UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

2017 Michal Koreček

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Webserver

Michal Koreček

Bakalářská práce

2017

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Michal Koreček**

Osobní číslo: **I13154**

Studijní program: **B2646 Informační technologie**

Studijní obor: **Informační technologie**

Název tématu: **Webserver**

Zadávající katedra: **Katedra informačních technologií**

Zásady pro vypracování:

Práce je zaměřena na tvorbu vlastního webserveru, na kterém budou demonstrovány základní principy a techniky, které technologie webserveru využívá.

V teoretické části bude práce klást důraz na server, klasifikace serverů, webserver a jeho použití, varianty serverů a na klíčové protokoly pro webserver. Budou definovány nároky na webserver.
V praktické části bude implementován vlastní webserver, včetně všech klíčových technologií. Praktická část pak bude představovat a demonstrovat jednotlivé technologie a jejich využití, případně techniky pro jejich implementaci.

V případové studii musí být implementováno především: Využití konfiguračních souborů, více vláknové přístupy při obsluze portů, naslouchání portů, obsluha portu statickou webovou stránkou, korektní práce s alespoň pěti stavovými kódy protokolu http, podpora alespoň pěti webových stránek, podpora obrázků a multimédií.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

***GIULIO ZAMBON. Beginning JSP, JSF and Tomcat: Java web development. 2nd edition. s.l.: Springer, 2012. ISBN 9781430246237.**
***KOLEKTIV. PHP5, MySQL, Apache. Brno: Computer Press, 2006. Programujeme profesionálně. ISBN 80-251-1073-7.**
***PECINOVSKÝ, Rudolf. OOP: Naučte se myslet a programovat objektově. Brno: Computer Press, a.s., 2010. ISBN 978-80-251-2126-9.**

Vedoucí bakalářské práce: **Ing. Josef Brožek**
Katedra informačních technologií

Datum zadání bakalářské práce: **31. října 2016**
Termín odevzdání bakalářské práce: **12. května 2017**

Univerzita Pardubice Česká republika -8-

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

prof. Ing. Antonín Kavička, Ph.D.
pověřený vedením katedry

V Pardubicích dne 31. března 2017

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 5. 5. 2017

Michal Koreček

PODĚKOVÁNÍ

Rád bych poděkoval Ing. Josefu Brožkovi za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování bakalářské práce. Také bych chtěl poděkovat své rodině za morální podporu a pomoc, kterou mi poskytli nejen při psaní bakalářské práce, ale během celého studia.

ANOTACE

Cílem této bakalářské práce bylo vytvořit vlastní webserver. Bakalářská práce je rozdělena na část teoretickou a praktickou. Teoretická část se zabývá základními principy a technikami, které využívá webserver. Dále je popsána klasifikace serverů, varianty serverů, webserver a jeho využití, klíčové protokoly pro webserver. V praktické části je implementován vlastní webserver v programovacím jazyce Java, včetně všech klíčových technologií. Je využito konfiguračních souborů, vícevláknových přístupů při obsluze portů, obsluhy portu statickou webovou stránkou, korektní práce se stavovými kódy protokolu HTTP, podpory webových stránek, podpory obrázků a multimédií.

KLÍČOVÁ SLOVA

server, webserver, protokol, Java, konfigurace, soubor, vlákno, port, HTTP

TITLE

Webserver

ANNOTATION

The aim of this bachelor thesis was to create a webserver. The bachelor thesis is split into a theoretical and a practical part. The theoretical part deals with the basic principles and techniques that are used by webserver. There are also described classification of servers, variants of servers, webserver and its applications, key protocols for webserver. In the practical part there is implemented author's own webserver in Java programming language including all key technologies. There are used configuration files, multi-threaded approach in operating the ports, the port operator static web page, fair labor with status codes HTTP, support web sites, support for images and multimedia.

KEYWORDS

server, webserver, protocol, Java, configuration, file, threat, port, HTTP

OBSAH

1 Úvod 13

2 Server 14

2.1 Klasifikace serverů 14

2.2 Hardware serveru 15

2.3 Software serveru 16

2.3.1 Webhosting 17

2.3.2 Virtuální privátní server 18

2.3.3 Dedikovaný server 18

2.3.4 Cloud hosting 19

2.4 Poskytování služeb 20

2.4.1 Klient-server 20

2.4.2 Peer-to-peer 21

2.5 Varianty serverů 22

3 Webový server 28

3.1 Princip webserveru 28

3.2 Využití webserveru 29

3.3 Obecné vlastnosti webových serverů 29

3.3.1 Protokoly pro webserver 30

3.3.2 Stavové kódy protokolu HTTP 35

4 Vlastní webserver 36

4.1 Struktura 36

4.2 Klíčové funkce 37

4.2.1 Hlavní okno 37

4.2.2 Logování serveru 38

4.2.3 Konfigurační soubory 39

4.2.4 Vlákna 42

4.2.5 Obsazení portů 44
4.2.6 Předávání obsahu 45
4.2.7 Stavové kódy 48
4.3 Testování serveru 49
5 Závěr 51
6 Použitá literatura 52
7 Přílohy 57

SEZNAM ILUSTRACÍ A TABULEK

Obrázek 1 - Rozdělení serveru....14
Obrázek 2 - Virtuální server....18
Obrázek 3 - Komunikace architektury klient-server pomocí protokolu HTTP....21
Obrázek 4 - Ukázka sítě pomocí modelu klient-server a peer-to-peer....22
Obrázek 5 - Databázový systém....23
Obrázek 6 - Tiskový server TP-LINK TL-PS110U....24
Obrázek 7 - Princip mapového serveru....27
Obrázek 8 - Architektura TCP/IP....30
Obrázek 9 - Aktivní FTP spojení....31
Obrázek 10 - Pasivní FTP spojení....32
Obrázek 11 - Ukázka hlaviček a těla požadavku klienta a odpovědi serveru....33
Obrázek 12 - Princip elektronické pošty....35
Obrázek 13 - GUI webového serveru....36
Obrázek 14 - Komunikace klient-server pomocí portů....45
Obrázek 15 - Menu softwaru LOIC, nastaveno pro 20 vláken webového serveru a 50 vláken softwaru LOIC....50

Tabulka 1 - Příklady výdrže serverů s pomocí zálohy napájení UPS....16
Tabulka 2 - Počet nevyřízených požadavků při 20 vláknech webového serveru a *n* vláknech softwaru LOIC....49
Tabulka 3 - Počet nevyřízených požadavků při *n* vláknech webového serveru a 50 vláknech softwaru LOIC....50

SEZNAM ZKRATEK A ZNAČEK

AJAX	Asynchronous JavaScript and XML
AMD	Advanced Micro Devices
CD-ROM	Compact Disc Read-Only Memory
CR	Carriage Return
CSS	Cascading Style Sheets
DCP	Digital Cinema Package
DLNA	Digital Living Network Alliance
DNS	Domain Name System
DVI	Digital Visual Interface
FOP	Fyzická operační paměť
FTP/TFTP	File Transfer Protocol/Trivial FTP
GIS	Geografické informační systémy
GUI	Graphical User Interface
GWS	Google Web Server
HDMI	High-Definition Multi-media
HP	Hewlett-Packard
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
IBM	International Business Machines
IIS	Internet Information Services
IMAP	Inernet Message Access Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IT	Informační technologie
J2EE	Java Platform, Enterprise Edition
LaaS	Logging as a Service
LAN	Local Area Network
LF	Line Feed
LOIC	Low Orbital Ion Cannon
MAN	Metropolitan Area Network
MTA	Mail Transfer Agent
NAS	Network Attached Storage
NAT	Network Address Translation
NFS	Network File System
NT	New Technology
OS	Operační systém
P2P	Peer-to-Peer
PaaS	Platform as a Service

PAN	Personal Area Network
PHP	Hypertext Preprocessor
POP	Post Office Protocol
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
RFC	Request For Comments
RSS	Rich Site Summary
SaaS	Software as a service
SCSI	Small Computer Systém Interface
SMB	Server Message Block
SMTP	Simple Mail Transfer Protocol
SPDY	Pronounced Speedy
SŘBD	Systém řízení báze dat
SSH	Secure Shell
TB	Terabyte
Tcl	Tool Command Language
TCP/IP	Transmission Control Protocol/Internet Protocol
TELNET	Telecommunication Network
TLS	Protokol Transport Security
UDP	User Datagram Protocol
UPS	Uninterruptible Power Supply
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
VA	Voltampér (zdánlivý výkon)
VGA	Video Graphics Array
VPS	Virtuální privátní server
W	Watt (činný výkon)
WAN	Wide Area Network
WWW	World Wide Web
XML	Extensible Markup Language

1 ÚVOD

Cílem bakalářské práce je vysvětlit, popřípadě doplnit informace týkající se serverů, obzvláště pak webového serveru. Tato bakalářská práce může dobře posloužit jako výukový materiál.

Práce je rozdělena na dvě části. V první části je popsáno, co je to server, možnosti jeho využití, nároky na hardware a možnosti softwaru. Jsou definovány jednotlivé druhy serverů s důrazem na webový server. V práci jsou také vysvětleny společné vlastnosti webových serverů.

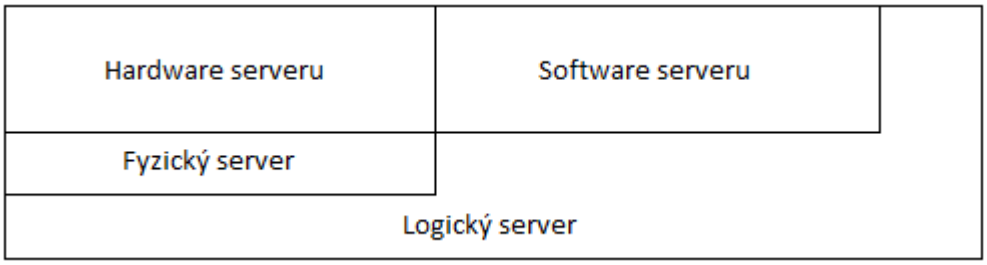
V praktické části je vyhotoven vlastní webový server implementovaný ve vývojovém prostředí NetBeans pomocí jazyku Java. Tato aplikace (webový server) dobře demonstruje funkčnosti webového serveru. Uživatel (student) se prakticky seznámí s principem webového serveru, se spuštěním serveru a s jeho inicializací pomocí konfiguračního souboru. Následně mu budou představeny výpisy, které slouží k tomu, co přesně webový server udělal, a také požadavky, které přijdou na server, a následně serverovou odpovědí zpět klientovi (webový prohlížeč). Je předvedeno vyřizování požadavků pomocí vláken, což zaručuje větší efektivitu vyřizování požadavků. Také je demonstrována práce s několika různými webovými stránkami, které má server uloženy na svém úložišti a následně je poskytuje klientovi. V neposlední řadě jsou implementovány stavové kódy protokolu HTTP a chybové stránky (error pages), které se zobrazí při příslušné chybě.

Po prostudování této práce by měl mít čtenář základní znalosti o principu a fungování serverů, obzvláště pak webového serveru. Na těchto informacích může stavět, implementovat a rozšiřovat další technologie.

2 SERVER

Server se architekturou svého hardwaru neliší od běžného domácího počítače, většinou však klade důraz na vyšší výpočetní výkon a upozaďuje výkon grafický. Server poskytuje služby, které jiný počítač může využívat. Existují základní rozlišení, kterými jsou architektura klient-server a peer-to-peer, které budou vysvětleny dále. Služby serveru mohou být nabízeny individuálně (lokálně), tyto služby jsou poskytovány jednomu počítači. Těmito službami jsou například správa tiskárny, automatických aktualizací a další. Servery se nacházejí nebo mohou nacházet ve všech druzích sítí (PAN, LAN, MAN, WAN). V místní síti (LAN) může mít skupina počítačů sdílený přístup například k diskům, tiskárnám, server může ověřovat jména a hesla uživatelů, může spravovat databáze, vyřizovat e-maily a další. U větších sítí, jako je internet, poskytují servery webové stránky a další služby, jako je například DNS[1]. Různé služby serverů budou popsány u jednotlivých druhů serverů. [8]

Server se rozděluje na fyzickou část (hardware) a logickou část. Logická část vzniká nakonfigurováním softwaru na fyzickou část serveru, viz obrázek 1. Jsou-li nakonfigurované softwary alespoň dvou serverů na jedné fyzické části, potom se tento server nazývá virtuální server (VS).



Obrázek 1 - Rozdělení serveru[2]

V kapitole je čerpáno ze zdrojů: [1]-[9], [11], [13], [14], [20], [22], [24], [26], [27], [29], [31], [32], [34]-[36], [39], [40], [42], [43], [51].

