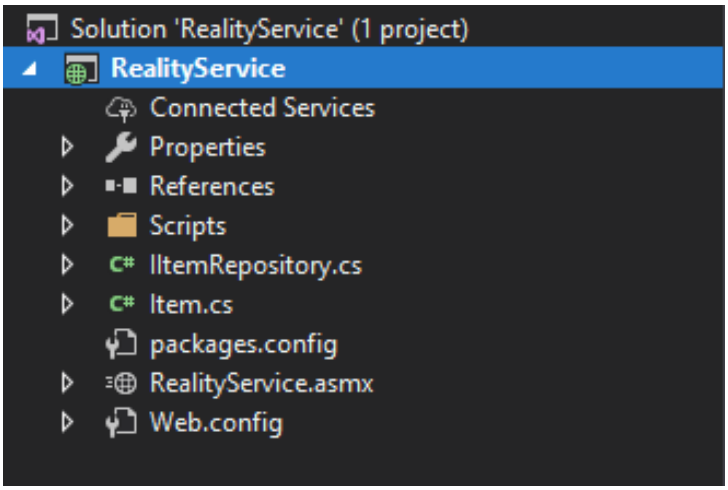
```
<Item>
  <RealId>15</RealId>
  <Name>Mala garzonka vhodna na AirBNB</Name>
  <Rooms>1</Rooms>
  <Category>byt</Category>
  <Price>60000</Price>
  <Location>Nitra</Location>
  <Size>16</Size>
  <Age>25</Age>
  <Isolation>ano</Isolation>
  <TimeOfCreation>2019-04-25T00:00:00+02:00</TimeOfCreation>
</Item>
```

Zdroj: Vlastné spracovanie

4.1.2. Štruktúra projektu webovej služby

Projekt našej webovej služby sa nachádza v riešení (solution) s názvom RealityService.sln

Obrázok č. 7: Solution explorer z Visual Studia



Zdroj: Vlastné spracovanie

Naša webová služba sa nachádza v súbore *RealityService.asmx*, v ktorom sa nachádza trieda *RealityService*. Telo triedy *RealityService* obsahuje webové metódy, ktoré bude klient vzdialene volať.

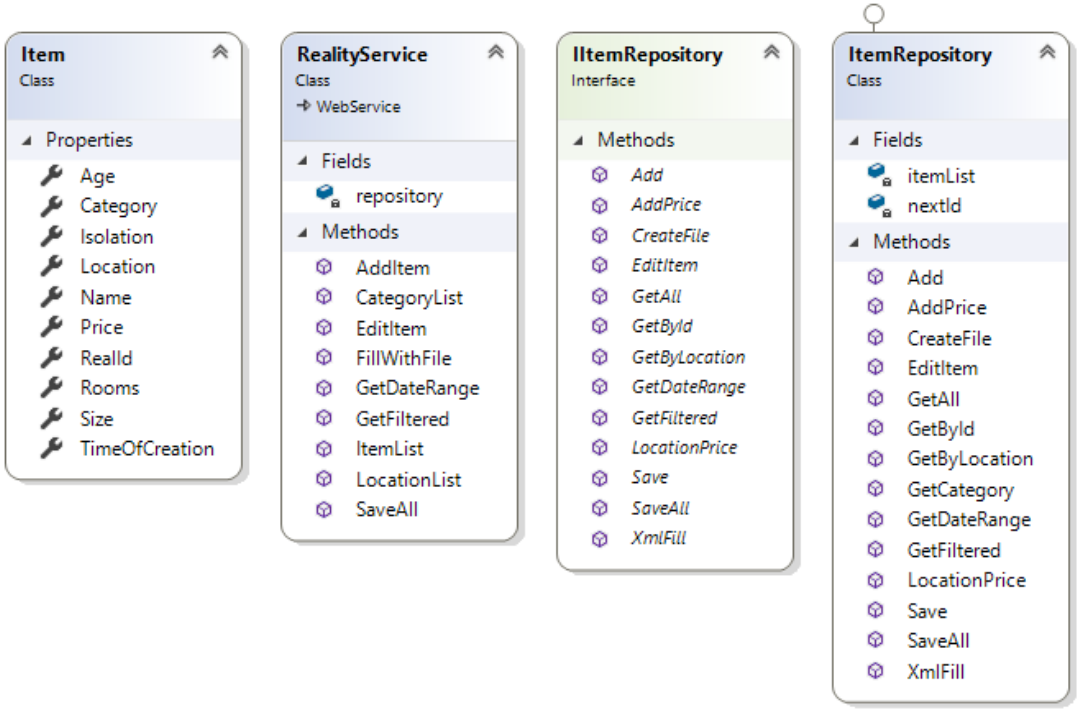
V tabuľke nižšie sa nachádza zoznam tried v projekte.

Tabuľka č. 3: Zoznam použitých tried

Názov triedy	Popis
RealityService	Trieda, ktorá obsahuje vystavené metódy webovej služby.
ItemRepository	Trieda, ktorá obsahuje servisné metódy webovej služby. Dedí z rozhrania *IItemRepository*.
Item	Obsahuje vlastnosti realitnej položky, ktoré budeme získavať metódou get a nastavovať metódou *set*.

Zdroj: Vlastné spracovanie

Obrázok č. 8: Diagram tried a ich metód vygenerovaný z Visual Studio 2017



Zdroj: Vlastné spracovanie

Projekt obsahuje 9 vystavených metód a 12 servisných metód, ktoré podrobne popíšeme v ďalších podkapitolách.

4.2. Metódy webovej služby

4.2.1. Vystavené metódy webovej služby

Vystavené metódy v kóde identifikujeme ako metódy s atribútom [WebMethod]. Môžeme ich volať z rozhrania webovej služby alebo vzdialene zo strany klienta. Návod na volanie zo strany klienta je popísaný priamo v rozhraní webovej služby.

Webová služba *KrajniakRealityService* má 9 vystavených metód. Slúžia na pridanie položky, úpravu položky, zobrazenie celého zoznamu realít, zobrazenie položiek pridaných v danom rozsahu dátumov, zobrazenie realít z danej kategórie, zobrazenie realít z danej lokality, zobrazenie špecifickej položky a naplnenie zoznamu realtít z XML súboru.

Prvou vystavenou metódou je metóda **AddItem**. Jej stručné fungovanie je popísané aj atríbútom *Description* atribútu [WebMethod]. Po zavolaní metóda pridá položku triedy *Item* do adresára, ktorým je *List<T>,* teda list objektov, v našom prípade ide o objekt typu *Item*. Kód vystavenej metódy *AddItem* vyzerá nasledovne:

```
[WebMethod(Description = "Prida polozku do zoznamu realit.")]
    public int AddItem(string name, string category, string location, int price, int age, int rooms, int size, string iso)
    {
        var item = new Item { Name = name, Category = category, Location = location, Price = price, Age = age, Rooms = rooms, Size = size, Isolation = iso, TimeOfCreation = DateTime.Now.Date };

        repository.Add(item);
        return item.RealId;
    }
```

Ako môžeme z kódu vyčítať, metóda pracuje so vstupnými parametrami dátového typu *string* v prípade parametrov *name, category, location, iso* a parametrami dátového typu *integer* pri zadaní vstupov *price, age, rooms a size*. Po prijatí vstupných parametrov metóda vytvorí lokálnu inštanciu triedy *Item* a priradí jej parametrom vstupné premenné. Parameter *TimeOfCreation* nie je získaný ako vstup metódy. Jeho definovanie prebehne vytvorením objektu typu *DateTime*, ktorého hodnota sa nastaví na aktuálny dátum na serveri, vyjadrený ako lokálny časový údaj, pomocou vlastnosti *DateTime.Now.Date*. Takto vytvorená lokálna inštancia objektu triedy *Item* je použitá ako vstupný parameter metóda *Add* triedy *ItemRepository*, ktorej fungovanie je definované v súbore *ItemRepository.cs*. Návratovou hodnotou vystavenej metódy je identifikátor aktuálne pridanej položky *RealId*, ktorý sa môže byť užitočný pri budúcom rozširovaní aplikácie.

EditItem je vystavená metóda, ktorá slúži na úpravu existujúcej položky v zozname realít. Logicky jej fungovanie môžeme popísať ako prepis prepis vlastností objektu triedy *Item*, nachádzajúceho sa v zozname objektov pod unikátnym identifikátorom *realId*. Metóda upraví všetky vlastnosti až na *TimeOfCreation*, ktorá nemôže byť z dôvodu pracovnej

transparentnosti zmenená. Manipulácia s časom pridania položky by totiž mohla spôsobiť konflikty medzi užívateľmi aplikácie.

```
[WebMethod(Description = "Upravi existujucu polozku v zozname realit.")]
    public void EditItem(int readlId, string newName, int newSize, string newCategory, int newPrice, string newLocation,
        int newAge, int newRooms, string iso)
    {
            repository.EditItem(readlId, newName, newSize, newCategory, newPrice, newLocation, newAge, newRooms, iso);
    }
```

Vystavená metóda posunie vstupné premenné metóde repository.EditItem, ktorá vykoná potrebnú logiku.

