**TECHNICKÁ UNIVERZITA V LIBERCI**
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2612 – Elektrotechnika a informatika

Studijní obor: 1802R022 – Informatika a logistika

# Vizualizace DNS transakcí

# DNS transactions visualisation

**Bakalářská práce**

Autor: **Ondřej Slavík**

Vedoucí práce: doc. RNDr. Pavel Satrapa, Ph.D.

Konzultant: Mgr. Martin Slavík

V Liberci 17. 5. 2013

Originální zadání

## Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum : 17. května 2013

Podpis

## Abstrakt a klíčová slova (v českém jazyce)

Bakalářská práce popisuje tvorbu programu pro vizualizaci komunikace mezi DNS servery a získávání dalších informací distribuovaných pomocí systému DNS a možnosti jejich využití. Práce vznikala v roce 2013 i s přihlédnutím k aktuálním tématům a potřebám na TUL.

V úvodní části autor vysvětluje volbu tématu, rozebírá zadání a cíl práce a soustředí se na teoretické aspekty v oblasti výzkumu DNS transakcí, např. na vazbu mezi „jménem počítače“ a IP adresou.

V druhé části již popisuje samotné DNS transakce, hierarchii domén a syntaxi pro domény. Dále se věnuje resolveru, jmenným serverům, zdrojovým záznamům a použitým programovacím jazykům.

Ve třetí části autor popisuje vývoj aplikace, zdrojové kódy jednotlivých částí a způsob využití aplikace.

Autor se pokouší o rozčlenění programovací jazyků dle relevance a vlastního užití v rámci projektu, který probíhal před samotnou tvorbou bakalářské práce. Popisuje vývojová prostředí jak pro HTML, JavaScript, PHP, tak pro CSS. V této části se bakalářská práce rovněž věnuje grafickému rozhraní celé aplikace, systému souborů v programu a přikládá ukázky kódu pro ïndex.php, ajax.php, prototype.js a style.css“. Dále je ověřena vlastní funkčnost celé aplikace na příkladu několika domén.

V závěrečné části autor popisuje možnosti dalšího využití či případného rozšíření programu o možnosti interaktivní volby a sumarizuje výsledky celého projektu.

**Klíčová slova:** DNS, vizualizace, transakce, jmenný server, doména

## Abstrakt a klíčová slova (v anglickém jazyce)

This bachelor's thesis describes the development of programs for visualising the communication between DNS servers and obtaining other information distributed via DNS and its potential applications. The thesis was written in 2013 with regard to current issues and needs at TUL.

In the introductory part the author explains the choice of topic, and discusses the assignment and its goal, focusing on the theoretical aspects of DNS transactions, such as the relation between a computer's "hostname" and IP address.

The second part describes the DNS transaction as such, domain hierarchy and domain syntax. It also includes discussion of the resolver, name servers, source records and programming languages in use.

In the third part, the autor describes the development of the application, the source codes of the individual parts and way to use the application.

The author attempts to sort programming languages according to their relevance and use in the project, which preceded the actual creation of this thesis. It describes development environments for both HTML, JavaScript and PHP, and CSS. In this part of the thesis he also deals with the graphical interface of the applications and its file system. Attached is a code sample for Ajax.php, prototype.js, Index.php and style.css. The application functionality is verified using several domains as an example.

In the final part the author describes the possibilities of further use of the program or its expansion with interactive options. The results of the whole project are summarized.

**Key words**: DNS, visualisation, transactions, name server, domain

# Obsah

## Seznam obrázků

## Seznam použitých zkratek

DNS - Domain Name System (systém doménových jmen)

HTML - HyperText Markup Language (značkovací jazyk pro hypertext)

HTTP - HyperText Transfer Protocol (protokol pro výměnu dokumentů v HTML)

CSS - Cascading Style Sheets (kaskádové styly)

XML - Extensible Markup Language

AJAX - Asynchronous JavaScript and XML

PHP - Hypertext Preprocessor (původně Personal Home Page )

RR - Resource Records (zdrojové záznamy, věty)

TTL - Time To Live

TLD - Top Level Domain (doména nejvyššího řádu)

SW – Software

HW - Hardware

NS - Name Server (jmenný server)

ROOT NS - Root Name Server (kořenový jmenný server)

TCP/IP - Transmission Control Protocol/Internet Protocol (primární transportní protokol - TCP/protokol síťové vrstvy - IP).

FTP - File Transfer Protocol (protokol pro přenos souborů)

UI designer – návrhář „webového“ rozhraní

localhost – lokální server, odkaz na právě používaný počítač

Pop-up okno – vyskakovací okno

# 1 Úvod

V úvodní kapitole této práce je nastíněna problematika, kterou se autor zabývá, a jsou stanoveny cíle práce.

## 1.1 Co je to DNS

Veškeré současné aplikace, které na Internetu zajišťují komunikaci mezi počítači, používají k identifikaci komunikujících uzlů IP adresu. Pro člověka jako uživatele jsou však IP adresy těžko zapamatovatelné. Proto se používá místo IP adresy název síťového rozhraní. Pro každou IP adresu máme zavedeno jméno přesněji řečeno doménové jméno. Toto doménové jméno můžeme používat ve všech příkazech, kde je možné použít IP adresu. (Výjimkou, kdy se musí jedině použít IP adresa, je specifikace samotného jmenného serveru.)

Jedna IP adresa může mít přiřazeno i několik doménových jmen a naopak pod stejným jménem se může skrývat několik IP adres.

Vazba mezi jménem počítače a IP adresou je definována v DNS databázi. DNS (Domain Name System) je celosvětově distribuovaná databáze. Databáze DNS obsahuje jednotlivé záznamy, které se také nazývají „DNS věty“ (Resource Records – RR). Jednotlivé části této databáze jsou umístěny na tzv. jmenných serverech. [1 str. 245]

## 1.2 Cíl práce

Cílem práce je vytvořit program na zobrazení komunikace mezi DNS servery, tedy zobrazit hierarchii serverů a jimi poskytovaných informací, které postupně vedou ke správné odpovědi na uživatelem položený dotaz. Pro splnění tohoto úkolu si autor vybral prostředí webových aplikací a s ním spojené programovací jazyky JavaScript, PHP, značkovací jazyky HTML a XML, použití kaskádových stylů (CSS) a využití AJAXu (Asynchronous JavaScript and XML), které kombinuje XML a JavaScript a dodává aplikaci „dynamičnost“. Samotné výsledky práce jsou poté prezentovány ve webovém rozhraní, kde uživatel zadá doménu, o které chce zjistit více informací. Poté se skrze strukturu DNS serverů dostane k požadovaným informacím o doméně.

## 1.3 Proč autor zvolil toto téma?

Problematika DNS autora zajímala již delší dobu, ale nikdy neměl příležitost se jí hlouběji věnovat. Dalším důvodem, proč si autor vybral toto téma, byla snaha vylepšit své znalosti protokolů TCP/IP, systému DNS a také využít a následně dále prohloubit své předchozí zkušenosti v programování webových aplikací a zlepšit své schopnosti při použití jazyka PHP, JavaScript, použití technologie AJAX a dalších.

# 2 Rozbor zadání, použité prostředky pro tvorbu aplikace

V druhé kapitole se tato práce věnuje problematice DNS, poté jsou zde popsány jazyky a technologie, které autor v práci použil. Na závěr kapitoly je přiblíženo samotné rozhraní aplikace a hlouběji analyzován kód programu.

## 2.1 Principy činnosti DNS

V této části práce je nastíněna problematika DNS, dále je popsána stromová struktura domén (NS) a IP adres. Dále se tu autor věnuje samotné funkčnosti NS a jednotlivým typům DNS záznamů.

### 2.1.1 Hierarchie domén

K pochopení problematiky sítí a DNS transakcí, je potřeba nejdříve vědět, jak Internet funguje a jak se v něm komunikuje.

Celý jmenný prostor DNS je rozdělen do tzv. domén, tj. skupin jmen, která k sobě logicky patří a která jsou uspořádána do stromové struktury. Z doménových jmen lze odvodit, patří-li jména jedné firmě, jedné zemi apod. V rámci domény můžeme vytvářet podskupiny, tzv. subdomény, např. doméně firmy lze vytvořit subdomény pro oddělení. Z jednotlivých „jmenovek (label)“ je pak složeno doménové jméno uzlu. Např. uzel se jménem jakub.firma.cz je uzel se jménem jakub v subdoméně firma domény.cz.

Doménové jméno se skládá z řetězců vzájemně oddělených tečkou. Jméno se zkoumá zprava doleva. Nejvyšší instancí je tzv. kořenová (root) doména, která se vyjadřuje tečkou zcela vpravo (tato tečka bývá často vypouštěna). V kořenové doméně jsou definovány domény nejvyšší (první) úrovně (Top Level Domains – TLD). Pro Českou republiku je vyhrazena doména .cz. [1, str. 246]

TLD se dále dělí na tyto:

- **Národní TLD** (*country-code TLD, ccTLD*) seskupující domény jednoho státu. Mají dvoupísmenný název, až na výjimky odpovídající kódu země podle normy ISO 3166-1
- **Generické TLD** (*generic TLD, gTLD*) seskupující obecné domény (např. `org` pro *neziskové organizace*), které nejsou spojené s jedním určitým státem (až na výjimku TLD *mil*, *gov* a *edu* které jsou z historických důvodů vyhrazeny pro vojenské, resp. vládní počítačové sítě v USA). Tyto domény jsou tedy využívány po celém světě, například pro globální projekty typu wikipedia.org.
- **Infrastrukturní TLD** používané pro vnitřní mechanismy Internetu. V současné době existuje pouze jediné takové TLD: *arpa*

Z výše uvedených tří kategorií TLD jsou nejrozmanitější generické domény nejvyššího řádu (anglicky generic top-level domain, gTLD). Jsou to domény nejvyššího řádu (alespoň teoreticky), které jsou společné pro určitou oblast zájmu a jejichž název by měl být složený nejméně ze tří písmen.

Dělení gTLD – v první řadě to jsou domény tzv. **neomezené**, to znamená, že jejich použití není nijak omezeno. Jedná se o domény: „.com .info .net .org .eu.“ Dále to jsou domény **vyhrazené** – ty jsou určeny pro specifický účel, ale není to vyžadováno, například doména „.name“. Poté existují ještě domény **vymezené**, které mohou být použity pouze pro daný účel. Jedná se o domény jako : „.biz .int .pro .xxx.“. Pak následují ještě domény **garantované**, které jsou specifické tím, že mohou být využity pouze pro daný účel, navíc mají garanta – organizaci stanovující podmínky registrace a dohlížející na provoz. Označují se sTLD (podle sponsored TLD, kde sponzor znamená garant) a jsou to tyto domény: „.aero .cat .coop .jobs .mobi .museum .travel.“ V současné době ICANN (Internet Corporation for Assigned Names and Numbers) a jeho oddělení IANA (The Internet Assigned Numbers Authority), které je zodpovědné za mnoho klíčových funkcí pro bezproblémový běh internetu (včetně správy DNS Root, atd.), pořádá rozsáhlý výběr, jehož cílem je zavedení řady nových TLD. Zbytek domén již pouze v bodech:

- Exkluzivní - tyto domény smějí využívat pouze servery v USA, navíc s určitými restrikcemi - .edu .gov .mil

- Infrastrukturní - .arpa
- Zrušené - .nato
- Rezervované - .example .invalid .localhost .test
- Připravované - .shop .app .food a mnoho dalších (od ledna 2014 by mělo být dostupných až několik stovek nových TLD)

### 2.1.2 Syntaxe pro domény

Celé jméno domény může obsahovat maximálně 255 znaků, jednotlivá „jmenovka“ poté maximálně 63 znaky. Řetězec může být složen z písmen, číslic a pomlčky. Pomlčka se nesmí vyskytovat na začátku ani na konci řetězce. Existují i rozšíření určující bohatší repertoár znaků možných pro tvorbu jmen. V doméně CZ se však těmto i dalším znakům snažíme vyhnout, protože tato rozšíření v doméně CZ nejsou povolena.