2.1 Klasifikace serverů

Servery je možné klasifikovat z několika základních hledisek, kterými jsou:

- fyzická architektura serveru,
    - 1:1 (jeden fyzický stroj s jedním logickým serverem)
    - 1:N (jeden fyzický stroj s N logickými servery)

[1] Hierarchický systém doménových jmen, který převádí doménová jména a IP adresy uzlů sítě
[2] Zdroj: autor

    - N:1 (N fyzických strojů seskládajíce se v jeden logický server)
    - N:M (N fyzických strojů s M logickými servery)
- cílového využití serveru,
    - Klient – server (např. webserver)
    - Klient – klient (např. SMTP servery)
- počet síťových služeb poskytovaných serverem.
    - Server určený pro jednu službu
    - Server poskytující více služeb

2.2 Hardware serveru

Výběr komponent je důležitý pro celkový chod serveru, kvalitní výkon a výdrž. Klíčovým parametrem při výběru hardwaru je určení serveru. Standardně je u serveru kladen důraz na výkon procesorů, operační paměti a diskových polí. Grafické karty jsou zpravidla integrované, protože k serveru není ve většině případů připojen monitor, tím pádem odpadá vykreslování složitých 3D prvků a animací. [42] Hlavní požadavek na server je vysoká životnost komponent, je zapotřebí vysoká úroveň síťové komunikace, pokud se serverem bude komunikovat více klientů. Měl by obsahovat vhodný diskový systém, důraz je kladen na rychlost disků spolu s kapacitou. Procesor by měl tedy zvládat zpracovávat instrukce v požadovaném čase. Hardware serveru se skládá z několika komponent, kterými jsou mikroprocesor, operační paměť, pevné disky, disková pole, CD-ROM, zásuvné karty, disketová mechanika, pásková mechanika, záložní zdroj UPS a hardwarové zabezpečení serveru vůči fyzickému napadení.

U mikroprocesoru serveru je důležitý vztah mezi cenou a výkonem. Důležitými atributy jsou sběrnice, frekvence procesoru, počet procesorů. Největší hráči na poli mikroprocesorů jsou Intel a AMD, které pro servery vytváří speciální editace procesorů. Některé firmy živící se prodejem serverů, si samy vyvíjí procesory většinou cílené na uživatele pro posixových operačních systémů.

Disková pole jsou připojena pomocí řadičů SCSI s různými možnostmi zapojení RAID. Při zapojení externích disků pomocí optických vláken je využíváno tzv. hot swapping[3] a hot plugging[4]. Obě tyto možnosti pracují za běhu systému, není zapotřebí disk odpojit od zdroje napájení.

[3] Systémové části (pevný disk, síťové karty, RAM, procesory, základní desky a další) jsou nahrazeny jinými
[4] Do systému je za běhu přidaná nová součást (pevný disk, myš, klávesnice a další)

Důležitou a nezbytnou součástí je pásková mechanika. Pásková mechanika se vybírá podle počtu dat v časovém intervalu. Je vyžadováno časté zálohování, například u herních serverů. Po pádu serveru není žádoucí znovu opakovat, co bylo uděláno před patnácti minutami. Na páskové mechanice by se nemělo šetřit, je to jedna z nejdůležitějších komponent. Cena páskové mechaniky šplhá řádově do tisíců korun.

Záložní zdroj UPS slouží ke krátkodobému nahrazení odpojeného primárního napájení. Vždy je nutné počítat s výkonem všech součástí serveru. Činný výkon (W) je vyjadřován jako výkon serveru. Zdánlivý výkon (VA) je vyjadřován jako výkon UPS. Veškerý výkon (VA) bude serverem vyzářen v podobě tepla, je proto nutné kvalitní chlazení.

Tabulka 1 - Příklady výdrže serverů s pomocí zálohy napájení UPS

VA – výkon UPS	W – výkon serveru	Výdrž chodu serveru (min.)
620	390	6 – 14
700	450	5 – 17
1000	650	6 – 18
1400	950	7 – 18
2200	1600	8 – 24
3000	2250	5 – 15

Pro server je zapotřebí zajistit nepřetržitý a kvalitní přísun energie. UPS vydrží pár minut, maximálně pár hodin. Po vybití UPS se následně vypne i server.

Operační systém Linux klade nejmenší nároky na procesor a FOP, také nepotřebuje tolik prostoru na disku. NT server klade průměrné požadavky na procesor, FOP a celkem vysoké požadavky na diskový prostor. Novell klade vysoké požadavky na FOP, průměrné nároky na procesor a nízké požadavky na diskový oddíl. [51]

2.3 Software serveru

Dnešní operační systémy jako Windows, Linux a další mohou sloužit jako klasické osobní počítače stejně tak jako server. U osobního počítače je kladen důraz na interakci. Po osobním počítači je požadovaná dobrá odezva vůči uživateli, u serveru je požadována dobrá stabilita a dosažení vysokých výkonů. Z hlediska obchodu je možné rozdělit operační systémy na dva základní typy. Prvním typem jsou počítače pro domácnost (např. Windows Vista Home Basic, Windows 7 Home Basic, Windows 8 Home Basic, Windows Vista Home Premium, Windows 10 Home a další), nebo firemní osobní počítače (Windows 7 Professional, Windows 8 Professional, Windows 10 a další). Operační systém pro domácnost je opět o něco levnější než

pro firemní účely. Domácím operačním systémům chybí některé komponenty. Například Windows Vista Home Basic se nemůže připojit do firemní sítě, nemá podporu vzdálené plochy apod. Druhým typem jsou operační systémy pro server, např. Windows Server. Tyto edice poskytují navíc síťové služby. Tyto operační systémy jsou logicky dražší, protože se používají v komerční sféře. Linux, Solaris, FreeBSD jsou open-source software. Uživatel těchto operačních systémů může měnit konfiguraci systému, přidávat aplikace. Nicméně ani tato možnost nezabránila k rozdělení osobních počítačů a serverů. [18] Na tomto open-source staví například americká firma Red Hat svoji linuxovou distribuci Red Hat Enterprise Linux. Za tuto distribuci se platí pouze podpora, servis, přístup k webové službě Red Hat Network. [39]

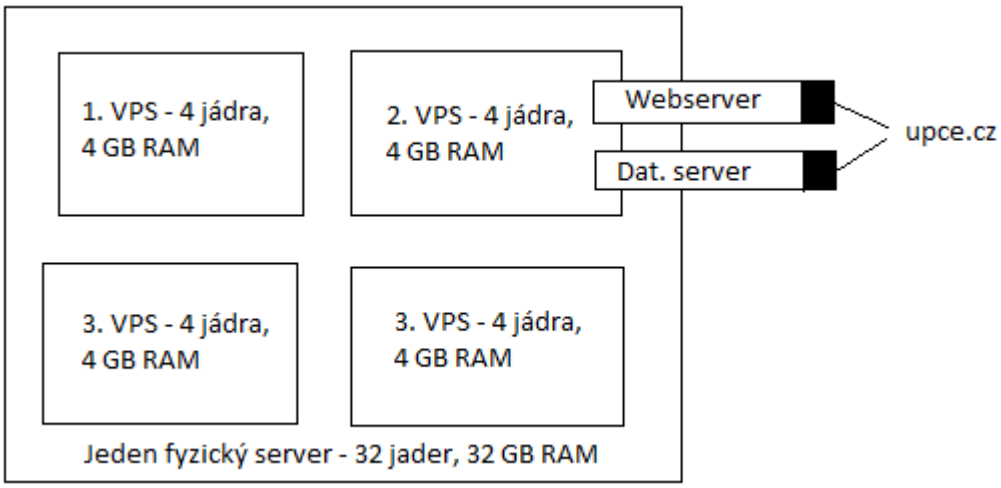
Servery se nacházejí na různých místech, například v podnicích, domácnostech, školách a v dalších institucích. Servery jsou často umístěny v tzv. hostingových centrech, kde je nad nimi ustavičný dohled a kde jsou kvalitně zajištěny základní principy chodu serveru. Pokud je zapotřebí ušetřit a zbytečně neplýtvat svými prostředky, je možné si za poplatek pořídit některou serverovou službu. Toto řešení se nazývá webhosting. Dalším způsobem je stále populárnější využívání virtuálního stroje, popř. využití dedikovaného serveru.

2.3.1 Webhosting

Pokud vlastník webových stránek chce mít stránky přístupné pomocí internetu, potřebuje umístit tyto stránky na server, který mu to umožní. Pokud vlastník server nemá, nebo se o něj nechce starat, popřípadě nemá hlubší znalosti v tomto směru, má možnost si nechat provozovat své webové stránky na serveru pronajímatele. Této možnosti se říká webhosting. Nejjednodušším způsobem uložení webových stránek na server umožňující webhosting je ten, že si zákazník pošle pomocí FTP protokolu svůj web na server. Dále naplní databázi, vytvoří si e-mailovou schránku pro svoji doménu a poté už nemusí nic řešit. O údržbu, nastavení, funkčnost webových, databázových, e-mailových služeb serveru se stará poskytovatel. Výhoda webhostingu je tedy ta, že osoba vlastnící web se nemusí starat o běh webových stránek, nemusí si kupovat server a nemusí zbytečně platit za celou spotřebu a údržbu celého serveru. Webhosting je služba sdílená, takže se na jednom serveru může nacházet několik stovek až tisíců webových stránek různých zákazníků. Při chybě či přetížení některých webů může dojít ke zpomalení, poškození, dokonce i nefunkčnosti jiných webů na serveru. Další nevýhoda je v tom, že někteří poskytovatelé nabízí jen určité technologie, jako je například podpora PHP aplikace, databáze MySQL. Pokud má osoba web propojený s jinou databází, například od Oracle, a nechce zvolit dražší řešení, musí svoji databázi předělat do MySQL. [47]

2.3.2 Virtuální privátní server

Virtuální server (VPS) je pro zákazníky, kterým už nedostačuje nabídka webhostingů. Buď potřebují vyšší výkon nebo další speciální nastavení, které sdílený webhosting neumožňuje, zároveň zákazník nechce měsíčně platit vyšší poplatek za dedikovaný server, kde by byl virtuálním vlastníkem pouze on sám. Zákazník už musí mít lepší vědomosti ohledně spravování serveru (zabezpečení, konfigurace apod.). Od poskytovatele dostane pouze prázdný server s instalačními balíčky a s požadovaným operačním systémem, případně dalšími základními balíčky například s Apache, MySQL, PHP, mailovým serverem a dalšími službami. Virtuální server není vhodný pro aplikace, které potřebují více než 4 GB RAM, protože je nadále sdílený s ostatními zákazníky, kterým by mohla tato aplikace zpomalit či zastavit jejich virtuální server. Virtuální privátní server je komerční pojem pro situaci, kdy si zákazník pronajme virtuální server. [47]



Obrázek 2 - Virtuální server[5]

2.3.3 Dedikovaný server

Dedikovaný server zpravidla nepotřebuje přímý přístup uživatelů. Dedikovaný server se od virtuálního serveru liší tím, že zákazník má celý server pro sebe. Může si požádat o přípravu serverového hardwaru na míru. Dedikovaný server funguje tak, že existuje osoba, nebo společnost vlastnící serverovnu. Serverovnou se rozumí místnost, kde je uložený server s dobrým vybavením, jako je rack (skříň, ve které je umístěn server), záloha napájení (UPS), klimatizace a další vlastnosti k dobrému chodu serveru. Vlastník serveru se stará pouze o chod serveru po hardwarové stránce, jako je výměna disků, porucha pamětí či procesorů. Tato osoba

[5] Zdroj: autor

pronajímá zákazníkovi dedikovaný server. Zákazník se stará o jeho software počínaje instalací operačního systému, nastavením, zálohováním, zabezpečením a jinými operacemi po softwarové stránce. [6] V praxi jsou dedikované servery používány náročnějšími zákazníky, kteří chtějí a potřebují celý výkon serveru pro své účely. [47]

Nededikovaný server

Nededikovaný server slouží uživateli zároveň jako počítač. Tím pádem se uživatel musí starat i o hardware.