ItemList je vystavenou metódou zobrazujúcou kompletný zoznam existujúcich položiek v systéme. Existujúcimi položkami v systéme myslíme celý zoznam objektov triedy *Item*, ktorý je uložený v premennej typu *List<Item>* s názvom *itemList* v triede *ItemRepository*. Hodnotu tejto premennej vracia metóda s názvom *GetAll()*. Vstupné perametre pre vypísanie celého zoznamu objektov nie sú potrebné.

```
[WebMethod(Description = "Vylistuje vsetky reality v zozname.")]
    public void ItemList()
    {
        Context.Response.Write(repository.GetAll());

    }

    [WebMethod(Description = "Ulozi vsetky polozky do XML suboru.")]
    public string SaveAll(string path)
    {
        repository.SaveAll(path);
        return "hotovo";

    }
```

Vystavená metóda *ItemList* nemá návratovú hodnotu, jej úlohou je vypísať odpoveď na požiadavku, ktorou je obsah premennej *itemlist*, ktorý vracia metóda *GetAll()*.

SaveAll je jednoduchá vystavená metóda, ktorá uloží všetky nehnuteľnosti zo systému do XML súboru. Na tento úkon používa servisnú metódu *repository.SaveAll*, ktorej posúva parameter *path*.

GetDateRange zobrazí položky zo zoznamu realít patriace do zadaného rozsahu dátumov. Rozsah dátumov je definovaný vstupnými premennými typu *DateTime*, *DateTime low* a *DateTime high*.

```
[WebMethod(Description = "Zobrazi polozky, ktore splnaju rozsah datumov.")]
    public void GetDateRange(DateTime low, DateTime high, string path)
    {
       Context.Response.Write(repository.GetDateRange(low, high, path));
    }
```

Vstupnou metódou vystavenej metódy je aj *string path*. Ide o cestu k adresáru na serveri, kam má byť súbor s výstupnými údajmi uložený.

CategoryList po zavolaní zobrazí vybrané objekty zo zoznamu objektov, spĺňajúce podmienku špecifikovanej kategórie vo vstupnej premennej *string category*. Vstupná premenná *path* má vo všetkých metódach rovnaký účel. Ak je jej hodnota metóde odoslaná, logika metódy, s ktorou vystavená metóda pracuje spracuje výstup vo formáte XML do súboru a uloží ho na server.

```
    [WebMethod(Description = "Vylistuje polozky zo zadanej kategorie.")]
    public void CategoryList(string category, string path)
    {
       Context.Response.Write(repository.GetCategory(category, path));
    }
```

LocationList je takmer identickou vystavenou metódou ako *CategoryList*. Namiesto kategórie však pracuje s premennou *location*.

```
[WebMethod(Description = "Vylistuje polozky splnajuce lokalizacny parameter.")]

    public void LocationList(string location, string path)
    {
        Context.Response.Write(repository.GetByLocation(location, path));
    }
```

Vystavená metóda **FillWithFile** má za úlohu naplniť zoznam realít položkami z XML súboru.

```
[WebMethod(Description = "Naplni zoznam polozkami z xml suboru")]

    public void FillWithFile(string path)
    {
        Context.Response.Write(repository.XmlFill(path));
    }
```

K súboru vo formáte .xml pristupuje pomocou zadanej premennej *path*, typu *string*. Cestu k zdrojovému súboru posúva metóde *repository.XmlFill*, ktorá vykoná potrebnú logiku.

GetFiltered slúži na zobrazenie položky, ktorá spĺňa zadané vstupné parametre vystavenej metódy metódy. Umožňuje zobraziť položku s konkrétnou kategóriou, počtom izieb v konkrétnom meste. Cena požadovanej položky sa špecifikuje intervalom *minprice* a *maxprice*.

```
[WebMethod(Description = "Zobrazi polozku splnajucu zadane parametre. Filtruje.")]
    public void GetFiltered(string category, int room, string city, int minprice, int maxprice, string path)
    {
        Context.Response.Write(repository.GetFiltered(category, room, city, minprice, maxprice, path));
    }
```

Vystavená metóda posúva zadané parametre servisnej metóde *GetFiltered* a zobrazuje výstup. V prípade potreby uloženia výstupu do Xml súboru je možné definovať vstupnú premennú *path*.

4.2.2. Servisné metódy webovej služby

Z funkčného hľadiska sú servisné metódy, ktoré uchovávajú aplikačnú logiku omnoho zaujímavejšie ako vystavené metódy v *.asmx* súbore, preto ich fungovanie popíšeme podrobnejšie.

Webová služba *RealityService* má 12 servisných metód.

Pre ich správne pochopenie si musíme popísať návrh triedy *ItemRepository*. Trieda *ItemRepository* dedí vlastnosti rozhrania *IItemRepository*, ktoré definuje nutné metódy, ktoré musí mať trieda *ItemRepository* implementované. Trieda má dve súkromné premenné.

- **Private static readonly List<Item> itemList** je súkromným zoznamom objektov typu *Item*, ktorý predstavuje úplný zoznam realít existujúcich v našom systéme. Zobrazenie zoznamu volá vystavená metóda *ItemList*.
- **Private static int nextId** je identifikátor pridanej položky. Prvotne nastavujeme identifikátor na hodnotu 0.

Pri uchovávaní objektov, respektíve našich realitných položiek v premenných sme zvolili premennú referenčného typu *List<T>*. Možnou alternatívou bol referenčný typ *ArrayList*, keďže majú podobnú funkcionalitu. Rozhodujúcim faktorom bolo odporúčanie na stránkach spoločnosti Microsoft, ktorý vyzdvihuje použitie „Listu“ kvôli vyššej rýchlosti vo väčšine prípadov a jeho typovej bezpečnosti.[29]

Niektoré servisné metódy vracajú vystaveným metódam výstup vo formáte JSON. Na konverziu výstupného formátu XML na JSON sme použili populárny JSONový

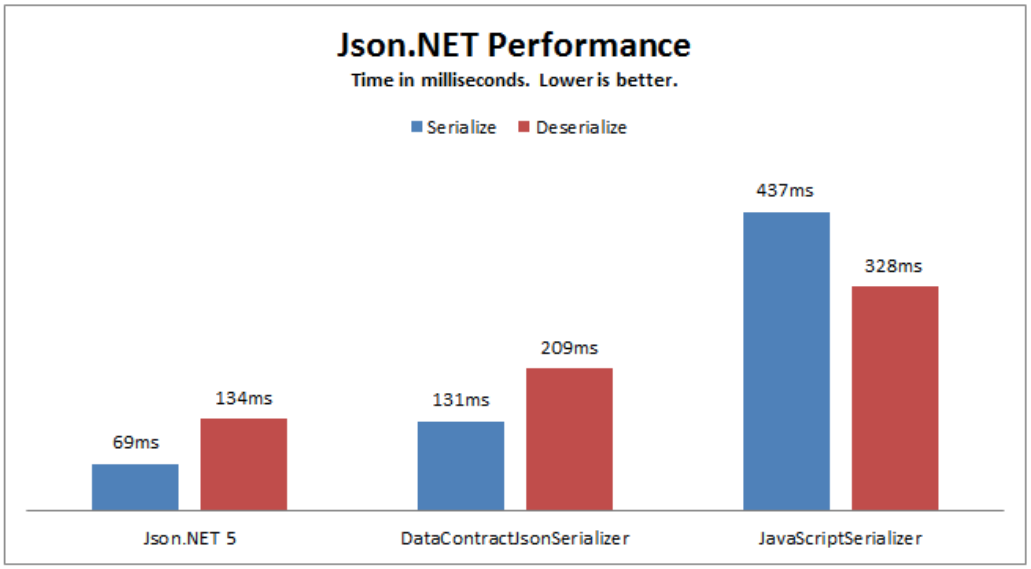
[29] Microsoft [Online]. Dostupné na internete: <https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1?view=netframework-4.8>. [2.4.2019].

framework Json.NET od spoločnosti Newtonsoft[30], ktorý je otvorený pod licenciou MIT a bezplatný pre komerčné použitie. Json.NET sme pridali do nášho vývojového prostredia pomocou správcu balíkov NuGet, príkazom:

PM> Install-Package Newtonsoft.Json

Konverzia výstupu z formátu XML na JSON je bežnou záležitosťou a na jej vykonanie existujú viaceré frameworky. Pre framework od spoločnosti Newtonsoft sme sa rozhodli kvôli jeho rýchlosti a jednoduchosti použitia.

Obrázok č. 9: Porovnanie výkonu Json.NET s konkurenciou



Zdroj: https://www.newtonsoft.com/json

Naše servisné metódy pracujú s XML údajmi a prehľadávajú ich koreňové štruktúry, preto v súbore *ItemRepository.cs* používame menný priestor *System.Linq*.

Servisné metódy a ich fungovanie si popíšeme v nasledovných riadkoch.