Mohou se použít velká i malá písmena, ale není to zase tak jednoduché. Z hlediska uložení a zpracování v databázi jmen (databázi DNS) se velká a malá písmena nerozlišují. Tj. jméno Newark.com bude uloženo v databázi na stejné místo jako NewArk.com nebo NEWARK.com atp. Tedy při překladu jména na IP adresu je jedno, kde uživatel zadá velká a kde malá písmena. Avšak v databázi je jméno uloženo s velkými a malými písmeny, tj. bylo-li tam uloženo například NewArk.com, pak při dotazu na newark.com databáze vrátí NewArk.com. Poslední tečka je součástí jména. [1, str. 247]

### 2.1.3 Reverzní domény - stromová struktura IP - adres

Reverzními doménami jsou nazvány ty domény, které jsou odvozeny z IP adres. Stejně jako jmenné domény vytvářejí stromovou strukturu. Pro potřeby reverzního překladu byla utvořena pseudodoména „in-addr.arpa“, pro IPverze6 pak „IP6.arpa“. Tato pseudodoména má jméno historického původu, jde o zkratku „inverse addresses in the Arpanet“.

Pod doménou in-addr.arpa jsou domény, které se jmenují jako první číslo z IP adresy sítě. Tj. doména in-addr.arpa má subdomény 0 až 255. Každá z těchto subdomén obsahuje nižší subdomény opět 0 až 255 atd. Např. síť 195.47.37.0/24 patří do domény 37.47.195.in-addr-arpa. Ta sama patří do domény 47.195.in-addr.arpa atd. Všimněte si, že domény jsou zde tvořeny jakoby IP-adresami sítí psanými ale pozpátku. [1, str. 248]

### 2.1.4 Rezervované domény a pseudodomény

Později se ukázalo, že jako TLD je možné využít i jiné domény. Některé další TLD byly rezervovány RFC-2606:

- doména .test pro testování,
- doména .example pro vytváření dokumentace a příkladů,
- doména .invalid pro navozování chybových stavů,
- doména .localhost pro softwarovou smyčku.
- Obdobně byla rezervována doména .local pro intranety. Význam této domény je obdobný jako význam sítě 10.0.0.0/8. [1, str. 250]

### 2.1.5 Resolver

A nyní, když je popsána datová struktura DNS, tak se můžeme dostat k popisu samotné funkčnosti a pokládání dotazů. Přeložení jména na IP adresu zprostředkovává tzv. resolver. Resolver není samostatná aplikace, ale je to klient (obvykle součást operačního systému), který se dotazuje jmenného serveru. Jelikož je databáze celosvětově distribuována, nemusí nejbližší jmenný server znát konečnou odpověď a může požádat o pomoc další jmenné servery. Získaný překlad pak jmenný server vrátí jako odpověď resolveru. Veškerá komunikace se skládá z dotazů a odpovědí.

Jak jsem již dříve zmínil resolver není konkrétní program. Jedná se o skupinu knihovních funkcí, která se přizpůsobuje (linkuje) konkrétní aplikaci, která si vyžádala tuto službu. Může se jednat o aplikace jako je například ftp, telnet, www prohlížeč a další. Každá z těchto aplikací si v případě potřeby vybere z knihovních funkcí ty potřebné a s jejich pomocí vytvoří dotaz, který je poté odeslán na DNS server.

Dalším aspektem při překladu jsou časová omezení. Je totiž možnost, že na dotaz nedostane resolver odpověď, ale při opakování toho stejného dotazu po nějaké, relativně krátké době, již dostaneme korektní odpověď. Tato situace je způsobena tím, že server v mezičase získal odpověď, ale protože ta z jiného jmenného serveru dlouho nepřicházela, tak první dotaz nebyl zodpovězen.

Resolvery můžeme rozdělit na dva typy:

- Plnohodnotný resolver je implementován místním jmenným serverem, na který se obracejí klienti z dané sítě, zná adresy kořenových serverů a dokáže kompletně vyřešit dotaz.
- Koncový resolver je zjednodušený, je obsažen v operačních systémech koncových počítačů a zná jen adresu místního serveru (s plnohodnotným resolverem), kterému má přeposílat dotazy svých aplikací

Ještě bych se rád stručně zmínil o vlastnostech resolveru a jeho konfiguraci. Možnosti konfigurace závisí na operačním systému, na kterém pracujeme. Nejdůležitější atribut, jak při konfiguraci na Unixu, Windows a dalších systémech, je korektní nastavení adresy (IP adresy, ne doménové jméno!) jmenného serveru, ke kterému bude resolver přistupovat.

### 2.1.6 Jmenný server

Na jmenném serveru jsou uloženy informace pro překlad názvů počítačů na IP adresy (resp. i naopak pro reverzní překlad). Jmenný server se stará o určitou část z prostoru jmen všech počítačů. Tento prostor, o který se stará, nazýváme zóna. Tato zóna je tvořena celou doménou, a to buď včetně všech subdomén, nebo jen vybranými subdoménami.

Jmenný server během svého startu načte do paměti data pro zónu, kterou spravuje. Primární jmenný server načte data z lokálního disku, sekundární jmenný server získá pro spravované zóny dotazem zone transfer data z primárního jmenného serveru a rovněž je uloží do paměti. Tato data primárního a sekundárního serveru se označují jako autoritativní (nezvratná). Dále jmenný server načte z lokálního disku do paměti data zóny cache/hint, která nejsou součástí dat jeho spravované zóny, ale umožní mu spojení s kořenovými (root) jmennými servery. Tato data se označují jako neautoritativní. [1, str. 251]

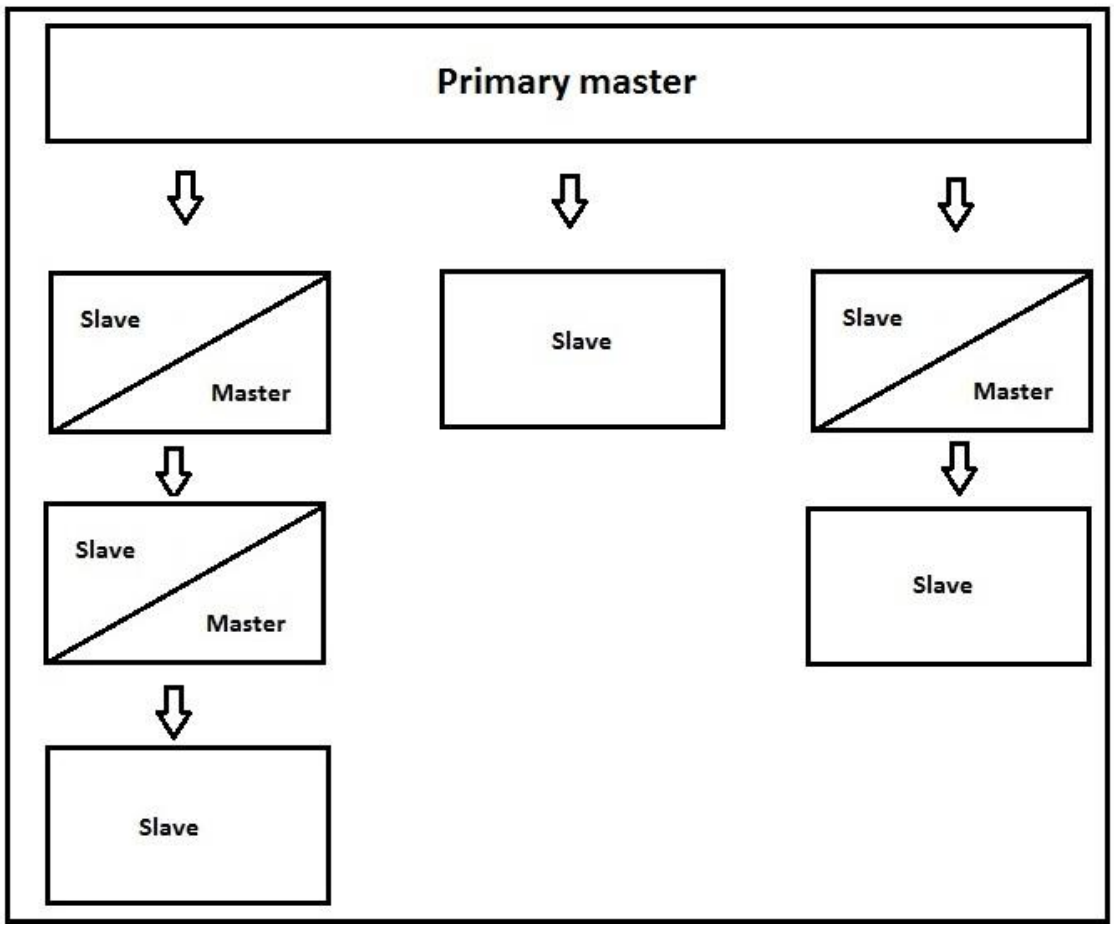
Pomocí věty NS (v zónových datech) je totiž možné delegovat udržování subdomény na jmenný server nižší úrovně. Tím je subdoména vyjmuta ze zóny serveru a je delegována na server „nižší úrovně “, který ji nyní bude spravovat. Jmenné servery můžeme rozčlenit podle typu uložení dat. Máme v podstatě 5 typů serverů. (viz Tab. 1)

**Tab. 1: Rozdělení jmenných serverů podle typu uložených dat**

| Typ serveru | Typ uložení dat |
|---|---|
| Primární jmenný server | Je to pro zónu primární zdroj, autoritativní server zóny. Data o své zóně čerpá z databází, které jsou uloženy na lokálních discích. Bývá také označován „master“. Databáze tohoto serveru jsou vytvářeny správcem dané domény. Tento primární server musí být uveden v NS záznamu jako autoritativní jmenný server (pro každou zónu je právě jeden). |
| Sekundární jmenný server | Data o doméně extrahuje z primárního jmenného serveru. Sekundární server má svou kopii dat (databáze), kterou aktualizuje při změně; na tu je buď aktivně upozorněn primárním serverem (NOTIFY) nebo se sám dotáže na SOA záznam a pokud došlo ke změně sériového čísla, stáhne si aktuální verzi (AXFR/IXFR). Když jsou tyto databáze uložené na disku, tak se již dále needitují, protože se v další intervalu vše stáhne z primárního serveru znovu. Tyto servery jsou autoritativní pro svou zónu. Používá se pro něj také označení „slave“. |
| Caching only server | Pro danou doménu neplní funkci ani primárního ani sekundárního jmenného serveru. Používá obecnou vlastnost jmenných serverů, tzn., že data, která jím putují, ukládá ve své paměti (existuje ale i řada autoritativních serverů bez cache - např. software NSD žádnou cache nemá) Tato data se poté označují jako neautoritativní. Server může vykonávat i více funkcí najednou. Může být pro některé domény primární, sekundární pro další a caching only pro zbytek domén. |
| Root name server | Autoritativní jmenný server starající se o root doménu. Všechny tyto root servery jsou zároveň i primární servery a právě tím se od ostatních jmenných serveru odlišuji. |
| Stealth server | Tajný autoritativní server, který není nikde zveřejňován. Vědí o něm pouze ty servery, které mají jeho IP adresu zanesenu v konfiguraci. Používá se z důvodu bezpečnosti (není na očích, tudíž se snižuje riziko útoku) nebo proto, že mezi ním a veřejnými servery probíhá nějaké další zpracování DNS dat, typicky DNSSEC podpis. |

Zdroj: Selection and Operation of Secondary DNS Servers. Dostupný z WWW: http://tools.ietf.org/html/rfc2182

Jeden jmenný server může být pro určitou zónu master server a pro nějaké je slave serverem. Pro každou zónu musí existovat právě jeden primární a alespoň jeden sekundární server. Každá zóna je tedy povinna mít alespoň dva autoritativní servery, pro případ výpadku jednoho z nich. Tuto architekturu více přiblíží diagram na Obr.1.




