

EKONOMICKÁ UNIVERZITA V BRATISLAVE

FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 103004/I/2015/2482926410

**ASP.NET XML WEBOVÁ SLUŽBA POSKYTUJÚCA
USPORIADANÉ INFORMÁCIE O BANKOVOM ÚČTE**

Diplomová práca

2015

Bc. Andrej Vysočáni

EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

**ASP.NET XML WEBOVÁ SLUŽBA POSKYTUJÚCA
USPORIADANÉ INFORMÁCIE O BANKOVOM ÚČTE**

Diplomová práca

Študijný program: Manažérske rozhodovanie a informačné technológie
Študijný odbor: 6258 Manažérske rozhodovanie a informačné technológie
Školiace pracovisko: Katedra aplikovanej informatiky
Vedúci záverečnej práce: Igor Košťál, Ing., PhD.

Bratislava 2015

Bc. Andrej Vysočáni

Čestné vyhlásenie

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

Dátum: 15.05.2015

.....
(podpis študenta)

Pod'akovanie

Na tomto mieste by som rád poďakoval mojej mame ako aj celej rodine za to, že ma podporovali počas celého štúdia na vysokej škole. Taktiež vedúcemu práce Ing. Igorovi Košťálovi, PhD. za trpezlivosť, pochopenie a celkovú pomoc pri tvorbe tejto diplomovej práce.

ABSTRAKT

Vysočáni, Andrej: ASP.NET XML webová služba poskytujúca usporiadané informácie o bankovom účte – Ekonomická Univerzita v Bratislave. Fakulta hospodárskej informatiky, katedra aplikovanej informatiky. – Vedúci záverečnej práce: Ing. Igor Košťál, PhD. – Bratislava: FHI EU, 2015, 57s

Cieľom diplomovej práce je zanalyzovať problematiku a úroveň platforiem podporujúcich dynamické generovanie webových stránok a naplno sa zasvätiť do sveta masívnej výmeny elektronických dát. Vo vybratom riadenom programovacom jazyku navrhnuť kompletne riešenie ASP.NET XML webovej služby poskytujúcej jej klientovi usporiadané informácie o jeho bankovom účte a následne riešenie aj implementovať. Popri tom aj celú štruktúru ktorú si riešenie vyžaduje.

Kľúčové slová:

Analýza, návrh, implementácia, asp.net, webová služba, c#

ABSTRACT

Vysočáni, Andrej: An ASP.NET XML web Service Providing Sorted Information on an Offered Bank Drafts – The University of Economics in Bratislava. The Faculty of Economic Informatics; Department of Applied Informatics. – The supervisor of final thesis: Ing. Igor Košťál, PhD. – Bratislava: FHI EU, 2015, 57p

The aim of the thesis is to analyze problematic and level of platforms that support dynamic generating of web pages and fully understand the world of massive interchange of electronic data. In chosen driven programming language design whole solution for ASP.NET XML webservice to offer to the client summary of his bank account balance and implement the solution. Also whole infrastructure with it.

Keywords:

Analysis, design, implementation, asp.net, webservice, c#

OBSAH

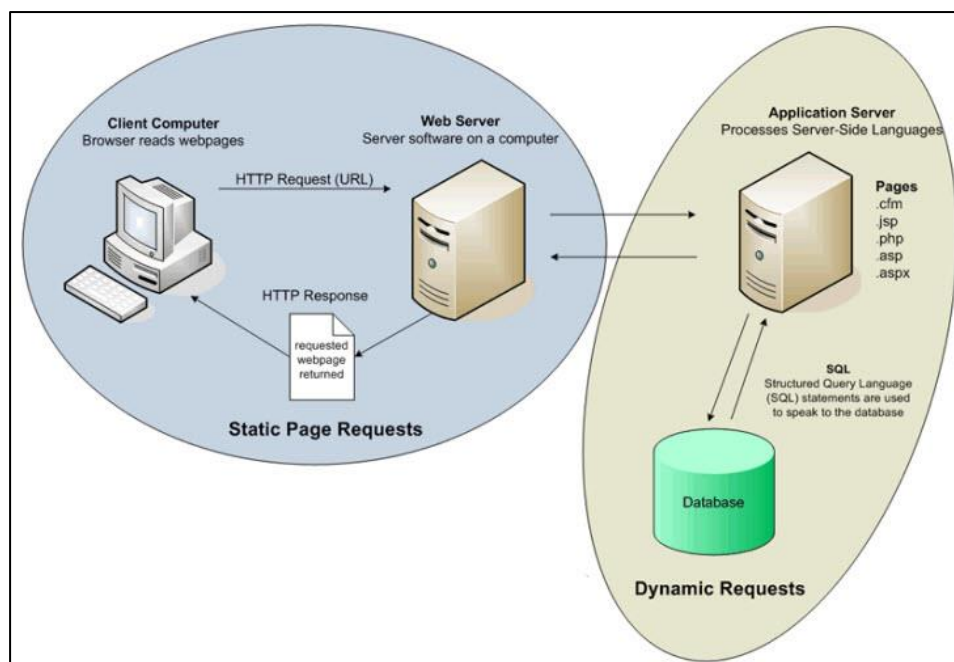
Úvod.....	9
1. Základné princípy webových služieb	10
1.1. Dynamicky generované stránky	10
1.2. Web server	10
1.3. Databázový server	11
1.4. Webová služba a jej elementy	11
1.5. Architektúra webovej služby.....	16
1.6. XML (extensible markup language)	17
1.7. DTD(Document Type Definition).....	20
1.8. XML Schema	20
1.9. XML namespaces.....	21
1.10. Metadáta	21
1.11. XML-RPC (remote procedure call).....	22
1.12. SOAP (Simple Object Access Protocol)	22
1.13. REST (Representational State Transfer)	23
1.14. JSON (Javascript Object Notation)	24
1.15. WSDL (Web Services Description Language).....	25
1.16. UDDI (Universal Description, Discovery and Integration)	25
2. ASP.NET	26
2.1. Výhody technológie .NET	26
2.2. Alternatívne platformy	30
2.2.1. JSP (Java Server Pages)	30
2.2.2. Nevýhody ASP/JSP/PHP	30
2.3. ADO.NET	32
3. Cieľ a metodika práce	34
4. Výsledky práce a diskusia.....	34

4.1. Analýza a návrh.....	34
4.1.1. Návrh a požiadavky na systém.....	34
4.1.2. Funkčné požiadavky	35
4.1.3. Návrhové zobrazenie.....	36
4.1.4. Opis aplikácie pomocou UML modelov	39
4.2. Implementácia	41
4.2.1. Databáza.....	41
4.2.2. Webový server a programovacie nástroje	45
4.2.3. Náhľad aplikácie	47
Záver	54

Úvod

Rozhodnutie či webová stránka má byť implementovaná ako statická stránka alebo ako dynamicky generovaná stránka, záleží od samotnej funkcionality a určenia stránky. Internetová stránka kde jednotlivé strany existujú ako konečné súbory na serveri predstavuje statickú stránku, naproti tomu stránka kde sú strany výsledkom výpočtov zbiehajúcich zakaždým keď je poslaná požiadavka na dokument musí byť implementovaná ako dynamicky generovaná stránka. Väčšina obsahu na internete je statická a nemení sa alebo sa mení veľmi sporadicky, na rozdiel od generovania z templejtv a databázových systémov. V týchto prípadoch je celý obsah stránky akoby skompilovaný pred nasadením a samotné statické strany sú uložené ako súbory, ku ktorým server priamo pristupuje. Príkladom takýchto stránok by mohli byť stránky prezentujúce firmy. Akonáhle je raz stránka vytvorená so všetkými základnými údajmi a údajmi, ktoré bližšie popisujú činnosti firmy pre potenciálneho zákazníka, nie je väčšinou potrebné meniť obsah stránky po dobu čo je online. V ostatných prípadoch keď sa obsah stránky často mení a záleží od inputu používateľa, nie je tento prístup vyhovujúci. Príkladom pre takúto stránku môže byť internet banking, kde musí byť obsah vytváraný za pochodu vzhľadom na požiadavky od používateľa. Obsah ktorý sa má zobrazit' sa veľmi často mení a závisí od zadaných vstupov ako napríklad výpis účtu. Používateľ zadáva číslo účtu a následne sa mu zobrazí dynamicky vygenerovaná stránka s výpisom transakcií nad účtom. Samozrejme čo sa týka výhod a nevýhod jednotlivých prístupov obe majú to svoje. Výhody statických stránok sú rýchle a efektívne, keďže nevyužívajú prístup k databáze nie je veľký priestor pre vznik bezpečnostných rizík a sú kompatibilné s akýmkoľvek typom webového servera. Na druhej strane musia byť nanovo nasadené po vykonaní určitých zmien, inak ich používatelia neuvidia a celkovo ich rozvoj je pomerne obmedzený. Dynamické stránky zasa majú výhodu, že ponúkajú obsah šitý na mieru používateľovi a zmeny sú ľahko aplikovateľné. No takéto stránky sú náročnejšie na zdroje a tým, že pracujú nad databázou môže dôjsť ľahko k problémom so zabezpečením potenciálne citlivých údajov.

Na obrázku (obr.1) nižšie možno vidieť na modeloch rozdiel medzi staticky generovanou stránkou a dynamicky generovanou stránkou. Toľko k možnostiam a prístupom k tvorbe webových stránok a aplikácií. Ako je už pravdepodobne aj z uvedených príkladov vyššie jasné, táto práca pojednáva o dynamickom prístupe.



obr. 1 Staticky generovaná webová stránka vs. Dynamicky generovaná webová stránka [1]

1. Základné princípy webových služieb

1.1. Dynamicky generované stránky

Platforma poskytujúca dynamické stránky s prístupom k databáze pozostáva najmä z webového servera a databázového servera. Na webovom serveri beží aplikácia ktorá pristupuje k databázovému serveru.

1.2. Web server

Webový server je v podstate program, ktorý poskytuje sieťový prístup k webovým stránkam. Jeho hlavnou náplňou je odpovedať na HTTP požiadavky od webových klientov. Požiadavka je vo forme URL (Uniform Resource Locator). V momente keď sa pošle požiadavka, web server kontroluje či súbor existuje a či je prístupný a následne vracia stránku do okna klienta. Každý počítač na internete, na ktorom je uložená webová stránka musí mať program predstavujúci webový server, aby bol schopný reagovať na požiadavky. Bližšie si webový server možnosti výberu ako aj konkrétny výber servera pre túto prácu popíšeme trocha neskôr.

1.3.Databázový server

Databázový server predstavuje program, ktorý má na starosti databázu a odpovedanie na požiadavky vo formáte SQL (Structured Query Language). Databáza uchováva dáta pre aplikáciu, organizovanou formou, takže sú ľahko prístupné, manažovateľné a aktualizované. Mnohé programovacie jazyky majú databázové rozhranie používané na komunikáciu s databázou. Databáza beží nezávisle od ostatných programov a potrebuje iba rozhranie aby ho ju mohol program dotazovať. Databázový server je v tejto práci tiež priblížený ešte neskôr.

1.4.Webová služba a jej elementy

Skôr ako si presne definujeme pojem webová služba pomocou rôznych exaktných výrazov, skúsme si utvoriť jednoduchú predstavu o čo ide a najmä nazrieť na ich praktické využitie. Problematika webových služieb nie je ani taká komplikovaná ako ju môžu vyobrazovať všetky zložité pojmy a termíny s ňou spojené. Povedzme, že delíme ľudí na dve skupiny. Jednu tvoria ľudia, ktorí disponujú informáciami a majú vôľu ich poskytnúť iným. Druhá skupina sú logicky tí, ktorí majú o tieto informácie záujem. Vezmime si napríklad internetové denníky či spravodajské portály. Hľadám na každom takomto portáli je niekde v rohu zobrazená predpoveď počasia. Zdrojom údajov býva slovenský hydrometeorologický ústav (SHMÚ) prípadne iná organizácia poskytujúca dáta o počasi. Ako však tieto dáta dostanem na svoj portál? Poskytovateľ by mohol zasielať dáta napríklad v obyčajnom textovom formáte a administrátori portálov by ich po prípadnej konverzii s grafickou úpravou vkladali na stránky. Na druhej strane musí predsa existovať nejaký jednoduchší, efektívnejší a najmä štandardizovaný spôsob získavania požadovaných dát. Spôsob ktorý by v prvom rade bolo možné zautomatizovať. Taký spôsob samozrejme existuje a dáta môžeme získavať pomocou webovej služby za pomoci XML. Podobne by sme sa mohli zmýšľať aj pri získavaní aktuálnych menových kurzov od zdroja ako je napríklad Európska centrálna banka, no pole využitia je veľmi široké.

Bližšie sa technológií XML venujeme neskôr v texte. Zatiaľ nám stačí vedieť, že ide o určitý štandardizovaný formát slúžiaci práve na prenos údajov medzi aplikáciami a publikovanie dokumentov a okrem samotných údajov popisuje aj ich štruktúru. XML je teda dobre čitateľný pre počítače aj pre ľudí, no ideálne je, keď si stroje vymenia dáta a používateľ o tom ani nevie. Návštevník spravodajského portálu tak napríklad ani netuší

ako a odkiaľ sa berú hodnoty teplôt či aktuálnych menových kurzov, koniec koncov prečo by aj mal.

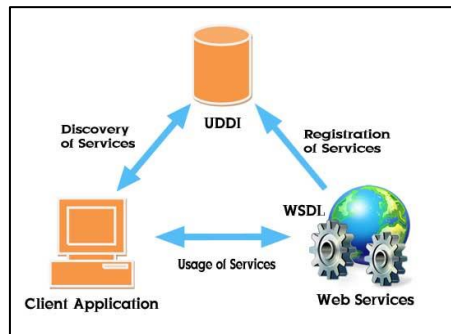
Webová služba umožňuje vytvárať dynamické stránky. Obyčajným HTML kódom by sme nedokázali poskytovať používateľovi dáta šité na mieru. Vďaka webovej službe sa požiadavka na dáta, napríklad nejaké osobné údaje v databáze, spracuje na strane servera a výsledok je integrovaný do statického HTML a takáto stránka sa zobrazí používateľovi.

Konzorcium W3C, ktoré predstavuje medzinárodnú organizáciu určujúcu webové štandardy, definuje webovú službu ako: *softvérový systém navrhnutý na podporu spolupráce počítačov cez sieť. Má rozhranie popísané v strojovo-spracovateľnom formáte (konkrétne WSDL). Ostatné systémy komunikujú s webovou službou spôsobom definovaným v popise služby za použitia SOAP správ, typicky za použitia HTTP s XML serializáciou v spojení s inými webovo orientovanými štandardmi.* [2]

Webová služba, jednoducho povedané, predstavuje mechanizmus typu požiadavka – odpoveď, ktorý umožňuje vzdialený prístup k dátam, prípadne ich modifikáciu. Jeden počítač (klient) požaduje nejaké dáta a druhý počítač (server) reaguje na požiadavku.