Add je metóda pre pridanie položky do zoznamu *itemList*. Vstupným parametrom je objekt item, typu Item, teda všetky vlastnosti triedy *Item.cs*, ktoré sa k metóde dostanú napríklad vstupom cez vystavenú metódu *AddItem*. Metóda *Add* najskôr nastaví hodnotu idenfitikátora

[30] Newtonsoft [Online]. Dostupné na internete: < https://www.newtonsoft.com/json>. [18.4.2019].

položky na úroveň hodnoty premennej nextId, ktorú následne navíši o jednotku. Následne je zavolaná, v mennom priestore *System.Collections.Generic*
Preddefinovaná, metóda *Add* triedy *List<T>,* ktorá pridá objekt na koniec zoznamu *itemList*.

```
public void Add(Item item)
    {
       item.RealId = nextId; //pri pridavani nepytame iD, to sa musi setnut automaticky
       nextId++;
       itemList.Add(item);
    }
```

CreateFile je metóda, ktorej účelom je vytvorenie súboru, do ktorého vo formáte XML uloží vstupný zoznam objektov, reprezentovaný premennou *content* vstupujúcou do metódy. Premenná *path* reprezentuje cestu, kam má byť súbor uložený.

```
public void CreateFile(string path, List<Item> content)
    {

       FileStream fs = File.Open(@path, FileMode.Create);

       var listik = content;
       string toWrite;

                   var emptyNamespaces = new XmlSerializerNamespaces(new[] {
       XmlQualifiedName.Empty });
       var settings = new XmlWriterSettings();
       settings.Indent = true;
       settings.OmitXmlDeclaration = true;

       XmlSerializer serializer = new XmlSerializer(typeof(List<Item>));

       using (var stream = new StringWriter())
       using (var writer = XmlWriter.Create(stream, settings))
```

```
        {
            serializer.Serialize(writer, listik, emptyNamespaces);
            toWrite = stream.ToString();
        }

        byte[] byteArr = Encoding.Default.GetBytes(toWrite);
        fs.Write(byteArr, 0, toWrite.Length);
        fs.Close();
    }
```

Na úpravu súboru používame triedu *FileStream*, pod menným priestorom *System.IO*, ktorá poskytuje dátový prúd pre súbor podporujúci synchrónne aj asynchrónne operácie čítania a písania.[31]

Metóda súbor otvorí a pristupuje k nemu v móde vytvor (*Create*). Vytvorí inštanciu triedy *XmlSerializerNamespaces*, ktorá obsahuje menné priestory a predpony, ktoré používa XmlSerializer (serializujúci a deserializujúci objekty v xml súbore) na tvorbu kvalifikovaných mien v inštancii XML dokumentu.[32]

Ďalej vytvorí inštanciu triedy *XmlWriterSettings()* z menného priestoru *System.Xml*, ktorá špecifikuje vlastnosti objktu *XmlWriter*. Naše nastavenia slúžia na vynechanie úvodnej deklarácie hlavičky XML súboru tak, aby sme pracovali s čistými XML údajmi.

Do premennej stream typu *StringWriter()* ukladáme údaje zo zoznamu *listik,* ktorý potrebujeme do súboru zapísať. Tieto údaje uložíme do premennej *toWrite* vo formáte *string*, ktorý ďalej konvertujeme na pole typu *Byte*.

Na záver trieda *filestream* zapisuje do súboru pole typu *Byte* s názvom *byteArr* a zatvára súbor.

---

[31] Microsoft [Online]. Dostupné na internete: <https://docs.microsoft.com/en-us/dotnet/api/system.io.filestream?view=netframework-4.8>. [18.4.2019].

[32] Microsoft [Online]. Dostupné na internete: <https://docs.microsoft.com/en-us/dotnet/api/system.xml.serialization.xmlserializer?view=netframework-4.8>. [18.4.2019].

GetDateRange slúži ako servisná metóda na získanie výstupu spĺňajúceho dátumové ohraničenie. Vstupnými premennými je spodná hranica *low* a horná hranica *high*. Obe hodnotového typu *DateTime*. Možným parametrom je aj cesta *path*.

```
public string GetDateRange(DateTime low, DateTime high, string path)
    {
      var temporaryList = new List<Item>();
      foreach (var item in itemList)
      {
         if (item.TimeOfCreation >= low && item.TimeOfCreation <= high)
         {
            temporaryList.Add(item);
         }
      }
      if (!string.IsNullOrWhiteSpace(path))
      {
         CreateFile(path, temporaryList);
      }

      var json = JsonConvert.SerializeObject(temporaryList);
      return json;
    }
```

Po zavolaní metódy sa vytvorí inštancia triedy *List<Item>* s názvom *temporaryList*, ktorá bude slúžiť ako lokálna premenná metódy, respektíve dočasný zoznam objektov. Pomocou cyklu *foreach* prehľadáme každý objekt v zozname objektov *itemList* a podmienkou *if* overíme, či je vlastnosť konkrétneho objektu *item.TimeOfCreation* v požadovanom rozsahu. Ak je podmienka splnená, objekt je pridaný do dočasného zoznamu *temporaryList*. Ak je zadaná cesta, na ktorú má byť výstup uložený, metóda zavolá druhú servisnú metódu *CreateFile* s parametrami *path* a *temporaryList*, ktorý predstavuje výstup metódy *GetDateRange*.

Metóda vráti výstup vo formáte JSON. Konverzia z formátu XML do JSON je reprezentovaná serializáciou objektu *temporaryList* do premennej *json*, hodnotového typu *string*.

GetAll je jednoduchou servisnou metódou bez vstupných premenných. Vracia obsah zoznamu *itemList* konvertovaný do formátu JSON identickým spôsobom ako v predchádzajúcej metóde *GetDateRange*.

```
public string GetAll()
    {
        var json = JsonConvert.SerializeObject(itemList);
        return json;
    }
```

GetFiltered, GetCategory a GetByLocation sú funkčne takmer identické servisné metódy, ktoré príjímajú premenné z príslušných vystavených metód a overením podmienky napĺňajú dočasný zoznam objektov *temporaryList*, ktorý pred vrátením konvertujú do formátu JSON.

```
public string GetFiltered(string category, int room, string city, int minprice, int maxprice,
string path)
    {
        var temporaryList = new List<Item>();
        foreach (var item in itemList)
        {
         if (item.Category == category && item.Rooms == room && item.Location == city
        && item.Price < maxprice && item.Price > minprice)
            {
                temporaryList.Add(item);
            }
        }

        if (!string.IsNullOrWhiteSpace(path))
        {
            CreateFile(path, temporaryList);
```

```
        }

        var json = JsonConvert.SerializeObject(temporaryList);
        return json;
    }
```

XmlFill je servisnou metódou pre vystavenú metódu FillWithFile, ktorá jej posúva hodnotu premennej path, z ktorej si má metóda prebrať vstupný súbor.

Použitá metóda *Xdocument.Load* vytvorí XML document zo súboru špecifikovanom cestou *path*. Pomocou metódy *Root.Elements* pristupujeme ku koreňovému elementu *Item* v našom XML súbore. Pre každý koreňový element vyberáme zo súboru nový objekt *Item*, ktorého vlastnosti nastavujeme podľa elementov *Name, Price, Location, Category, Size, Age, Rooms, Isolation* a *TimeOfCreation* v XML súbore. Vybratý objekt pridávame do zoznamu newlist pomocou metódy *ToList.*

Následne každú položku v zozname *newlist* pridáme do zoznamu *itemList* v cykle *foreach* a keďže v navrhnutom XML súbore nefiguruje vlastnosť objektu *Item*, *RealId*, musíme ho v cykle každému objektu priradiť podľa potrebného identifikátora *nextId*.

Celá funkcionalita metódy je umiestnená do bloku *Try*, ktorý sleduje či kod nevyhodí výnimku, kvôli nesprávnej štruktúre dát XML súboru.

Pri výskyte výnimky v bloku *Try* hľadá Common Laguange Runtime (CLR) blok *Catch*, ktorý ošetruje, ako sa má s námietkou vysporiadať.[33]

V našom metóda vraciama reťazec *"zly format xml",* v prípade výskytu *System.ArgumentNullException,* objavujúcej sa pri posune neplatného alebo prázdneho argumentu metóde.

[33] Microsoft [Online]. Dostupné na internete: <*https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/try-catch*>. [19.4.2019].

```
public string XmlFill(string path)
        {
            var newlist = new List<Item>();
            try
            {
                newlist = (from e in XDocument.Load(@path).Root.Elements("Item")
                           select new Item
                           {
                               Name = (string)e.Element("Name"),
                               Price = (int)e.Element("Price"),
                               Location = (string)e.Element("Location"),
                               Category = (string)e.Element("Category"),
                               Size = (int)e.Element("Size"),
                               Age = (int)e.Element("Age"),
                               Rooms = (int)e.Element("Rooms"),
                               Isolation = (string)e.Element("Isolation"),
                               TimeOfCreation = (DateTime)e.Element("TimeOfCreation"),
                           })
                    .ToList();
                foreach (var item in newlist)
                {
                    item.RealId = nextId;
                   nextId++;
                    itemList.Add(item);
                }
            }
            catch (System.ArgumentNullException)
            {
                return "zly format xml";
            }

            var json = JsonConvert.SerializeObject(newlist);
           return json;
        }
```

Nasledovné servisné metódy spolupracujú za účelom zvolenia, zmeny a uloženia konkrétneho objektu zo zoznamu objektov.

GetById slúži na zvolenie konkrétneho objektu podľa jeho identifikátora, ktorý je jej vstupným parametrom.

```
public Item GetById(int id)
 {
    var item = itemList.SingleOrDefault(item1 => item1.RealId == id);
    //vyberie jedno zo zoznamu ktore splna podmienku
    if (item == null)
    {
       return null;
    }

        return new Item {RealId = item.RealId, Category = item.Category, Name = item.Name, Location = item.Location, Price = item.Price, Size = item.Size, Age = item.Age, Rooms = item.Rooms, Isolation = item.Isolation};
 }
```

Na voľbu konkrétneho objektu podľa jeho vlastnosti *RealId* je použitá preddefinovaná metóda triedy *List<T> SingleOrDefault()*. Vlastnosti nájdeného objektu uloží do lokálneho objektu s názvom *item*. Metóda vráti nový objekt typu *Item*, ktorého vlastnosti nadefinuje podľa vlastností vytvoreného lokálneho objektu.