2.3.4 Cloud hosting

Je to poměrně mladá metoda, která využívá online virtuálních serverů. Tato metoda pracuje na modelu nazývaný cloud computing. Tento model spojuje několik serverů za účelem sdílení pracovní zátěže nějaké úlohy. Je-li vyžadováno spuštění složité úlohy, cloud může rozdělit úlohu na jednotlivé procesy do několika menších počítačů, díky tomuto řešení není třeba jednoho výkonového stroje. Umožňuje přes cloud („Internet“) služby, jako je poskytování serverů, softwaru, podpora úložiště a zálohování dat, databáze, vytváření nových aplikací, streamování videa a zvuk a také mnoho dalšího. Za služby cloudu se neplatí klasickým měsíčním pronájmem jako u virtuálního serveru. U služeb cloudu je placeno pouze za služby, které jsou opravdu skutečně používané. Služby cloudu šetří náklady, peníze, produktivitu, výkon a spolehlivost. Základní cloudové služby se dají rozdělit do tří základních druhů, kterými jsou: infrastruktura jako služba (LaaS), platforma jako služba (PaaS), software jako služba (SaaS). LaaS je určeno pro pronájem serverů a virtuálních počítačů, sítě, úložiště, zálohu dat. PaaS poskytuje testování, vývoj, správu a dodávání softwarových aplikací, rychlé vytváření webů a mobilních aplikací. Dále zahrnuje úložiště, databáze, sítě pro potřebný vývoj. Poskytovatelé služby SaaS hostují a dodávají softwarové aplikace přes internet, většinou se tak děje na úkor vyžádání. Poskytovatelé zajišťují veškerou údržbu, upgrady, opravy a zabezpečení. Tyto veškeré služby jsou většinou poskytovány na základě předplatného. Nasazené cloudy se dělí na veřejný, privátní, nebo hybridní cloud. U veřejného cloudu dodávají dodavatelé výpočetní prostředky jako servery, úložiště pomocí internetu. U veřejných cloudů spravuje veškerou infrastrukturu poskytovatel. Uživatel pouze spravuje svůj účet pomocí webu. Prostředky privátních serverů používá pouze jediná organizace. Cloud server může být umístěn v centru firmy, je také možné pronajímat privátní cloud server jiné společnosti. Hybridní cloud kombinuje veřejný a privátní cloud za účelem větším společnostem dávat větší flexibilitu a rozsáhlejší možnost nasazení. [5]

2.4 Poskytování služeb

Server zprostředkovává základní síťové služby, kterými jsou:

- Server může sdílet technická zařízení, jako jsou disky, tiskárny.
- Poskytuje přístup na ostatní počítače pomocí utility rlogin nebo protokolu Telnet.
- Umožňuje přenos zpráv (e-mail, news - RSS[6]).
- Server poskytuje přenos souborů pomocí WWW, protokolů NFS, FTP a další.
- Je nutno zmínit síťovou službu, kterou je správa sítě pomocí DNS. [29]

Server poskytuje služby pomocí architektury (modelu) klient-server a zřídka je použit v architektuře peer-to-peer. Pomocí architektury klient-server pracují operační systémy NT server, Linux server, Novell a další. [51] Pokud je nežádoucí provozovat vlastní server, je možné využívat služby, kterými jsou webhosting, virtuální a dedikovaný server, v neposlední řadě také cloud hosting.

2.4.1 Klient-server

Klient-server je základní síťová architektura (model), která odděluje klienta od serveru. Klientem se rozumí nějaká grafická aplikace, jako je například webový prohlížeč. Na druhé straně program zvaný server vyřizuje požadavky klienta. Odeslání požadavku klienta se nazývá request a odezva serveru klientovi se nazývá response.

Příklad: Uživatel má k dispozici webový prohlížeč (klient). Uživatel by rád věděl svůj zůstatek na bankovním účtu. Zadá webovou stránku banky. Pomocí požadavků a odpovědí mezi klientem a serverem se provede jeho autentizace. Dále server zašle dotaz databázovému programu, který zašle požadavek databázovému serveru. Databázový server vrátí hodnotu databázovému programu, který pak vrátí hodnotu serveru. Server následně vrací odpověď klientovi o zůstatku na bankovním účtu.

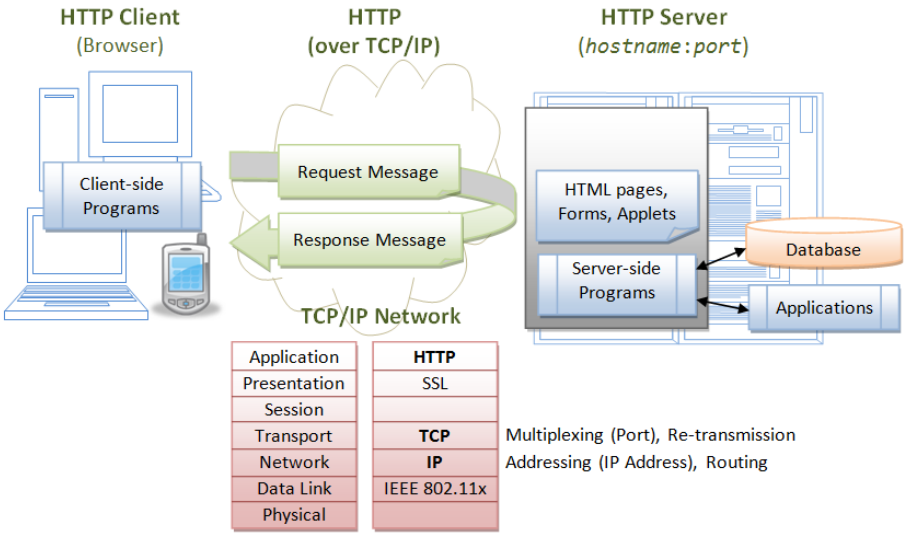
Tento model je z většiny používán obchodními a firemními aplikacemi. Model využívá protokoly, jako jsou HTTP pro webový server, SMB pro sdílení disků a tiskáren ve Windows, SMTP, DNS, Telnet a další.

Výhoda modelu klient-server je v centralizaci místa poskytování služeb a možnosti optimalizace výkonu. K centralizovanému místu je snadný přístup. Díky tomu je snadnější uskutečnit údržbu. Klienti si nevšimnou například přemístění, opravy a modernizace serveru.

[6] Rodina XML formátů určených pro čtení novinek na webových stránkách

Data jsou ukládaná na servery, kde by měla být bezpečněji uložena než v případě struktury peer-to-peer. [27]

Tento model má ovšem i nevýhody a ty nastávají v okamžiku přetížení a výpadku sítě. U přetížené sítě server nebude stíhat vyřizovat větší počet požadavků najednou nad určitý limit. Tento model není natolik robustní, jako je architektura peer-to-peer. Pokud dojde k výpadku serveru (klient-server), žádosti klientů nemohou být splněny. U peer-to-peer se šířka propustnosti zvětšuje s počtem klientů.



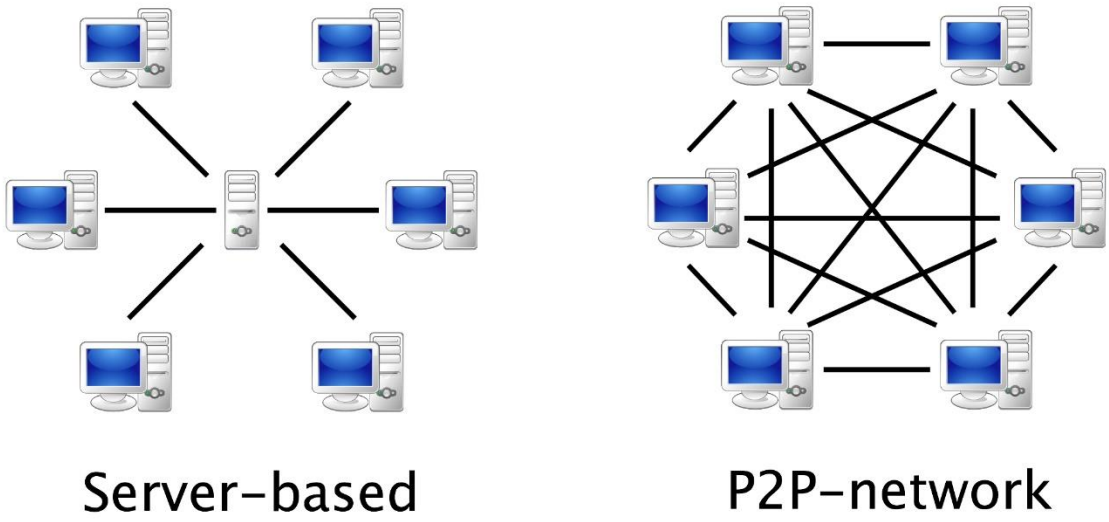
Obrázek 3 - Komunikace architektury klient-server pomocí protokolu HTTP[7]

2.4.2 Peer-to-peer

Model peer-to-peer (rovný s rovným) je další ze způsobů poskytování služeb. Tento model se znatelně liší od modelu klient-server. Server, zdroj dat a centrum informací odpadá. Pokud server existuje, je to pouze kvůli zjednodušení komunikace. Klienti jsou propojeni a sdílejí mezi sebou informace. Architektura peer-to-peer se používá pro datové sítě a souborové servery. U klient-server platí to, že čím více klientů se připojuje na server, tím pomaleji dochází k přenosu dat. U modelu peer-to-peer je to naopak. Čím více klientů je připojeno, tím dochází k větší rychlosti sítě. Například peer-to-peer sítí lze uvést torrenty, mailové servery, clustery pro aktualizace dat (např. Microsoft, který Windows 10 umožňuje aktualizovat prostřednictvím seedu P2P sítě). Lidé mezi sebou sdílejí různá data, čímž se dostáváme k výhodám a nevýhodám peer-to-peer modelu. Výhoda peer-to-peer sítí je ve stabilitě spojení. Pokud vypadne jeden uzel (klient), dají se většinou data pořídit jinde. Nevýhoda této architektury je ta, že uživatel může

[7] Zdroj: [24]

vlastnit neoprávněný obsah dat, který dál rozšiřuje. Tím se uživatelé dostávají mimo hranice zákona. Internet není anonymní a uživatelé se dají dohledat. Může následovat odpojení od internetu, vysoká pokuta, v krajních případech odnětí svobody. [36]



Obrázek 4 - Ukázka sítě pomocí modelu klient-server a peer-to-peer[8]

2.5 Varianty serverů

Servery mohou poskytovat různé typy služeb. Většina druhů serverů je popsána v následujících podkapitolách, přičemž za nejznámější a nejpoužívanější servery jsou pokládány webové a aplikační servery.

Doménový server

Počítače rozumí IPv4 a IPv6 adresám. Nicméně pro člověka jsou tyto řetězce čísel těžko zapamatovatelné. Proto DNS server k určité IP adrese přiřadí požadovanou doménu, jako je například *www.google.com*. Pokud je zadána do prohlížeče zmiňovaná adresa, počítač pošle požadavek nejprve na jeden z kořenových serverů, jestliže zná požadovanou doménu. Server odpoví, že nezná, ale ví, na který další server má zaslat požadavek o nalezení webové stránky. Tímto rekurzivním dotazováním počítač dostane kýženou webovou stránku. Nicméně pokud by se server dotazoval vždy kompletně, docházelo by k velkým časovým ztrátám. Proto se počítač dotazuje tzv. vyrovnávacího serveru, na který se ukládá záznam, kde je doména umístěna, díky uživateli, který se na zmiňovanou adresu dotazoval dříve. [20]

[8] Zdroj: [1]

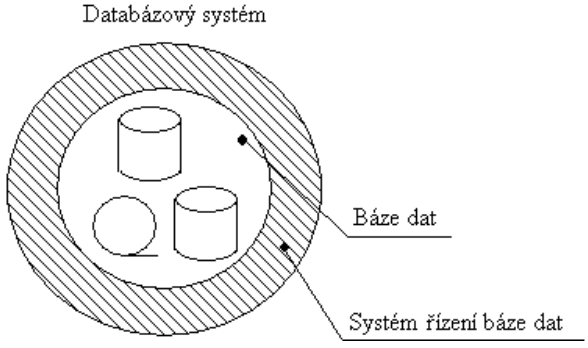
Souborový server

Souborový server neboli File Server slouží jako úložiště dat. V malých a domácích sítích stačí použít standartních OS balíčků pro data. Nicméně správci sítí často volí známé a moderní softwarové balíky, jako jsou Windows Server, Mac OS Server, UNIX a pestrou platformu Linux serverů.

Jelikož souborový server zastává funkci úložiště dat, v dnešní době je běžná celková velikost jednoho, nebo více disků s kapacitou nad 1 TB. Z toho plyne, že u souborových serverů je kladen důraz na stabilitu a rychlost disků. Dále mohou být přidávány další disky, které jsou připojeny přes sériovou sběrnici FireWire nebo USB. Disky se seskupují do RAID polí, je tím docíleno lepších výkonů či zálohy. Je-li server správně nakonfigurován a je zveřejněn na síti, lze použít techniku „namapování“ disků. To znamená mít k dispozici složku z disku serveru, se kterou se lokálně pracuje. Tato složka se zobrazí jako disk v počítači s určitým názvem disku. Při správné konfiguraci a oprávnění může uživatel číst, vytvářet, měnit data uložená na souborovém serveru. Tímto vzniká lepší efektivita při hromadné práci lidí. [13]

Databázový server

Databázový server slouží jako úložiště databází. Na tomto serveru jsou uloženy databázové soubory a databázový systém. Tento systém se skládá z bází dat a z SŘBD, které tvoří rozhraní mezi aplikačními programy a uloženými daty. [43] Tento systém je permanentně k dispozici. Tímto systémem je například MySQL, Oracle, Microsoft Access a další. Tyto systémy zpracovávají a kontrolují veškerou manipulaci s daty. Kvůli bezpečnosti je žádoucí umístit databázový server na samostatně běžícím počítači, který běží pro účely databáze. K tomuto počítači je kladen důraz na fyzické zabezpečení, proto jsou databázové servery umisťovány do profesionálních serveroven. [11]



Obrázek 5 - Databázový systém[9]

[9] Zdroj: [14]

Tiskový server

Tiskový server je počítač nebo zařízení, které spravuje tiskárny na konkrétní síti. Je potřeba jeden počítač a tiskárna nebo více počítačů a tiskáren. Většina tiskových serverů je připojena přes LAN konektor a jeden nebo více fyzických portů, jako je například USB kabel.