Iná definícia môže znieť odlišne ako tá od W3C no podstata ostáva zachovaná: *Webové služby sú transakčné (transactional) medziaplikačné (cross application) nástroje, ktoré sa riadia všeobecnými štandardmi a protokolmi. Webová služba typicky poskytuje jednu funkciu, no tieto funkcie môžu byť ľubovoľne zložené a zložené z viacerých služieb.* [3]

Ďalšia definícia hovorí o webovej službe ako o Softvérovej aplikácii identifikovanej prostredníctvom URI (Uniform Resource Identifier), ktorej interfejsy a väzby je možné definovať, opísať a vyhľadávať ako artefakty XML. Podporuje priamu interakciu s inými softvérovými aplikáciami prostredníctvom správ zapísanými v jazyku XML a prenášanými protokolmi internetu. [4]



obr. 2 Jednoduchý model fungovania webových služieb [5]

Vráťme sa späť k nášmu predošlému príkladu kde sa ľudia delili na dve skupiny a teda tí čo ponúkajú informácie a tí čo žiadajú informácie. Informácia nech je teda určitá webová služba. Aby táto služba mohla pracovať a aby bolo možné s ňou pracovať, na oboch stranách stoja hardvérové a softvérové nástroje implementujúce túto službu. Rozdelené skupiny označme ako žiadateľ (klient) a poskytovateľ. Jeden teda službu poskytuje a druhý sa jej dožaduje, aby mohol využiť jej výstup pre svoje záujmy. Vo všeobecnosti existujú dva typy klientov webových služieb resp. dva spôsoby prístupu k nim [6]:

1. *Pomocou knižníc implementujúcich protokol SOAP. Tieto knižnice sú dodávané s produktmi implementujúcimi webové služby. Príkladom takýchto produktov je Sun JWSDP, Apache Axis, MS SOAP Toolkit a podobne. Tento spôsob je určený pre externé aplikácie.*
2. *Prostredníctvom klientských aplikácií určených pre koncových používateľov.*

Webové služby umožňujú rôznym aplikáciám využívať rôzne technológie komunikácie bez potreby nákladného dodatočného programovania. Väčšina webových služieb využíva pri komunikácii XML. Webové služby sa neviažu na jeden konkrétny programovací jazyk, hardvér alebo operačný systém. To znamená, že napríklad Java aplikácia využívajúca Oracle, dokáže komunikovať s Windows aplikáciou využívajúcou SQL server. Táto flexibilita je dosiahnutá práve vďaka využívaniu otvorených štandardov.

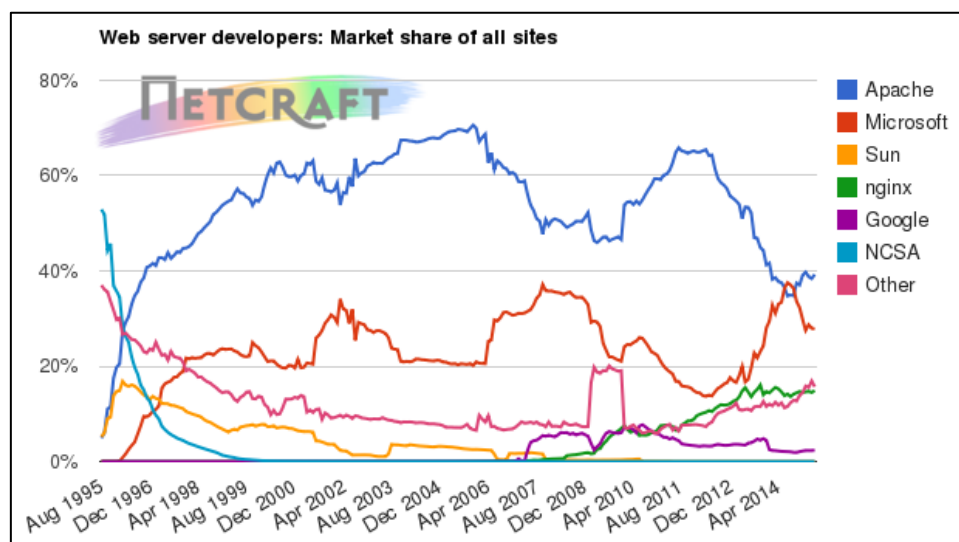
Ak by sme to mali zosumarizovať, tak webová služba je každá služba, ktorá:

- Je dostupná cez internet alebo súkromné (intranet) siete
- Používa štandardizované XML

- Nie je viazaná na žiaden operačný systém alebo programovací jazyk
- Popisuje samú seba prostredníctvom spoločnej XML syntaxe
- Je možné ju nájsť pomocou простého vyhľadávacieho mechanizmu

Pri riešení môžeme teda využívať či už produkty od Microsoftu, ktoré sú však väčšinou spoplatnené, alebo môžeme nájsť vhodnú bezplatnú alternatívu. S operačným systémom Windows ide ruka v ruke webový server Internet Information Server (IIS), databázový server MSSQL server a skriptový systém ASP.NET spolu s najčastejšie používanými programovacími jazykmi C# či Visual Basic. Alternatívou pre prevažne korporátne riešenie od Microsoftu je operačný systém Linux s využitím webového servera Apache, databázového servera MySQL a skriptového systému PHP. Nie je to však pravidlo a môžu sa využiť rôzne nástroje. Napríklad existuje aj Windows verzia webového servera Apache či databázového servera MySQL. Podľa prieskumov portálu Netcraft v rámci porovnania využitia webových serverov je jasným lídrom Apache, ktorý v apríli 2015 dosiahol trhovú podiel 39,25%. Microsoft s ich serverom IIS si drží druhú pozíciu s podielom 27,83%. [7]

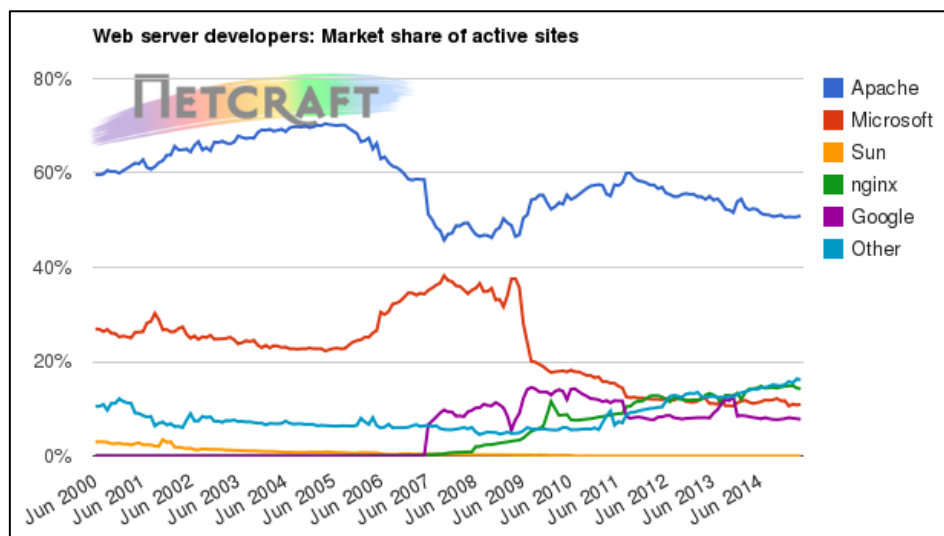
Spoločnosť Netcraft robí prieskum využitia webových serverov na trhu na mesačnej báze a výsledky za apríl 2015 možno vidieť nižšie na obrázkoch 1-6.



obr. 3 Trhový podiel webových serverov v rámci všetkých stránok[4]

Developer	March 2015	Percent	April 2015	Percent	Change
Apache	337,175,536	38.39%	333,285,741	39.25%	0.87
Microsoft	245,496,533	27.95%	236,288,843	27.83%	-0.12
nginx	127,191,696	14.48%	126,274,778	14.87%	0.39
Google	20,097,702	2.29%	20,051,433	2.36%	0.07

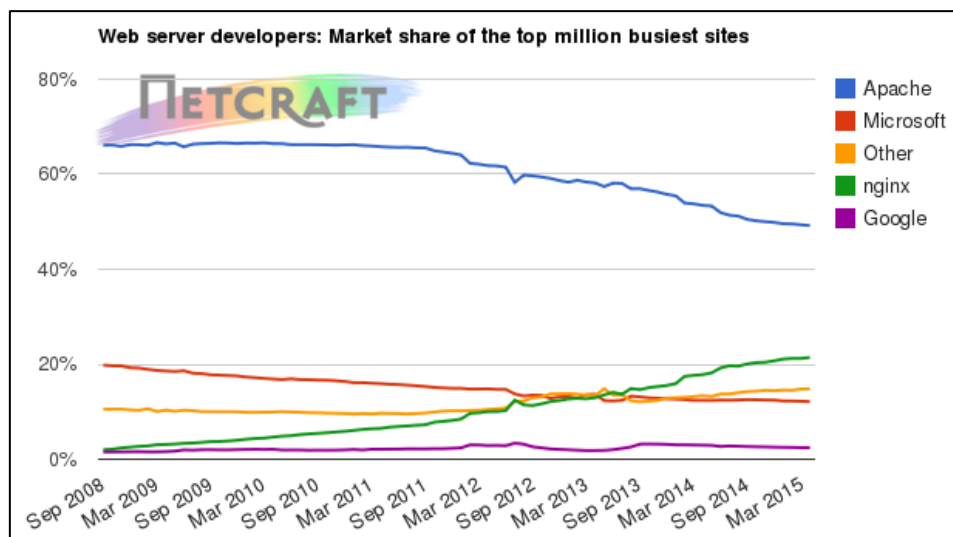
obr. 4 Tabuľkový prehľad trhového podielu webových serverov v rámci všetkých stránok[4]



obr. 5 Trhový podiel webových serverov v rámci aktívnych stránok[4]

Developer	March 2015	Percent	April 2015	Percent	Change
Apache	90,254,111	50.66%	90,001,047	50.91%	0.25
nginx	25,591,447	14.36%	25,174,837	14.24%	-0.12
Microsoft	19,160,056	10.75%	19,353,327	10.95%	0.19
Google	14,041,104	7.88%	13,712,694	7.76%	-0.12

obr. 6 Tabuľkový prehľad trhového podielu webových serverov v rámci aktívnych stránok[4]



obr. 7 Trhový podiel webových serverov v rámci top milión najvyťaženejších stránok[4]

Developer	March 2015	Percent	April 2015	Percent	Change
Apache	493,463	49.35%	491,868	49.19%	-0.16
nginx	212,151	21.22%	214,310	21.43%	0.22
Microsoft	122,069	12.21%	121,210	12.12%	-0.09
Google	24,434	2.44%	24,293	2.43%	-0.01

obr. 8 Tabuľkový prehľad trhového podielu webových serverov v rámci top milión najvyťaženejších stránok[4]

1.5. Architektúra webovej služby

Ako uvádza portál Tutorialspoint existujú dva pohľady na architektúru webovej služby [8]:

- Z hľadiska rolí v rámci webovej služby
- Z hľadiska súboru protokolov webovej služby

V rámci webovej služby ako takej rozoznávame 3 hlavné role:

- *Poskytovateľ služby* – poskytuje webovú službu. Implementuje ju a sprístupňuje ju na internete.
- *Žiadateľ služby* – akýkoľvek používateľ služby. Vytvára sieťové spojenie a posiela XML požiadavku.
- *Registrátor služby* – centralizovaný súbor služieb. Poskytuje centrálny priestor kde môžu vývojári zverejňovať nové služby alebo nájsť existujúce.

Ako bolo spomenuté druhým spôsobom ako nazerať na architektúru webovej služby je z hľadiska jednotlivých vrstiev protokolov ktoré celkovo zabezpečujú fungovanie služby:

- *Transport služby* – vrstva zodpovedná za prenos správ medzi aplikáciami. Typickými protokolmi sú hypertext transfer protocol (HTTP), Simple Mail Transfer Protocol (SMTP), file transfer protocol (FTP). Vďaka využívaniu protokolu http dokážu webové služby poľahky preniesť dáta cez firewall, nakoľko daný protokol mu je známy a nepovažuje ho za prípadnú hrozbu.
- *Správy XML* – vrstva zodpovedná za kódovanie správ do bežného XML formátu aby mohli byť správy na oboch stranách prečítané. Táto vrstva v súčasnosti zahŕňa XML-RPC a SOAP.
- *Popis služby* – vrstva zodpovedná za popis verejného rozhrania pre konkrétnu webovú službu. Popis je realizovaný na základe Web Service Description Language (WSDL).
- *Objavenie služby* – vrstva zodpovedná za centralizáciu služieb v spoločnom registri a poskytujúca jednoduchú funkciu publikácie a hľadania. V súčasnosti je táto funkcionálna zabezpečená pomocou Universal Description, Discovery, and Integration (UDDI).

1.6.XML (extensible markup language)

XML je skratka pre eXtensible Mark-up Language a bol vyvinutý v roku 1998 štandardizačnou organizáciou pre web W3C(World Wide Web Consortium). XML je v podstate štruktúrovaný textový dokument, ktorý je pomerne ľahko čitateľný pre človeka a spracovateľný pre stroj resp. aplikáciu. Predstavuje okresanejšiu a viac flexibilnú verziu svojho predchodcu SGML(Standard General Mark-up Language), ktorý vznikol v osemdesiatych rokoch po tom ako ho americký štandardizačný inštitút ANSI(American National Standards Institute) a tiež medzinárodná štandardizačná organizácia ISO(International Organization for Standardization) uznali ako štandard.

SGML pritom bol ešte odvodený od jazyka GML(Generalized Markup Language), ktorý vznikol niekedy v šesťdesiatych rokoch. [9]

Nevýhodou SGML bola neschopnosť reagovať na požiadavky webu. Vznikol v dobe pomalých a drahých počítačov. Aby bolo možné z týchto systémov získať maximum, štandard SGML bol vybavený minimalizačnými nástrojmi. Tie mali výsledné súbory stlačiť na čo najmenšiu veľkosť. Výsledkom boli drahé, náročné a pomalé analyzátory a finančná náročnosť zavádzania SGML do praxe.

Miesto priameho použitia štandardu SGML veľké rozšírenie získala iba jeho aplikácia HTML(Hyper Markup Language) ktorú vytvoril Tim Berners-Lee v roku 1995. Aby však dáta mohli spracovávať aj stroje bolo potrebné dôslednejšie formátovať údaje a mať jasné pravidlá a keďže HTML k tomu nestačí a SGML je zložitý, tak prichádza na scénu XML. [4]

XML je značkovací jazyk a teda používa tzv. „tagy“ na označenie, kategorizáciu, a organizovanie informácií jasne určeným spôsobom. Pozostáva z elementov a každý musí mať okrem otváracieho tagu aj zatvárací.(`<meno>Andy</meno>`). Text, dáta, či obrázky môžu byť obsiahnuté v značkovacích tagoch. Neexistuje limitácia na konkrétny súbor značiek, naopak tvorca vytvára vlastné značky, ktoré spĺňajú jeho potreby a požiadavky na dokument. Flexibilita XML zabezpečila jeho široké rozšírenie v oblasti výmeny dát na internete.