EditItem slúži na úpravu existujúcej položky v systéme. Vytvorí lokálnu premennú, do ktorej uloží konkrétny objekt zo zoznamu, identifikovaný podľa jeho *realId* za pomoci metódy *GetById*. Jeho vlastnosti následne zmení aby sa rovnali hodnotám premenných prijatých z vystavenej metódy. Po zmene zavolá servisnú metódu *Save*, ktorá dočasný objekt *tempItem* uloží.

```
public void EditItem(int readlId, string newName, int newSize, string newCategory, int newPrice, string newLocation, int newAge, int newRooms, string iso)
    {
        var tempItem = GetById(readlId);

        tempItem.Name = newName;
        tempItem.Size = newSize;
        tempItem.Category = newCategory;
        tempItem.Price = newPrice;
        tempItem.Location = newLocation;
        tempItem.Age = newAge;
        tempItem.Rooms = newRooms;
        tempItem.Isolation = iso;
        Save(tempItem);

    }
```

Save je metóda používaná na uloženie zmenených údajov. Jej vstupom je objekt s želanými vlastnosťami. Metóda pristúpi k objektu v zozname *itemList*, ktorý ma identický identifikátor ako vstupný objekt a prepíše jeho vlastnosti tak, aby sa rovnali vlastnostiam želaného objektu.

```
public void Save(Item updateItem)
    {
        var oldItem = itemList.SingleOrDefault(i => i.RealId == updateItem.RealId);
        if (oldItem == null)
        {
            return;
        }
        oldItem.Location = updateItem.Location;
        oldItem.Name = updateItem.Name;
        oldItem.Category = updateItem.Category;
        oldItem.Price = updateItem.Price;
```

```
    oldItem.Isolation = updateItem.Isolation;
    oldItem.Size = updateItem.Size;
    oldItem.Age = updateItem.Age;
    oldItem.Rooms = updateItem.Rooms;
}
```

4.3. Údaje, ktoré webová služba spracováva

4.3.1. Trieda Item

Trieda Item slúži na definovanie objektu - realitnej položky. Má 10 vlastností. Nachádza sa v samostatnom súbore *Item.cs*, ktorý zdieľa menný priestor *RealityService*.

Tabuľka č. 4: Zoznam vlastností triedy Item

Názov	**Typ**	**Popis**
RealId	integer	Unikátny identifikátor
Name	string	Meno položky
Size	integer	Rozloha nehnuteľnosti
Category	string	Kategória nehnuteľnosti
Price	integer	Cena
Location	string	Poloha
Age	integer	Vek nehnuteľnosti
Rooms	integer	Počet izieb
Isolation	string	Izolácia, zateplenie
TimeOfCreation	DateTime	Čas vytvorenia položky

Zdroj: Vlastné spracovanie

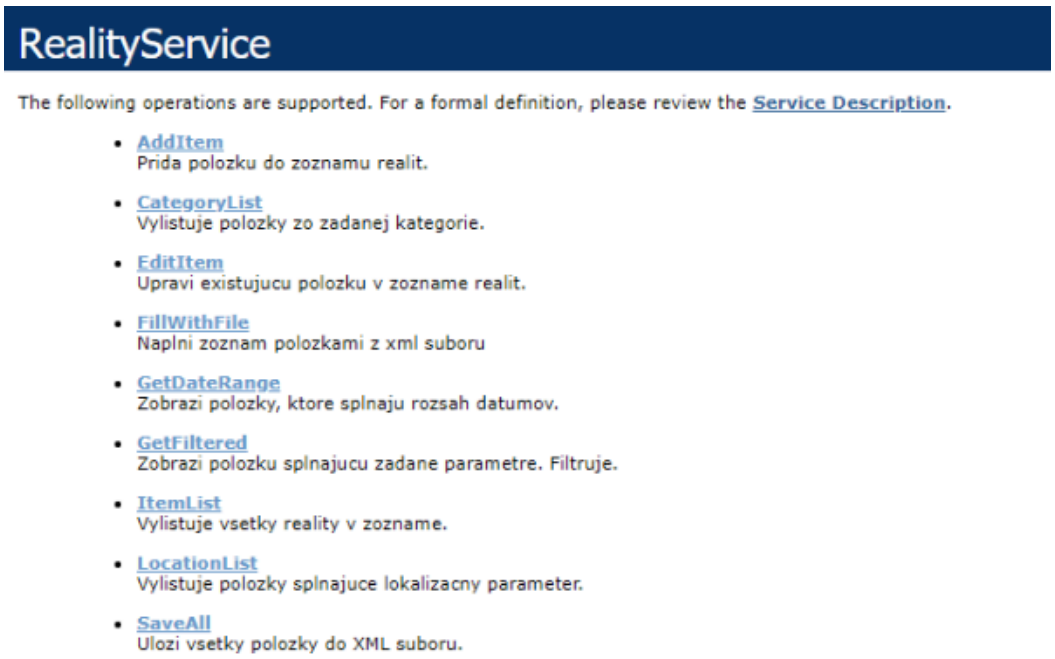
Kód triedy *Item*:

```
public class Item
  {
     public int RealId { get; set; }
     public string Name { get; set; }
     public int Size { get; set; }

     public string Category { get; set; }
     public int Price { get; set; }
     public string Location { get; set; }
     public int Age { get; set; }
     public int Rooms { get; set; }
     public string Isolation { get; set; }

     public DateTime TimeOfCreation { get; set; }
  }
```

4.4. Testovací klient webovej služby

Na obrázku nižšie vidíme testovacieho klienta, ktorého pripravilo vývojové prostredie Vistual studio po zostavení .asmx stránky s vystavenými metódami webovej služby.

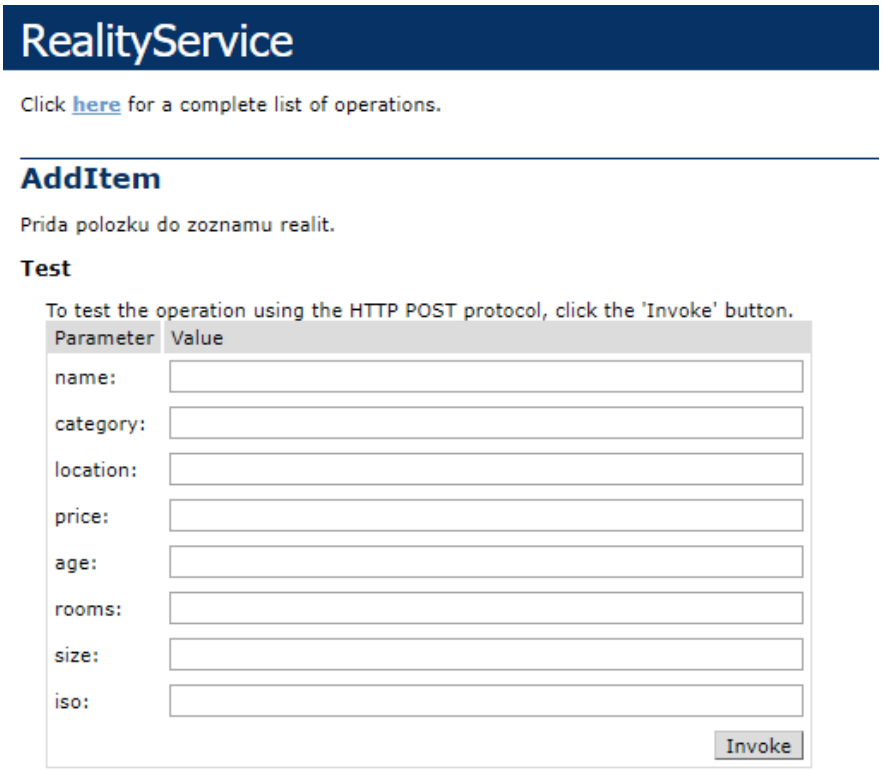
Obrázok č. 10: Rozhranie testovacieho klienta



Zdroj: Vlastné spracovanie

Po rozkliknutí niektorej z vystavených metód prechádzame na informačnú stránku s formulárom pre zadanie vstupných premenných webovej metódy a s tlačidlom Invoke, ktorý webovú metódu zavolá a vykoná jej telo.

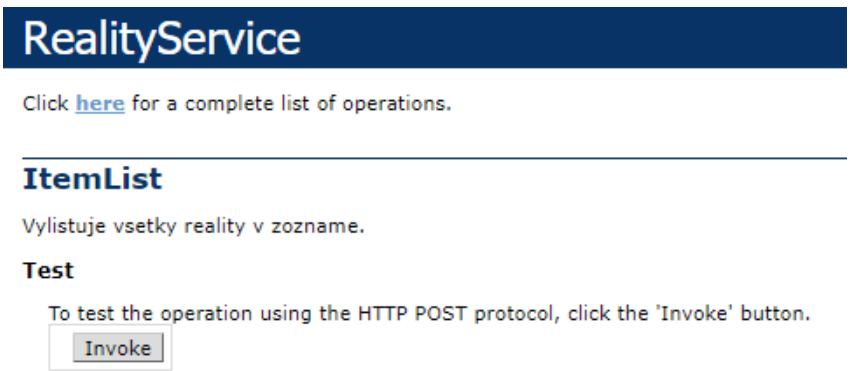
Obrázok č. 11: Ukážka rozhrania metódy AddItem



Zdroj: Vlastné spracovanie

Ak webová metóda nemá vstupné parametre ako napríklad metóda ItemList, zavolá sa len tlačidlom Invoke prostredníctvom HTTP POST protokolu, ktorý je použitý pri volaní všetkých vystavených metód.

Obrázok č. 12: Ukážka volania metódy AddItem



Zdroj: Vlastné spracovanie

4.5. Umiestnenie webovej služby

4.5.1. IIS server Expres

Na zostavovanie webovej služby a jej testovanie sme používali HTTP webový server IIS server Express, ktorý pri zostavovaní riešenia vývojové prostredie Vistual Studio automaticky spustí. Express verzia IIS servera je vhodná na použitie počas vývoja aplikácie hotovú webovú službu sme umiestnili na IIS server.