Výhoda tohoto serveru spočívá v jednoduché instalaci a konfiguraci. Prakticky kterýkoli počítač z místní sítě se může připojit na tento server. Každý uživatel nemusí mít svoji vlastní tiskárnu. Stačí jedna tiskárna na chodbě pro celé oddělení. Tiskový server šetří čas a peníze.

Konfigurace a instalace je velmi snadná. Pomocí USB kabelu je připojen tiskový server spolu s tiskárnou. Ethernetový kabel je připojen jedním koncem do routeru a druhým koncem na tiskový server. Následně adaptér napájí tiskový server. Pomocí CD-ROM je nakonfigurován tiskový server. V položce průvodce nastavením lze zadat název serveru a nastavit buď automaticky, nebo ručně IP adresu serveru. Je potřeba nainstalovat ovladač k příslušné tiskárně pro každý počítač v síti. [22]

Obrázek 6 - Tiskový server TP-LINK TL-PS110U[10]

Proxy server

Proxy server je program, prostředník mezi klientem a serverem. Proxy server překládá požadavky klienta a přebírá odpověď ze serveru, kterou následně pošle klientovi. Proxy server odděluje lokální síť (intranet) od veřejné sítě (internet). Proxy server si ukládá populární stránky do vyrovnávací paměti cache. Buď klientovi pošle kopii stránky z paměti cache, nebo předá klientovi webovou stránku poslanou serverem. Díky ukládání kopií zlepšuje výkon a dokáže

[10] Zdroj: [45]

odfiltrovat reklamní bannery, zachytit malware[11] či blokovat stránky podle podezřelého obsahu. Pro soukromé procházení na veřejné síti lze použít takzvaného anonymního proxy serveru. [7]

Díky anonymnímu serveru může klient skrýt svoji IP adresu. Na internetu existuje několik set tisíc anonymních serverů. Ovšem většina internetových služeb vnímá anonymní IP adresy za problematické, protože se snadno dají zneužít pro činnosti, které nedovolují zákony. [2]

Aplikační server

Aplikační server, též APS, je server, který provozuje sdílené aplikace. Tento server se používá zejména u náročnějších aplikací ve firemní sféře. Jedna část aplikace je nainstalovaná na serveru a druhá část aplikace je nainstalována na uživatelském počítači, tato část slouží jako klient. V současnosti většina aplikačních serverů vychází ve standardu J2EE. [3]

Aplikační server je používán, když je zapotřebí provozovat aplikaci bezpečně, zaručit neztracení dat, mít aplikaci dostupnou 24 hodin denně odkudkoliv z internetu. Pronajímané aplikační servery šetří provozní cenu, odpadá potřeba vlastního hardwaru a starost o opravy a rozšiřování serveru.

Aplikační server začne uživatel využívat v případě, kdy má novou aplikaci, potřebuje ji kdekoliv mimo kancelář, přičemž mu ušetří čas i peníze, protože si uživatel není jist, jestli ji bude dlouhodobě užívat, ale potřebuje kvalitní provoz za dobrou cenu. Také ji využije v případě, kdy aplikaci již má, ale potřebuje ji dále rozšířit, zvýšit počet uživatelů a rychlost aplikace, zkvalitnit a zlevnit její provoz. [26]

Herní server

Servery se dají rozdělit na oficiální a neoficiální. Oficiálním serverem je obvykle vlastník nebo firma, která hru vytvořila. Většinou se za služby serveru platí. U neoficiálních neboli freeserverů je server udržován komunitou, nikoli oficiálními vývojáři. Oficiální i neoficiální servery jsou nejčastěji zastoupené herními servery. [35]

Herní server funguje jako databázový a aplikační server. U těchto serverů, jak by se mohlo zdát, není kladen takřka žádný důraz na grafickou kartu. Tento server opět posílá data skrz síť a žádné grafické rozhrání nezobrazuje. U těchto serverů je kladen důraz na výkonný procesor a velmi rychlé RAM paměti stejně tak jako u databázových a aplikačních serverů. [42]

[11] Škodlivý program určený k vniknutí nebo poškození systému

Media server

Media server se specializuje na média, jako je video, zvuk, fotografie, DCP[12] a jejich metadata. Server lze rozšířit o zabezpečení nebo zpoplatnění obsahu. Media server se dá rozdělit na media servery, které se specializují na určitou věc.

- Video server se zaměřuje pouze na video a metadata uložená v souborové formě. Může mít převodníky pro vstup a výstup videa po signálových kabelech.
- Produkční video server se specializuje na ukládání a sdílení videoobsahu při výrobě.
- Vysílací neboli odbavovací server se zaměřuje na výstup vysílaného videoobsahu. Je zapotřebí přesný a plynulý videoobsah.
- Stream server se specializuje na vysílání v on-line nebo off-line režimu do veřejné sítě. [40]

V domácnostech přibývá větší počet počítačů. S rostoucím počtem počítačů je žádoucí si zálohovat a ukládat data. Tento problém se dá vyřešit pomocí NAS serveru. NAS server funguje na principu malého počítače se sdílenými pevnými disky, na které se lze připojit z lokální sítě. Je usnadněno sdílení fotografií, videí, zvuků atd. Kvalitní NAS servery umí zálohovat data všem počítačům v lokální síti, popřípadě zpřístupnit společnou tiskárnu pomocí USB kabelu. [32]

Pokud není žádoucí propojit například televizi s počítačem pomocí kabelu HDMI, VGA, DVI, je možné využít DLNA server. Standard DLNA pro poskytování multimediálního obsahu vznikl v roce 2003 firmou Sony společnou domluvou s ostatními výrobci. DLNA server je zprovozněn například na počítači, kde se vyskytují média. Poté se stačí připojit pomocí WiFi a následně je možno vybírat, procházet složky stejně jako v počítači, na kterém je zprovozněn DLNA server. Doporučeným serverem je PS3 Media server. Tento server byl původně určený pro streamování obsahu na konzoli Playstation 3, ale díky standardu DLNA je možné přehrávat videa, prohlížet fotky například na XBOXu 360. [4]

Licenční server

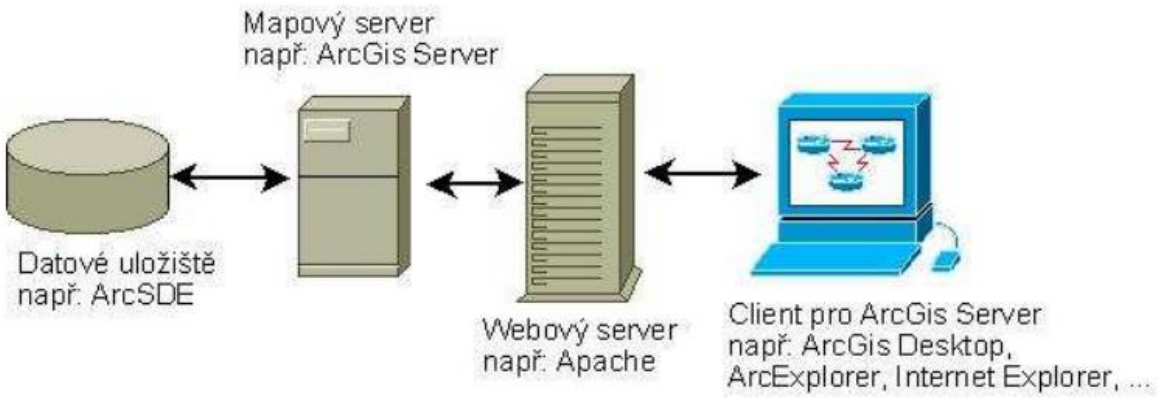
Licenční servery umožňují přes síť certifikaci klientů na daný požadovaný software. Je využíváno tzv. plovoucích licencí, kdy licenční sériové číslo není přiděleno napevno na určený počítač, ale je přidělováno dynamicky po síti pro potřeby klientů. Výhoda plovoucích licencí je ta, že zákazník platí pouze za počet využívaných licencí. Firma má *n* zaplacených licencí, pokud je překročeno *n* licencí, další uživatel nemůže používat daný licencovaný program. Po ukončení

[12] Filmový standard používaný pro filmovou distribuci v digitalizovaných kinech

práce jednoho z uživatelů se může jiný připojit, protože se jedna licence uvolnila. Je možné poskytnout sériové číslo dlouhodobě pro některého uživatele. Uživateli může být přiřazena off-line licence administrátorem pro dlouhodobé účely. [31]

Mapový server

Mapový server poskytuje služby GIS přes internet. Grafické informační systémy publikují geografická data, sbírají a aktualizují, synchronizují geografická data, poskytují webové mapové služby nebo webové analytické služby. Mezi známé služby mapových serverů patří *mapy.cz, mapy.atlas.cz, maps.google.cz* a další. Mapový server funguje na principu klient-server. [34]



Obrázek 7 - Princip mapového serveru[13]

Redakční server

Redakční server obsahuje aplikaci nazývanou redakční systém. Tato aplikace umožňuje jednoduše vytvářet, měnit a upravovat webové stránky bez znalosti programování. Jak bude kód vypadat, zajišťuje redakční systém. Redakční systém šetří čas jak na vytváření webu, protože psaní kódu bude vždy pomalejší, tak čas na aktualizaci stránek a publikaci informací. Na stránky je možné se připojit kdykoliv a odkudkoliv. Šetří peníze, protože není nutné platit odborníky za školení nebo za správu webu. Aktualizaci webu může provádět několik lidí najednou. Redakční systém patří mezi dražší aplikace, ale pokud je tento systém používán dlouhodobě, rychle se tyto investované peníze vrátí. Hostingový server má větší požadavky na redakční systém než statické stránky. Nicméně zaručením kvalitní hostingové služby je docílena stejná rychlost jako u statických stránek. Redakční systém je výhodné používat na firemní stránky, stránky měst a obcí, portály a informační weby, e-shopy, blogy, intranety. [9]

[13] Zdroj: [34]

3 WEBOVÝ SERVER

Roku 1989 Tim Berners-Lee vytvořil nový projekt za účelem zjednodušit komunikaci mezi vědci. Tím vznikl první webový prohlížeč a první webový server.

Webovým serverem se rozumí program, který nabízí své služby jiným programům (klientům). Klient je nejčastěji webový prohlížeč. Aby mohly programy spolu komunikovat, jsou stanovena pravidla pro komunikaci. Tato pravidla jsou známá jako protokoly. Přenos webových stránek umožňuje HTTP protokol. Tento protokol zpřístupňuje pomocí aplikačních bran další protokoly, jako je FTP nebo SMTP.

Webový server je také počítač, který poskytuje své služby dalším počítačům. Tento počítač má v sobě uloženy webové stránky jako přímé soubory, nebo předpisy (programy, skripty), jak webové stránky vygenerovat. Na webovém serveru běží program, který komunikuje s webovými prohlížeči počítačů na základě HTTP protokolu.

Pro lepší výkon webových serverů jsou určeny počítače např. firem HP, IBM nebo SUN. Tyto operační systémy nejčastěji pracují na operačních systémech Unix nebo na operačním systému Windows server. Nejpoužívanější software pro webové servery je Apache server, tento server je zdarma a přitom se jedná o velmi kvalitní produkt. [28] Za tímto softwarem se řadí další produkty, kterými jsou ISS od Microsoftu, Nginx od Igora Sysoeva a GWS od Google. [48]

V kapitole je čerpáno ze zdrojů: [10], [12], [15]-[19], [21], [23], [28], [30], [37], [38], [41], [44], [45]-[47].

3.1 Princip webserveru

Webový server pracuje na principu modelu klient-server, který je popsán v bodě 2.4.1. Ke komunikaci mezi klientem a serverem se používá protokol HTTP. Klient se připojí na server, pošle požadavek (request). Klient chce například stránku *www.abc.cz/zpravy.html*. Server se podívá do složky např. *C:\www*, ve které má uloženy veškeré složky (jména webů), ve kterých jsou uložené soubory jednotlivých webů. Z řetězce *www.abc.cz* zjistí jméno webu, které je *abc*, a že se tento web nachází v *C:\www\abc*. Soubor *zpravy.html* by se měl nacházet na cestě *C:\www.abc.cz/zpravy.html* [46] a měl by být psaný HTML jazykem, který vytváří webovou stránku. Pokud je vše v pořádku, server posílá (response) hlavičku se stavovým kódem 200 OK, typem souboru, popřípadě další informace. Dále server zasílá tělo, ve kterém jsou data, která si vyžádal klient, v tomto případě HTML soubor.