XML je dokument, ktorý má jasné syntaktické pravidlá a špecifikácie nakoľko ide o štandard. Ak dokument spĺňa tieto pravidlá či už v rámci syntaxe alebo logiky, tak hovoríme o tzv. „well-formed“ dokumente, čo by sme mohli preložiť ako správne štruktúrovaný dokument. Aby toto dokument dosiahol musí spĺňať základné pravidlá [10]:

- XML dokument musí mať koreňový element
- XML elementy musia obsahovať zatvárací tag
- XML tagy sú citlivé na veľké písmená
- XML elementy musia byť správne vnorené
- hodnoty atribútov musia byť v úvodzovkách

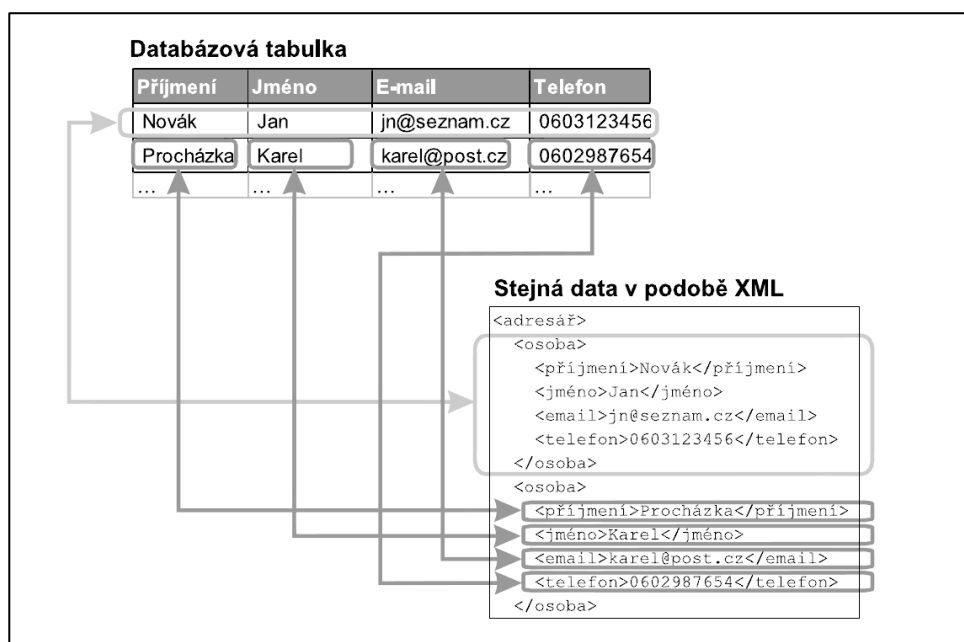
Ak je dokument well-formed a zároveň spĺňa definíciu v dokumente DTD alebo XML schéma, potom môžeme dokument označiť tiež ako validný.

XML bolo navrhnuté aby zvládlo elektronické publikovanie vo veľkých merítkach a hrá dôležitú rolu vo výmene širokého spektra dát na webe. XML tvorí základ webových služieb a aj vďaka tomu je to pomerne mocný značkovací jazyk aj napriek tomu, že je už starý asi 17 rokov.

Jednoduchý príklad XML formátu:

```
<HR>
  <Zamestnanec>
    <Meno>Peter</Meno>
    <Vek>35</Vek>
    <Mesto>Bratislava</Mesto>
  </Zamestnanec>
  <Zamestnanec>
    <Meno>Jozef</Meno>
    <Vek>43</Vek>
    <Mesto>Košice</Mesto>
  </Zamestnanec>
</HR>
```

Nižšie na obrázku (obr.9) môžeme vidieť, že XML si dokáže poradiť aj s interpretáciou dát z relačnej databázy.



obr. 9 Znáozornenie interpretácie databázových dát do XML [11]

XML obsahuje samotné dáta. Jeho štruktúru popisujú DTD alebo XML Schema a prezentačnú vrstvu má na starosti štandard XSL. O nich si niečo bližšie popíšeme nižšie.

Podstatu samotnej komunikácie umožňujú protokoly, ktoré XML využívajú. Najčastejšími sú XML-RPC, SOAP či REST.

1.7.DTD(Document Type Definition)

XML umožňuje definovať vlastnú sadu značiek, ktoré sa budú v dokumente využívať. Slúži nám na to dokument DTD, ktorým definujeme aké značky môže XML obsahovať. Následne je možná kontrola pomocou parsera, či XML obsahuje iba značky, ktoré sú žiadané. Ide teda o sadu značkovacích deklarácií ktoré definujú typ dokumentu pre značkovacie jazyky XML ale aj HTML či SGML, keďže sú akoby príbuzné. Pri tvorbe XML dokumentu nie je DTD nevyhnutne potrebné, keďže XML ako také dodržiava striktné pravidlá vytvárania, no DTD pomáha udržať dokument prehľadný a konzistentný. DTD sa deklaruje v „DOCTYPE“ deklarácii pod XML deklaráciou obsiahnutou v XML dokumente.

```
<?xml version="1.0"?>  
<!DOCTYPE documentelement [definition]>
```

DTD sa v súčasnosti používa menej, v oveľa väčšom merítku sa používa XML schéma. Niektorí môžu dokonca tvrdiť že úplne vytlačila DTD, no to má stále svoje minoritné zastúpenie a fanúšikov a firemných aplikácií, ktoré používajú XML dokumenty a boli vytvorené v dobe pred XML schémami.

1.8.XML Schema

Keď sa dve strany rozhodnú zdieľať dáta a komunikovať prostredníctvom XML správ, musí byť jasne definovaná syntax a sémantiku správ. XML schéma alebo tiež XSD (XML Schema Definition) je jazyk ktorý slúži na opis štruktúry XML dokumentu. Ako už bolo spomenuté je to v podstate náhradník za DTD. Schéma sa definuje v externom XML dokumente a poskytuje väčšiu flexibilitu ako DTD. Základným predpokladom prečo je XML schéma populárnejšia je fakt, že je napísaná v jazyku XML, kdežto DTD je

odvodené od SGML a teda ide o prakticky iný jazyk. Tento fakt samotný mnohých vývojárov odstrašil, keďže by sa museli učiť nový jazyk. Existujú však aj ďalšie rozdiely, ktoré pekne zhrnul používateľ na známom vývojárskom fóre stackoverflow [12]:

- XML schéma definuje dátové typy pre elementy a atribúty, zatiaľ čo DTD nepodporuje dátové typy.
- XML schéma poskytuje podporu pre menné priestory, zatiaľ čo DTD nie.
- XML schéma definuje číslo a poradie detských elementov, kdežto DTD nie.
- XML schémou môže byť manipulované pomocou XML DOM, no s DTD to možné nie je.
- V rámci XML schémy môže odosielateľ opísať dáta tak aby príjemateľ porozumel, no v prípade DTD môže dôjsť k zlej interpretácii dát.
- XML schéma je rozšíriteľná, kdežto DTD nie je.

V XML schéme definujeme dátové typy, ktoré môžu byť jednoduché alebo komplexné. *Jednoduchý* dátový typ definuje elementy a atribúty textovou hodnotou a neobsahuje žiadne sub-elementy. *Komplexný* dátový typ môže naopak obsahovať ďalšie elementy s atribútmi.

1.9.XML namespaces

XML namespaces, alebo menný priestor, umožňuje unikátnu identifikáciu elementov a atribútov v XML dokumente, ich spájaním s menným priestorom identifikovaným pomocou URI(Uniform Resource Identifier). Pomáha eliminovať nejednoznačnosť medzi rovnako pomenovanými elementmi alebo atribútmi, najmä v prípade práce s rôznymi XML dokumentmi z rôznych zdrojov. Je možné ho priradiť určenému elementu alebo bloku elementov.

1.10. Metadáta

Metadáta predstavujú určité dáta o dátach. Sú to údaje ako autor dokumentu, dátum vytvorenia, druh dokumentu a podobne. Je dobré poznať čo najviac metadát o dokumente či už pre jednoduchšie vyhľadávanie alebo pre klasifikáciu dokumentu. Jedným

z najznámejších formátov pre zápis a výmenu metadát je RDF(Resource description Framework), ktorý umožňuje k ľubovoľnému dokumentu pripojiť ľubovoľné metadáta. Od Microsoftu zasa existuje formát CDF(Chanell Definition Format), ktorý svojou jednoduchou syntaxou, založenou na XML, umožňuje definovať zaujímavé internetové zdroje. [11]

1.11. XML-RPC (remote procedure call)

XML-RPC (remote procedure call) je ako z názvu vyplýva protocol určený na volanie vzdialených procedúr.

XML-RPC správa predstavuje HTTP-POST požiadavku, obsahujúcu názov volanej metódy a parametre, pričom jej telo je vo formáte XML, ktoré definuje jej dátové typy. Parametre procedúr môžu tvoriť čísla, dátumy, stringy ako aj komplexné záznamové a zoznamové štruktúry. [13]

Príklad XML-RPC požiadavky:

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181

<?xml version="1.0"?>
<methodCall>
  <methodName>examples.getStateName</methodName>
  <params>
    <param>
      <value><i4>41</i4></value>
    </param>
  </params>
</methodCall>
```

1.12. SOAP (Simple Object Access Protocol)

SOAP protokol vznikol ako nasledovník XML-RPC, ktorý nesie prvky predstavujúce bezpečnostné riziko a problém s kompatibilitou. Firewall a proxy servery

takýto tok dát bežne blokujú. Lepším spôsobom komunikácie medzi aplikáciami je cez HTTP, nakoľko ten podporujú všetky internetové prehliadače. Práve SOAP prenáša správy založené na XML hlavne pomocou HTTP. Používa sa pritom metóda POST. Pozostáva zo základných elementov [2]:

- Obálka (envelope element) umožňuje identifikovať XML dokument ako SOAP správu.
- Hlavička (header element) obsahuje nejaké doplnkové informácie.
- Telo (body element) je nutný element obsahujúci informácie o samotnej požiadavke resp. odpovedi.
- Chybový (fault) element obsahuje informácie o chybách.

Jednoduchý príklad SOAP správy:

```
<Envelope>
  <Header/>
  <Body>
    <Platba>
      <Meno>Andy</Meno>
      <Ucet>1234 5678 9000 1234</Ucet>
      <Transakcia>platba</Transakcia>
      <Suma>12</Suma>
    </Platba>
  </Body>
</Envelope>
```

1.13. REST (Representational State Transfer)

REST (Representational State Transfer) v podstate spočíva v jednoduchom využití http na volania metód a návratom je XML. používa metódy GET, POST, PUT a DELETE na vykonanie CRUD (Create, Read, Update, Delete) operácií na danom objekte. Parametre sú posielané ako bežné http parametre v dotazovacom stringu GET alebo POST požiadavky. REST je viac menej štýl architektúry resp. dizajnový model webovej služby a je považovaný za prevládajúci model v súčasnosti. Dôvodom jeho expanzie a vytlačania SOAP je jeho jednoduchosť. [14]

Webové služby využívajúce REST zakladajú na 4 základných dizajnových princípoch:

- Používanie HTTP metód explicitne

- Bezstavovosť
- URI vo forme adresárovej štruktúry
- Prenos XML, JSON(Javascript Object Notation) alebo oboch

1.14. JSON (Javascript Object Notation)

JSON ako alternatíva k XML je otvorený štandard s textovým základom navrhnutý so zreteľom na dobrú čitateľnosť prenášaných dát.

JSON v súčasnosti zažíva rozkvet pre jeho jednoduchosť najmä pri spolupráci s javascriptom, čo sa dá dedukovať už zo samotného názvu. Napriek tomu však je nezávislý od programovacieho jazyka a platformy. JSON má jednoduchšiu syntax, ktorá sa ľahšie číta a kde prakticky ide o textovú reprezentáciu javascript objektu. Vzhľadom na to, že v XML je potrebné aplikovať uzatváracie tagy kdežto v JSON nie, tak zaberá aj o niečo menej priestoru.

Jednoduchý príklad JSON formátu:

```
{
  "Zamestnanec":
  {
    "Meno": "Peter",
    "Vek": "35"
  }
}
```

Zaujímavosťou a možno istou prekážkou pri využívaní JSON, ako poukázal na svojom blogu jeden používateľ, je klauzula v samotnej licencií používania, v ktorej stojí, že softvér využívajúci JSON bude použitý pre dobro a nie pre zlo. [15]

Na jednej strane ide pravdepodobne o žart, no na strane druhej je to právny dokument a právnici majú často slabý zmysel pre humor.

1.15. WSDL (Web Services Description Language)

Ďalším pojmom, ktorý ide ruka v ruke s webovou službou je jazyk WSDL, ktorý opisuje, aké funkcie ponúka webová služba a spôsob, ako sa jej to opýtať. Zapisuje sa vo formáte XML. Spravidla teda opisuje SOAP komunikáciu.

Podporované operácie a správy sú opísané abstraktne a potom sa obmedzujú na konkrétny sieťový protokol a formát správy. Toto znamená, že WSDL opisuje verejné rozhrania webovej služby.

WSDL sa často používa v kombinácii so SOAP a XML, aby poskytovalo webové služby cez internet. Klientský program pripájajúci sa k webovej službe vie čítať WSDL, aby zistil, aké funkcie sú dostupné na serveri. Akékoľvek použité špeciálne dátové typy sú uložené vo WSDL súbore vo formáte XML. Program môže používať SOAP na zavolanie funkcií napísaných vo WSDL.

1.16. UDDI (Universal Description, Discovery and Integration)

UDDI predstavuje akýsi platformovo nezávislý register webových služieb, kde je možné registrovať a lokalizovať aplikácie webových služieb. Umožňuje firmám zverejniť svoje webové služby a zároveň dozvedieť sa o službách iných. Poskytuje špecifikácie k jednotlivým službám a definície ako služby či softvérové aplikácie navzájom spolupracujú na internete. [17]

UDDI využíva štandardy ako XML, http a DNS protokoly. Ďalej používa WSDL na popis rozhraní k webovým službám a pre zabezpečenie komunikácie medzi rôznymi platformami prijíma SOAP protocol. [18]

Z UDDI profituje podnik nech je akokoľvek veľký. Pred UDDI neexistoval žiaden internetový štandard pre informovanie zákazníkov či partnerov o produktoch a službách. A nebol ani spôsob akým integrovať vzájomné systémy a procesy.

Čo to vlastne UDDI umožňuje? Povedzme, že by bol napríklad vytvorený UDDI štandard pre letové rezervácie, letecké spoločnosti by potom mohli registrovať ich služby

do UDDI zložky. Cestovné kancelárie by následne mohli vyhľadať rozhranie rezervácií leteniek a komunikovať priamo s danou službou.

2. ASP.NET

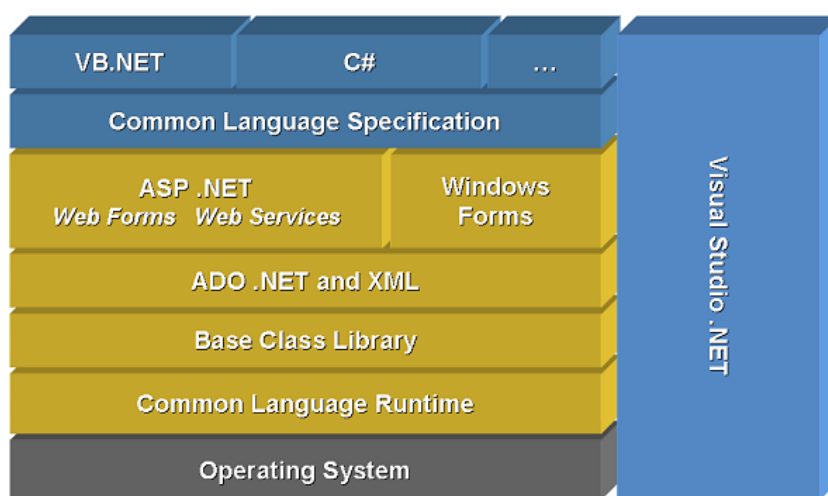
V dnešnej dobe sú hlavnými hráčmi vývojových platforiem Microsoft či Sun Microsystems. Základným rozdielom je zrejme v tom, že Sun Microsystems je na rozdiel od Microsoftu voľne dostupný. Aj napriek tomu Microsoft ponúka voľne dostupnú verziu pre vyskúšanie programovania v ich vývojovom prostredí. Táto platforma sa nazýva Visual Studio Express. Každá z týchto platforiem má svoj programovací jazyk. U Sun Microsystemu je to JAVA a Microsoftu buď klasické C ++ či technológia .NET. JAVA má tú výhodu, že je možné ju použiť pre viac operačných systémov. Na rozdiel od toho možno technológii .NET použiť iba v systéme Windows.

Framework od Microsoftu je vývojový systém, ktorý slúži na podporu programovania ako webových aplikácií, tak aj klasických aplikácií pre MS Windows.

2.1.Výhody technológie .NET

Platforma .NET Framework je od počiatku kompletne založená na objektovo orientovaných princípoch. Použité jazyky v technológii .NET, tj. Visual Basic, C #, J #, a spravované C ++, kompilujú do spoločného sprostredkujúceho jazyka (Intermediate Language). Oproti prostrediu ASP je lepšia podpora dynamických webových stránok. .NET ponúka integrovanú podporu webových stránok pomocou novej technológie ASP.NET, v rámci ktorej je kód stránok zostavený a možno ho písať v jazyku vysokej úrovne, kompatibilným s technológiou .NET. Sada komponentov .NET súhrnne označovaná ako ADO.NET zaisťuje efektívny prístup k relačným databázam a rôznym zdrojom dát. Navyše sú k dispozícii komponenty, ktoré poskytujú prístup k súborom a k adresárom. Najmä je do technológie .NET integrovaná podpora jazyka XML, čo umožňuje manipuláciu s dátami, ktoré možno importovať alebo exportovať na iné platformy ako Windows. Zásadným spôsobom mení technológia .NET zdieľanie kódu medzi aplikáciami,

kde zavádza koncepciu zostavenia (assembly), ktoré nahrádzajú tradičné knižnice DLL. Zjednodušene môžeme povedať, že technológia .NET je jednoduchá na vývoj - ten je úplne objektovo orientovaný, umožňuje automatické riadenie životného cyklu objektov, je zameraný na aplikačnú logiku nie na kontrolu aplikačnej infraštruktúry. .NET Odstraňuje komunikačné bariéry medzi komponentmi a aplikáciami - zjednodušuje integráciu, uľahčuje znovupoužiteľnosť kódu a čiastkových alebo aj celých komplexov riešenia. Bezpečnosť je zaistená identifikáciou kódu, ďalej je prítomná typová bezpečnosť. Odstraňuje problémy s komponentmi COM. Umožňuje vyvíjať aplikácie aj pre mobilné zariadenia. [19]



obr. 10 schema .NET framework [20]

Základným stavebným kameňom .NET Frameworku je modul Common Language Runtime (CLR). Akýkoľvek zdrojový kód vyžaduje pred spustením v module CLR kompiláciu, ktorá musí v prostredí .NET prebiehať dvojfázovo:

- Kompilácia zdrojového kódu do jazyka IL
- Kompilácia jazyka IL do kódu špecifického pre cieľovú platformu pomocou modulu CLR

Ako som už spomenul vyššie, existencia jazyka IL je jednou zo zásadných výhod platformy .NET. Jedná sa o nízkoúrovňový jazyk s jednoduchou syntaxou, ktorá nepracuje s textom, ale s číselnými kódmi. Jeho prioritou je nezávislosť, ktorá sa prejavuje predovšetkým tým, že rovnaký súbor s inštrukciami v bajtové kóde možno preniesť na ľubovoľnú platformu. V čase spustenia je možné ľahko vykonať záverečnú fázu kompilácie, ktorá zaistí spustenie kódu na príslušné platforme. Jazyk IL sa vždy kompiluje

metódou Just-In-Time (JIT), čo znamená, že miesto kompilácie celej aplikácie v jednom priechoďte JIT zjednodušené kompiluje každú z častí kódu v okamihu jej volanie. Po prvej kompilácii kódu je výsledný natívny spustiteľný súbor uložený, kým nie je aplikácia ukončená. Preto ho nie je nutné pri opakovanom spustení rovnaké časti kódu kompilovať znova. Najmenej rovnako dôležitá ako nezávislosť na platforme je výhoda možnosti spolupráca medzi jazykmi. Môžeme kompilovať do jazyka IL z jedného programovacieho jazyka a výsledný kód by mal spolupracovať s kódom, ktorý bol do IL zostavený z iného vyššieho jazyka.

ASP.NET je súčasť .NET Frameworku firmy Microsoft pre tvorbu webových aplikácií a služieb. Je nástupcom technológie ASP (Active Server Pages), od ktorého je odvodený, a priamym konkurentom JSP (Java Server Pages)

Hlavným rozdiel medzi technológiou ASP a ASP.NET je ten, že kódy na stránkach ASP.NET sú kompilované. Tým sa odstráni potreba analýzy a interpretácie jednotlivých riadkov pri každom prístupe klienta. Vznikne kompilovaný kód, ktorý je samozrejme oveľa rýchlejší. Klasické ASP stránky pomocou skriptov na strane servera priamo generujú HTML stránky, ktoré sa zašlú klientovi. Stránky sa interpretujú od začiatku do konca bez možnosti ošetrovať vzniknuté stavy a udalosti. [21]

Ďalšie rozdiely ASP a ASP.NET:

ASP	ASP.NET
Nie je objektovo orientované v pravom význame. Používa síce funkcie triedy a iné objekty, no používa viac menej štýl zhora nadol, kde sa kód začne na vrchu a vykonáva sa postupne smerom dole.	Je v pravom zmysle objektovo orientované. Kód sa vykonáva ako reakcia na nejaký spúšťač (napr. kliknutie, načítanie stránky,...)
Limitované iba na VBScript alebo JScript	Podporuje mnoho programovacích jazykov ako Visual Basic, C#, J#, Delphi.NET,...
Premenné sú typu variant. Čo v podstate znamená, že pri deklarácii	Premenné sú silne typizované s pravidlami typickými pre hlavné

premennej je jedno čo je to za typ a môže byť do nej následne vložený či už string, integer alebo objekt.	programovacie jazyky.
	bohatý výber ovládacích prvkov a knižnic veľmi zrýchľuje vývoj aplikácií

[22]

ASP.NET je založený na CLR (Common Language Runtime), ktorý je zdieľaný všetkými aplikáciami postavenými na .NET Frameworku. Programátori tak môžu realizovať svoje projekty v akomkoľvek jazyku podporujúcom CLR, napr. Visual Basic.NET, JScript.NET, C#, Managed C++, ale aj mutácie Perlu, Pythonu a ďalších.

ASP.NET uľahčuje programátorom prechod od programovania klasických aplikácií pre Windows do prostredia webu: stránky sú poskladané z objektov, ovládacích prvkov (Controls), ktoré sú obdobou ovládacích prvkov vo Windows. Pri tvorbe webových stránok je teda možné používať ovládacie prvky ako tlačidlo (Button), nápis (Label) a ďalšie. Týmto prvkom je možné priradovať určité vlastnosti, zachytávať na nich udalosti, apod. Tak, ako sa ovládacie prvky pre Windows samy kreslia do formulárov na obrazovku, kompilátor produkuje z webových ovládacích prvkov HTML kód, ktorý tvorí časť výslednej stránky poslanej do klientovho prehliadača.

Schopnosť ASP.NET cachovať celú stránku alebo jej časť podstatne zvyšuje výkon serveru. Dá sa prevádzkovať na rôznych operačných systémoch aj webových serveroch, napr. IIS (Windows), Apache (Windows, Linux s open source implementáciou .NETu Mono).

Aj keď webový protokol HTTP je sám o sebe bezstavový (tj. jednotlivé požiadavky od užívateľa medzi sebou nie sú previazané), zachovanie kontextu medzi jednotlivými požiadavkami vyžaduje. ASP.NET tento problém rieši kombináciou HTML a JavaScriptu pomocou dvoch základných techník

ViewState - uchováva informácie medzi opakovaným odosielaním formuláru na server (postbackom) v zakódovanom tvare a v skrytých formulárových poliach. Jeho výhodou je, že využíva len HTML a nevyžaduje žiadnu špeciálnu podporu na strane

servera ani klienta. Nevýhodou je, že sa medzi serverom a klientom prenáša väčší objem dát, hlavne ak je ViewState využívaný nesprávne.

Session State - oproti tomu ukladá všetky informácie na strane servera a predáva (typicky ako cookie alebo súčasť URL) len jednoznačný identifikátor. To síce znižuje objem prenášaných dát, ale kladie vyššie nároky na výkon serveru. Pokiaľ sa sessions používajú nesprávne, môže byť server náchylný aj k Denial of Service útokom. Oproti ASP umožňuje ASP.NET ukladanie session state do samostatného procesu alebo na SQL server. To zjednodušuje použitie session vo webových farmách, zvyšuje výkon a umožňuje stav zachovať aj pri reštarte serveru.

2.2. Alternatívne platformy

2.2.1. JSP (Java Server Pages)

Ako hlavného konkurenta si predstavme JSP. JSP je technológia Java, ktorá pomáha softvérovým vývojárom obsluhovať dynamicky generované webové stránky, založené na HTML, XML alebo ostatných typoch dokumentov. Patrí medzi technológiu tvorby dynamických stránok na webovom serveri. JSP dokument je zmesou kódu Javy, XML, JDBC, HTML a JavaScriptu. Ide o serverovo a platformovo nezávislú technológiu.

Z hľadiska architektúry môžeme JSP vnímať ako vysokú úroveň abstrakcie java servletov. JSP sú prekladané do servletov za behu, každý servlet je cachovaný a znovu použitý až kým nie je zmenené pôvodné JSP. Kompilované stránky používajú Java bytecode miesto natívneho softvérového formátu a musia byť vykonávané v rámci Java Virtual Machine (JVM), ktorá je integrovaná v rámci operačného systému servera. Tak je poskytnutá abstraktné prostredie nezávislé od platformy.

2.2.2. Nevýhody ASP/JSP/PHP

Aby sme mohli dostatočne oceniť novú platformu, musíme uviesť aj hlavné nevýhody ASP stránok, PHP skriptov a JSP stránok. Nevýhody uvedieme vo forme heslovitého prehľadu.

- kód vytvorený v spomínaných skriptových systémoch predstavuje lineárny bezstavový programový modul, so všetkými nevýhodami s tým spojenými.

- skriptový kód ASP, PHP a JSP sa v týchto systémoch mieša s HTML kódom. Nevýhoda je zrejmá. Na väčších webových projektoch pracujú obvykle tímy vývojárov, dizajnérov, grafikov a podobne. Pre vývojára môže predstavovať problém kód vytvorený grafikom, hlavne ak je webová stránka rozdelená do viacerých rámcov, prípadne ak sa používajú moderné technológie, ako napríklad Macromedia Flash. Pre grafika webových stránok je zasa záhadou skriptový kód, ktorý nachádzajú vo svojich predtým vytvorených stránkach, a to na tých najneočakávanejších miestach, napríklad vo vnútri HTML kódu pre vytvorenie tabuliek.

- dochádza k miešaniu jednotlivých architektonických vrstiev projektu

- vytvoriť vývojové prostredie pre takúto technológiu nie je práve jednoduché. Vývoj webových aplikácií na spomínaných platformách môže byť pomerne náročný a zdĺhavý a teda aj drahý.

- programátori sú zvyknutí využívať osvedčené moduly v rôznych projektoch. Spomínané technológie však takúto modularitu neumožňujú, jedine kopírovanie zdrojového kódu z jedného zdrojového súboru do nového súboru a jeho následné prispôbenie.

- Po vygenerovaní HTML kódu na strane servera, bol tento kód odoslaný ku klientovi a následne server na všetko zabudol.