4.5.2. Nasadenie webovej služby na IIS server

Webové servery IIS poskytujú úplnú modulárnu architektúru slúžiacu ku kumponentizácii, rozširovaniu a integrácii ASP.NET aplikácií.[34]

Po dokončení vývoja sme hotový projekt vložili na IIS server pomocou funkcie Publish vo Visual studiu. Výsledok bol uložený do súborového adresára prislúchajúcemu k novej webstránke vytvorenej v správcovskom prostredí IIS servera, IIS server manager. Pre prístup k webstránke sme si zvolili port 8080.

Na IIS serveri môže naša webová služba fungovať kontinuálne, bez nutnosti spustenia vývojového prostredia Visual studio a ladenia .asmx stránky.

[34] Microsoft. [Online] Dostupné na internete <https://docs.microsoft.com/en-us/iis/get-started/introduction-to-iis/iis-web-server-overview>. [18.4.2019].

4.6. Návrh klienta používajúceho webovú službu

Našim hlavným cieľom bol vývoj webovej služby, no aby s ňou vedel zákazník pohodlne interagovať, potrebujeme mu ponúknuť prehľadného, intuitívneho klienta, ktorý výstupy vystavených metód webovej služby spracuje do usporiadanej podoby.

Usporiadanou podobou chápeme tabuľku, ktorej stĺpce budú predstavovať vlastnosti realitných položiek.

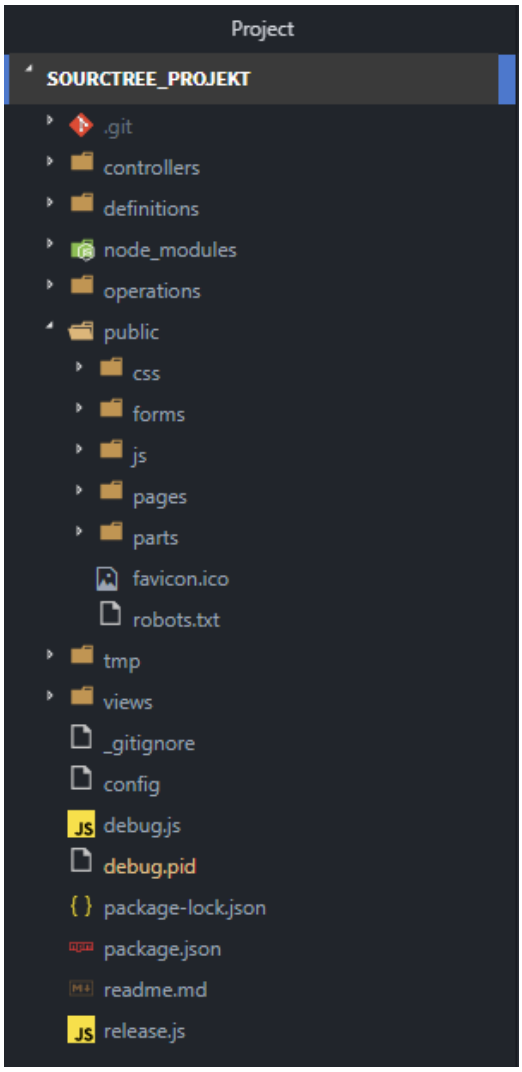
Existuje mnoho spôsobov takzvanej "konzumácie" webovej služby, väčšina z nich zdieľa použitie scriptovacieho jazyka JavaScript, hypertextového značkovacieho jazyka HTML a kaskádových štýlov CSS.

Klienta webovej služby sme pripravovali v integrovanom vývojovom prostredí Atom. Na spojenie klienta s webovou službou sme použili framework *Total.js* fungujúci na JavaScriptovom vykonávacom prostredí *Node.js*. Na tvorbu komponentov pre stranu klienta sme použili knižnicu *jComponent*. Jednotlivé elementy na obrazovke klienta, ako sú tlačidlá, tabuľka či formuláre sú použité ako predpripravené komponenty dostupné na componentator.com.

4.6.1. Štruktúra klienta

Názov projektu klienta je SOURCETREE_PROJEKT. Pri jeho tvorbe sme využívali softvér na kontrolu a uchovávanie verzií *github* a na ich posielanie sme využívali GUI softvér *Sourcetree*. Je naprogramovaný v IDE Atom.

Obrázok č. 13: Štruktúra projektu v IDE Atom

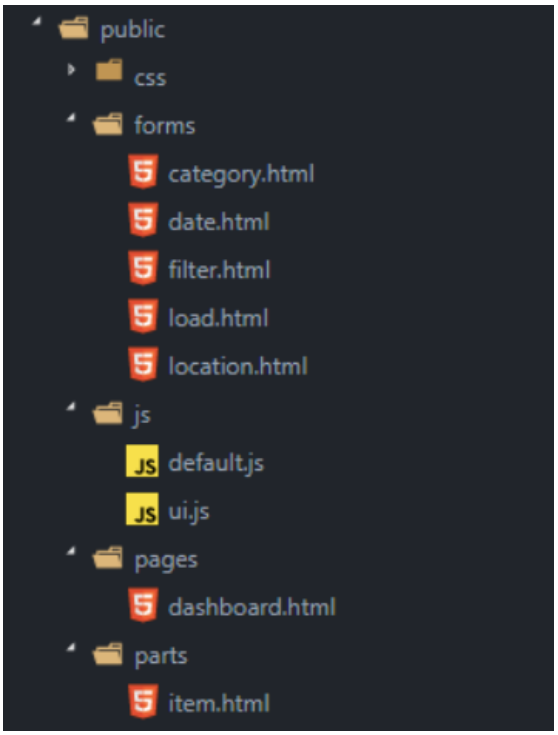


Zdroj: Vlastné spracovanie

V zložke public sa nachádzajú súbory s JavaScriptovým, HTML alebo CSS kódom pre stranu klienta. V zložkách *operations* a *controllers* sa nachádzajú javascriptové súbory, s kódom, ktorý sa pripája na webovú službu umiestnenú na IIS servery cez RESTové volania. Prístupovú vrstvu k vystaveným metódam webovej služby definuje súbor *api.js* v zložke *controllers*. Operácie v súbore *oper.js* predstavujú metódy definujúce spôsob pripojenia sa na konkrétnu vystavenú metódu webovej služby.

Prístupová vrstva *api.js* a oper.js je napísaná vo frameworku *Total.js*, čo je backendový framework písaný v JavaScripte.

Obrázok č. 14: Štruktúra zložky public s html a js súbormi



Zdroj: Vlastné spracovanie

V súbore *ui.js* sa nachádzajú komponenty, ktoré sú sťahované z *componentator.com* prostredníctvom Content Delivery Networku.

4.7. Spojenie klienta s webovou službou

Spojenie klienta s webovou službou je vykonané prostredníctvom volania vystavených operácií apinou v adresári *controllers/api.js*, ktorá definuje prístupovú vrstvu k backendu. Vystavené operácie v adresári *operations/oper.js* komunikujú s metódami webovej služby pristupovaním k ich url. Návod na komunikáciu s vystavenými metódami prostredníctvom HTTP POST protokolu nám poskytuje samotná webová služba, ktorá sa takto sama popisuje.

Obrázok č. 15: Automaticky generovaný opis HTTP post metódy v testovacom rozhraní webovej služby

HTTP POST

The following is a sample HTTP POST request and response. The placeholders shown need to be replaced with actual values.

```
POST /RealityService.asmx/AddItem HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: length

name=string&category=string&location=string&price=string&age=string&rooms=string&size=string&iso=string
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<int xmlns="http://tempuri.org/">int</int>
```

Zdroj: Vlastné spracovanie

4.7.1. API (Aplication Programming Interface)

Na obrázku nižšie vidíme nami nadefinované aplikačné programové rozhranie API, ktoré používa HTTP požiadavky GET, PUT a POST na manipuláciu s údajmi. Ide o RESTové api, ktoré je viac robustné ako SOAP.

Požiadavka GET je používaná na získanie údajov zo zdroja, PUT pre aktualizáciu údajov a POST pre vytvorenie údajov.[35] V prípade potreby je k dispozícii aj požiadavka DELETE na vymazanie údajov, s ktorou ale nepracujeme.

Obrázok č. 16: Vystavené RESTové api

```
exports.install = function() {
    ROUTE('GET     /api/items/                          * --> @ItemList');
    ROUTE('GET     /api/items/category/                 * --> @CategoryList');
    ROUTE('GET     /api/items/location/                 * --> @LocationList');
    ROUTE('GET     /api/items/date/                   * --> @DateList');
    ROUTE('GET     /api/items/filter/                   * --> @FilterList');
    ROUTE('PUT     /api/items/{id}/                     * --> @UpdateItem');
    ROUTE('POST    /api/items/                          * --> @AddItem');
    ROUTE('POST    /api/path/                           * --> @SendPath');

};
```

Zdroj: Vlastné spracovanie

[35] Techtarget. [Online] Dostupné na internete: <https://searchmicroservices.techtarget.com/definition/RESTful-API>. [18.4.2019].

4.7.2. Ukážka operácie

Operácia *ItemList* vytvára inštanciu objektu *RESTBuilder*, ktorému definuje url pre spojenie sa s webovou službou *http://localhost:8080/RealityService.asmx/ItemList* a definuje hlavičku, podľa návodu, ktorý poskytuje webová služba ako *Content-Type: application/x-www-form-urlencoded*. Na záver uloží výstup do funkcie *callback*.