3.2 Využití webserveru

Webový server vrací klientům statický nebo dynamický obsah. Statický obsah jsou např. webové stránky uložené formou souborů na disku. Tato webová stránka je rychlejší, protože nepotřebuje měnit obsah webové stránky za chodu na požadavek uživatele. Pro vytvoření stránek není vyžadována znalost programování, je tu možnost ukázání kopie komukoli, nejsou kladeny žádné speciální požadavky na webhosting. Tato stránka může být zobrazena bez nutnosti webserveru a aplikačního serveru. Stránka může být uložena přímo na disku počítače nebo jiném paměťovém médiu. Nevýhoda je ta, že veškerý měnitelný obsah musí běžet na straně klienta. Údržba velkého počtu statických stránek může být prakticky nepoužitelná bez automatizačních nástrojů. [41]

Webový server může vytvářet také dynamický obsah. To je dáno tím, že různí klienti chtějí různá data. Například chtějí data ze souborů, databází apod. Tyto dynamické stránky jsou tvořeny webovými aplikacemi. Jednou z webových aplikací je například Facebook. Dynamické stránky používají skriptovací jazyky, v některých případech společně s databází.

Mezi nejznámější skriptovací jazyky patří PHP, JavaScript, Ruby, Python, Perl, Tcl. Skriptovací jazyk většinou pracuje na straně serveru, například PHP. Výhodou těchto skriptů je bezpečnost dat, protože je odeslán pouze výsledek, zdrojový kód je skryt koncovému uživateli (jiní vývojáři nemohou duplikovat). Je nežádoucí mít kompletní přístup do databází koncovým uživatelem. Webová aplikace například v PHP se chová vždy stejně bez ohledu na druh klienta. Při horší rychlosti připojení může docházet k horší režii mezi klientem a serverem, protože skripty na straně serveru zatěžují komunikaci klient-server. Nemusí vědět, co se děje na straně klienta, tím pádem nemusí zareagovat například na stisk tlačítka.

Pak jsou jazyky pracující na straně klienta, jím je např. JavaScript. V některých případech snižují propustnost mezi klientem-serverem a tím mohou urychlit režii. Jsou nevhodné pro přihlašovací údaje.

Třetími skripty jsou jazyky, které pracují jak na straně serveru, tak na straně klienta např. AJAX. Tuto technologii používají například Google mapy. [15]

3.3 Obecné vlastnosti webových serverů

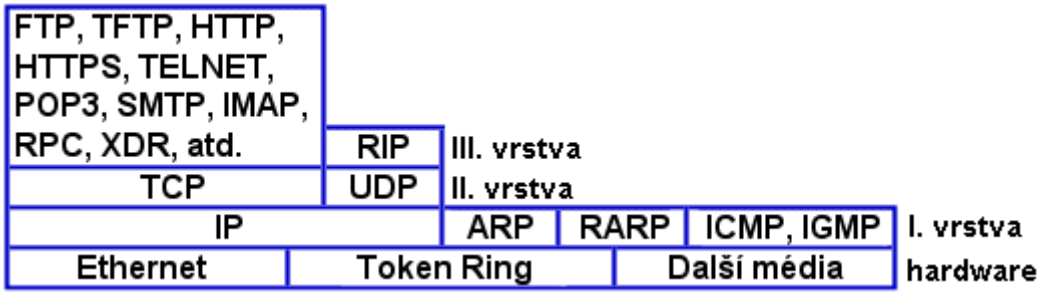
Webservery se můžou v určitých částech od sebe lišit, ovšem některé části mají společné. Webové servery potřebují síťové připojení pro komunikaci mezi počítači. Požadavky jsou posílány pomocí HTTP protokolu. Odpověď serveru je znovu realizována pomocí HTTP.

Odpověď obsahuje hlavičku se stavovým kódem a poté následuje požadovaný obsah (tělo), pokud je vše v pořádku. Tento obsah je dokument ve formátu HTML. Mohou se posílat i jiné formáty, jako jsou obrázky, skripty a jiné.

3.3.1 Protokoly pro webserver

Protokol je soubor norem či pravidel, které musí daný program nebo zařízení splňovat, aby bylo schopno komunikovat s jiným zařízením nebo programem. Protokoly jsou stále vyvíjeny, jsou zveřejňovány pomocí RFC dokumentů. Developer tvořící programy používající síťovou komunikaci musí nutně znát tyto protokoly. Klasický uživatel by měl mít povědomí o těchto protokolech, protože se s nimi může setkat v různých nastaveních. Protokoly se nacházejí v rodině protokolů, která se nazývá TCP/IP. Model TCP/IP je rozdělen do čtyř hlavních vrstev, kterými jsou: aplikační, transportní, síťová a vrstva síťového rozhraní. Tento model se liší od referenčního modelu ISO/OSI, který se snaží standardizovat počítačové sítě pomocí sedmivrstvového modelu. [44]

Ke každému protokolu se řadí takzvaný síťový port. Tento port je číslo od 1 do 65535 a slouží jako hlavní identifikace při síťové komunikaci mezi programy využívajícími TCP, UDP protokoly. Nejznámější síťové porty jsou 20 a 21 pro FTP, 22 SSH, 23 TELNET, 25 SMTP, 23 DNS, 80 HTTP, 110 POP3. [38] Komunikace je realizovaná pomocí tzv. socketů. Socket je koncový bod komunikace. Každý klient i server má svůj socket, ke kterému je přiřazen port. Tento port otevírá komunikaci a následně na něm odposlouchává takzvaný démon. Démon je program, který čeká na nějakou událost. Následně ji obslouží, bez nutnosti uživatele. Jako démon běží například databázový, webový server.



Obrázek 8 - Architektura TCP/IP[14]

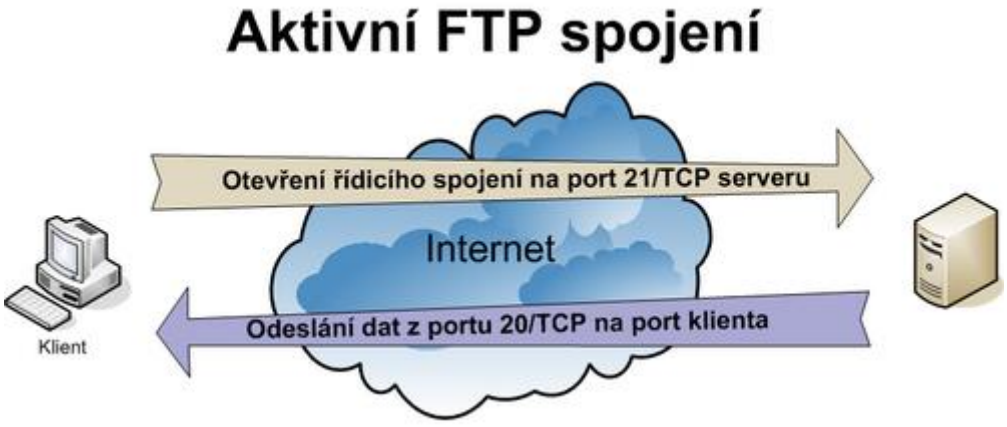
[14] Zdroj: [37]

Nejdůležitější protokoly pro webový server spadají do třetí vrstvy TCP/IP modelu. V této vrstvě jsou protokoly, jako je FTP, TFTP, HTTP, HTTPS, TELNET, POP3, SMTP, IMAP a další. [37]

FTP protokol

FTP protokol je jeden z nejstarších protokolů rodiny TCP/IP a využívá právě tuto bázi. Tento protokol je určen pro posílání souborů mezi dvěma počítači připojenými v síti. Jeho výhodou je vysoká rychlost, ale zároveň nemá prakticky žádné zabezpečení. Přenos tohoto protokolu není šifrovaný. Původně byl FTP protokol navržen v aktivním režimu s nedostatkem IPv4 a nástupem technologie NAT, byl navržen pasivní režim, který zjednodušuje práci uživatelům v lokální síti za NATem. NAT překládá privátní IP adresy na veřejné IP adresy. Server nevidí na koncového klienta, ale pouze na veřejnou adresu privátní sítě.

Aktivní spojení funguje tak, že klient nad vyšším portem, než je 1024/TCP, propojí spojení se serverem na portu 21/TCP. Zde probíhá řídící spojení. Poté co se klient prokáže pomocí loginu a hesla, může posílat serveru požadavky. Server tyto požadavky vyřizuje a výsledky posílá přes port 20/TCP zpět klientovi na klientem nastavený port. Aktivní FTP snižuje režii, protože je oddělena příkazová a datová část. Odeslaná data se nemusí odesílat zpět klientovi, ale klidně někam jinam. Nevýhoda je, když se klient připojuje z privátní sítě za NATem. Klient musí mít povoleno příchozí spojení s portem serveru 20/TCP na předem nespecifikovaný cílový port.



Obrázek 9 - Aktivní FTP spojení[15]

[15] Zdroj: [21]

U pasivního FTP spojení otevírá datové spojení klient nad náhodným portem nad 1024/TCP. Na klienta nejsou kladeny žádné nadstandartní nároky na spojení se serverem. [21]

Obrázek 10 - Pasivní FTP spojení[16]

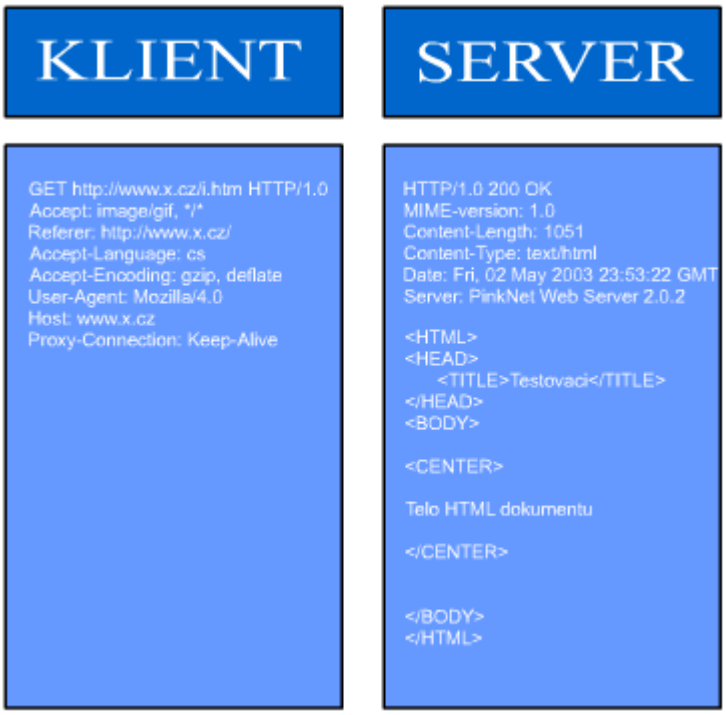
Pokud je zapotřebí šifrovaného přenosu, slouží k tomu protokoly SFTP a SCP.