**Obr. 1 –** Master/slave architektura jmenných serverů

Z hlediska uživatelů je jedno zda komunikuje s master či slave serverem, pro určenou zónu jsou oba dva autoritativními servery, mají data stejné váhy. Klient ani neví (z DNS odpovědi se ani nedozví), s jakým typem serveru komunikuje. Oproti tomu caching server poskytuje neautoritativní odpovědi a pokud odpověď nezná, kontaktuje autoritativní server pro danou zónu. Jmenný server (označovaný též DNS server) naplňuje svou paměť několika způsoby. Autoritativní data načte ze souborů na disku, nebo je získá pomocí dotazu zone transfer z paměti jiného serveru. Neautoritativní data jmenný server získává postupně z odpovědí jiných serverů, tak jak vyřizuje jednotlivé DNS dotazy. [1, str. 263]

### 2.1.7 Zdrojové záznamy (věty) - RR

Informace o doménových jménech a jim příslušných IP adresách, stejně tak jako všechny ostatní informace distribuované pomocí DNS, jsou uloženy v paměti jmenných serverů ve tvaru zdrojových vět (Resource Records – RR).

V případě, že DNS klient potřebuje získat informace z DNS, pak požaduje po jmenném serveru věty RR podle zadaných požadavků. [1, str. 263]

Klientem, o kterém mluvíme, bývá vždy resolver, buď na straně uživatele, případně resolver použitý jiným jmenným serverem, který na daný dotaz nezná odpověď. V protokolu DNS mají všechny zdrojové věty stejnou strukturu a skládají se z následujících polí:

- NAME – jméno domény
- TYPE – typ věty
- CLASS – třída věty
- TTL – Time to live. Doba, po kterou bude tento záznam v cache serveru uváděn jako platící. Po vypršení této doby je záznam neplatný
- RDLENGHT – určuje délku pole RDATA
- RDATA – řetězec měnící se délky. Ta závisí na typu a třídě RR

Pro větší přehlednost je níže uveden přehled vybraných RR vět podle typu v Tab. 2.

## Tab. 2: Vybrané Typy RR vět

| **Typ** | **Název (anglický)** | **Popis pole** |
|---|---|---|
| A | address | Obsahuje IPv4 adresu přiřazenou danému jménu (32 bitů) |
| AAAA | IPv6 address | Obsahuje IPv6 adresu (128 bitů) |
| NS | name server | Obsahuje název autoritativního DNS serveru, který je autoritou pro danou doménu |
| CNAME | canonical name | Alias - jiné jméno pro již existující a známé jméno, využívá se pro servery zavedených služeb, jako je například WWW. Jeho určení pomocí přezdívky jej později dovoluje jednoduše přemístit na jiný počítač |
| MX | mail exchange | Ukazuje adresu a prioritu serveru pro příjem elektronické pošty pro danou doménu. Tentokrát obsahuje dvě pole - preference (přirozené číslo, menší znamená vyšší prioritu) a doménové jméno e-mailového serveru |
| SOA | **s**tart **o**f **a**uthority | Je uvozující záznam zónového souboru. Obsahuje název primárního serveru, adresu elektronické pošty jejího správce (zavináč nahrazen tečkou) a dále:<br>*Serial* - sériové číslo (potřeba zvýšit s každou změnou v záznamu), podle něj sekundární server pozná, že je doména změněna;<br>*Refresh* - interval, v kterém se má sekundární server ptát na novou verzi zóny (v sekundách);<br>*Retry* - jak často má sekundární server opakovat své pokusy, pokud se mu nedaří spojit s primárním<br>*Expire* - čas po kterém jsou záznamy na sekundárním serveru označeny za neaktuální (nedaří se kontaktovat primární server)<br>*TTL* - implicitní doba platnosti záznamů |
| PTR | pointer | Zvláštní typ záznamu pro reverzní zóny (použití pro reverzní překlad) |
| SRV | | specifikace umístění konkrétní služby (podle čísla portu) na dané doméně |
| TXT | TeXT | Textová poznámka |
| NSEC | Next Secure Record | Část DNSSEC, informuje klienta o neexistenci hledaného záznamu |
| AXFR | | Požadavek na získání transferu celé zóny |
| IXFR | Incremental Zone Transfer | Požadavek na získání inkrementálního zone transferu |
| DNSKEY | DNSSEC public key | Veřejný klíč pro ověření identity serveru (aby nešla podvrhnout webová stránka) |
| NSEC3 | Next-Secure record | Rozšíření DNSSEC, „hashované“ klíče |
| RRSIG | DNSSEC signature | Digitální podpis pro ověření identity serveru, využívá NSEC3 |
| * | | Získání všech vět |

Zdroj: Domain Name System (DNS) Parameters Dostupný z WWW: http://www.iana.org/assignments/dns-parameters/dns-parameters.xml

## 2.2 Použité programové vybavení (programovací jazyky, nástroje pro programování)

V této části práce je popsáno programové vybavení, které autor využil pro dosažení cílů práce.

### 2.2.1 HTML

HyperText Markup Language, označovaný zkratkou HTML, je značkovací jazyk pro hypertext. S jeho pomocí se vytvářejí stránky v systému World Wide Web, který umožňuje prezentaci na Internetu. Jazyk je aplikací dříve vytvořeného rozsáhlého univerzálního značkovacího jazyka SGML (Standard Generalized Markup Language). Vývoj HTML zpětně ovlivňovaly webové prohlížeče.

V roce 1990 byl navržen jazyk HTML a protokol pro jeho přenos v počítačové síti – HTTP (HyperText Transfer Protocol – přenosový protokol hypertextu). Zároveň také Tim Berners-Lee vytvořil první webový prohlížeč, který nazval WorldWideWeb.

HTML je charakterizován množinou značek (tzv. *tagů*) a jejich atributů lišících se podle verze. Mezi tagy se uzavírají části textu dokumentu a tím se určuje jejich význam (*sémantika*). Názvy jednotlivých značek se umisťují mezi úhlové závorky < a >. Část dokumentu skládající se z otevírací značky, nějakého obsahu a odpovídající ukončovací značky tvoří tzv. *element* (prvek) dokumentu.

Dokument v jazyku HTML má předepsanou strukturu:

- Kořenový element – element `html` (značky `<html>` a `</html>`) uvozuje a končí celý dokument.
- Hlavička dokumentu – obsahuje metadata, vztahující se k celému dokumentu. Definují např. název dokumentu, jazyk, kódování, klíčová slova, popis, použitý styl zobrazení. Hlavička je uzavřena mezi tagy `<head>` a `</head>`.
- Tělo dokumentu – obsahuje vlastní text dokumentu. Tvoří se značkami `<body>` a `</body>`.

### 2.2.2 Cascading Style Sheets (CSS) - Kaskádové styly

CSS popisuje vzhled webů, odděluje obsah a prezentaci webové stránky. Styly jsou kaskádové, protože je možno je aplikovat z externího souboru, zevnitř sekce stylů

webové stránky nebo z řádky a každá z nižších úrovní stylů přepisuje jakoukoliv dříve definovanou stylovou charakteristiku. [2, str. 192]

Soubor kaskádových stylů se sestává z několika pravidel. Každé pravidlo zahrnuje selektor a blok deklarací. Každý blok deklarací pak má seznam deklarací oddělených středníky ; a každá deklarace sestává z identifikátoru vlastnosti, následuje dvojtečka : a hodnota vlastnosti. Nepovinně ještě může následovat označení !important, které zvýší sílu deklarace.

CSS definuje mnoho různých selektorů, které obvykle můžeme kombinovat. Mezi základní patří:

- `body` – Tyto deklarace platí pro všechny výskyty elementu `body`.
- `body p` – Tyto parametry budou určující pro všechny elementy `p`, které se nachází v elementu `body`, v jakékoliv hloubce.
- `body>div` – Tyto deklarace budou určovat všechny elementy `div`, které jsou potomky elementu `body`. Tudíž pokud bychom měli element `div`, který se nachází v `<body><blockquote><div>…`, tyto deklarace by pro něj neplatily, protože tento `div` není přímým dítětem elementu `body`.
- `.trida` – Tyto deklarace budou platit pro všechny elementy, které mají v HTML nastavenou třídu `trida`. Toho se využívá pomocí tagu atributem `class`.
- `#identifikátor` – Tyto deklarace budou platit pro všechny elementy, které mají v HTML nastaveno id `identifikátor`.

### 2.2.3 JavaScript a AJAX

JavaScript je multiplatformní, objektově orientovaný skriptovací jazyk, jehož Autorem je Brendan Eich z tehdejší společnosti Netscape.

JavaScript je určen především k programování aktivních částí WWW stránek. V omezené míře jej lze použít i k vytváření skriptů na straně serveru či dokonce pro normální aplikace. [3, str. 269]

Jsou jím obvykle řízeny různé aktivní prvky GUI (tlačítka, textová políčka) nebo tvořeny animace a efekty obrázků. Jeho syntaxe poněkud připomíná C/C++/Java.

Slovo Java je v jeho názvu pouze z marketingových důvodů a s Javou jako programovacím jazykem jej vedle názvu spojuje jen místy podobná syntaxe. JavaScriptu byl ustaven standart asociací ECMA (Europen Computer Manufacturers Association) v červenci 1997 a v srpnu 1998 organizací ISO (International Organization for Standardization).

Program v JavaScriptu obvykle nabíhá až po stažení WWW stránky z Internetu (tzv. na straně klienta), na rozdíl od ostatních jiných interpretovaných programovacích jazyků (např. PHP a ASP), které pracují na straně serveru ještě před stažením z Internetu. Z toho plynou jistá bezpečností omezení, JavaScript např. nemůže pracovat se soubory, aby tím nenarušil soukromí uživatele

JavaScript je jednoduchý skriptovací jazyk, využívající objekty, určený pro snadné webové programování. Ačkoliv se i v JavaScriptu objevují prvky objektově orientovaných jazyků, má blíže ke skriptovacím jazykům jako je Perl. Hodí se k provádění jednoduchých úkolů, například ke kontrole údajů ve formuláři, dynamickému generování HTML kódu a k základním výpočtům, týkajícím se data a času nebo typu prohlížeče. [4, str. 59]

JavaScript je ale také využíván pro technologii AJAX.