Okrem týchto základných nevýhod na ktoré narazia aj vývojári jednoduchých aplikácií typu hobby, obsahujú technológie ASP, PHP a čiastočne aj JSP ďalšie nevýhody na ktoré narazíme pri vývoji podnikových aplikácií. Napríklad ako používať stav na webových farmách, aby stavová informácia prežila pád aplikácie, a aby stav fungoval bez cookie. Pri aplikáciách s masívnym prístupom potrebujeme zvýšiť výkon použitím cache pre pseudostatické data. Väčšina webových aplikácií využíva databázy a tým pádom vzniká problém autentifikácie používateľa voči databáze. Riešenia týchto a mnohých iných problémov musia v ASP/PHP/JSP riešiť vývojári webových aplikácií. U ASP.NET sú príslušné bloky, ktoré riešia naznačené problémy súčasťou infraštruktúry. [23]

2.3.ADO.NET

Vývojári pred vznikom platformy .NET Framework používali technológie pre prístup k dátam vo forme ODBC, OLE DB, alebo ADO (ActiveX Data Objects). Spoločnosť Microsoft s uvedením platformy .NET Framework vytvorila nový spôsob práce s dátami nazývaný ADO.NET. Na počiatku je nutné poznamenať, že prítomnosť ADO.NET v .NET Framework nenahradzuje súčasnú verziu ADO. Tá je neustále dôležitá pre klasické COM aplikácie. Dôvodom je spôsob výmeny dát. Zatiaľ čo COM aplikácie spracovávajú objekty typu VARIANT s ktorými ADO pracuje. ADO.NET (ActiveX Data Objects .NET) obsahuje kolekciu tried, ktoré programátorom využívajúcich platformu .NET poskytujú služby pre prístup k dátam, čím im poskytujú sadu komponentov pre tvorbu distribuovaných aplikácií zdieľajúcich dáta a tvorbu databázových aplikácií. Knižnica ADO.NET je neoddeliteľnou súčasťou rozhrania .NET Framework, ktoré poskytujú prístup k relačným, aplikačným a XML dátam. Triedy knižnice ADO.NET sa nachádzajú v súbore System.Data.dll. Pod dátami si predstavuje prevažne informácie uložené v databázach. Či už sa jedná o dáta v databázach napríklad na Microsoft SQL Serveru či dáta sprístupnené cez OLE DB alebo XML. Medzi jeho prednosti patrí predovšetkým jednoduchý spôsob použitia, rýchlosť pri spracovaní a pod. Stačí vytvoriť spojenie so serverom, s ktorým budeme chcieť pracovať, pomocí zvoleného adaptéru a zadaného dotazu získať z databázy dáta a tie potom načítať do niektorej z pripravených konštrukcií pre prácu s dátami z tabuliek.

ADO.NET ale nemusí pracovať iba s databázami na nejakom serveri. Táto technológia bola vyvíjaná súčasne s XML triedami v prostredí .NET Framework, vďaka tomu umožňuje dáta načítať a zapisovať aj vo formáte XML spolu s definičným súborom XSD definujúceho schému danej databázy. Nástroje ADO.NET boli navrhnuté tak, aby sa oddelil spôsob prístupu k dátam od manipulácie s dátami. K prvej skupine patrí .NET Framework Data Provider obsahujúci množinu komponent zahrňujúcich podmnožiny:

- Connection – zabezpečujúce pripojenie,
- Command – obsahujúci množinu príkazov pre vyberanie dát,
- DataReader – zabezpečujúce načítanie dát,

- **DataAdapter** - užitočný predovšetkým pre spoluprácu s prvkami dočasnej databázovej pamäti typu **DataSet** a navyše rieši zmeny v zdroji dát pomocou príkazov **Select**, **Insert**, **Update**, **Delete**.

Do druhej skupiny priradujeme okrem iných objekt pamäťovej reprezentácie dát - **DataSet** (skladajúci sa z objektov napr. **DataTable**, **DataRow**, **DataSet** apod.). To znamená, že tieto objekty uchovávajú v pamäti načítané dáta z databáze. To umožňuje pracovať s týmito dátami rovnako ako s dátami v databáze, zatiaľ čo pripojenie k databáze je v tom čase odpojené. Po ukončení modifikácie dát v **DataSete** sa tieto zmeny pomocou **Data Providerov** aktualizujú aj v samotnom úložisku dát [24].

3. Cieľ a metodika práce

Ciele tejto práce môžeme zadefinovať ako na jednej strane analýza, návrh a implementácia požadovanej aplikácie a na strane druhej zozbieranie dostatočných vedomostí nielen o vybranej ale aj konkurenčnej platforme a preskúmať silné a slabé stránky vybranej MS platformy a na základe zozbieraných údajov o programovacích prostriedkoch, navrhnúť vhodnú kombináciu pre využitie pri implementácii. Popri implementácii môžeme pomocou diagramov modelovacieho jazyk analyzovať vytváraný systém.

Súčasťou tohto vypracovania je aj využitie vedeckých metód, ktorými sú systémová analýza dedukcia, či syntéza, ktoré sú využívané nasledovne:

- Metódy vzťahovej analýzy – tieto metódy našli uplatnenie pri popise UML diagramov a vzťahov medzi nimi
- Metódy deskriptívnej analýzy – tie sú zasa využívané pri popisoch jednotlivých častí aplikácie
- Syntéza – využíva sa pri výbere konkrétnych diagramov pri analýze aplikácie

V práci čerpám vedomosti z knižných zdrojov, k tomu rôzne webové portály či dokumentácie. Pri práci som okrem iného využil Enterprise architect pre tvorbu diagramov na analýzu aplikácie, MS Visual Studio 2013 pri tvorbe programovacieho kódu, Microsoft Word pri tvorbe tejto práce.

4. Výsledky práce a diskusia

4.1. Analýza a návrh

4.1.1. Návrh a požiadavky na systém

Každý systém alebo aplikácia predtým ako sa nad ňou začnú pariť hlavy skúsených programátorov musí byť v prvom rade zadaná. Zadaná teda najčastejšie od klienta. Ten si

za ňu platí ako za produkt prípadne službu. Tieto podmienky a žiadosti klienta musia byť samozrejme hneď jasné. Aj keď realita býva taká, že to tak jasné nie je. Klient vie že chce aplikáciu, ktorá mu prinesie vyššie zisky, no čo má tá aplikácia všetko robiť ostáva často nejasné až do posledných chvíľ a prípadne aj ako dodatočná implementácia. Preto si rozoberme čo sa a ako sa bude robiť a všetky požiadavky už teraz. Pomocou rôznych modelov môžeme tak klientovi názorne ukázať a vysvetliť čo sa dá spraviť ľahko a čo naopak bude stáť viac času aj peňazí.

V tejto kapitole si preberieme kompletne návrh implementácie našej aplikácie a uvidíme nakoľko sa v závere podarí priblížiť k očakávanému výsledku. V tomto konkrétnom prípade má táto práca za úlohu za úlohu zhotovenie webovej služby, vďaka ktorej v aplikácii získame výpis účtu klienta. V podstate teda sa žiada vytvoriť simplifikovaná verzia internet bankingu. Implementáciu vykonávam tak povediac from scratch a teda nie len programátorskú časť ale aj nasadenie na webový server a spravovanie databázy.

4.1.2. Funkčné požiadavky

Navrhovaný systém má plniť základnú požiadavku internet bankingu a to možnosť sledovať svoje pohyby na účte. Podmienkou je aby toto bolo docielené pomocou webovej služby čo predstavuje dnes štandard pri výmene dát. Najväčšia výhoda webových služieb je platformová nezávislosť a flexibilita. Keď viaceré systémy medzi sebou komunikujú nie je pri tom podstatné na akej platforme fungujú práve vďaka univerzálnosti XML jazyka o čom sme si už vraveli na začiatku.

Webová služba bude teda získavať zákazníkove transakcie a výsledok bude vracať do vytvorenej aplikácie banky. Výpis bude obsahovať klasické položky ako číslo účtu , ktorého sa výpis týka, číslo účtu ktorý pôsobí ako protistrana, samozrejme tiež sumu ktorá na účet prišla alebo z neho odišla. Okrem toho ešte dátum transakcie, zostatok po transakcii a prípadne popis k transakcii. Klient si bude môcť z daných transakcií aj filtrovať na základe určitých kritérií.

Klient sa do aplikácie samozrejme dostane iba po úspešnom zadaní platných používateľských údajov ktoré mu vystaví banka. Po prihlásení bude automaticky premostený na stránku s transakciami. Okrem toho si používateľ bude môcť zísť do

svojho profilu a skontrolovať svoje údaje poskytnuté banke ako napríklad meno, priezvisko, telefónne číslo alebo email.

4.1.3. Návrhové zobrazenie

Na obrázku nižšie možno vidieť predpokladaný vzhľad prihlasovacieho okna aplikácie. Dizajn je veľmi minimalistický a jednoduchý. Zadané prihlasovacie meno a heslo sa bude porovnávať s tými uloženými v databáze a v prípade zhody bude užívateľ prihlásený.



The image shows a login window titled "Prihlásenie". It contains two input fields: "Prihlasovacie meno" with the value "throwaway" and "Heslo" with the value "*****".

obr. 11 Návrh prihlasovacieho okna

Na ďalšom obrázku je znázornené v podstate defaultné okno z ktorého sa používateľ bude pohybovať ďalej v aplikácii. Z okna bude mať prehľad o svojom zostatku a účtoch ktoré má aktívne. Kliknutím na číslo účtu sa presunie na ďalšie okno. Tiež sa dostane do svojho profilu alebo sa bude môcť odhlásiť.

Domov

Menu

Klient: Jozef Novák

Aktuálny zostatok: 777.77€

profilodhlásiť

Účty

1. bežný účet:009704512

2. bežný účet: 007912356

3. sporiaci účet: 00087125

obraty

info o účte

obr. 12 Návrh okna s výpismi transakcií

Ďalšie okno tvorí v podstate základ aplikácie a používateľ s ním bude prichádzať do kontaktu okrem prihlasovania najčastejšie. Po tom ako klikol na niektorý aktívny účet, sa mu otvorili transakcie pod týmto účtom. Používateľ má k dispozícii filter, ktorým môže obmedziť výber ukazovaných transakcií. Taktiež sa z daného okna dostane na okno domov a tiež do svojho profilu.

Obraty

Menu

Klient: Jozef Novák

Aktuálny zostatok: 777.77€

profilodhlásiť

domov

info o účte

Filtrer:

mena: EUR

typ transakcie: všetky

dátum: od: 01.01.2012 do: 01.01.2013

dátum	popis transakcie	suma	mena
01.01.2013	Platba kartou	15	EUR
31.12.2012	poplatok za vedenie účtu	1	EUR

obr. 13 Návrh okna obraty

Ďalším oknom, do ktorého sa používateľ môže dostať je okno „info o účte“. Tu vidí v prehľadnej tabuľke všetky údaje charakterizujúce daný účet.

Info o účte

názov účtu:	
číslo účtu:	
typ účtu:	
produkt:	
aktuálny zostatok:	
stav účtu:	
dátum založenia:	
denný limit:	
bezpečnostný prvok:	
typ výpisu:	
notifikácia sms:	

úrokové miery

kreditná:	0,10%
debetná:	
trestná:	5,00%

obr. 14 Návrh okna info o účte

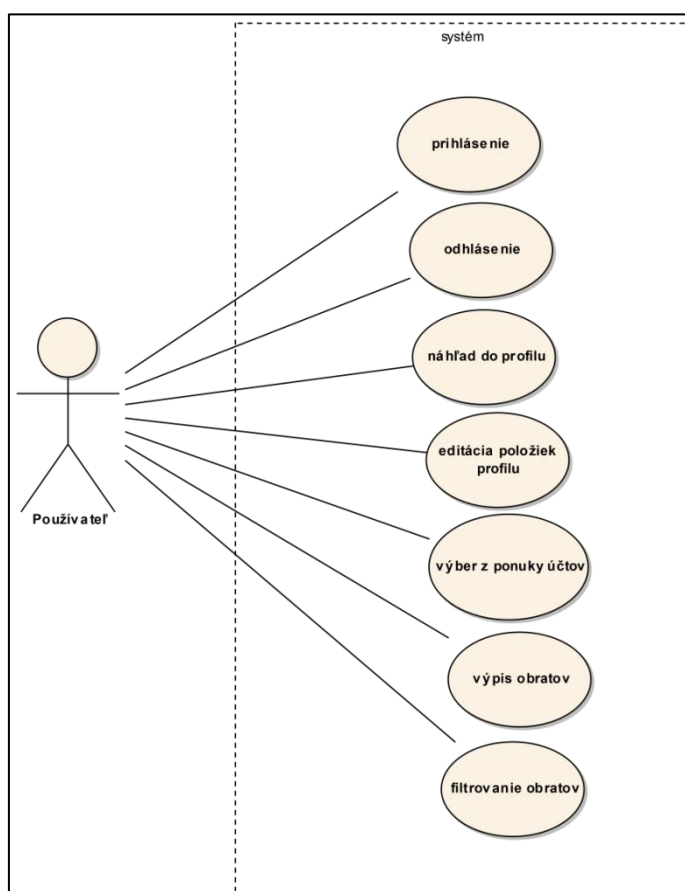
Okno „profil“ poskytne používateľovi náhľad na jeho osobné údaje a prípadne aj možnosť niektoré pozmeniť.

Profil

prihlasovacie meno:	515456754
Meno:	Jozef
Priezvisko:	Novák
email:	novak@gmail.com
telefón:	0904 123 456
jazyk:	slovenský ▼

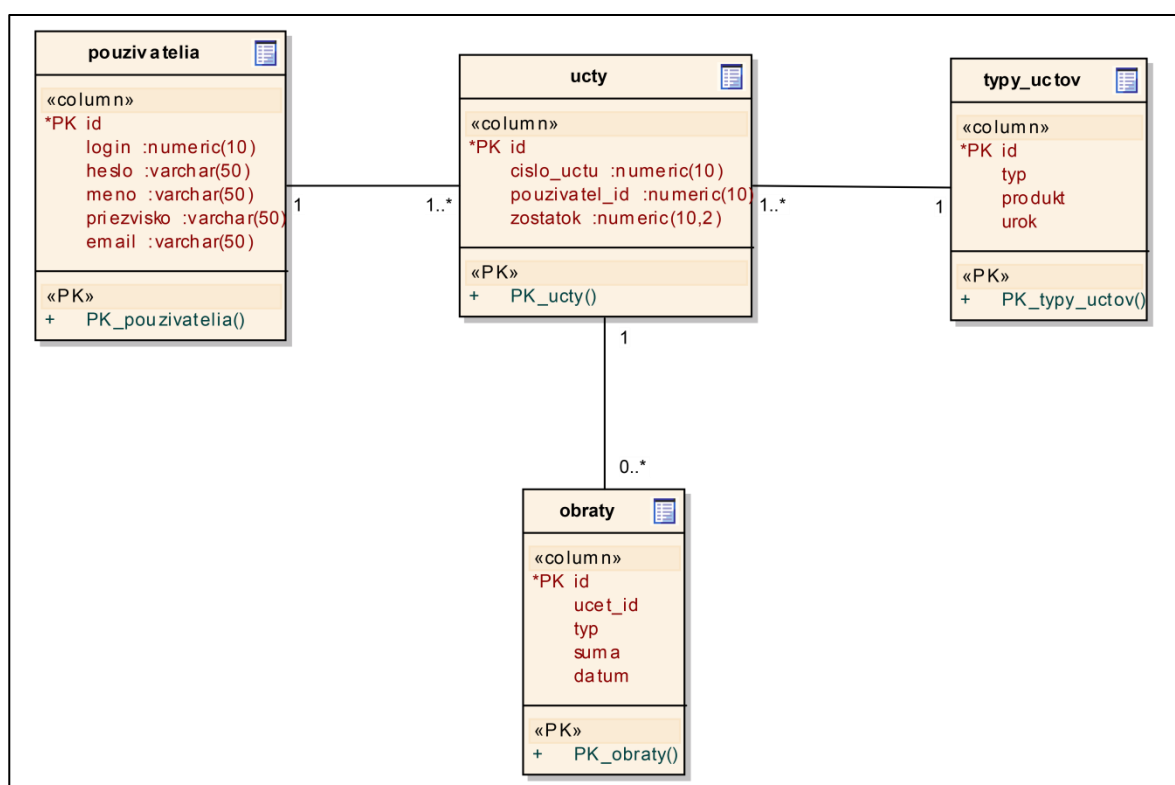
obr. 15 Návrh okna profilu

4.1.4. Opis aplikácie pomocou UML modelov



obr. 16 Use case model navrhovanej aplikácie

Na obrázku je znázornený use case diagram pre našu aplikáciu. Hlavnú rolu v našom systéme bude hrať samozrejme používateľ. Ten bude mať možnosť prihlásenia sa do systému na základe prihlasovacieho mena a hesla uloženého v databáze. Po prihlásení sa bude môcť samozrejme aj odhlásiť kedy stratí všetky práva prislúchajúce konkrétnemu účtu. Po prihlásení bude môcť používateľ nazrieť do svojho profilu kde budú zobrazené jeho údaje. Následne bude mať možnosť niektoré položky aj editovať. Používateľ môže mať viacero účtov v rámci jedného konta. Z menu si jednoduchým kliknutím vyberie požadovaný účet. Následne sa mu zobrazia obrátové položky v prehľadnej tabuľkovej forme. Bude mať možnosť filtrovať tento výstup podľa špecifických požiadaviek (napríklad dátum, suma a pod.)



obr. 17 Navrhovaný dátový model

Na obrázku máme znázornený navrhovaný dátový model pre náš systém. Pozostáva zo štyroch tabuliek pouzivatel, ucty, typy_uctov a obraty. Používateľ môže vlastniť jeden a viac účtov a konkrétny účet môže patriť iba jednému konkrétnemu používateľovi. Účet má jeden konkrétny typ a viacero účtov môže nadobúdať jeden a ten istý typ. Jeden účet môže mať žiaden alebo viac obratov a každý obrat spadá pod jeden konkrétny účet.

Tabuľka „používatelia“ obsahuje atribúty predstavujúce základné informácie o používateľovi resp. majiteľovi účtu. Obsahuje tiež atribút login a heslo, ktoré sa porovnáva so zadanými hodnotami v rámci prihlasovacieho formulára. Tabuľka „účty“ zasa nesie základné informácie o účtoch ako sú napríklad číslo účtu či aktuálny zostatok. Obsahuje tiež cudzí kľúč ktorým sa každý účet viaže na používateľa. Tabuľka „typy účtov“ nesie záznam o všetkých typoch účtu ktorými banka disponuje vo svojej ponuke. Každý účet je nejakého typu z určitej produktovej línie na ktorú sa viažu špecifické úroky. Posledná je tabuľka „obraty“, ktorá nesie záznamy o jednotlivých príjmoch, platbách a poplatkoch realizovaných nad konkrétnym účtom.

4.2. Implementácia

4.2.1. Databáza

Nakoľko táto práca spočíva na .NET frameworku, netreba veľmi vysvetľovať dôvod výberu SQL servera od Microsoftu za môj databázový server. Okrem iného s ním prichádzam do kontaktu denne v práci a čisto zo subjektívneho hľadiska som si ho obľúbil a zabehal sa v ňom pomerne bez väčších problémov aj napriek tomu že som k nemu prechádzal od SQL developera pre ORACLE. Konkrétne som teda rozbehal verziu 2008 R2. Aj keď a existuje novšia verzia táto je stále obľúbená a používaná v reálnom svete IT. A to sa netreba vôbec čudovať. Cena za licenciu síce nedosahuje výšky ORACLE no tiež vie zaťažiť peňaženku nejednej firme. No kvality tomuto serveru od tvorca Windowsu nemožno vytknúť, server pracuje veľmi rýchlo a vďaka všestrannému IDE je radosť s ním pracovať. Umožňuje napríklad aj tvorbu entitno-relačného modelu. Presne taký môžeme vidieť nižšie na obrázku. Ten bol vygenerovaný pomocou SQL manažéra z už existujúcej databázy. Databáza bolo vytvorená jednotným súborom SQL príkazov.