Obrázok č. 17: Zdrojový kód operácie ItemList

```
NEWOPERATION('ItemList', function($) {
    RESTBuilder.make(function(builder) {
        builder.url('http://localhost:8080/RealityService.asmx/ItemList');
        builder.header('Content-Type', 'application/x-www-form-urlencoded');
        builder.post();
        builder.exec(function(err, response) {
          console.log(response);
            if (!err) {
                $.callback({ items: response });
            }
        });
    });
```

Zdroj: Vlastné spracovanie

4.8. Zobrazenie výstupu v okne klienta

Po zavolaní vystavenej metódy sa jej odpoveď zobrazí do tabuľky (datagridu) v sekcii Dashboard. Jej fungovanie si priblížime pri opise používateľského rozhrania.

Obrázok č. 18: Zobrazený výstup v okne klienta

	Meno	Kategória	Cena	Poloha	Čas	Počet izieb	Vek	Veľkosť	Zateplenie
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	Krasna rodinna vila s dvorom a fo…	dom	120000	Levice	01.05.2019 00:00:00	7	20	555	ano
2	Mezonet v historickej budove	byt	149900	Levice	10.03.2019 00:00:00	3	10	150	ano
3	Panelakovy byt	byt	80000	Levice	11.03.2019 00:00:00	3	50	67	ano
4	Vila na kopci	dom	300000	Levice	12.03.2019 00:00:00	7	18	523	nie
5	Bungalov na predmesti	dom	110000	Levice	13.03.2019 00:00:00	4	1	99	ano
6	Velke urodne pole s kukuricou	pozemok	10000	Levice	14.03.2019 00:00:00	0	0	100000	0
7	Novostavba v historickom centre	byt	249900	Bratislava	15.03.2019 00:00:00	3	60	65	nie
8	Garzonka v Petrzalke	byt	75000	Bratislava	16.03.2019 00:00:00	1	45	25	ano
9	Zahradny domcek vhodny na byv…	dom	159000	Bratislava	17.03.2019 00:00:00	2	30	59	nie
10	Renovovana mestska vila na Pali…	dom	999500	Bratislava	18.03.2019 00:00:00	9	70	600	nie
11	Parkovacie miesto v podzemnej …	garaz	21900	Bratislava	19.03.2019 00:00:00	0	9	18	0
12	Byt s balkonom	byt	30000	Humenne	20.03.2019 00:00:00	2	34	51	ano
13	Panelakovy byt po poziari	byt	20000	Humenne	21.03.2019 00:00:00	4	45	98	nie
14	Samostatne stojaca zateplena ga…	garaz	12500	Humenne	22.03.2019 00:00:00	0	10	25	ano
15	Obrovska novostavba na samote	dom	111000	Humenne	23.04.2019 00:00:00	12	15	509	ano
16	Stary dom na zburanie, zaujimav…	dom	10000	Humenne	24.04.2019 00:00:00	3	105	150	nie

Zdroj: Vlastné spracovanie

4.9. Opis používateľského rozhrania

Po zadaní URL do prehliadača sa používateľovi zobrazí používateľské rozhranie. Je veľmi jednoduché a prehľadné.

V ľavom postrannom paneli sa nachádzajú možnosti filtrovania:

- všetky, ktoré vyvolá metódu *ItemList,*
- kategórie, ktoré vyvolá metódu *CatagoryList*,
- lokalita, ktoré vyvolá metódu *LocationList*,
- dátum, ktoré vyvolá metódu *GetDateRange*,
- filter, ktoré vyvolá metódu *GetFiltered*.

Ďalej dve tlačidlá:

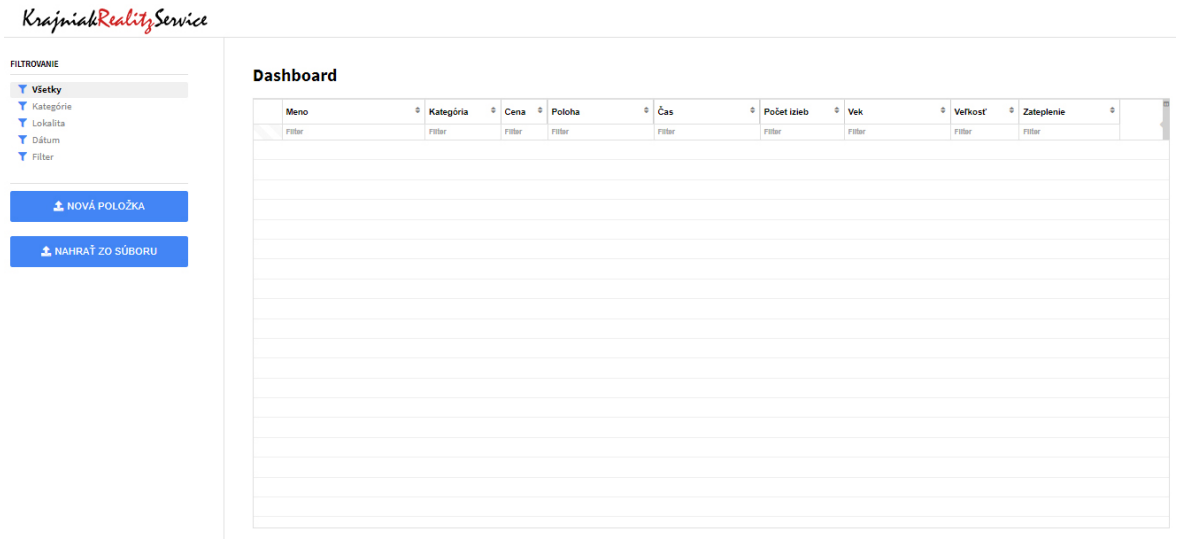
- nová položka, ktoré vyvolá metódu *AddItem,*
- nahrať zo súboru, ktoré vyvolá metódu *FillWithFile*.

Funkcionalita úpravy položky je vyvolaná formulárom, ktorý sa zobrazí po kliknutí na riadok v tabuľke.

4.9.1. Ukážka rozhrania

Na obrázku nižšie je vidieť celé rozhranie klienta. V ľavom hornom rohu sa nachádza navrhnuté logo aplikácie KrajniakRealityService. V ľavom stĺpci vidíme tlačidlá pre voľbu metód. V sekcií Dashboard vidíme tabuľku, do ktorej zobrazujeme výstup z vystavených metód webovej služby, tak ako sme ho ukázali na obrázku číslo 18.

Obrázok č. 19: Ukážka používateľského rozhrania klienta



Zdroj: Vlastné spracovanie

Pre navrhnuté logo sme použili font Mistral. Ukážka loga:

Obrázok č. 20: Navrhnuté logo

KrajniakRealityService

Zdroj: Vlastné spracovanie

4.9.2. Formuláre

Údaje, ktoré posielame webovej službe získavame na strane klienta prostredníctvom formulárov. Náš klient poskytuje 5 HTML formulárov v zložke public/forms.

Category.html je formulár vyvolaný po kliknutí na položku Kategórie v postrannom stĺpci.

Date.html je vyvolaný kliknutím na položku Dátum.

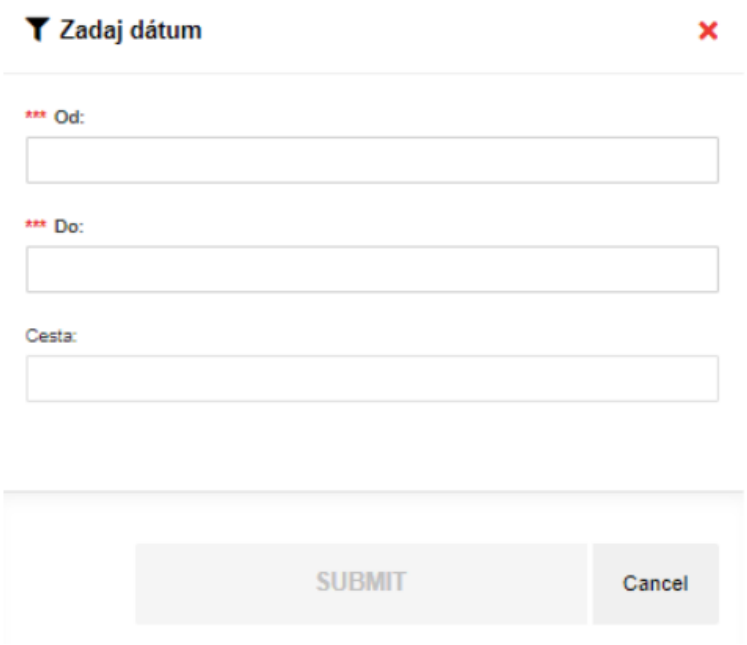
Filter.html je vyvolaný kliknutím na položku Filter.

Load.html je vyvolaný kliknutím na tlačidlo Nahrať zo súboru.

Location.html je vyvolaný kliknutím na položku Lokalita.

Na obrázku nižšie vidíme ukážku formuláru Date.html.

Obrázok č. 21: Formulár Date



Zdroj: Vlastné spracovanie

Každý formulár má dve základné funkcie exports.reload a exports.submit.

Funkcia reload slúži na opätovné načítanie formulára.

Funkcia submit posiela jeho obsah do príslušnej apiny prostredníctvom AJAXu. Po potvrdení tlačidlom *submit* definujeme oneskorenie 500 milisekúnd, kým sa formulár skryje.

Na obrázku nižšie vidíme funkcie formulára vyvolaného po kliknutí na tlačidlo *Nahrať zo súboru,* posielajúceho údaje do apiny */api/path/*.