Ajax byl jako zkratka pro asynchronní JavaScript a XML vytvořen Jamesem Garrettem v článku publikovaném na webové stránce společnosti Adaptive Path. Tento článek publikoval v únoru 2005 na webu s názvem Ajax: A New Approach to Web Applications (Ajax: nový přístup k webovým aplikacím).

Ajax se doslova vřítil na scénu webových aplikací a nabídl vysoce interaktivní webové aplikace náramně se podobající těm desktopovým, neboť mění obsah svých stránek bez nutnosti jejich znovunačtení.

Často se pomocí Ajaxu realizují tzv. našeptávače. Když píšete něco do textového políčka, tak je váš text průběžně odesílán na server, zpětně pak dostáváte nápovědná slova, která byste mohli chtít napsat. [5]

Podobným způsobem byl využit i v aplikaci autora, kde také neustále kontroluje a odesílá obsah textového pole. A má i další využití, např. vyhodnocování již zadaných dat uživatelem do formuláře na straně serveru, zatímco je vyplňován jeho zbytek. Aplikace od společnosti Google jsou toho velmi dobrým příkladem.

Ovšem ani takové řešení jako AJAX není zcela dokonalé. Vyhledávače nemusí vždy indexovat všechny stránky Ajaxové aplikace a ne vždy jdou tyto stránky uložit do záložek. V praxi se autor setkal s problémy u komerčně zaměřených webových projektů, které nemohli odkazovat své reklamy (bannery, textové reklamy, atd.) na konkrétní URL adresy s jimi propagovanou nabídkou, ale pouze na nějaké obecnější části „webu“. Tím samozřejmě tyto reklamy ztrácí na účinnosti, protože potenciální zákazník ihned nenachází to, co hledal, a proto se musí k požadovaným stránkám ještě „proklikat“. Také se Ajaxová aplikace stává nefunkční, pokud má uživatel vypnutý JavaScript.

### 2.2.4 PHP

Ačkoliv se na vývoji PHP podílí několik autorů, jeho skutečným duchovním otcem je Rasmus Lerdorf. První parser pro PHP napsal v roce 1995 jako CGI skript v Perlu, kterému říkal „Personal Home Page“, nebo jen PHP. Původně jej používal pro webovou knihu návštěv.

PHP je vedle Active Server Pages (ASP), a Perlu jedním z nejrozšířenějších serverových skriptovacích jazyků. PHP se vyznačuje neuvěřitelnou rozmanitostí a lze jej použít i pro samostatné, s webem nesouvisející, aplikace. Nejčastěji ale bývá používán pro dynamické předzpracování stránek na unixových serverech (především u Apache; viz http://www.apache.org). Jedná se tedy o nejpoužívanější rozšíření pro Apache. [4, str. 43]

Syntaxe jazyka čerpá z několika programovacích jazyků (Perl, C, Pascal a Java). PHP je nezávislý na platformě, rozdíly v různých operačních systémech jsou zanedbatelné a omezují se na několik OS-závislých funkcí a skripty lze většinou mezi nimi přenášet bez jakýchkoli změn.

PHP podporuje mnoho knihoven pro různé účely - např. zpracování textu, grafiky, práci se soubory, přístup k většině databázových systémů (mj. MySQL, ODBC, Oracle, PostgreSQL, MSSQL), podporu celé řady internetových protokolů (HTTP, SMTP, SNMP, FTP, IMAP, POP3, LDAP, …).

PHP je vedle ASP, jedním ze dvou nejrozšířenějších skriptovacích jazyků pro web na světě. Oblíbeným se stal hlavě díky jednoduchosti použití, bohaté zásobě funkcí, a tomu, že kombinuje vlastnosti více programovacích jazyků a je tak tolerantnější

k vývojáři, který má částečnou svobodu v syntaxi. V kombinaci s operačním systémem Linux, databázovým systémem (obvykle MySQL nebo PostgreSQL) a webovým serverem Apache je často využíván k tvorbě webových aplikací. Pro tuto kombinaci se vžila zkratka LAMP – tedy spojení Linux, Apache, MySQL a PHP nebo Perl. V PHP se tvoří i ty největší internetové projekty.

Výhod PHP je velmi mnoho, za hlavní považuje autor tu, že PHP je specializované na webové stránky a navíc obsahuje rozsáhlý soubor funkcí v základní knihovně PHP (přes pět a půl tisíce) s možností doinstalování dalších knihoven funkcí. Další velkou výhodou je nativní podpora mnoha databázových systémů (SQl, MySQL atd..). Navíc je PHP multiplatformní programovací jazyk (zejména Linux a Microsoft Windows). Zbylé výhody jsou popsané již pouze v bodech:

- Šance použití nativních funkcí operačního systému (poté ale hrozí nekompatibilita s jiným OS).
- Velká podpora na hostingových službách – PHP je v podstatě standardem, který najdeme všude.
- Velké množství projektů a kódů, které lze zdarma využít (WordPress, phpBB a další).
- Velice slušná dokumentace.

### 2.2.5 Oracle VM Virtual box, Unix – Ubuntu server, BIND

Na vývoj aplikace autor využíval aplikaci Virtual box, která vytvoří virtuální počítač, na který byl nainstalován Unix Ubuntu pro servery. Na tomto serveru běžel „localhost“, na kterém byla aplikace testována a díky unixovému DNS softwaru BIND i dokončena. Ale k tomuto rozhodnutí bylo potřeba dospět. Nejdříve autor využíval volně dostupné „hostingy“, ale ty časem začaly být nevyhovující, jak rychlostí, tak i nemožností využívat všechny možnosti jazyka PHP, které byly potřeba pro dokončení aplikace. Poté ještě autor testoval využíti „Localhostu“ přímo pod Windows a to v podobě balíku aplikaci XAMPP, který obsahuje Apache, PHP, mySQL a další aplikace. Bohužel, ani XAMPP plně nevyhovoval představám autora, proto přešel na unixový server.

### 2.2.6 Vývojové prostředí pro HTML, JavaScript a PHP – PSPad

PSPad je celosvětově rozšířený freewarový textový editor a editor zdrojových kódů pro platformu Microsoft Windows vyvíjený v prostředí Delphi. Program vyvíjí český programátor Jan Fiala, první verze vyšla v roce 2001.

Program neobsahuje nekonečnou řadu zbytečných funkcí, které většina uživatelů stejně nikdy nevyužije a soustředí se spíše na jednoduchost, přehlednost a rychlost, to je také jeden z důvodů, proč si ho autor oblíbil při tvorbě webových aplikací. Tento program má mnoho výhod, které ušetří mnoho práce. Jedna z hlavních výhod je funkce zvýraznění syntaxe s automatickým nastavením dle typu souboru a zvýraznění skriptů v rámci HTML, které výrazně zlepšuje přehlednost kódu. Pro urychlení operací při kopírování souborů je to integrovaný FTP klient. Také velmi často využívaná funkce je otevření více dokumentů současně, případně přeformátování, komprese a kontrola HTML kódu. Pro doplnění shrnuji v bodech i další výhody tohoto řešení:

- neomezená délka editovaného textu
- porovnání obsahu textových souborů,
- export včetně zvýraznění do RTF a HTML
- definice externích programů, ve kterých je možné soubor otevřít, kompilace…,
- sloupcové a řádkové bloky, záložky v textu,
- změny velikosti písmen, odstranění a přidání diakritiky, formátování kódu

### 2.2.7 Vývojové prostředí pro CSS – TopStyle Lite

TopStyle je program vytvořený v první řadě pro tvorbu CSS. Hlavní výhodou je, že program pomáhá k tvorbě takového kódu, který se správně zobrazuje ve všech předem definovaných prohlížečích. Výhody použití tohoto softwaru jsou v první řadě přehlednost a rychlost práce. Možnost integrace do jiných editorů a také integrovaný seznam všech hodnot a všech jejich atributů. Pro doplnění ještě umožňuje:

- Průvodce vytvářením selektorů.
- Zjištění kompatibility s mnoha verzemi prohlížečů.
- Paleta CSS barev. Tj. barvy, které se vyskytují ve vašem CSS dokumentu.

# 3 Popis aplikace, jejího rozhraní a technické realizace

Nyní se již začneme zabývat samotným jádrem této práce, webovou aplikací pro popis funkce DNS transakcí. V první kapitole je stručně popsán vývoj aplikace, v jehož průběhu se vyskytly problémy a překážky, které vyžadovaly řešení. V dalších kapitolách bude zmíněna technické realizace a blíže analyzován kód aplikace a jeho funkčnost a využití.

## 3.1 Průběh vývoje aplikace

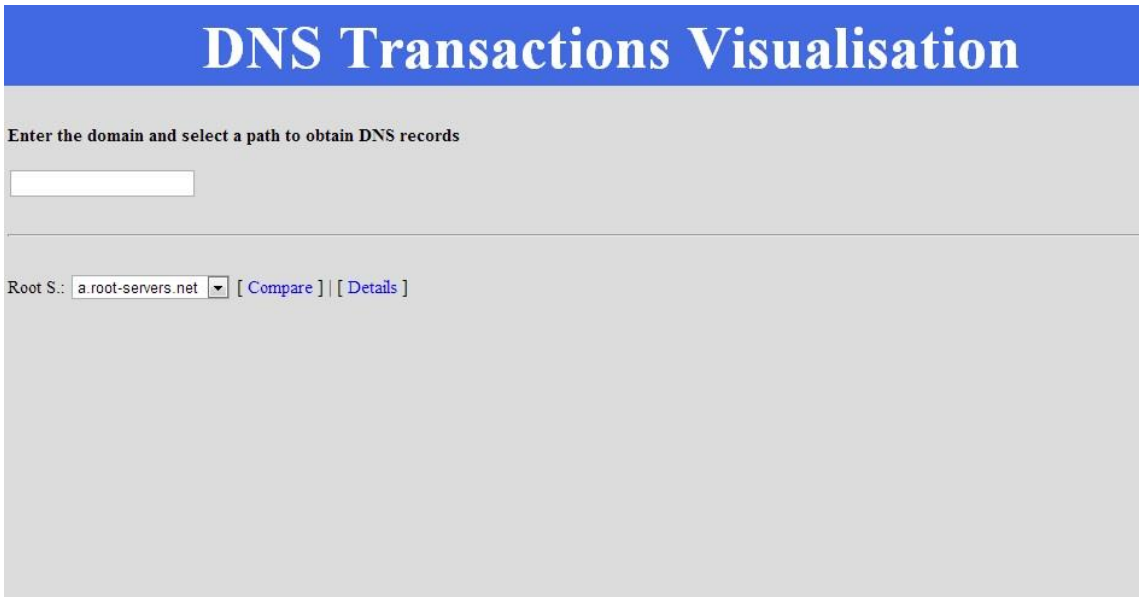
Na počátku bylo vytvořeno grafické rozhraní aplikace, u čehož se autor práce zaměřil hlavně na tři věci: přehlednost, jednoduchost a funkčnost. Autor práce se již delší dobu zabývá vyvíjením webových aplikací a tyto teoretické i praktické dovednosti mohl aplikovat při vyvíjení této aplikace. Největší problém byl s dotazy na konkrétní NS. Funkce PHP (**dns_get_record**), kterou k tomu autor využíval, uměla pouze získat DNS záznamy z NS, ale neumožňovala zeptat se konkrétního serveru. Autor nalezl knihovnu funkcí **Net_DNS**, kterou si ale na „free hostingu“ nemohl nainstalovat.