```
Create Table USERS
(
  UserID nvarchar(255) not null PRIMARY KEY,
  Password nvarchar(255) not null,
);

Create Table Profile
(
  UserID nvarchar(255) not null UNIQUE FOREIGN KEY REFERENCES
USERS(UserID),
  Meno nvarchar(50),
```

```

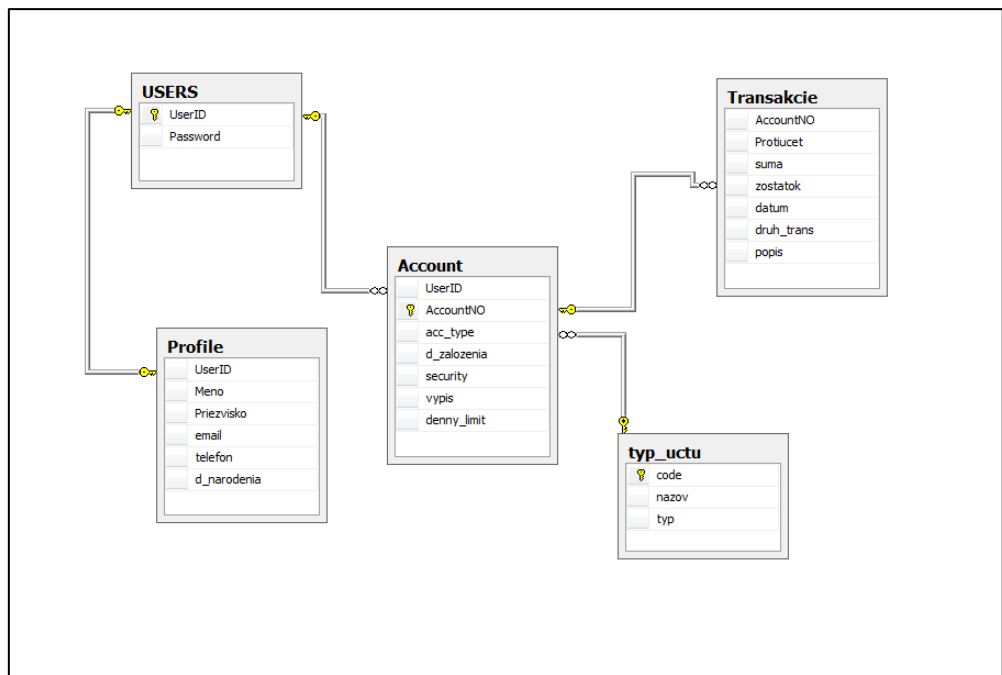
Priezvisko nvarchar(50),
email nvarchar(50),
telefon nvarchar(50),
d_narodenia date
);

Create Table Account
(
UserID nvarchar(255) not null FOREIGN KEY REFERENCES USERS(UserID),
AccountNO nvarchar(255) not null PRIMARY KEY,
acc_type int,
d_zalozenia date,
security nvarchar(50),
vypis nvarchar(50),
denny_limit decimal(18,2)
);

Create Table typ_uctu
(
code int not null PRIMARY KEY,
nazov nvarchar(255),
typ nvarchar(50),
);

Create Table Transakcie
(
AccountNO nvarchar(255) not null FOREIGN KEY REFERENCES
ACCOUNT(AccountNO),
Protiucet nvarchar(255),
suma decimal(18,2),
zostatok decimal(18,2),
datum date,
druh_trans nvarchar(50),
popis nvarchar(255)
);

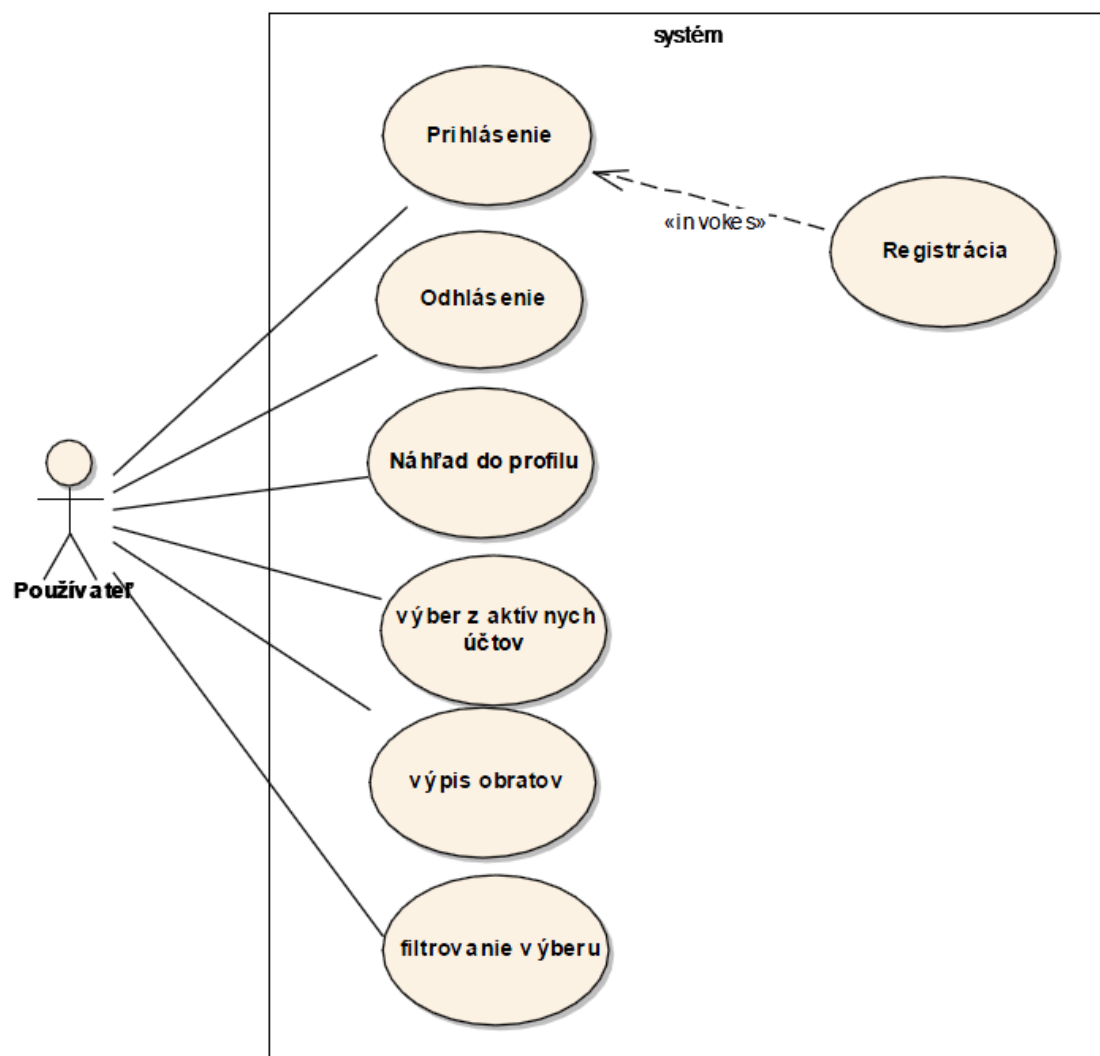
```



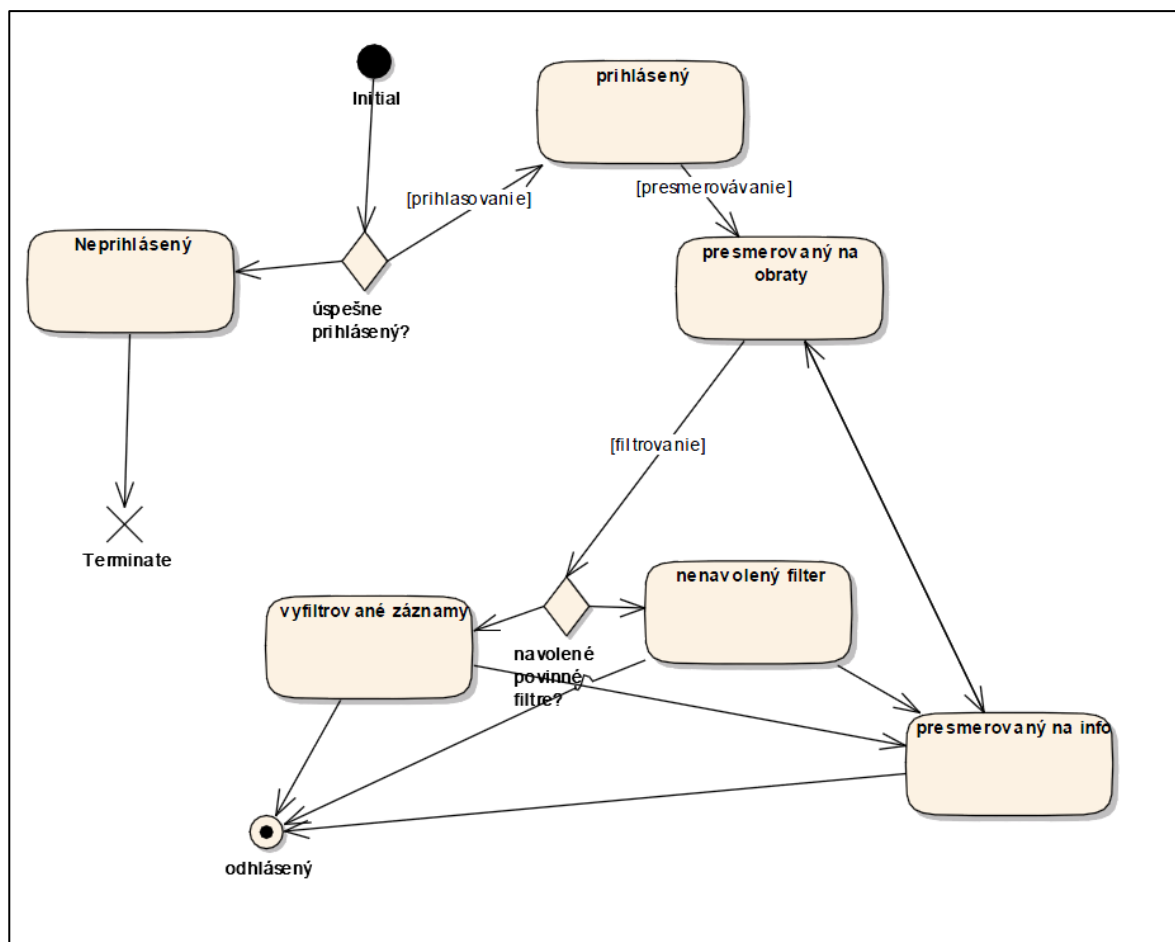
obr. 18 ERD diagram

Môžeme vidieť, dátový model sa od návrhu veľmi nelíši. Má podobné entity ako aj atribúty. Základnú tabuľku predstavuje tabuľka Account. Tá obsahuje informácie o účte vrátane čísla účtu. Na tabuľku Account sa viaže konfiguračná tabuľka „typ_účtu“. V nej sú uvedené typy účtov z portfólia banky. Ďalej sa na Account viaže tabuľka Transakcie. Táto tabuľka obsahuje záznamy o pohyboch na jednotlivých účtoch. Účty vlastní klienti ktorých prihlasovacie údaje sú uvedené v tabuľke Users. Nakoniec v tabuľke Profile nájdeme rôzne základné info o majiteľovi účtu.

Ako vidíme na use case modely, používateľ má k dispozícii základné funkcionality účtu a to autentizáciu, náhľad do profilu a obrátov a ich filtráciu.



obr. 19 Implementačný use case model



obr. 20 stasavový diagram aplikácie

4.2.2. Webový server a programovacie nástroje

Za webový server som rovnako nerozmýšľal dlho keďže celý projekt sa realizuje pod .net frameworkom, najlepšie mi poslúži IIS (Internet Information Services). IIS má v sebe každý Windows iba sa automaticky neinštaluje pri inštalácii windowsu. Je potrebné si ho manuálne doinštalovať, no nie je to nič zložité a podľa návodu by to zvládla aj nejedna mamička. Keď už som nainštaloval IIS a trocha sa s ním zoznámil, je dobré začať plánovať trocha konkrétnejšie. Takže zo zadania je jasné, že sa programuje v ASP.NET a takisto že bude involvnutá webová služba. Keď príde na Microsoft ľudia dost' zvyknú sem tam udrieť pod pás. No jedno im treba nechať a to vývojové prostredie Visual Studio. Ja konkrétne som použil Visual Studio 2013. Toto prostredie sa mi veľmi páči v posledných edíciách dokonca dbali aj na vizuál ponúkli na výber tri základné témy, pričom tmavá sa im veľmi vydarila. Popri ASP.NET som využíval objektovo orientovaný

jazyk C#. Okrem neho bol na výber aj VB okrem iných. Pri implementácii kalendára som využil CSS štýly a javascript a to konkrétne voľne šíriteľný template pikaday.

Implementovaný fragment kódu s pikaday:

```
</script>
</div>

<div>
    <asp:Label ID="LabelDatum2" runat="server"
Text="do:"></asp:Label>
    <asp:TextBox ID="Datum2" runat="server"></asp:TextBox>
    <script type="text/javascript">
        var picker = new Pikaday(
        {
            field: document.getElementById('Datum2'),
            firstDay: 1,
            format: 'YYYY-MM-DD',
            minDate: new Date('1990-01-01'),
            maxDate: new Date('2020-12-31'),
            yearRange: [1990, 2020],
            numberOfMonths: 1,
            theme: 'dark-theme'
        });
    </script>
```

Pri riešení som v podstate pracoval s dvoma projektmi. Išlo vlastne o jeden kde jednu časť tvorila implementácia samotnej web servisy a v druhej časti som implementoval webovú aplikáciu, ktorá túto servisu konzumuje. Webová služba prakticky poskytuje viacero metód a teda výsledok ktorý vráti závisí od vložených parametrov. Servisa sa teda pripája k databáze s využitím knižnice system.data.sqlclient a aplikáciu môže potom danú servisu iba zavolať.

V ASP.NET netreba dokola vypisovať spojenie na databázu miesto toho sa zapíše do konfiguračného súboru web.config a potom sa na neho iba referuje:

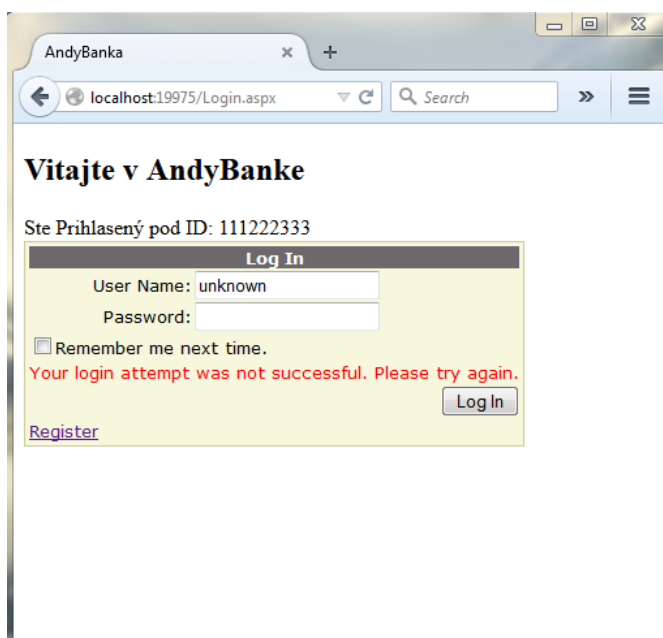
```
<connectionStrings>
    <add
        name="DBConnectionStr"
        connectionString="Data Source=SEMIR-PC;Initial
Catalog=AndyBank;Integrated Security=true;"
        providerName="System.Data.SqlClient"/>
</connectionStrings>
```

Implementoval som webovú službu ešte tak povediac starým spôsobom. Tvorí ju súbor s koncovkou .asmx. Dnes sa považuje takáto web servisa v rámci .net za starú

a neflexibilnú. Nahradila ju technológia WCF tiež známe ako Windows Communication Foundation. Je to novší a pružnejší spôsob ako vytvárať webovú službu. Hlavný rozdiel je v tom, že stará webservisa vie komunikovať iba cez HTTP pomocou SOAP správ. Na druhej strane WCF je flexibilnejšie a je schopná komunikovať napríklad aj cez TCP. V podstate akoby stará web servisa je iba súčasťou WCF balíčka a dnes sa určite odporúča implementovať riešenia najnovšími technikami.

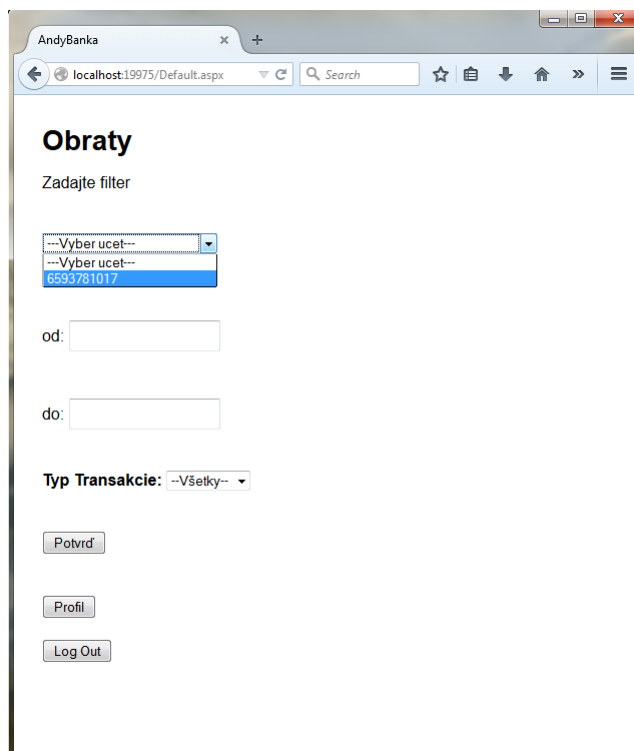
4.2.3. Náhľad aplikácie

Nižšie sú uvedené ukážky z aplikácie, na ktorých si priblížime ich funkcionality. Aplikácia má samozrejme autentizačný systém, kontroluje či má daná osoba prístup do aplikácie. Používateľ zadá používateľské meno v tvare číselného kódu, ktorý mu prideli banka. Následne zadá heslo a v prípade že sa pomýli alebo nie je v systéme, ukáže sa mu chybová hláška. V prípade, že je používateľ už prihlásený, nad oknom je zobrazené pod akým účtom je práve prihlásený.



obr. 21 Neuspesny login

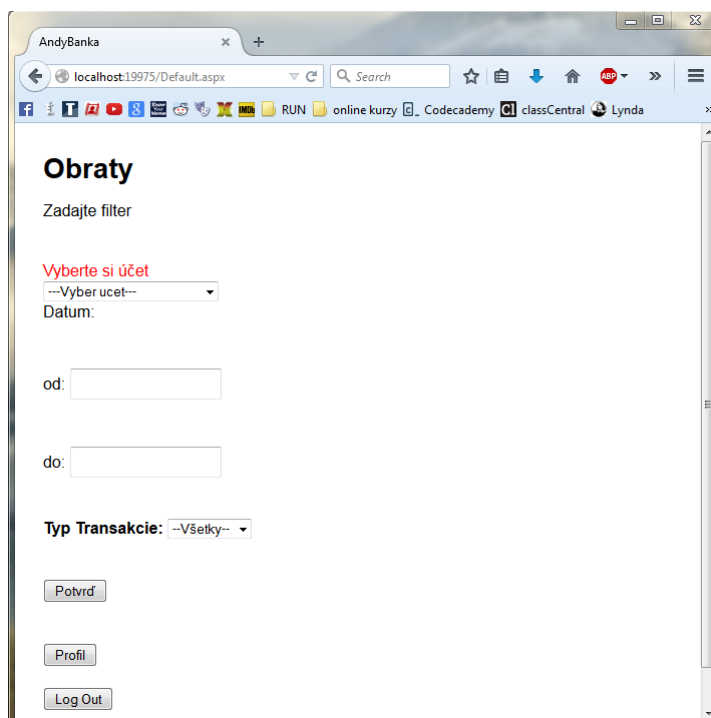
Ak prihlásenie prebehne úspešne, je používateľ presmerovaný na hlavné okno a to okno Obraty. V tomto okne sa mu zobrazujú všetky aktivity nad účtom, či už platby alebo príjmy. K dispozícii mu je viacero filtrov podľa ktorých môže vyberať záznamy. Prvý z filtrov predstavuje číslo účtu. Používateľovi sa v dropdownliste ponúkajú iba účty ktoré patria jemu.



The screenshot shows a web browser window titled 'AndyBanka' with the address bar displaying 'localhost:19975/Default.aspx'. The main content area is titled 'Obraty' (Transactions). Below the title, there is a section labeled 'Zadajte filter' (Enter filter). This section contains a dropdown menu for account selection, currently showing '---Vyber ucet---' with a blue highlight on the selected option '6593781017'. Below the dropdown are two date input fields labeled 'od:' and 'do:'. Further down is a dropdown for 'Typ Transakcie:' (Transaction Type) with the option '--Všetky--' (All). At the bottom of the filter section are three buttons: 'Potvrď' (Confirm), 'Profil' (Profile), and 'Log Out'.

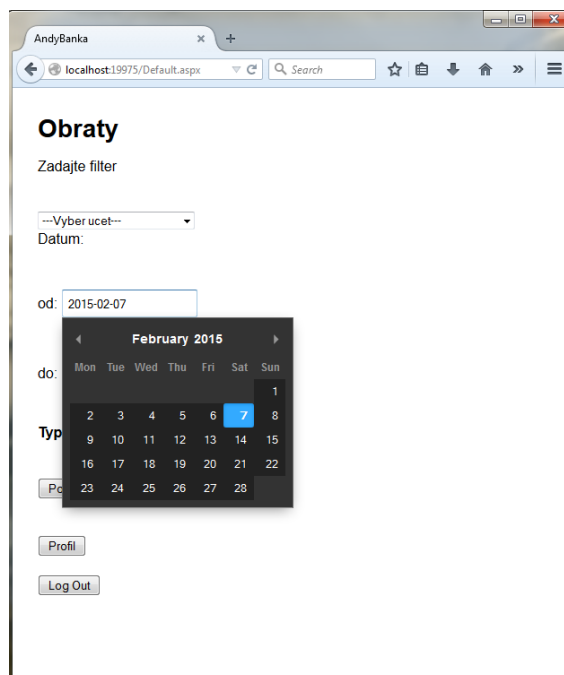
obr. 22 Okno obraty filter1

V prípade, že používateľ nevyberie žiaden filter a stlačí potvrdzovacie tlačidlo vyhodí sa chyba keďže bez vybraného účtu nie je kde pozerat' transakcie.



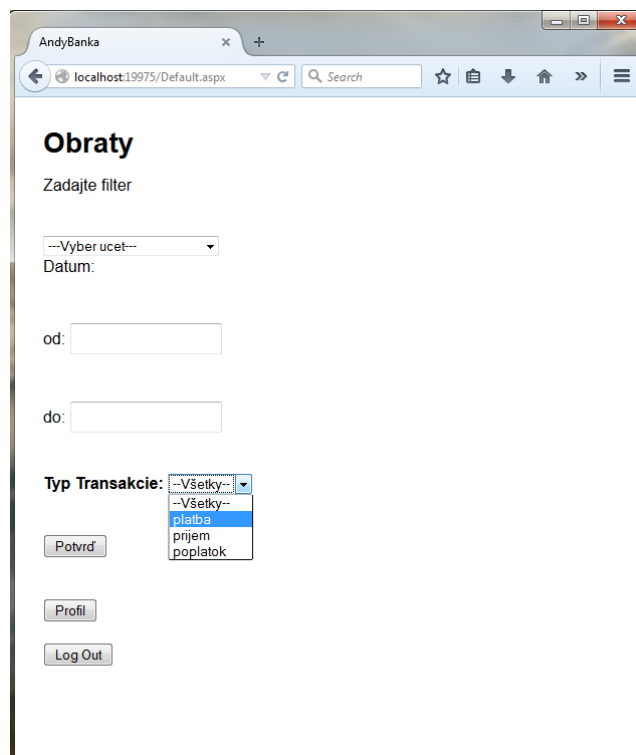
obr. 23 Nespravny filter

Ďalším filtrom v poradí je kalendár, ktorý vyskakuje interaktívne keď používateľ klikne na príslušný box. Používateľ si tu môže filtrovať transakcie podľa dátumu.



obr. 24 Okno Filter2

Posledným filtrom je možnosť vybrať transakcie podľa typu. A teda môže pozerat' iba platby alebo príjmy, prípadne poplatky.



obr. 25 Okno Obraty Filter3

V prípade, že nejaké záznamy spĺňajú podmienky filtra, sú zobrazené v prehľadnej tabuľke. Používateľ sa z tohto okna dokáže presunúť aj do okna profilu alebo sa odhlásiť.

Obraty

Zadajte filter

6593781017
Datum:

od:

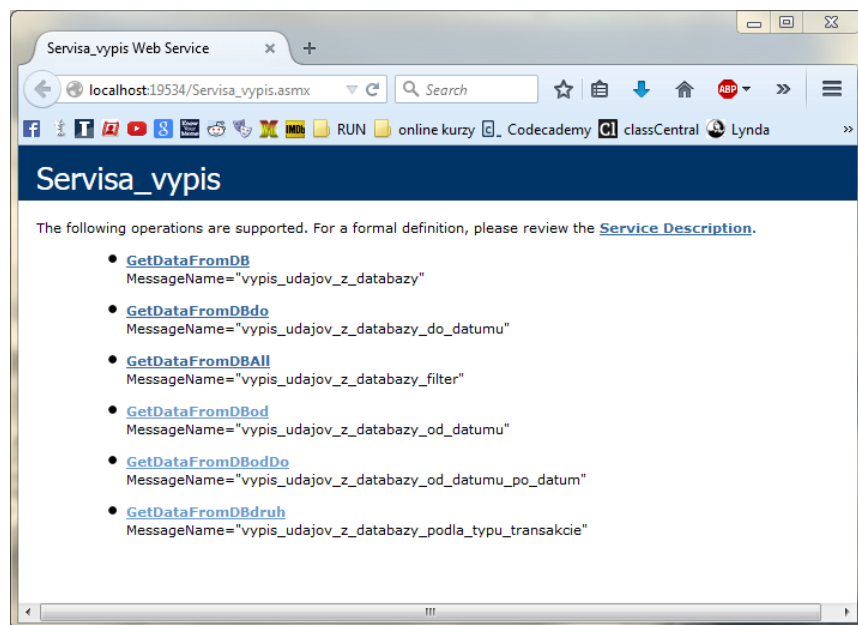
do:

Typ Transakcie: --Všetky--

AccountNO	Proticet	suma	zostatok	datum	druh_trans	popis
6593781017	4684846462	1000,00	1000,00	6. 5. 2015 0:00:00	prijem	pociatocny vklad
6593781017	975423164	-73,20	926,80	8. 6. 2015 0:00:00	platba	
6593781017	7985645411	-26,00	900,80	17. 6. 2015 0:00:00	platba	
6593781017	98745233254	5,40	906,20	22. 6. 2015 0:00:00	prijem	
6593781017	96854653673	-13,00	893,20	7. 7. 2015 0:00:00	platba	sypane caje TeaPot

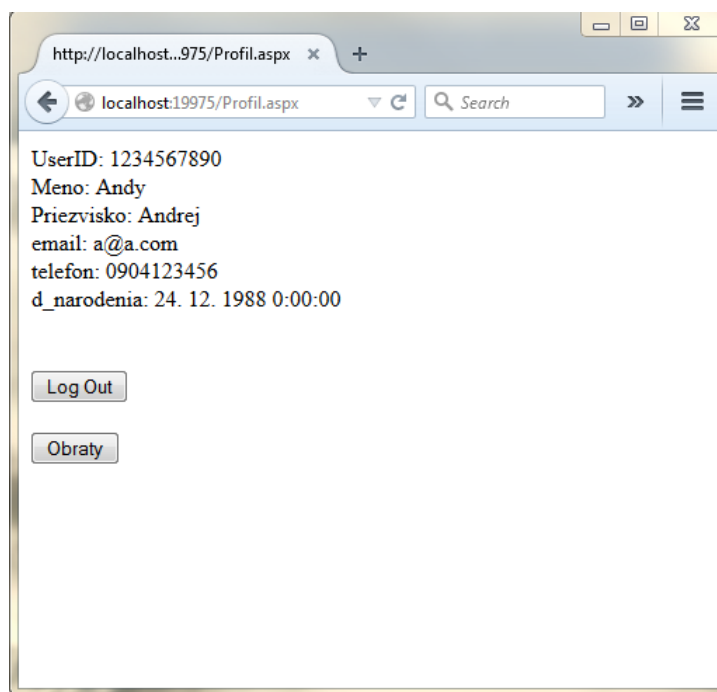
obr. 26 Okno Transakcií

Samotná tabuľka s jednotlivými transakciami je výsledkom volania webovej služby. V podstate je projekt rozdelený na dve časti. Jednu tvorí samotná webová služba, ktorá vracia transakcie v závislosti od posunutých parametrov.



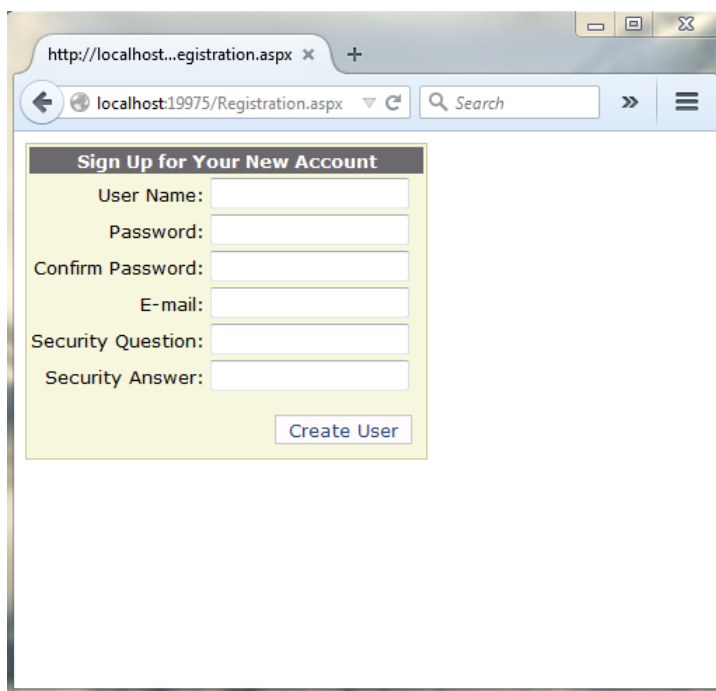
obr. 27 Volanie samotnej webovej služby

V profile sú zobrazené údaje o používateľovi, ktoré o ňom vlastní banka. Údaje sú zobrazené v prehľadnej formulárovej štruktúre.



obr. 28 Okno Profil

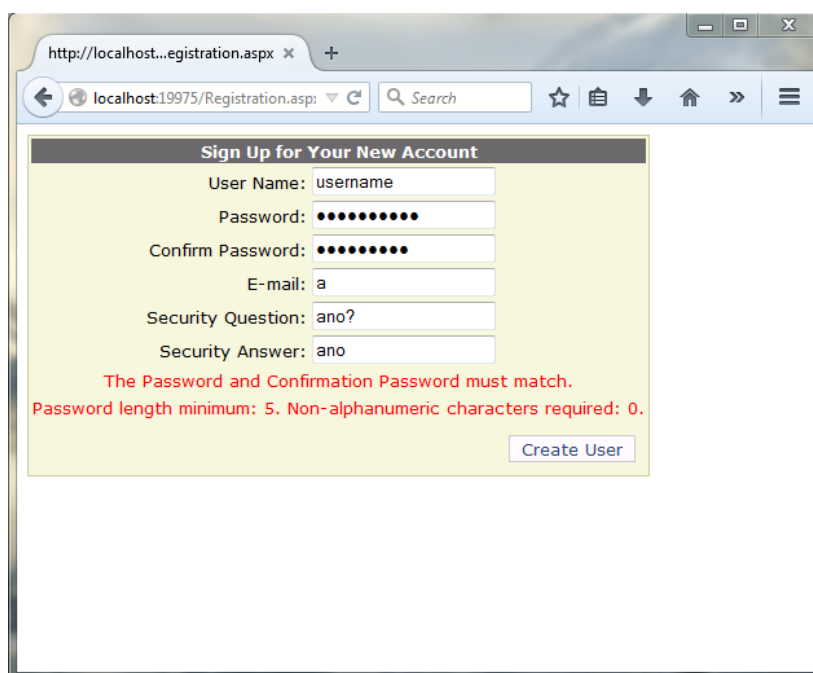
Aj keď aplikácia počítala s tým že používateľov pridáva banka, no existuje možnosť registrácie pomocou registračného formulára.



The screenshot shows a web browser window with the address bar displaying 'http://localhost...egistration.aspx'. The page title is 'Sign Up for Your New Account'. The form contains the following fields: 'User Name:', 'Password:', 'Confirm Password:', 'E-mail:', 'Security Question:', and 'Security Answer:'. Each field has a corresponding text input box. At the bottom right of the form is a button labeled 'Create User'.

obr. 29 Okno Registracia

Registračný formulár má viacero validačných checkov a v prípade nesprávneho inputu varuje používateľa výraznou červenou errorovou hláškou.