Obrázok č. 22: Zdrojový kód funkcií export.submit a exports.reload

```
PLUGIN('loadform', function(exports) {

  exports.reload = function(com) {
    NULL('loadform.form');
  };

  exports.submit = function(com) {
    // Reset form
    var form = GET('loadform.form');
    RESET('loadform.form.*');
    SETTER('loading', 'show');

    AJAX('POST /api/path/', form, function(response) {
      SETTER('loading', 'hide', 500);
      console.log(response);
    });
    EXEC('dashboard/refresh');
    com.hide();
  };
});
```

Zdroj: Vlastné spracovanie

4.9.3. Dashboard

Dashboard je tabuľkou, na ktorú sa cez AJAXové volania zobrazuje výstup operácií. Tabuľka (DataGrid) je predpripraveným komponentom, ktorý je dostupný ako j-DataGrid na webe componentator.com.

Obrázok č. 23: Definované stĺpce tabuľky (datagrid)

```
<div class="row">
  <div class="col-lg-12">
    <div data-jc="datagrid__dashboard.grid__checkbox:false;click:dashboard/select;highlight:true;autoselect:false">
      <script type="text/plain">
        [
          { name: 'Name', text: 'Meno', width: 300 },
          { name: 'Category', text: 'Kategória' },
          { name: 'Price', text: 'Cena' },
          { name: 'Location', text: 'Poloha' },
          { name: 'TimeOfCreation', text: 'Čas', format: '[ts]' },
          { name: 'Rooms', text: 'Počet izieb' },
          { name: 'Age', text: 'Vek' },
          { name: 'Size', text: 'Veľkosť' },
          { name: 'Isolation', text: 'Zateplenie' }
        ]
      </script>
    </div>
  </div>
</div>
```

Zdroj: Vlastné spracovanie

5. Diskusia

Zistili sme, že realitné kancelárie a agenti na Slovenskom trhu pracujú najmä s improvizovanými riešeniami a pomáhajú si systémami, ktoré sú poskytované inzertným, e-commerce segmentom, ktorý neprítomnosť dostatočného množstva softvérových produktov pre menšie realitné kancelárie na trhu supluje. Webová služba môže byť, po odkonzultovaní s konkrétnym klientom, upravená podľa jeho potrieb použitá ako komerčný produkt.

Kompilovaná webová služba vyniká pred použitím rôznych CMS postavených na jazyku PHP, ako napríklad Wordpress najmä bezpečnosťou. V porovnaní s evidenciou individuálnych agentov v súboroch tabuľkových procesorov ide o reprezentatívnejšie a bezpečnejšie riešenie, ktoré nie je veľmi náročné ani nákladné na vývoj a môžu si ho tak dovoliť aj menšie spoločnosti.

Vhodným rozšírením aplikácie by bolo abstrahovanie od práce so zdrojovými a výstupnými XML súbormi, ktorých miesto by nahradila vhodná databáza. Ak by mala byť nami vyvinutá aplikácia použitá ako komerčný produkt, bude potrebné myslieť aj na zabezpečenie komunikácie medzi klientom a serverom, napríklad pomocou SSL certifikátu. Zabezpečenie samotnej webovej služby môže byť ošetrené napríklad zaheslovanými používateľskými kontami.

Nami vyvinutá aplikácia môže slúžiť na evidenciu realitných položiek, no vhodným doplnkom by mohla byť funkcia pridávania fotodokumentácie k nehnuteľnosti. Ak by existovala vôľa viacerých realitných agentúr, najmä najväčších hráčov združujúcich sa v asociáciách vytvoriť podobnú aplikáciu pre svojich členov, mohlo by byť ďalším logickým krokom lobovanie za integráciu s takýmto systémom u prevádzkovateľov inzertných portálov. Zjednodušene by mohli byť inzeráty nahrávané na inzertné portály priamo zo zdrojového systému.

Výsledkom by bolo stransparentnenie vlastníctva textov, aj obrázkov spracovaných konkrétnou realitnou agentúrou. Rovnako by sme odradili špekulantov od kradnutia

autorských diel maklérov, keďže ich vlastníctvo by bolo jasne dokázateľne a nebolo by presúvané na inzertný server.

Možnosti rozširovania webovej služby sú obmedzené len ľudskou kreativitou a vôľou zákazníka vývoj produktu financovať.

Záver

Aplikácie slúžiace na evidenciu informácií dôležitých pre fungovanie akejkoľvek firmy budú vždy žiadaným produktom. V dobe, kedy sú údaje najcennejším majetkom firmy musíme hľadať stále efektívnejšie spôsoby, ako s nimi pracovať. Realitné kancelárie sú príkladom firiem, ktoré pracujú s veľkým množstvom obchodných položiek podobného typu a stali sa preto vhodným objektom nášho skúmania.

V úvode skúmania sme sa zoznámili s fungovaním webových služieb ako s novým spôsobom vývoja na báze komponentov spĺňajúc podmienky interoperability, priateľskosti k sieti internet, so silne typizovaným rozhraním a so schopnosťou využiť existujúce internetové štandardy. Získali sme informácie o prístupoch, programovacích jazykoch a vývojových platformách a tieto poznatky sme využili pri návrhu a vývoji našej webovej služby, ktorú sme úspešne umiestnili na IIS server. K webovej službe sme pripravili prehľadného klienta.

V práci sa nám podarilo vytvoriť ASP.NET XML webovú službu, ktorá môže slúžiť potrebám realitnej kancelárie a tak splniť cieľ práce. Nami vytvorená webová služba umožňuje vytvárať obchodné položky a uchovávať ich. Okrem vytvárania položky po položke umožňuje prácu so zdrojovými súbormi vo formáte XML, z ktorých môžu byť obchodné položky, respektíve realitné inzeráty do systému nahrané, ďalej spracované a následne exportované a uložené do výstupného súboru rovnakého formátu. Aplikácia poskytuje jej klientovi 9 vystavených metód, pomocou ktorých môže vykonávať selekciu určitých druhov realitných položiek podľa ich vlastností alebo ich dokonca zobrazovať v určenom časovom intervale. Tieto dôležité vlastnosti našej webovej služby demonštruje vytvorený klient, ktorý v používateľsky priateľskom rozhraní výstup vystavených metód zobrazuje.

Ako sme poznamenali v kapitole Diskusia, nami vytvorená aplikácia má potenciál na rozvoj a jej používanie predstavuje prínos najmä pre menších realitných sprostredkovateľov a kancelárie, s ktorých potreby sme pochopili pri skúmaní súčasného stavu a odrazili sa na našej aplikácii.

Použitá literatúra

1. **ERASMUS, Michael**. *CoffeeScript Programming with jQuery, Rails, and Node.js.* Birmingham: Packt Publishing, 2012. 121 s. ISBN 978-1-84951-958-8.

2. **FREEMAN, Adam**. *Pro ASP.NET Core MVC.* Sixth Edition. New York: Apress, 2016. 1013 s. ISBN 978-1-4842-0398-9.

3. **HUME, Dean Allan**. *Fast ASP.NET websites*. New York: Manning Publications Co. 2013, 190 s. ISBN 9781617291258.

4. **MACDONALD, Matthew**. *Begining ASP.NET 4.5 in C#.* New York: Apress, 2012. s. 6-7 ISBN 978-1-4302-4251-2.

5. **MONSON-HAEFEL, Richard**. *J2EE™ Web Services.* Boston: Adison-Wesley. 2003. 928 s. ISBN 0-321-14618-2.

6. **SHORT, S**. *Building XML Web Services for the Microsoft .NET Platform.* Redmont: Microsoft Press, 2002. 426 s. ISBN 0-7356-1406-7.

7. **ŠKULTÉTY, Rastislav**. *JavaScript: Programujeme internetové aplikace.* 2. Aktualizované vydání. Brno: Computer Press, 2004. 224 s. ISBN 80-251-0144-4.

8. **THANGARATHINAM, Thiru**. *Professional ASP.NET 2.0 XML.* Indianapolis: Wiley Publishing, 2006. 539 s. ISBN 978-0-7645-9677-3.

Internetové zdroje

1. **BERKLEY** [Online]. Dostupné na internete: <http://courses.ischool.berkeley.edu/i290-14/s05/lecture-2/slide2.xhtml>. [20.3.2019].

2. **C# Corner** [Online]. Dostupné na internete: <https://www.c-sharpcorner.com/UploadFile/1d42da/web-service-basics/>. [16.2.2019].

3. **Freecodecamp** [Online]. Dostupné na internete: <https://guide.freecodecamp.org/javascript>. [16.4.2019].

4. **Guru99 Elearning** [Online]. Dostupné na internete: < https://www.guru99.com/web-service-architecture.html>. [12.2.2019].

5. **Internet World Stats** [Online]. Dostupné na internete: <https://www.internetworldstats.com/emarketing.htm>. [16.2.2019].

6. **JSON** [Online]. Dostupné na internete: <https://www.json.org>. [20.3.2019].

7. **JSON** [Online]. Dostupné na internete: <https://www.json.org/xml.html>. [20.3.2019].

8. **Microsoft** [Online]. Dostupné na internete: <https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet>. [2.4.2019].

9. **Microsoft** [Online]. Dostupné na internete: <https://dotnet.microsoft.com/learn/web/what-is-aspnet>. [2.4.2019].

10. **Microsoft** [Online]. Dostupné na internete: <https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1?view=netframework-4.8>. [2.4.2019].

11. **Microsoft** [Online]. Dostupné na internete: <https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-netod/4950065f-7a79-4021-85e6-33ee9b5e91ba>. [15.4.2019].