Proto byl pro další vývoj zvolen balíček aplikací Xampp, který obsahuje všechny potřebné programy pro vytvoření lokálního serveru (localhost) pod Windows. Bohužel, přes veškeré pokusy, ani tento přístup nepřinesl potřebnou funkčnost. Proto autor zvolil instalaci softwaru Virtual box, který umožňuje instalaci virtuálního počítače.

Pro virtuální počítač byla zvolena unixová distribuce Ubuntu server. Po zpřístupnění localhostu byla naprogramována webová aplikace, která již umožňovala oslovit s dotazem konkrétní NS. Byl pro to použit DNS software BIND, který je součástí této distribuce. Konkrétně je to využití funkce „dig“, s kterou je možné dotazovat se konkrétních Name Serverů. Poté byla s využitím této funkce vyvinuta aplikace pro vizualizaci DNS transakcí. Na závěr byla do této aplikace implementována technologie Ajax s použitím velice rozšířených knihoven prototype.js, scriptaculous.js a window.js. Z knihovny prototype.js byly využity metody „Ajax.Updater“ a „$F“. Pomocí knihoven window.js a scriptaculous.js jsou vytvářena „vyskakovací“ okna a upravovány jejich vlastnosti a animace. Samotná grafická úprava aplikace byla provedena pomocí implementace CSS stylů a tím byla vizuálně přizpůsobena do

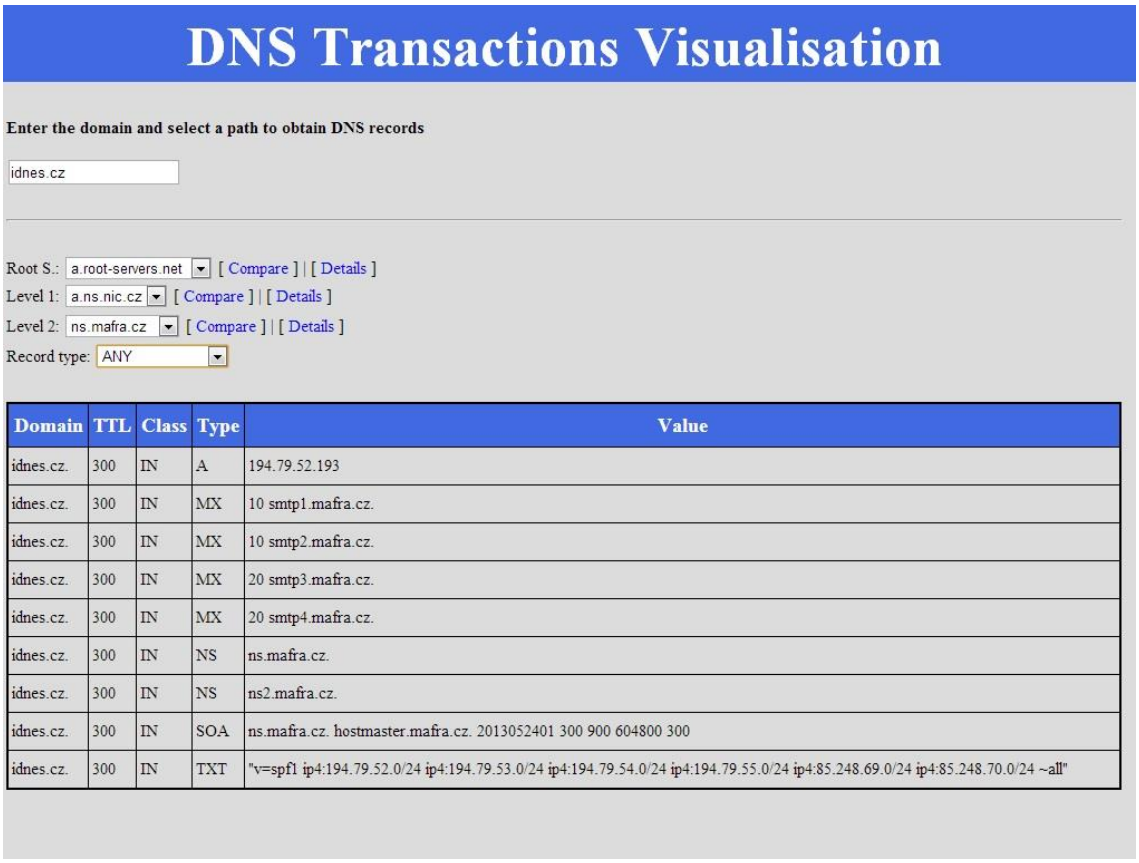
podoby, kterou vidíte na Obr. 2. Implementace CSS a jejich tvorba bude zmíněna v následujících kapitolách, kdy budou rozebrány jednotlivé součásti programu.

Při tvorbě tohoto grafického rozhraní a „ladění“ CSS nevzniklo mnoho problémů a až na několik krátkých prostojů, vše fungovalo jak má a bylo rychle připravené pro potřeby této aplikace.




**Obr. 2 –** Rozhraní aplikace před zadáním domény

Po připravení úvodní strany přišlo na řadu zpracování výstupu z této aplikace. Vzhledem k udržení přehlednosti zobrazení, bylo zvoleno generování roletek s výběrem NS, dvěma „tlačítky“ („Compare“ slouží pro porovnání odpovědí Name Serverů a „Details“ pro detailní zobrazení odpovědi aktuálního NS příslušné úrovně) a nakonec tabulky, do které se generují konkrétní DNS záznamy. I tuto část programu a použité CSS autor popíše podrobněji v následujících kapitolách. Samotné grafické rozhraní zobrazování výsledku je vidět na Obr. 3. V tomto případě je to vypsání DNS dotazu ANY (tzn., zobraz všechny záznamy o doméně) v přehledné tabulce. Tu lze také v CSS stylech upravovat.




**Obr. 3 –** Rozhraní, zobrazení výsledku dotazu ANY

## 3.2 Programová část

A nyní se již začneme zabývat samotným jádrem této práce, webovou aplikací. V první kapitole bude popsán systém souborů a v dalších kapitolách bude blíže rozebrán kód programu a jeho funkce

### 3.2.1 Systém souborů aplikace

Tato aplikace je sestavena ze tří typů souborů. V souborech *.php je programová část, v souborech *.css je uloženo nastavení kaskádových stylů pro grafické zpracování webu a konečně v souborech typu *.js jsou uloženy metody, které se používají při zpracování vstupních dat a zobrazování „vyskakovacích“ (Pop-up) oken. Pro přehlednost budou nyní jednotlivé soubory stručně popsány. Index.php je základním souborem, který se načte při zadání webu. Po jeho načtení je propojen se soubory prototype.js, window.js, scriptaculous.js, které se starají a správnou funkčnost Ajax technologií. Také je propojen se soubory kaskádových stylů Style.css (upravení celkového vzhledu stránek), default.css a alphacube.css (upravení vzhledu Pop-up

oken). Díky technologii Ajax již z této stránky nemusí uživatel odcházet a načítat další stránku (včetně hlavičky atd.) znovu, ale vše se již načítá, počítá a vytváří na pozadí. Uživateli se pouze načítají (aktualizují) požadovaná data, případně otevírají a zavírají pouze jím požadovaná Pop-up okna. Tyto změny a výpočty se dějí pouze při akci typu změna položky v roletce, změna textu v textovém poli, případně zvolením nějakého tlačítka. Všechny změny na obrazovce jsou neustále ovlivňovány CSS styly ze souborů typu *.css.

### 3.2.2 Kód souboru - index.php

Základní soubor webové aplikace je index.php. V první části souboru jsou uvedeny základní deklarace, meta tagy. V tagu link se „volají“ a načítají soubory kaskádových stylů (*.css)

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
    <title>DNS transactions visualisation</title>
    <link rel="stylesheet" href="style.css" type="text/css" />
    <link href="js/themes/default.css" rel="stylesheet" type="text/css" >
    <link href="js/themes/alphacube.css" rel="stylesheet" type="text/css" >
```

V dalším pokračování kódu se v těle (tag *body*) stránka rozděluje pomocí tagů * * na jednotlivé sekce, která má každá své vlastní *id*, podle kterého je bude možné v budoucnu rozeznat a ovlivňovat jejich vzhled pomocí CSS. Další zajímavá část je textové pole input. Zde se zadává doména, kterou chce uživatel prověřit. Jsou zde také vidět javascriptové kontroly (keyUp a updateDomain), ty si rozebereme později. A je tu také první „roletka“ (tag select, s JS kontrolou updateDomain ), s výběrem možností (tag Option), Implicitně je zvolen „a“ Root server a na všech dalších úrovních výběru je zvolen vždy první NS ze seznamu, ale po úvodním načtení všech informací o doméně (všech úrovní NS) si již uživatel může volit konkrétní NS dle svých zájmů. Také jsou zde generována „tlačítka“ *Compare* a *Details*, která pomocí funkcí rootCompare a rootDetails generují Pop-up okna s porovnáním NS a detailem odpovědi root NS.

```
<body>
  <div id="hlavni">
    <div id="logo"><h1>DNS Transactions Visualisation</h1>
```

```
    </div>
    <div id="obsah">
     <p><b>Enter the domain and select a path to obtain DNS records</b>
     </p>
     <input type="text" name="domain" id="domain" onchange="updateDomain()"
onkeyup="keyUp()" /><br />
     <div> 
     </div>
     <HR />
     <div> 
     </div>Root S.:
     <SELECT id="rootns" name="rootns" onchange="updateDomain()">
      <? foreach(array('a','b','c','d','e','f','g','h','i','j','k','l','m') as $r) { ?>
      <OPTION VALUE="<?= $r ?>.root-servers.net">
      <?= $r ?>.root-servers.net
      </OPTION>
      <? } ?>
     </SELECT> [
     <span class="compare" onclick="rootCompare()">Compare
     </span> ] | [
     <span class="compare" onclick="rootDetails()">Details
     </span> ] <br />
     <div id="l1" class='se'>
     </div>
     <div id="l2" class='se'>
     </div>
     <div id="l3" class='se'>
     </div>
     <div id="l4" class='se'>
     </div>
     <div id="l5" class='se'>
     </div>
     <div id="l6" class='se'>
     </div>
     <div id="l7" class='se'>
     </div>
     <div id="l8" class='se'>
     </div>
    </div>
   </div>
  </body>
```

V další části kódu vidíme propojení JS (**prototype.js**, **Scriptaculous.js** a **window.js**), poté deklarace potřebných proměnných a JS funkce pro kontrolu a pro zobrazení „Please wait“. První kontroluje textové pole, zda v něm někdo něco nepíše, pokud v něm nikdo nic nepsal více jak 1,5s (1500ms), přejde se k další kontrole. Druhá funkce hlídá, aby se při načítání nedala změnit hodnota textového pole, proti naposled vložené hodnotě a přidává animaci načítání při zobrazení „PleaseWait“, vice na Obr.4.

Poté je zde také funkce pro zrušení zobrazení načítání, která je volána při dokončení procesu.