HTTP protokol

Po FTP protokolu přišel HTTP protokol, který se spolu s protokoly elektronické pošty zasloužil o rozmach internetu. Tento protokol využívá port 80/TCP a posílá hypertextové dokumenty pomocí HTML jazyka. Pomocí standardu MINE může přenášet různé soubory, např. stejně jako elektronická pošta, používá se společně s formátem XML pro webové služby a pomocí aplikačních bran dokáže zpřístupnit i další protokoly, jako jsou SMTP a FTP. Pomocí lokátoru URL je zjištěno, jaký zdroj přesně klient vyžaduje. HTTP protokol neumožňuje šifrování, pro zabezpečení se využívá TLS protokolu a toto označení je známo jako HTTPS. [19] Protokol HTTP využívá architektury klient-server. Existují čtyři verze protokolu, kterými jsou 0.9, 1.0, 1.1, 2.0. Formáty response a request se v různých verzích liší. Nejpoužívanější verzí je 1.1. Ve verzi 1.0 klient pokaždé inicializuje TCP spojení pro každou komunikaci (požadavek/odpověď). Ve verzi 1.1 lze přidržet spojení pro více transakcí, čímž je zlepšena komunikace. Základem HTTP protokolu je načtení webové stránky, získání informací o webové stránce a odeslání uživatelských dat na server (formuláře na webu). Tyto operace jsou uskutečňovány pomocí příkazů GET, HEAD, POST. Pokud chce klient získat webovou stránku, použije příkaz GET, za ním bude lokátor URL a verze HTTP protokolu. Za těmito příkazy následuje hlavička se základními informacemi, každá položka pole hlavičky má svůj

[16] Zdroj: [21]

vlastní řádek. Po identifikátoru pole hlavičky následuje přiřazovací znak „:“, poté mezera a hodnota pole, řádek je ukončen sekvencí znaků CR, LF. CR posouvá kurzor na začátek řádku a LF posouvá kurzor na nový řádek. V těchto polích se dá nastavit mnoho parametrů. Po hlavičce následuje prázdný řádek (2x sekvence znaků <CR><LF>), který odděluje hlavičku od těla. V těle je posílán zadaný obsah, je-li požadován. [18]



Obrázek 11 - Ukázka hlaviček a těla požadavku klienta a odpovědi serveru[17]

Verze 1.1 rozšiřuje verzi 1.0 o další pole hlavičky. A přidává další příkazy PUT, DELETE, OPTIONS, TRACE. Příkaz PUT je podobný jako POST. U příkazu PUT klient zná, který URI je požadován, a nesmí odesílat požadavek na jiné zdroje. Pokud server chce poslat odpověď na jiné URI, musí odpovědět stavovým kódem 301 Moved Permanently. Metoda DELETE říká, aby server smazal požadovaný zdroj identifikovaný pomocí URL. Příkaz OPTIONS umožňuje klientovi určit možnost, omezení zdroje a schopnosti serveru. TRACE umožňuje klientovi vidět, co je přijato na straně serveru a použít data pro testování a diagnostiku. [16]

V roce 2015 byl vydán protokol HTTP/2. Společnost Google oznámila, že nahradí svůj rychlý protokol SPDY právě zmiňovaným protokolem HTTP/2. Podpora ze strany ostatních prohlížečů je prý otázkou času. Náročnější požadavek např. požadavek na obrázek je časově náročný, proto jsou požadavky zapouzdřeny a zkomprimovány, čímž je zaručena daleko nižší

[17] Zdroj: [17]

režie. Druhá novinka je cache pushing, kdy webový server posílá data ještě dříve, než si o ně klient řekne. Při vytvoření a odeslání dotazu na stránku server odpoví a navíc odesílá různé externí soubory, jako jsou CSS soubory, PHP a Java scripty. HTTP/2 usnadňuje TLS šifrování, které je méně náročné na systémové prostředky. [10]

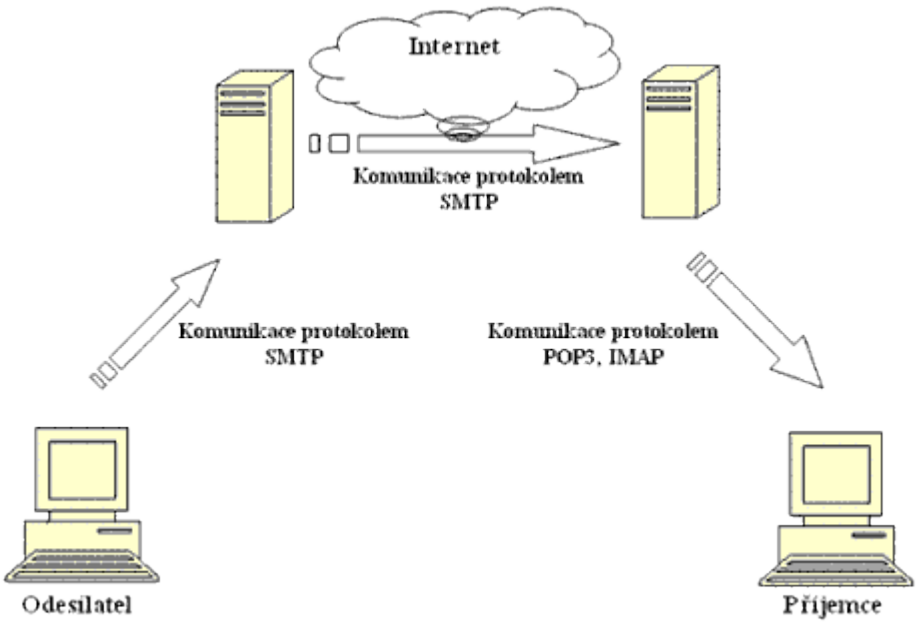
Protokoly elektronické pošty

Elektronickou poštu používá 90% uživatelů internetu. [12] Odeslání e-mailu je provedeno přes SMTP protokol klientem, například MS Outlook. SMTP je jedním z nejstarších protokolů rodiny TCP/IP. Používá port 25/TCP pro komunikaci mezi e-mailovými servery a port 587/TCP pro příjem e-mailů od e-mailových klientů. MTA zachytí tento e-mail. Poté zjistí, kde poslouchá MTA pro danou doménu příjemce, a pošle mu daný e-mail přes SMTP protokol. [30]

Každý klient potřebuje elektronickou schránku, která je uložena na e-mailovém serveru, kam právě posílá MTA e-mail přes SMTP protokol. Z této schránky si klienti přes IMAP nebo POP3 stahují e-maily.

E-mailový klient se dá nahradit webmailem. Webmail je e-mailový klient fungující ve webovém prohlížeči používající většinou protokol IMAP. [30] Protokol POP3 pracuje v off-line režimu, je vhodný pro uživatele, kteří nemají permanentní přístup k internetu. Uživatel si postahuje e-maily k sobě na úložiště. Výhoda i nevýhoda je v tom, že se tyto e-maily na serveru smažou. Výhodu mají klienti, kterým chodí velký počet zpráv, a ve své schránce budou mít místo pro další e-maily.

IMAP pracuje v on-line režimu. Se svojí schránkou může uživatel pracovat odkudkoli. Na počítač se stahují pouze důležité informace, jako jsou hlavičky zpráv. Při vybrání e-mailu se ze serveru stáhne celá požadovaná zpráva. U zpráv se uchovává, jestli je zpráva nepřečtená, důležitá, odvozená. Uživatel může libovolně složky mazat, vytvářet, přeskupovat apod. Protokolem je možné připojit více klientů. [23]



Obrázek 12 - Princip elektronické pošty[18]

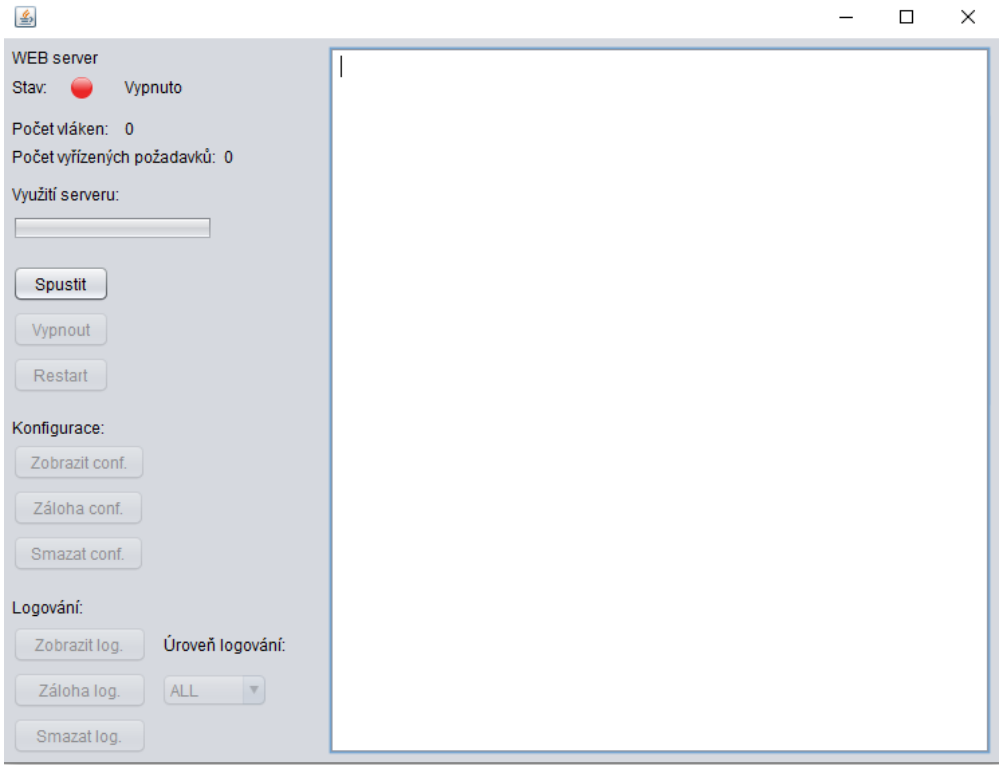
3.3.2 Stavové kódy protokolu HTTP

Stavové kódy jsou součástí HTTP protokolu. Jsou umístěny v hlavičce odezvy serveru na určitou událost. Po odpovědi (response) serveru se klient dozvídá, jak server vyhodnotil požadavek (request) - zda byl požadavek vyhodnocen kladně, záporně, nebo došlo k chybě. Klient díky této informaci může provést patřičné kroky. Každý stavový kód má tři čísla. První číslo říká, o jaký druh se jedná, a zbylá dvě čísla určují přesnou definici stavového kódu. Stavové kódy se dělí na 5 druhů: informační odpovědi 1xx, úspěšné reakce 2xx, přesměrování zpráv 3xx, chybové reakce klienta 4xx, chybové reakce serveru 5xx. Přehled stavových kódů se dá dohledat na internetu, například zde: *https://httpstatuses.com*. Mezi nejznámější stavové kódy patří 200 OK. Tento kód říká, že je vše v pořádku a server požadovaný dokument má. 301 Moved Permanently říká, že je dokument trvale přesunut, a 302 Found říká, že je dokument dočasně přesunut. 304 Not Modified znamená, že není modifikováno. Dokument klient nedostane, pokud nebyl do určité doby změněn. 403 Forbidden oznamuje neoprávněný přístup, například když je snaha o výpis nedovoleného adresáře. 500 Internal Server Error oznamuje, že je chyba na serveru a ne na straně klienta. Jedna z nejznámějších error page 404 Not Found znamená, že server nenalezl požadovaný dokument.

[18] Zdroj: [12]

4 VLASTNÍ WEBSERVER

Cílem ručně psaného webového serveru je snaha pochopit a demonstrovat základní principy funkce webového serveru. Tento webový server je napsán v programovacím jazyce Java, ovládaný pomocí GUI. [50] Ovládání webového serveru pomocí GUI je blíže popsáno a vysvětleno v příloze C. Díky velké oblibě a rozšířenosti tohoto jazyka by měla většina uživatelů (studentů) snadno pochopit implementaci a funkci webového serveru. Tento server využívá protokolu HTTP. Je žádoucí vědět, co přesně server dělá. K tomu slouží logování, které zapisuje hlášky jak do souboru, tak do textového panelu aplikace. Server si načítá konfigurační soubor například kvůli tomu, aby věděl, jaké webové stránky má k dispozici. Server by měl korektně reagovat na požadavky a tyto požadavky vyřizovat pomocí vláken.



Obrázek 13 - GUI webového serveru[19]

V kapitole je čerpáno ze zdrojů: [25], [33], [48]-[50].

4.1 Struktura

Aplikace je založena na modelu klient-server. Serverem se rozumí psaná aplikace v Javě, která komunikuje s klientem (webovým prohlížečem). Tato komunikace probíhá pomocí odkazu na právě používaný počítač, který se nazývá localhost. Tento odkaz má v každém počítači

[19] Zdroj: autor

vyhrazenou IP adresu *127.0.0.1*. Díky tomuto odkazu může počítač pracovat sám se sebou. Pomocí odkazu localhost může klient komunikovat s aplikacemi, které používají protokol TCP/IP v tom samém počítači. Ke každé aplikaci se řadí síťový port. K aplikaci se přiřadí síťový port pomocí konfiguračního souboru. Po spuštění serveru je možné požadovat od serveru webové stránky, které má uloženy na úložišti - konkrétně: *C:\www\nazevWebu\webovaStranka*. Do adresní řádky webového prohlížeče uživatel musí zadat *localhost:port/cesta,* místo *localhost* je možné napsat *127.0.0.1*. Například je požadováno načtení webové stránky *index.html* z webu *mujweb*. Uživatel tedy zadá do webového prohlížeče *localhost:8080/mujweb/index.html*. Server stránky pošle klientovi, pokud jsou na serverovém disku v adresáři *mujweb*. Díky číslu portu 8080 klient ví, že má komunikovat právě s webovým serverem, ke kterému se řadí číslo portu 8080. V případě chyby, kdy server nemůže najít požadovaný obsah, vrací klientovi chybovou stránku. Díky těmto stránkám uživatel zjistí, co je špatně, a může se podle toho patřičně zařídit. Je možné zadat do adresní řádky prohlížeče *localhost:8080/mujweb*, server přesměruje požadavek a uživateli vrátí webovou stránku, jejíž adresa je *localhost:8080/mujweb/index.html*. Pokud je zadána adresa *localhost:8080/mujweb/*, je požadavek doplněn o *index.html* a vrátí *localhost:8080/mujweb/index.html*. Pokud uživatel zadá pouze *localhost:8080*, server přesměruje požadavek na statickou webovou stránku *localhost:8080/home/index.html*.