12. **MOHAMED, Azam** [Online]. Dostupné na internete: < https://www.codeproject.com/Articles/8257/How-to-Make-a-Simple-WebService-and-Consume-It >. [15.4.2019].

13. **Newtonsoft** [Online]. Dostupné na internete: < https://www.newtonsoft.com/json>. [18.4.2019].

14. **Techrepublic** [Online]. Dostupné na internete: < https://www.techrepublic.com/article/make-sense-out-of-the-confusing-world-of-net-file-types/>. [15.4.2019].

15. **Techtarget** [Online] Dostupné na internete: <https://searchmicroservices.techtarget.com/definition/RESTful-API>. [18.4.2019].

16. **Stormath** [Online]. Dostupné na internete: <https://stormpath.com/blog/rest-vs-soap>. [2.4.2019].

Príloha: Zdrojové XML údaje


```
<ArrayOfItem>
 <Item>
  <RealId>0</RealId>
  <Name>Mezonet v historickej budove</Name>
  <Rooms>3</Rooms>
  <Category>byt</Category>
  <Price>149900</Price>
  <Location>Levice</Location>
  <Size>150</Size>
  <Age>10</Age>
  <Isolation>ano</Isolation>
  <TimeOfCreation>2019-03-10T00:00:00+01:00</TimeOfCreation>
 </Item>
 <Item>
  <RealId>1</RealId>
  <Name>Panelakovy byt</Name>
  <Rooms>3</Rooms>
  <Category>byt</Category>
  <Price>80000</Price>
  <Location>Levice</Location>
  <Size>67</Size>
  <Age>50</Age>
  <Isolation>ano</Isolation>
  <TimeOfCreation>2019-03-11T00:00:00+01:00</TimeOfCreation>
 </Item>
 <Item>
  <RealId>2</RealId>
  <Name>Vila na kopci</Name>
  <Rooms>7</Rooms>
  <Category>dom</Category>
  <Price>300000</Price>
  <Location>Levice</Location>
  <Size>523</Size>
  <Age>18</Age>
  <Isolation>nie</Isolation>
  <TimeOfCreation>2019-03-12T00:00:00+01:00</TimeOfCreation>
 </Item>
 <Item>
  <RealId>3</RealId>
  <Name>Bungalov na predmesti</Name>
  <Rooms>4</Rooms>
  <Category>dom</Category>
  <Price>110000</Price>
  <Location>Levice</Location>
  <Size>99</Size>
  <Age>1</Age>
  <Isolation>ano</Isolation>
  <TimeOfCreation>2019-03-13T00:00:00+01:00</TimeOfCreation>
 </Item>
 <Item>
  <RealId>4</RealId>
  <Name>Velke urodne pole s kukuricou</Name>
  <Rooms>0</Rooms>
  <Category>pozemok</Category>
  <Price>10000</Price>
```

```
  <Location>Levice</Location>
  <Size>100000</Size>
  <Age>0</Age>
  <Isolation>0</Isolation>
  <TimeOfCreation>2019-03-14T00:00:00+01:00</TimeOfCreation>
</Item>
<Item>
  <RealId>5</RealId>
  <Name>Novostavba v historickom centre</Name>
  <Rooms>3</Rooms>
  <Category>byt</Category>
  <Price>249900</Price>
  <Location>Bratislava</Location>
  <Size>65</Size>
  <Age>60</Age>
  <Isolation>nie</Isolation>
  <TimeOfCreation>2019-03-15T00:00:00+01:00</TimeOfCreation>
</Item>
<Item>
  <RealId>6</RealId>
  <Name>Garzonka v Petrzalke</Name>
  <Rooms>1</Rooms>
  <Category>byt</Category>
  <Price>75000</Price>
  <Location>Bratislava</Location>
  <Size>25</Size>
  <Age>45</Age>
  <Isolation>ano</Isolation>
  <TimeOfCreation>2019-03-16T00:00:00+01:00</TimeOfCreation>
</Item>
<Item>
  <RealId>7</RealId>
  <Name>Zahradny domcek vhodny na byvanie</Name>
  <Rooms>2</Rooms>
  <Category>dom</Category>
  <Price>159000</Price>
  <Location>Bratislava</Location>
  <Size>59</Size>
  <Age>30</Age>
  <Isolation>nie</Isolation>
  <TimeOfCreation>2019-03-17T00:00:00+01:00</TimeOfCreation>
</Item>
<Item>
  <RealId>8</RealId>
  <Name>Renovovana mestska vila na Palisadach</Name>
  <Rooms>9</Rooms>
  <Category>dom</Category>
  <Price>999500</Price>
  <Location>Bratislava</Location>
  <Size>600</Size>
  <Age>70</Age>
  <Isolation>nie</Isolation>
  <TimeOfCreation>2019-03-18T00:00:00+01:00</TimeOfCreation>
</Item>
<Item>
  <RealId>9</RealId>
  <Name>Parkovacie miesto v podzemnej garazi</Name>
  <Rooms>0</Rooms>
  <Category>garaz</Category>
  <Price>21900</Price>
```

```
  <Location>Bratislava</Location>
  <Size>18</Size>
  <Age>9</Age>
  <Isolation>0</Isolation>
  <TimeOfCreation>2019-03-19T00:00:00+01:00</TimeOfCreation>
</Item>
<Item>
  <RealId>10</RealId>
  <Name>Byt s balkonom</Name>
  <Rooms>2</Rooms>
  <Category>byt</Category>
  <Price>30000</Price>
  <Location>Humenne</Location>
  <Size>51</Size>
  <Age>34</Age>
  <Isolation>ano</Isolation>
  <TimeOfCreation>2019-03-20T00:00:00+01:00</TimeOfCreation>
</Item>
<Item>
  <RealId>11</RealId>
  <Name>Panelakovy byt po poziari</Name>
  <Rooms>4</Rooms>
  <Category>byt</Category>
  <Price>20000</Price>
  <Location>Humenne</Location>
  <Size>98</Size>
  <Age>45</Age>
  <Isolation>nie</Isolation>
  <TimeOfCreation>2019-03-21T00:00:00+01:00</TimeOfCreation>
</Item>
<Item>
  <RealId>12</RealId>
  <Name>Samostatne stojaca zateplena garaz</Name>
  <Rooms>0</Rooms>
  <Category>garaz</Category>
  <Price>12500</Price>
  <Location>Humenne</Location>
  <Size>25</Size>
  <Age>10</Age>
  <Isolation>ano</Isolation>
  <TimeOfCreation>2019-03-22T00:00:00+01:00</TimeOfCreation>
</Item>
<Item>
  <RealId>13</RealId>
  <Name>Obrovska novostavba na samote</Name>
  <Rooms>12</Rooms>
  <Category>dom</Category>
  <Price>111000</Price>
  <Location>Humenne</Location>
  <Size>509</Size>
  <Age>15</Age>
  <Isolation>ano</Isolation>
  <TimeOfCreation>2019-04-23T00:00:00+02:00</TimeOfCreation>
</Item>
<Item>
  <RealId>14</RealId>
  <Name>Stary dom na zburanie, zaujimava investicia</Name>
  <Rooms>3</Rooms>
  <Category>dom</Category>
  <Price>10000</Price>
```

```
  <Location>Humenne</Location>
  <Size>150</Size>
  <Age>105</Age>
  <Isolation>nie</Isolation>
  <TimeOfCreation>2019-04-24T00:00:00+02:00</TimeOfCreation>
</Item>
<Item>
  <RealId>15</RealId>
  <Name>Mala garzonka vhodna na AirBNB</Name>
  <Rooms>1</Rooms>
  <Category>byt</Category>
  <Price>60000</Price>
  <Location>Nitra</Location>
  <Size>16</Size>
  <Age>25</Age>
  <Isolation>ano</Isolation>
  <TimeOfCreation>2019-04-25T00:00:00+02:00</TimeOfCreation>
</Item>
<Item>
  <RealId>16</RealId>
  <Name>Mezonet nad stadionom</Name>
  <Rooms>3</Rooms>
  <Category>byt</Category>
  <Price>400000</Price>
  <Location>Nitra</Location>
  <Size>300</Size>
  <Age>10</Age>
  <Isolation>ano</Isolation>
  <TimeOfCreation>2019-04-26T00:00:00+02:00</TimeOfCreation>
</Item>
<Item>
  <RealId>17</RealId>
  <Name>Dom v uplnom centre polovicny podiel</Name>
  <Rooms>4</Rooms>
  <Category>dom</Category>
  <Price>359900</Price>
  <Location>Nitra</Location>
  <Size>250</Size>
  <Age>1</Age>
  <Isolation>nie</Isolation>
  <TimeOfCreation>2019-04-27T00:00:00+02:00</TimeOfCreation>
</Item>
<Item>
  <RealId>18</RealId>
  <Name>Pozemok na ubytovnu na okraji mesta</Name>
  <Rooms>0</Rooms>
  <Category>pozemok</Category>
  <Price>300000</Price>
  <Location>Nitra</Location>
  <Size>1500</Size>
  <Age>0</Age>
  <Isolation>0</Isolation>
  <TimeOfCreation>2019-04-28T00:00:00+02:00</TimeOfCreation>
</Item>
<Item>
  <RealId>19</RealId>
  <Name>Garaz vhodna na prenuservis</Name>
  <Rooms>0</Rooms>
  <Category>garaz</Category>
  <Price>34500</Price>
```

```
    <Location>Nitra</Location>
    <Size>45</Size>
    <Age>56</Age>
    <Isolation>nie</Isolation>
    <TimeOfCreation>2019-04-29T00:00:00+02:00</TimeOfCreation>
  </Item>
</ArrayOfItem>
```