```
<script type="text/javascript" src="js/prototype.js"></script>
<script type="text/javascript" src="js/scriptaculous.js"></script>
<script type="text/javascript" src="js/window.js"></script>
<script>
   var ld = '';
   var ns = '';
   var levels = [ '_', '_', '_', '_', '_', '_', '_', '_', '_', '_' ];
   var lastTimeout = false;
   var infowin;
   var cStarted = false;
   function keyUp() {
     if (lastTimeout != false) {
       clearTimeout(lastTimeout);
     }
     lastTimeout = setTimeout('updateDomain()', 1500);
   }
   function pleaseWait()
   {
     $('domain').disabled = true;
 infowin = Dialog.info('Please wait...', {className: "alphacube", width:250,
showProgress: true, showEffectOptions: {duration:0.0}, showEffect:Effect.BlindDown,
hideEffectOptions: {duration:0.0}});
   }
   function stopWait()
   {
     infowin.close();
     $('domain').disabled = false; }
```


**Obr. 4 –** Čekání na výsledek dotazu

Pokud je tedy do textového pole zadána doména zobrazí se: "Please wait..." viz. Obr. 4 a volá se přes metodu Ajax.Updater (z prototype.js) soubor ajax.php, funkce updateLevel, které se předají hodnoty proměnných. Po proběhnutí funkce a vrácení výsledku se funkce posune o úroveň dál (z root serveru máme na NS pro CZ doménu a poté opět ještě nižší úroveň NS) a poté probíhají další cykly, dokud se nedostaneme na poslední úroveň NS.

```
  function updateDomain()
 {
  if (($F('domain') == ld) && ($F('rootns') == ns)) { return; }
  ld = $F('domain');
  ns = $F('rootns');
  for (var i = 2; i < 9; i++) {
   $('l'+i).innerHTML = '';
   levels[i] = '_';
  }
  $('l1').innerHTML="Please wait...";
  pleaseWait();
  new Ajax.Updater('l1', 'ajax.php', {
   parameters: { domain: $F('domain'), ns: $F('rootns'), level: 1 }, evalScripts: true
  });
 }
 function updateLevel(level)
 {
  if (level > 7) { return; }
  if (levels[level] == $F('sl'+level)) { return; }
  for (var i = level; i < 9; i++) {
   $('l'+i).innerHTML = '';
   levels[i] = '_';
  }
  levels[level] = $F('sl'+level);
  $('l'+level).innerHTML="Please wait...";
  pleaseWait();
  new Ajax.Updater('l'+level, 'ajax.php', {
   parameters: { domain: $F('domain'), ns: $F('sl'+level), level: level }, evalScripts:
true
  });
 }
```

Pokud dojdeme na poslední úroveň NS, tak poté se zruší Pop-up okno s „Please wait“ a uživatel následně může z roletky vybrat typ záznamu, na který se chce dotázat a aplikace se dotáže přes jím vybraný Name Server.

```
function updateLevelType(level)
 {
```

```
    for (var i = level+1; i < 9; i++) {
      $('l'+i).innerHTML = '';
    }
    $('l'+(level+1)).innerHTML="Please wait...";
    pleaseWait();
    new Ajax.Updater('l'+(level+1), 'ajax.php', {
      parameters: { domain: $F('domain'), ns: $F('sl'+level), level: level, type: $F('type') }, evalScripts: true
    });
  }¨
```

Také jsou zde funkce pro tlačítka „Compare“ a „Details“ (na úrovni root name serverů - nelze spustit porovnání ani detail, pokud není zadána doména v textovém poli, Obr. 5). Tyto funkce jsou rozděleny podle úrovně NS, od kterého uživatel bude chtít více informací.

```
function rootCompare()
  {
    if ($F('domain') == '') {
      alert('Please enter domain first');
      return; }
    var win = new Window({className: "alphacube",  width:400, height:400, zIndex: 100, resizable: true, title: "Root result compare", showEffect:Effect.BlindDown, hideEffect: Effect.SwitchOff, draggable:true, wiredDrag: true, modal: true})
    win.getContent().innerHTML= '<br />Compare  ' +
                        '<SELECT id="rootns_c1" name="rootns" onchange="updateRootCompare()"><? foreach(array('a','b','c','d','e','f','g','h','i','j','k','l','m') as $r) { ?><OPTION VALUE="<?= $r ?>.root-servers.net"><?= $r ?>.root-servers.net</OPTION><? } ?></SELECT>' +
                        ' With <SELECT id="rootns_c2" name="rootns" onchange="updateRootCompare()"><? foreach(array('a','b','c','d','e','f','g','h','i','j','k','l','m') as $r) { ?><OPTION VALUE="<?= $r ?>.root-servers.net"><?= $r ?>.root-servers.net</OPTION><? } ?></SELECT>' +
                        '<div id="rcr"></div>';
    win.showCenter(true);
  }
  function rootDetails()
  {
     if ($F('domain') == '') {
     alert('Please enter domain first');
     return;
    }
      var win = new Window({className: "alphacube", title: "Details",
              width:700, height:500,
              url:
"details.php?domain="+escape($F('domain'))+"&ns="+escape($F('rootns')),
```

```
                showEffectOptions: {duration:1.5}})
        win.showCenter(true);
    }
    function levelDetails(level)
    {
        var win = new Window({className: "alphacube", title: "Details",
                width:700, height:500,
                url:
"details.php?domain="+escape($F('domain'))+"&ns="+escape($F('sl'+level)),
                showEffectOptions: {duration:1.5}})
        win.showCenter(true);
    }
    function levelCompare(level)
    {
      $('c_r'+level).innerHTML = '';
      var win = new Window({className: "alphacube", title: "DNS result compare",
                    width:400, height:400,
                    showEffectOptions: {duration:1.5}})
      win.setContent('compare_l'+level);
      win.showCenter(true);
    }
    function compareLevel(level)
    {
      if (cStarted) { return; }
      cStarted = true;
      var type = "ANY";
      if (null != $('type')) {
        type = $F('type');
      }
      pleaseWait();
      new Ajax.Updater('c_r'+level, 'ajax_compare.php', {
        parameters: { domain: $F('domain'), ns1: $F('comp1_'+level), ns2:
$F('comp2_'+level), type: type }, evalScripts: true
      });
```


**Obr. 5 –** Nezadaná doména

### 3.2.3 Kód souboru – ajax.php

V následujícím souboru jsou metody, které se volají z index.php. Předají se do těchto metod hodnoty $domain (doména), $ns (NS, kterého se ptát), $type (typ DNS dotazu), $level (úroveň domény…ROOT, CZ…) s pomocí AJAXu.

```
function _r($i) {
  return isset($_REQUEST[$i]) ? escapeshellarg(trim($_REQUEST[$i])) : '';
}
$domain = _r('domain');
$ns = _r('ns');
$type = _r('type');
$level = intval($_REQUEST['level']);
if (($type == "'_'") || ($ns == "'_'")) {
  print "<SCRIPT>\nstopWait();\n</SCRIPT>";
  exit(0);
}
```

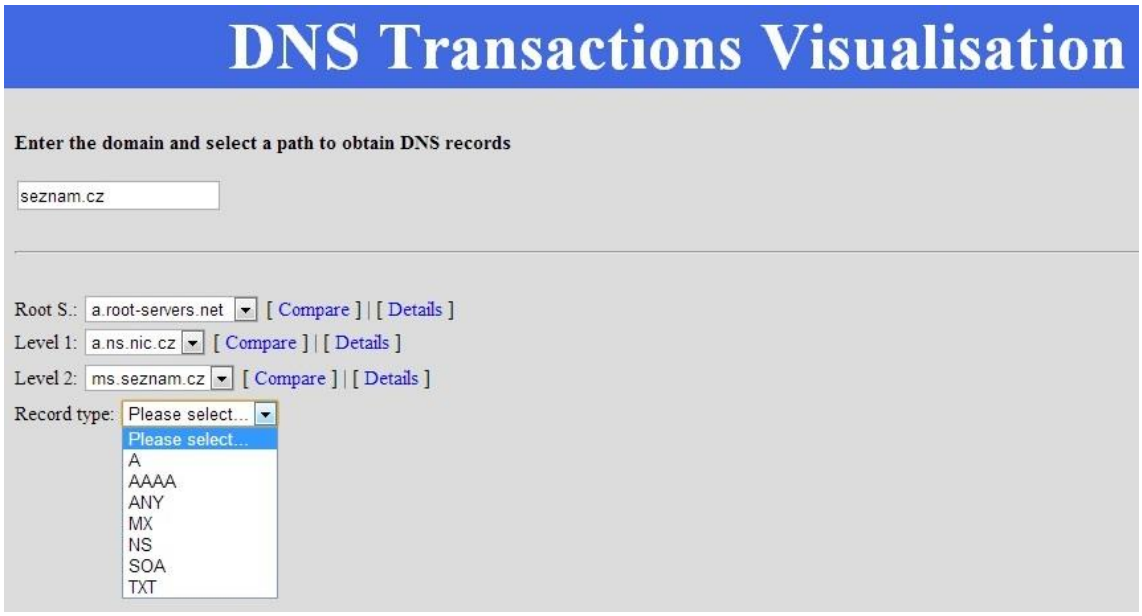
V následujícím části kódu se ptáme, zda už je vybraný typ DNS dotazu. Toto se volá přes metodu ***exec*** ta spustí externí program (funkci ***dig*** z BINDu), spustí se očištění o prázdné řádky, komentáře či další nevhodné řádky a poté se vypíše výsledek. Je zobrazeno na Obr. 3.

```
if (!empty($type)) {
   exec("dig -t $type $domain +noadditional +noauthority +nocomments @{$ns} | sort
| grep -v '^;' | grep -v '^\$'", $result);
   print "<div> </div><TABLE class=\"table\" border=\"1px\"
cellpadding=\"3px\"><TR
class=\"tableHead\"><TD>Domain</TD><TD>TTL</TD><TD>Class</TD><TD>
Type</TD><TD>Value</TD></TR>";
   foreach ($result as $line) {
     list($d,$ttl,$c,$t,$v) = preg_split('/\s+/', $line, 5);
     print "<TR class=\"tableRow\">";
     print "<TD>" . trim($d) . "</TD>";
     print "<TD>" . trim($ttl) . "</TD>";
     print "<TD>" . trim($c) . "</TD>";
     print "<TD>" . trim($t) . "</TD>";
     print "<TD>" . trim($v) . "</TD>";
     print "</TR>";
   }
   print "</TABLE>";
   print "<SCRIPT>\nstopWait();\n</SCRIPT>";
   exit(0);
 }
```

V následujícím části kódu se ptáme, zda už je vybraný NS konečný. Pokud ano,

pak se zjišťuje jaké typy DNS záznamů může poskytnout. Postupně se berou typy dns záznamů z pole $types (není problém přidat další typy záznamů), a na které dostaneme odpověď, tak ty se zařadí do výběru. Poté se vybrané typy porovnají s typy záznamů uvedenými v odpovědi na dotaz ANY a případně se o nové typy obsažené v ANY výběr rozšíří. Poté se vypíší v nabídce roletky (hned pod „Please Select…), která se zobrazí po 0,1 sekundě. Je lépe zobrazeno na Obr. 6.