4.2 Klíčové funkce

Aby webový server fungoval tak, jak má, potřebuje soubor funkcí, kterými jsou konfigurační soubory k prvotní inicializaci serveru, vlákna na vyřizování požadavků, správné nastavení portů, implementace předávání obsahu a práce se stavovými kódy.

Program obsahuje třídu *Log*, která slouží k různým výpisům systému. Tato třída umí ukládat výpisy serveru (hlášky serveru, co přesně se stalo). Tyto hlášky ukládá po vypnutí serveru do souboru. Třída *Log* umožňuje vytvořit zálohu logovacích hlášek a mazání uložených logovacích hlášek. Server obsahuje třídu *Config*, která načítá konfigurační soubor, vytváří zálohu konfiguračního souboru, obnovuje konfigurační soubor a maže konfigurační soubor. Třída *Server* pouští a zastavuje server. Třída *Handler* přijímá požadavky od klienta a vrací klientovi odpovědi.

4.2.1 Hlavní okno

Nad třídami programu operuje třída *MainFrame*, ve které je implementováno grafické rozhraní webového serveru. Při spuštění aplikace je vytvořena ve třídě *MainFrame* instance třídy *Log*

s parametrem *this* (instance třídy *MainFrame*), dále je vytvořena instance třídy *Config* s parametry právě vytvořené instance třídy *Log* a parametrem *this*. Při kliknutí na tlačítko Spustit se prověří, jestli byla konfigurace úspěšná, pokud ano, vytváří se instance serveru s parametry: instance třídy *Config*, *Log* a instance třídy *MainFrame*. Instance třídy *Server* tedy do sebe zabalí všechny ostatní důležité komponenty, a tudíž může korektně volat metody *start()* a *stop()*, díky kterým umožňuje spouštět a vypínat webový server.

4.2.2 Logování serveru

Je důležité vědět, mít zaznamenáno, co server přesně dělá a udělal. Tudíž při chodu serveru je požadován výpis, co přesně server udělal (na jakém portu běží, je-li požadavek vyřízen, nebo nevyřízen, změna úrovně hladiny a další). Při různých chybách je důležité dohledat, co za chybu se přesně stalo. To by mělo být možné dohledat v právě vypisovaných hláškách. Na počátku webového serveru byla implementována třída *Log*.

Třída Log

Ve třídě *Log* jsou řešeny výpisy, které jsou následně zobrazovány do textového panelu aplikace nebo uloženy do textového souboru. Za běhu serveru jsou logovací hlášky uloženy do proměnné *loggs* datového typu *LinkedBlockingDeque<String>*, tato proměnná je následně vypisovaná. Tato třída implementuje metodu *logPrint(String, String)*. Metodě je předáván textový řetězec zprávy a logovací úroveň *INFO, WARN, ERROR*. Tyto úrovně specifikují, co přesně se stalo. *INFO* například říká, že byl přijat požadavek od klienta. Hladina *WARN* varuje například, že hladina logování byla změněna a některé druhy logů nebudou vypisovány. Hladina *ERROR* říká, že nastala určitá chyba. Tato metoda je řešena pomocí konstrukce *switch*. Pro určení větve slouží název hladiny logování, kterými jsou *INFO* (informace), *WARN* (varování), *ERROR* (chyba). Do proměnné *level* datového typu S*tring* je uložena hladina z konfiguračního souboru (defaultně hladina *ALL*) nebo změněná hladina v checkboxu, kterými jsou další hladiny *SYSTEM* (systém), *NONE* (žádná hladina), *ALL* (všechny hladiny). Hladina *SYSTEM* umožňuje používat pouze hladiny *INFO* a *WARN*.

Třída *Log* implementuje další metody, které umožňují otevření, uložení, zálohování a smazání logovacího souboru.

Otevření logovacího souboru – Metoda *showLog()* načte data ze souboru *log.txt* ze zdrojové složky projektu. Zkouší otevřít soubor a následně přidává logovací hlášku, že byl soubor otevřen. Pokud soubor nelze otevřít, je zachycena výjimka hladiny *ERROR – Log.txt not found*.

V následujícím kódu je zobrazena metoda pro zobrazení logovacího souboru.

```
public boolean showLog() {
        File file = new File(logPath); //načtení souboru dané cesty
        try {
            Desktop.getDesktop().open(file); //zobrazení souboru
            logPrint("Log.txt opened.", "INFO"); //výpis informační hlášky
            return true; //pokud je vše v pořádku, tak vrať hodnotu true
        } catch (Exception ex) { //zachycení výjimky
            logPrint("Log.txt not found.", "ERROR"); //výpis chybové hlášky
            return false; //pokud je zachycena výjimka, potom vrať false
        }
    }
```

Uložení logovacího souboru – K uložení logovacích hlášek slouží metoda *saveToFile()*. Pokud soubor *log.txt* v *C:/www/* neexistuje, je vytvořen nový soubor *log.txt* do adresáře *www*. Do tohoto souboru pomocí cyklu *for-each* jsou ukládány jednotlivé logy z proměnné *loggs*. Tato metoda je volána po kliknutí na tlačítko Vypnout (vypnutí serveru).

Zálohování logovacího souboru – Je možné zálohovat logovací hlášky pomocí metody *createBackup()*. Metoda vytvoří soubor *log.txt* ve zdrojové složce projektu. Do tohoto souboru jsou následně pomocí *for-each* cyklu zapisovány logy z proměnné l*oggs*.

Smazání logovacího souboru – Metoda *removeLog()* smaže soubor *log.txt* ze zdrojové složky projektu. Pokud ne, metoda vrací *false*.

4.2.3 Konfigurační soubory

Při spuštění serveru server načte prvotní konfiguraci. Konfigurace je uložena pomocí textového souboru ve složce projektu v kořenovém adresáři. Webový server z konfigurace čte cesty k webům, kde má být uložen soubor s logovacími hláškami a kam se má ukládat záloha konfiguračního souboru. Dále čte, jaká má být nastavena prvotní úroveň logování, s kolika vlákny může webový server pracovat, číslo portu, ke kterému se řadí aplikace a případně chybové stránky, které jsou uloženy ve formátu HTML na úložišti serveru. Na další stránce je zobrazen konfigurační soubor. Znak # udává, že daný řádek je pouze komentář.

```
#Příkazy pro weby mohou být WEB (vždy se automaticky bude hledat index.html)

WEB home C:\www\home
WEB mujweb C:\www\mujweb
WEB pacmania C:\www\pacmania
WEB WEB C:\www\WEB
WEB zakaz C:\www\zakaz

#Příkazy pro další soubory mohou být LOG, CONFIG

LOG C:\www\log.txt
CONFIG C:\www\config.txt

#Příkazy nastavení mohou být LOGLEVEL (ALL, SYSTEM, ERRORS, NONE), MAXTHREADS (1-20)

LOGLEVEL ALL

#Maximální počet vláken

MAXTHREADS 20

#Port pro spuštění serveru

PORT 8080

#Příkazy pro errorpage mohou být: ERRORPAGE404, ERRORPAGE423, ERRORPAGE505

ERRORPAGE404 C:\www\errorpages\404.HTML
ERRORPAGE423 C:\www\errorpages\423.HTML
ERRORPAGE505 C:\www\errorpages\505.HTML
```

Třída Config

V třídě *Config* je implementovaná metoda *inicializeConfig()*, která zkouší načítat data z konfiguračního souboru. Pokud soubor neexistuje, nebo je adresářem, je vypsána hláška *Config.txt not found.* a metoda vrací *false*, v opačném přídě vrací *true*. Pomocí vstupního proudu *InputStreamReader* server načte a uloží soubor do proměnné *s* datového typu *BufferedReader*. Jsou postupně čteny řádky, přičemž jsou vynechány prázdné řádky a řádky, na kterých je první znak *#*. Za tímto znakem jsou psány komentáře. Pokud není prázdný řádek nebo znak *#*, je pomocí prvního slova určen řetězec (identifikátor), díky kterému se vykoná příslušná větev v konstrukci *switch*. Ve větvích se přiřazují parametry, s kterými je následně pracováno. Webové stránky jsou ukládány pomocí *HashMap<String, String>*, první parametr (klíč) je název webu a druhý parametr je cesta k tomuto webu. V následujícím kódu jsou ukázány některé větve (cesta zálohy logování, konfiguračního souboru, prvotní úroveň logování, počet vláken, s kterými webový server může pracovat a uložení čísla portu), díky kterým jsou ukládány parametry z konfiguračního souboru. V proměnné *idx* datového typu *int* je hodnota, která udává index, kde končí identifikátor a následuje mezera.

```
case "LOG":
    LOGPath = s.substring(idx + 1, s.length());
    break;
case "CONFIG":
    ConfigPath = s.substring(idx + 1, s.length());
    break;
case "LOGLEVEL":
    LOGLEVEL = s.substring(idx + 1, s.length());
    break;
case "MAXTHREADS":
    maxNumOfThreads = (Integer.parseInt(s.substring(idx + 1, s.length())));
    break;
case "PORT":
    port = Integer.parseInt(s.substring(idx + 1, s.length()));
    break;
```

Ve třídě *Config* jsou další metody datového typu *boolean* pro správnou funkčnost webserveru. Těmito metodami jsou metody pro otevření, obnovení, zálohování a mazání konfiguračního souboru.

Otevření konfiguračního souboru - Metoda *openConfig()* otevře konfigurační soubor *config.txt* ze zdrojové složky projektu. Pokud je vše v pořádku, vypíše se hláška *Config.txt opened.* A metoda vrací *true*, v opačném případě se vypíše hláška *Config.txt not found* a metoda vrací *false*. Konfigurační soubor je možné libovolně měnit a následně ho uložit.

Mazání konfiguračního souboru - Metoda *delete()* smaže *config.txt* ze zdrojové složky projektu. Pokud je soubor smazán, je vypsaná hláška *Config.txt deleted.* Pokud soubor není nalezen nebo je špatná cesta, je vypsána logovací hláška *Cannot delete – Config.txt not found.*

Vytvoření zálohy konfiguračního souboru - Metoda *createBackup()* načte do instanční proměnné *br* datového typu *BufferedReader* nový objekt *BufferedReader* s parametrem *InputStreamReader*. Proměnná *br* následně přes cyklus *while* ukládá řádky do pomocné *s* a následně jsou tyto řádky zapsány pomocí proměnné datového typu *PrintWriter* do souboru *C:/www/config.txt*. Pokud nejde, metoda vrací *false*.

Obnovení konfiguračního souboru – Metoda *restore()* funguje na stejném principu jako metoda *createBackup()* s tím rozdílem, že pokud nelze nalézt soubor *config.txt* ve zdrojové složce projektu, tak metoda *restore()* zkusí najít zálohu z *C:/www/config.txt*. a soubor *config.txt* zapíše do zdrojové složky projektu, pokud ne, metoda vrací *false*.

4.2.4 Vlákna

Server je vytvořen jako vícevláknový. První vlákno začne běžet při startu serveru (tlačítka Spustit/Vypnout). Toto vlákno pouze naslouchá na portu a pro každý požadavek, který na port přijde, vytvoří nové obslužné vlákno. Toto obslužné vlákno následně zjistí, o co prohlížeč požádal (html soubor, css soubor, obrázek apod.) a pokusí se ho najít. Pokud požadovaný soubor najde, odpoví prohlížeči (odešle požadovaný soubor s relevantní hlavičkou), pokud ho nenajde, odpoví chybou 404.

Třída *Thread* implementuje třídu *Runnable,* ve které je definována metoda *run()*. Metoda *start()* ve třídě *Thread* volá metodu *run()*. Následující kód demonstruje volání metody *run()* pomocí metody *start()*. Kód v metodách *run()* probíhá pro obě vlákna paralelně. [33]

```
public class Main{
public static class Main(String args[]){
Thread t1 = new MyThread();
Thread t2 = new MyThread();
t1.start();
t2.start();
        }

class MyThread extends Thread{
@Override
public void run(){
// co má vlákno udělat
                }
        }
}
```

Třída Server

V této třídě jsou implementovaná právě zmiňovaná vlákna. Po zavolání metody *start()* ve třídě *Server* je kontrolováno, zda není vytvořené vlákno a jestli server neběží. Pokud není vytvořeno nové vlákno a server neběží, tak server zkouší vytvořit první vlákno, spustit server (proměnná *running* nastavena na hodnotu *true*) a přiřadit serverovému socketu port, pokud nelze, server vypíše hlášku chyby.