The screenshot shows the same registration form as in the previous image, but with some fields filled out and error messages displayed. The 'User Name' field contains 'username', 'Password' and 'Confirm Password' fields contain masked text (dots), 'E-mail' contains 'a', 'Security Question' contains 'ano?', and 'Security Answer' contains 'ano'. Below the form, there is a red error message: 'The Password and Confirmation Password must match. Password length minimum: 5. Non-alphanumeric characters required: 0.' The 'Create User' button is still visible at the bottom right.

obr. 30 Okno registracia error

Záver

Hlavným cieľom tejto diplomovej práce bolo zanalyzovať problematiku a úroveň platforiem podporujúcich dynamicky sa generované webové stránky a naplno sa zasvätiť do sveta masívnej výmeny elektronických dát. Priblížiť aspekty, či už konkurenčné výhody alebo úskalía jednotlivých možností ktoré poskytujú rôzne platformy na trhu. Aby sme naozaj naplno pochopili čaro siete drahých ale zato prepracovaných vývojárskych nástrojov od Microsoftu bolo úlohou aj priamo privoňať k programátorskému remeslu odskúšať si na vlastnej koži čo so sebou prináša byť vývojár.

V úvodnej časti práce sme sa sústredili na objasnenie pojmov ktorých nebolo málo, no boli potrebné pre dotvorenie plného obrazu o stave a možnostiach na trhu pri vývoji webových aplikácií.

V ďalšej časti sme analyzovali požiadavky akoby od klienta na aplikáciu a pomocou rôznych UML diagramov sme aplikáciu aj následne rozobrali z funkčného hľadiska a popísali sme všetky možnosti a služby ktoré ponúka.

Výsledkom práce je aplikácia na sledovanie pohybu finančných prostriedkov na účte klienta. Zdrojové kódy, UML diagramy, obrazovky aplikácie je možné nájsť priamo v práci alebo prípadne v prílohe.

Zoznam použitej literatúry

1. **smartbtl** [Online][cit. 2015-03-05] Dostupné na internete <<http://smartbtl.us/component/content/article.html?id=67:static-webpage-vs-dynamic-webpage>>
2. **W3C** [Online][cit. 2015-05-05] Dostupné na internete: <<http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice>>.
3. **Gyrus**. Integration using web services. <http://www.gyrus.com>. [Online] Dostupné na internete: <<http://www.gyrus.com/products/gyrusaim/web-services/>>.
4. **Buranský, Imrich**. 2002. *XML a webové služby*. Praha : Microsoft, s.r.o., 2002. 131s
5. **<https://sqlhut.files.wordpress.com>**. [Online][cit. 2015-05-07]Dostupne na internete: <<https://sqlhut.files.wordpress.com/2011/11/web-services.jpg>>
6. **Bisták, Ľuboš**. 2006. *Technológie pre webové služby : diplomová práca*. Bratislava: UK FMFI, 2006. 91 s.
7. **Netcraft**. 2015. April 2015 Web Server Survey. [Online] 20. April 2015. [cit. 08.5.2015] Dostupné na internete: <<http://news.netcraft.com/archives/2015/04/20/april-2015-web-server-survey.html>>.
8. **TutorialsPoint.com**. [Online] [cit. 2015-04-13]. Dostupné na internete: <http://www.tutorialspoint.com/webservices/web_services_architecture.htm>.
9. **Group, SGML Users'**. [Online] [cit. 2015-05-07]Dostupné na Internet: <<http://www.sgmlsource.com/history/sgmlhist.htm>>.
10. **w3schools** [Online][cit. 2015-04-03] Dostupné na internete: <http://www.w3schools.com/xml/xml_doctypes.asp>.
11. **Kosek, Jiří**. *XML pro každého*. Praha : Grada Publishing, 2000. ISBN 80-7169-860-1.

12. **Perháč, Peter.** [Online][cit. 2015-05-11]. Dostupné na internete: <<http://stackoverflow.com/questions/1544200/what-is-difference-between-xml-schema-and-dtd?rq=1>>.
13. **Winer, Dave.** XML-RPC Specification. *XML-RPC.com*. [Online] 2003. Dostupné na internete: <<http://xmlrpc.scripting.com/spec.html>>
14. **Zimmerman, Jimmy.** SOAP, XML-RPC, and REST. [Online] Dostupné na internete: <<http://jimmyzimmerman.com/blog/2007/01/soap-xml-rpc-and-rest.html>>
15. **Fox, Brian.** Use JSON? Well you'd better not be Evil. *Sonatype*. [Online] Dostupné na internete: <<http://blog.sonatype.com/people/2012/03/use-json-well-you-d-better-not-be-evil/>>
16. —. UDDI. *wikipedia*. [Online] http://en.wikipedia.org/wiki/Universal_Description_Discovery_and_Integration.
17. WSDL and UDDI. *w3schools.com*. [Online] Dostupné na internete:<http://www.w3schools.com/wSDL/wSDL_uddi.asp>
18. **asp.net.** [Online] [cit. 2015-05-04]dostupné na internete: <<http://www.asp.net/get-started>>.
19. **asptreeview.com.** [Online][cit. 2015-04-12] Dostupné na internete: <http://www.asptreeview.com/show/show_slideshow.aspx>.
20. **Lacko, Ľuboslav.** *ASP.NET pre začiatočníkov*. Praha : Microsoft, s.r.o.
21. amazon [Online][cit. 2015-05-03]Dostupné na internete: <<http://askville.amazon.com/Difference-asp-net/AnswerViewer.do?requestId=17641467>>
22. **Lacko, Ľuboslav.** *ASP.NET pre začiatočníkov*. Praha : Microsoft, 2002.
23. **Kačmář, Dalibor.** *Programujeme .NET aplikace ve Visual Studiu .NET*. Praha : Computer Press, 2001. ISBN 80-7226-569-5.

ZOZNAM ILUSTRÁCIÍ

OBR. 1 STATICKY GENEROVANÁ WEBOVÁ STRÁNKA VS. DYNAMICKY GENEROVANÁ WEBOVÁ STRÁNKA [1]	10
OBR. 2 JEDNODUCHÝ MODEL FUNGOVANIA WEBOVÝCH SLUŽIEB [5].....	13
OBR. 3 TRHOVÝ PODIEL WEBOVÝCH SERVEROV V RÁMCI VŠETKÝCH STRÁNOK[4]	14
OBR. 4 TABUĽKOVÝ PREHĽAD TRHOVÉHO PODIELU WEBOVÝCH SERVEROV V RÁMCI VŠETKÝCH STRÁNOK[4]	15
OBR. 5 TRHOVÝ PODIEL WEBOVÝCH SERVEROV V RÁMCI AKTÍVNYCH STRÁNOK[4]	15
OBR. 6 TABUĽKOVÝ PREHĽAD TRHOVÉHO PODIELU WEBOVÝCH SERVEROV V RÁMCI AKTÍVNYCH STRÁNOK[4].....	15
OBR. 7 TRHOVÝ PODIEL WEBOVÝCH SERVEROV V RÁMCI TOP MILIÓŇ NAJVVYŽAŽENEJŠÍCH STRÁNOK[4].....	16
OBR. 8 TABUĽKOVÝ PREHĽAD TRHOVÉHO PODIELU WEBOVÝCH SERVEROV V RÁMCI TOP MILIÓŇ NAJVVYŽAŽENEJŠÍCH STRÁNOK[4]	16
OBR. 9 ZNÁZORNENIE INTERPRETÁCIE DATABÁZOVÝCH DÁT DO XML [11]	19
OBR. 10 SCHEMA .NET FRAMEWORK [20]	27
OBR. 11 NÁVRH PRIHLASOVACIEHO OKNA.....	36
OBR. 12 NÁVRH OKNA S VÝPISMI TRANSAKCIÍ	37
OBR. 13 NÁVRH OKNA OBRATY.....	37
OBR. 14 NÁVRH OKNA INFO O ÚČTE	38
OBR. 15 NÁVRH OKNA PROFILU	39
OBR. 16 USE CASE MODEL NAVRHOVANEJ APLIKÁCIE.....	39
OBR. 17 NAVRHOVANÝ DÁTOVÝ MODEL	40
OBR. 18 ERD DIAGRAM	43
OBR. 19 IMPLEMENTAČNÝ USE CASE MODEL.....	44
OBR. 20 STSAVOVÝ DIAGRAM APLIKÁCIE	45
OBR. 21 NEUSPESNY LOGIN	47
OBR. 22 OKNO OBRATY FILTER1	48
OBR. 23 NESPRAVNY FILTER.....	49
OBR. 24 OKNO FILTER2	50
OBR. 25 OKNO OBRATY FILTER3	50
OBR. 26 OKNO TRANSAKCIÍ	51
OBR. 27 VOLANIE SAMOTNEJ WEBOVEJ SLUŽBY.....	52
OBR. 28 OKNO PROFIL.....	52
OBR. 29 OKNO REGISTRACIA.....	53
OBR. 30 OKNO REGISTRACIA ERROR	53

Zoznam príloh

Príloha 1 – zdrojový kód aplikácie (na CD)