```
  if (0 < trim(`host $domain $ns | grep 'has address' | wc -l`)) {
    print 'Record type: ';
    exec("dig -t ANY $domain @{$ns} | grep -v '^;' | egrep '\sIN\s' | awk '" . '{ print $4;}'
. "'", $rt);
    $types                                                                    =
array('A','AAAA','ANY','CNAME','DS','DNSKEY','MX','NS','NSEC','NSEC3','PTR','RRSI
G','SOA','SRV','TXT','AFSDB','DNAME','SIG');
    foreach ($rt as $t) {
      if (!in_array(trim($t), $types)) {
        $types[] = trim($t);
      }
    }
    sort($types);
    print          "<SELECT          id=\"type\"          name=\"type\"          id=\"sl{$level}\"
onchange=\"updateLevelType({$level})\">";
    print "<OPTION VALUE=\"_\">Please select...</OPTION>";
    foreach ($types as $t) {
      exec("dig -t $t $domain +noadditional +noauthority +nocomments @{$ns} | grep -
v '^;' | grep -v '^\$'", $r);
      if (!empty($r)) {
        print "<OPTION VALUE=\"$t\">$t</OPTION>";
      }
      unset($r);
    }
    print "</SELECT>";
    print                "<SCRIPT>\nstopWait();\nsetTimeout('updateLevelType({$level})',
100);</SCRIPT>\n";
    exit(0);
  }
```

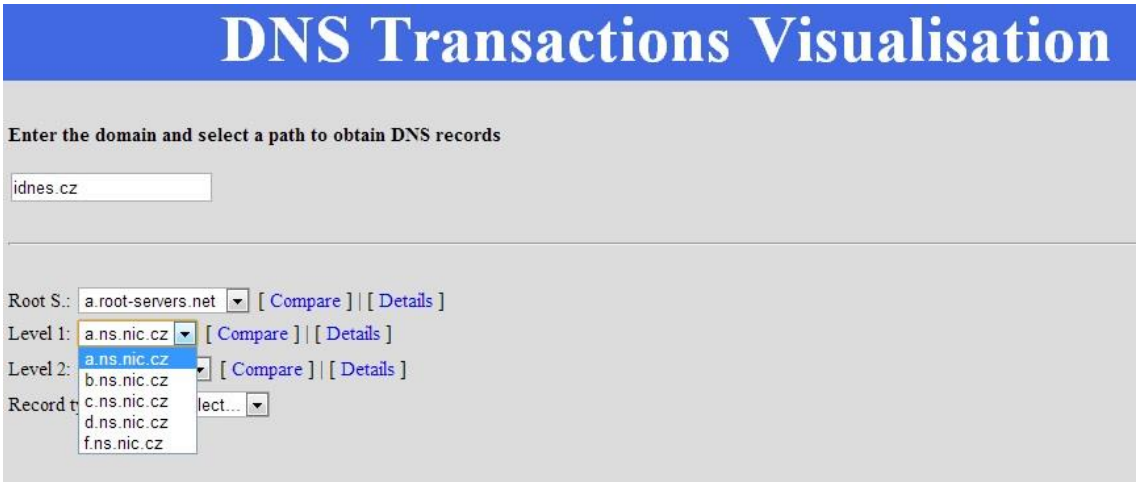



**Obr. 6 –** Výběr typu DNS záznamu

V další části kódu je přes dig (BIND) opět tázán NS na aktuální úrovni. Získáváme abecedně seřazený seznam NS další úrovně. Více je vidět na Obr.7.

A následně jsou přidaná další ošetření na „ořízntutí“ nepotřebných komentářů a také kontrola zda je správně vyplněna doména, pokud ne, tak je vypsáno: *"no valid data..."* viz Obr.8.

```
exec("dig {$domain} @{$ns} | egrep 'IN\tNS' | awk '{ print \$5; }' | sort ", $res);
$list = array();
foreach ($res as $line)
{
  list($line, $comment) = @explode(';', $line);
  if (empty($line)) continue;
   $list[] = trim($line, " \t\n\r\0\x0B." );
}
if (empty($list)) {
  print "no valid data...";
  exit(0);
}
$level++;
print "<SELECT name=\"sl{$level}\" id=\"sl{$level}\" onchange=\"updateLevel({$level})\" onclick=\"updateLevel({$level})\" >";
print "<OPTION VALUE=\"_\">Please select...</OPTION>";
foreach ($list as $name)
{
  print "<OPTION VALUE=\"{$name}\">$name</OPTION>\n";
}
print "</SELECT>";
```




**Obr. 7 –** Výběr NS

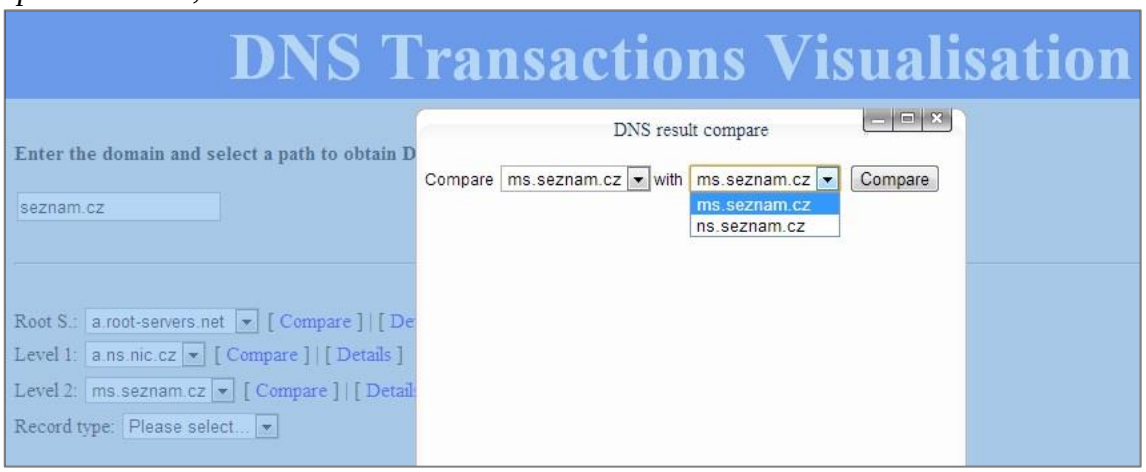

**Obr. 8 –** Špatně zadaná doména

### 3.2.4 Kódu souboru – ajax_compare.php

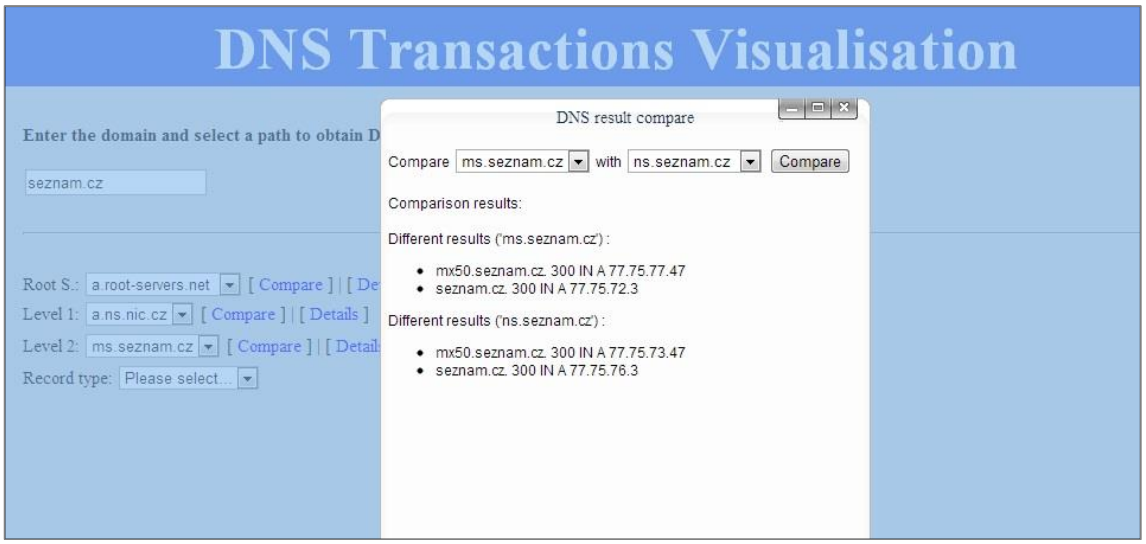
Tento soubor je použit pro porovnání odpovědí od konkrétních NS na dané úrovni a je volán pouze pokud uživatel klikne na tlačítko „Compare“. Předávají se sem údaje (hodnoty proměnných) potřebné pro porovnání NS. Srovnávají se zde odpovědi name serverů uživatelem vybrané úrovně, viz Obr. 9a. V případě NS poslední úrovně, který nemá vybrán konkrétní typ dotazu, se používá implicitně typ ANY. Poté je vypsán výsledek porovnání, zda jsou odpovědi identické, nebo či zda se nějakým způsobem liší a případně tyto rozdílné záznamy pro konkrétní NS vypíšeme, viz Obr 9.b.

```
 cStarted = false;
</SCRIPT><br />Comparison results: <br /><br />
<?
 function _r($i) {
  return isset($_REQUEST[$i]) ? escapeshellarg(trim($_REQUEST[$i])) : '';
 }
```

```
 $domain = _r('domain');
 $ns1 = _r('ns1');
 $ns2 = _r('ns2');
 $type = _r('type');
 if ($type == "'_'") { $type = "ANY"; }
 if (0 < trim(`host $domain $ns1 | grep 'has address' | wc -l`)) {
  if ($type == "ANY") {
   exec("dig -t {$type} {$domain} @{$ns1} | egrep 'IN\t' | sort ", $res1);
   exec("dig -t {$type} {$domain} @{$ns2} | egrep 'IN\t' | sort ", $res2);
  } else {
   exec("dig -t {$type} {$domain} @{$ns1} | egrep 'IN\t{$type}' | awk '{ print \$5; }' |
sort ", $res1);
   exec("dig -t {$type} {$domain} @{$ns2} | egrep 'IN\t{$type}' | awk '{ print \$5; }' |
sort ", $res2);
  }
 }
else {
  exec("dig {$domain} @{$ns1} | egrep 'IN\tNS' | awk '{ print \$5; }' | sort ", $res1);
  exec("dig {$domain} @{$ns2} | egrep 'IN\tNS' | awk '{ print \$5; }' | sort ", $res2);
 }
 if ($res1 == $res2) {
  print " No difference";
  exit();
 }
 $x1 = array_diff($res1, $res2);
 $x2 = array_diff($res2, $res1);
 print "Different results ($ns1) :<br /><ul>";
 foreach ($x1 as $d) {
  print "<LI>$d</LI>";
 }
 print "</ul>";
 print "Different results ($ns2) :<br /><ul>";
 foreach ($x2 as $d) {
  print "<LI>$d</LI>";
 }
 print "</ul>";
```




**Obr. 9a –** Výběr NS k porovnání

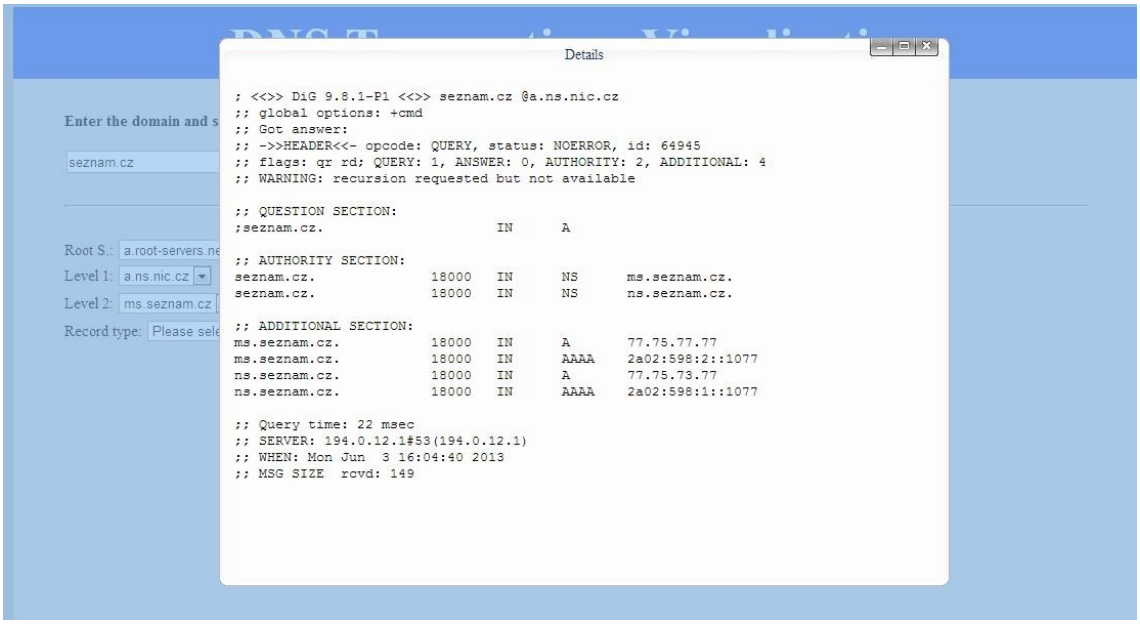



**Obr. 9b –** Výsledek porovnání odpovědí NS

### 3.2.5 Kód souboru – details.php

Tento soubor se zavolá, pouze pokud uživatel klikne na tlačítko „Details.“ Jsou zde přebírány hodnoty do proměných *$domain* a *$ns*, které jsou poté využity pro vytvoření dotazu a následný výpis odpovědi NS viz obr.10.

```
function _r($i) {
  return isset($_REQUEST[$i]) ? escapeshellarg(trim($_REQUEST[$i])) : '';
}
$domain = _r('domain');
$ns = _r('ns');
print "<PRE>\n";
passthru("dig {$domain} @{$ns}");
print "</PRE>\n";
```




**Obr. 10 –** Zobrazení kompletní odpovědi NS

### 3.2.6 Kód souboru - knihovny JS – příklad prototype.js

V následujícím souboru, knihovně JS prototype.js je mnoho metod, které jdou využít pro vytváření dalších aplikací s JS a také pro využití technologie AJAX. V aplikaci je prototype.js využita pro potřeby AJAXU metoda *Ajax.Updater,* která je volána v index.php. S pomocí této metody a metody $F se předávají proměné $domain (doména), $ns (NS, kterého se ptát) a další a díky nim aplikace neustále reaguje na změny, které provádí uživatel. Stejně tak využíváme i další metody z window.js a scriptaculous.js.

```
Ajax.Updater = Class.create(Ajax.Request, {
  initialize: function($super, container, url, options) {
    this.container = {
      success: (container.success || container),
      failure: (container.failure || (container.success ? null : container))
    };
    options = Object.clone(options);
    var onComplete = options.onComplete;
    options.onComplete = (function(response, json) {
      this.updateContent(response.responseText);
      if (Object.isFunction(onComplete)) onComplete(response, json);
    }).bind(this);
    $super(url, options);
  },
  updateContent: function(responseText) {
    var receiver = this.container[this.success() ? 'success' : 'failure'],
        options = this.options;
    if (!options.evalScripts) responseText = responseText.stripScripts();
    if (receiver = $(receiver)) {
      if (options.insertion) {
        if (Object.isString(options.insertion)) {
          var insertion = { }; insertion[options.insertion] = responseText;
          receiver.insert(insertion);
        }
        else options.insertion(receiver, responseText);
      }
      else receiver.update(responseText);
    }
  }
});
```

### 3.2.7 Ukázky kódu souboru – style.css

Jako poslední je ukázka souboru kaskádových stylů, které byly použity a s jejichž pomocí je tvořeno grafické rozhraní aplikace. Jsou tu klasická nastavení od fontů přes zarovnání šířky a řádkování atd. Vybral jsem pouze ty důležitější třídy, které se starají o korektní zobrazení celku a výstupu.

```
#hlavni {
        width: 1100px;
        margin: auto;
        text-align: left;
        background-color: #DCDCDC;
        height: 100%;
        position: relative;
}
#logo {
        color: #ffffff;
        background-color: #4169E1;
        height: 70px;
        text-align: center;
        font-size: 25px;
}
#obsah {
        padding: 15px 50px ;
        font-size: 15px;
}
.se  {
        text-align: left;
        line-height: 1.8;
        font-size: 15px;
}
.table {
  border-collapse: collapse;
        width: 100%;
        border: 2px;
  border-style: solid;
 }
.tableHead  {
        text-align: center;
        font-size: 18px;
  font-weight: bold;
```

## 3.3 Způsob využití aplikace

Na závěr je třeba ještě ověřit funkčnost celé aplikace a popsat způsob jejího využití. Nejdříve zadáme adresu n webové aplikace (localhost) a poté je třeba do formuláře vyplnit žádané údaje, tzn. doménu, kterou chceme vyhledat, více na Obr. 2.

Poté si uživatel vybere, kterého ROOT serveru se chce dotázat. V případě, že byla špatně zadaná doména, tak aplikace vypíše chybové hlášení: *"no valid data..."*, viz. Obr. 8. V případě, že byla správně zadaná doména, tak aplikace zobrazí: *"Please wait..."* viz Obr. 4 a vytvoří se struktura roletek dle úrovní NS (implicitně je vždy zvolen první NS ze seznamu) a také tlačítka „Compare a „Details.“

Ze zobrazených rozbalovacích roletek s NS, si může uživatel vybrat jakýkoliv dostupný NS, kterého by se chtěl „zeptat“ (více na Obr. 7) a ve chvíli kdy si nějaký zvolí, tak se opět začnou načítat všechny úrovně NS.

V momentě, kdy uživatelé dosáhou poslední úrovně NS, aplikace to zjistí, a tak je poté nabídnuta roletka se všemi dostupnými záznamy k zadaná doméně, více zobrazeno na Obr.6. Pokud si vybere uživatel záznam ANY, tak jsou vypsány všechny DNS zápisy k dané doméně. Jak již bylo napsáno výše, pokud by se objevil nový záznam, který není v připraveném seznamu a ani se nezobrazuje v dotazu ANY, tak je poté třeba daný typ přidat přímo do kódu aplikace.

Pokud uživatel v průběhu tohoto procesu změní textové pole s doménou (nepočítá se kliknutí či pohyb kurzoru v něm), začíná si volit hierarchii domén od začátku. Pokud zasáhne do stromu NS, začíná si volit od úrovně změněného NS.

# 4 Závěr

Tato aplikace byla vytvořena pro vizualizaci DNS transakcí, které probíhají mezi Name Servery. Díky této aplikaci si může uživatel zvolit vlastní cestu, to znamená jím vybrané servery, kterých se může zeptat na další cestu, až se nakonec dostane k jím žádaným DNS záznamům. Ze všech typů záznamů byla vybrána skupina, která pokrývá běžné potřeby uživatelů pro DNS dotazy a pokud by se objevil nový typ záznamu, který je viditelný v DNS záznamu ANY, tak jej aplikace automaticky přidá do výběru možných záznamů. Pokud by se tento záznam v ANY nezobrazoval (byl skrytý), poté je třeba ho přidat do kódu aplikace. Aplikace také umožňuje porovnání odpovědí z různých NS (tlačítko Compare) a také zobrazení kompletního dotazu (tlačítko Details).

Oproti původní verzi aplikace se jde již dotazovat na konkrétní Name Server a navíc většinu uživatelských akcí aplikace zpracovává v technologii AJAX. Takže zaznamenává změny, které uživatel provedl na aktuální stránce a po splnění kontrolních podmínek, je rovnou zpracovává na straně serveru, aniž by uživatel musel použít jakékoliv tlačítko. Co se týče grafického provedení, aplikace je zaměřená na jednoduchost a nijak to nepřehání s barevností. Řídím se heslem že: „Méně je někdy více.“ Ani povahou ani zaměřením nejsem návrhář UI/UX, takže co se této oblasti týče, tam určitě ještě nějaké rezervy budou. Při testování a vyvíjení této aplikace jsem poznal, že technologie Ajax je velmi „mocná“, ale i tak má své nevýhody, které jsem popsal v kapitole 2.2.3. Není proto vždy a na všechny typy webových stránek použitelný. Na druhou stranu je to velice elegantní a rychlé řešení a spíše převažují výhody než nevýhody.

.

# 5 Bibliografie


[1] DOSTÁLEK, L. a KABELOVÁ, A. *Velký průvodce protokoly TCP/IP a systémem DNS* Brno : CCP Books, a.s. 2005 ISBN 80-7226-675-6.

[2] LAVIN, P. *PHP objektově orientované* Praha : Grada Publishing, a.s. 2009 ISBN 978-80-247-2137-8.

[3] SATRAPA, P. *Web design* Havlíčkův Brod : Neokortex spol. s.r.o. 1997 ISBN 80-902230-1-X.

[4] MCCLURE, S. a SHAH, S. *Web Hacking: Útoky a Obrana* Praha : SoftPress s.r.o., 2003 ISBN 80-86497-53-4.

[5] LÁSLO, P. *Ajax – Úvod - Programujte.com.* Programujte.com. [Online] 26. 6 2008 Dostupný z WWW: http://programujte.com/clanek/2008062101-ajax-uvod/.

[6] SURÝ, O. *www.root.cz* [Online] 22. 7 2010, Dostupný z WWW: http://www.root.cz/clanky/hotovo-dns-je-podepsano/.

[7] ZAJÍC, P. *www.linuxsoft.cz*. [Online] 8. 11 2004. Dostupný z WWW: http://www.linuxsoft.cz/article.php?id_article=502.

[8] VRÁNA, J. *www.interval.cz*. [Online] 10. 8 2007. Dostupný z WWW: http://interval.cz/clanky/vicestrankovy-formular-v-php-a-javascriptu/.

[9] GRIMMICH, Š. *www.tvorba-webu.cz*. [Online] 2008. Dostupný z WWW: http://www.tvorba-webu.cz/php/pole.php.

[10] STEVENS, W. R. *TCP/IP Illustrated Vol. 1 - The Protocols*,. místo neznámé : Addison-Wesley, 1994.

[11] LAHVIČKA, J. *www.interval.cz.* [Online] 12. 6 2000. Dostupný z WWW: http://interval.cz/clanky/php-zakladni-informace/.

[12] KOSEK, JIŘÍ. *PHP a XML*. Praha : Grada Publishing, a.s., 2009. ISBN 978-80-247-1116-4.

[13] SOSINSKY, B. *Mistrovství - počítačové sítě.* Brno : Computer press, a.s., 2010. ISBN 978-80-251-3363-7.

[14] php.net. *PHP: Hypertext Preprocessor*. [Online] 2013. Dostupný z WWW: http://php.net/manual/en.

[15] Oracle *VM VirtualBox. VirtualBox.org*. [Online] Oracle. Dostupný z WWW: https://www.virtualbox.org/.

[16] *Prototype JavaScript framework: a foundation for ambitious web applications.* http://prototypejs.org/. [Online] Dostupný z WWW: http://prototypejs.org/.

[17] *Programujte.com - web o programování, webdesignu, počítačové grafice, databázích, elektrotechnice a designu*. programujte.com. [Online] Dostupný z WWW: http://programujte.com.