Metoda *run()* ve třídě *Server* zkouší projít metodou, dokud se nezmění *running* na *true* (server běží), jinak metodou neprojde. Pokud je proměnná *running* nastavena na hodnotu *true*, tak proměnná *clientSocket* datového typu *Socket* čeká na požadavek od webového prohlížeče. Jakmile se do proměnné *clientSocket* uloží požadavek, tak je vytvořeno nové obslužné vlákno s parametrem nový *Handler* s parametry: *Socket, Config, Server, Log, MainFrame*. Toto vlákno

následně zavolá metodu *start()*, která poté zavolá metodu *run()* ve třídě *Handler*. Metoda *run()* ve třídě *Handler()* slouží k vyřizování požadavků. [25]

Kód metody *start()*:

```
public void start() { //start serveru
    if (firtThread == null && !runnig) { //pokud není vlákno vytvořeno a neběží server
        firtThread = new Thread(this); //vytvoření nového vlákna

        log.logPrint("Webserver starting up on port " + port, "INFO");
        frame.setTextArea(null); //vypsání logovací hlášky
        try {
            socket = new ServerSocket(port); //přiřazení portu serverovému socketu
            firtThread.start(); //spuštění prvního vlákna, následně volaná metoda run()
            runnig = true; //server běží

            numOfRunningThreads.incrementAndGet(); //počítadlo - přidání běžícího vlákna
            vytizeni(); //grafické znázornění vytíženosti serveru
            numOfThreads = new AtomicInteger(0); //počítadlo - vyřízená vlákna

            frame.setLables(getNumOfRunninfThreads(), getEnedThreads());
            //vypsání počtu běžících a celkově ukončených vláken
        } catch (Exception e) {

            log.logPrint("Server cannot start: " + e.toString(), "ERROR");
            frame.setTextArea(null);
        }
    }
}
```

Kód metody *run()*:

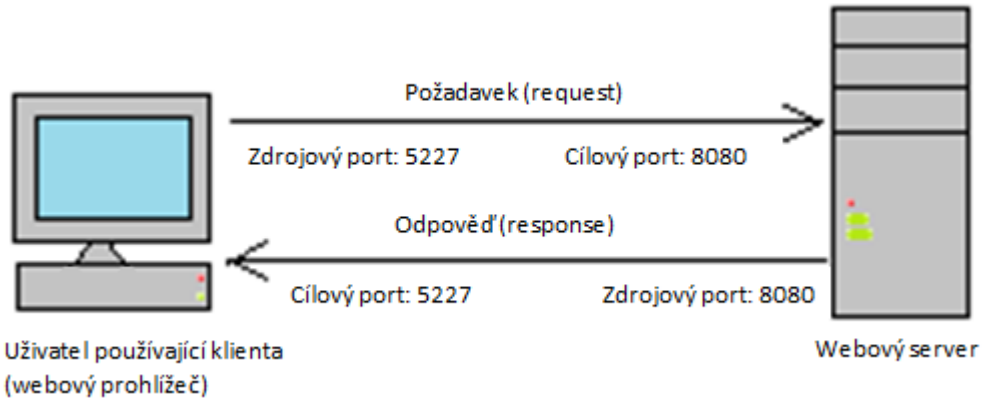
```
@Override
 public void run() {

     try {
         while (runnig) {
             if (getIntNumOfRunningThreads() <= config.getMaxPocetVlaken()) {
             //pokud je překročen maximální počet vláken, tak server nečeká na další požadavky
                 Socket clientSocket = socket.accept(); //čeká na požadavek od prohlížeče
                 new Thread(new Handler(clientSocket, config, this, log, frame)).start();
             //vytvoření a spuštění nového vlákna pro každý požadavek,
             //poté program pokračuje v metodě run() ve třídě Handler pro každé vlákno
             }
         }

         log.logPrint("Srever stopped", "INFO");
         frame.setTextArea(null); //vypsání logů do textového panelu
     } catch (Exception e) {

         log.logPrint("Error while server run -> " + e.toString(), "ERROR");
         frame.setTextArea(null);
     }

 }
```

Metodou *stop()* ve třídě *Server* je možné server vypnout. Pokud je první vlákno nenulové, server se pokusí zavřít *serverSocket* a následně zastavit server nastavením proměnné *running* na hodnotu *false*.

4.2.5 Obsazení portů

Ke každé síťové aplikaci je přiřazen síťový port, na tomto portu poslouchá tzv. démon (program). Některá čísla portů používají známé protokoly. Například protokol HTTP používá ke komunikaci port 80 (8080). Některé aplikace mohou být špatně napsané, například aplikace Skype používá také port 80, přičemž když je požadována instalace Apache serveru, tak Apache server hlásí chybu, protože chce ke komunikaci právě port 80, proto je zapotřebí přepsat číslo portu v konfiguračním souboru Apache například na 8080, pokud není požadována odinstalace Skypu.

Serverový port je nastaven podle konfiguračního souboru (*int serverPort* = 8080). Tato hodnota je předána jako parametr, při vytvoření objektu typu *ServerSocket*. Díky tomuto je určen koncový bod webového serveru, na který mají přicházet požadavky od klienta. Pomocí zavolání metody *serverSocket.accept()* program čeká, dokud nepřijde požadavek od klienta (démon naslouchá na portu 8080 na příchod požadavku, následně vkládá požadavek do proměnné *clientServer* datového typu *Socket*), kde je mimo jiné také vložen port klienta (server posílá odpovědi na tento port). Poté je proměnná *clientServer* předána jako parametr společně s parametry datového typu *Log, Config,* Main*Frame* a *Server*, při vytvoření nového objektu typu *Handler*, který je vložen jako parametr nového vlákna, toto vlákno následně vykoná požadovaný požadavek. Do proměnných *input*, *output* datového typu *InputStream* a *OutputStream* jsou uloženy vstupní a výstupní proudy proměnné *clientSocket*. Následně je pomocí vstupního proudu vyčten požadavek a server postupně posílá vyřízené požadavky pomocí proměnné *output.write("jeden řádek požadavku")*. Po odeslání celého výsledku je vstupní a výstupní proud datového typu *clientSocket* zavřen pomocí metody *close()*. Celý cyklus se provádí pro každý požadavek (html, css, obrázek atd.).



Obrázek 14 - Komunikace klient-server pomocí portů[20]

4.2.6 Předávání obsahu

K vyřizování požadavků slouží třída *Handler*, ve které je implementovaná metoda *run()*. Tato metoda načte požadavek od klienta a následně odešle výsledek klientovi.

Třída Handler

Po příchodu požadavku od klienta je vytvořeno nové vlákno ve třídě *Server*. V novém vlákně je vytvořen a vložen nový objekt datového typu *Handler*, který volá parametrický konstruktor. Tento konstruktor následně inicializuje proměnné ve třídě *Handler*. Do proměnných *in*, *out* datových typů *BufferedReader*, *BufferedWriter* jsou uloženy vstupní a výstupní proudy klient socketu, díky čemuž může server číst požadavek, následně ho vyřídit a poslat zpět klientovi. Zavoláním metody *start()* je zavolaná metoda *run()* ve třídě *Handler*, která vyřizuje požadavky.

```
Socket clientSocket = socket.accept(); //čeká na požadavek od prohlížeče
//po příchodu požadavku program pokračuje
new Thread(new Handler(clientSocket, config, this, log, frame)).start();
//vytvoření nového vlákna s parametrem nového objektu
//po zavolání metody start() je následně spuštěna metoda run() ve třídě Handler
```

Metoda run

Tato bezparametrická metoda funguje tak, že jsou pomocí *in.readLine()* postupně vyčteny všechny řádky požadavku. Pokud jeden z řádků obsahuje řetězec *GET* (požadavek klienta) například *GET /mujweb/index.html HTTP/1.1*, tak je nejdříve ze řetězce odstraněn podřetězec *GET* a *HTTP/1.1*. Následně je do proměnné *temp* vložen název složky, ve kterém se nachází celý web. V tomto případě *mujweb*. Do proměnné *get* je uložena webová stránka, která by měla být načtena (*index.html*). Do proměnné *path* je uložena proměnná *temp*. Proměnná *path* je klíč, podle kterého je vyhledána absolutní cesta do složky webu. Proměnná *path* je předávána jako

[20] Zdroj: autor

parametr do metody *getWebPath(String)* ve třídě *Config*, je vrácena absolutní cesta do složky webu. Pokud uživatel zadá pouze *index* (*temp = "index"*), *index.html* (*temp = "index.html"*), nebo nezadá žádnou cestu (*temp = ""*), je mu načtena domovská stránka serveru. Následně je vykonán tento kód:

```
File file = new File(config.getWebPath(path) + "/" + get);
//vytvořena nová instance na nový soubor s cestou do složky,
//kde se nachází web + konkrétní soubor ve složce webu
BufferedReader fw = new BufferedReader(new FileReader(file));
//pokus o přečtení souboru, pokud je cesta špatná,
//je zachycená výjimka a následně poslána error page 404 klientovi
```

Pokud program pokračuje a nevytvoří výjimku, pomocí proměnné *out* je nejdříve do klient socketu zapsán stavový kód, že je vše v pořádku. Následně je z proměnné *get* (jaký soubor se má načítat) uložena koncovka do proměnné *ending*. Následně je tato koncovka porovnávána s koncovkami, které je webový server schopen vyřizovat. Pokud je koncovka *html*, tak server zapíše do socketu klienta *out.write("Content-Type: text/html\r\n\r\n")*. První *\r\n* umístí kurzor zpět na začátek řádku a nový řádek, druhé *\r\n* je potřeba k ukončení hlavičky. Následně je odesláno tělo odpovědi. U html, css a dalších textových souborů jsou postupně zapsány řádky do socketu klienta. Odeslání obrázků je uskutečněno pomocí následujícího kódu.

```
BufferedImage img = ImageIO.read(file); //uložení obrázku do proměnné img
ImageIO.write(img, "jpg", clientSocket.getOutputStream());
//zapsání obrázku do klient socketu s koncovkou jpg
```

Na další stránce je kód programu, kde je zobrazeno, jaké možné formáty podporuje a následně vyřizuje ručně psaný webový server. [49]

```
if (ending.equals("html")) { //porovnání koncovky
    out.write("Content-Type: text/html\r\n\r\n");
    //zapsání do klient socketu, o jaký soubor se jedná
    log.logPrint("Pripojen klient:" + clientSocket.getInetAddress(), "INFO");
    frame.setTextArea(null); //výpis loginů
} else if (ending.equals("js")) {
    out.write("Content-Type: text/javascript\r\n\r\n");
} else if (ending.equals("ttf")) {
    out.write("Content-Type: font/ttf\r\n\r\n");
} else if (ending.equals("css")) {
    out.write("Content-Type: text/css\r\n\r\n");
} else if (ending.equals("png")) {
    out.write("Content-Type: image/png\r\n\r\n");
    writeImg = true;
    BufferedImage img = ImageIO.read(file);
    ImageIO.write(img, "png", clientSocket.getOutputStream());
    //zapsání obrázku do klient socketu
} else if (ending.equals("jpg")) {
    out.write("Content-Type: image/jpg\r\n\r\n");
    writeImg = true;
    BufferedImage img = ImageIO.read(file);
    ImageIO.write(img, "jpg", clientSocket.getOutputStream());
} else if (ending.equals("mp4")) {
    out.write("Content-Type: video/mp4\r\n\r\n");
    writeImg = true;
    OutputStream output = clientSocket.getOutputStream();
    byte[] data = new byte[1024];
    int len = 0;
    FileInputStream input = new FileInputStream(file);
    while ((len = input.read(data)) > -1) {
    output.write(data, 0, len);
    //postupné zapisování po 1 KiB do klient socketu
    }
} else if (ending.equals("gif")) {
    out.write("Content-Type: image/gif\r\n");
    writeImg = true;
    BufferedImage img = ImageIO.read(file);
    ImageIO.write(img, "gif", clientSocket.getOutputStream());
} else if (ending.equals("bmp")) {
    out.write("Content-Type: image/bmp\r\n\r\n");
    writeImg = true;
    BufferedImage img = ImageIO.read(file);
    ImageIO.write(img, "bmp", clientSocket.getOutputStream());
} else if (ending.equals("ico")) {
    writeImg = true;
    out.write("Content-Type: image/x-icon\r\n\r\n");
    BufferedImage img = ImageIO.read(file);
    ImageIO.write(img, "ico", clientSocket.getOutputStream());
}
if (!writeImg) { //pokud není soubor obrázek nebo video,
    //program zapisuje postupně text souboru do socketu klienta
    while ((line = fw.readLine()) != null) {
        out.write(line);
        out.newLine(); } }
```

Po zapsání odpovědi do klientova socketu, musí být vyprázdněn výstupní proud pomocí *out.flush()* a následně jsou zavřeny vstupní (*in.close()*) a výstupní proudy (*out.close()*) klient socketu.

4.2.7 Stavové kódy

V aplikaci je použito 5 stavových kódů. Pokud je zadána špatná cesta k souboru, je poslán klientovi stavový kód 404 Not Found a následně je mu zaslána chybová stránka html. Pokud je načtená cesta v pořádku a lze číst ze souboru, je klientovi poslán stavový kód 200 OK následovaný typem souboru a tělem odpovědi. Pokud není uveden soubor, který má server poslat klientovi například pouze *localhost:8080/mujweb*, server pošle klientovi stavový kód 302 Found a následně řádek *Location: "C:\www\mujweb\index.html"*. Tento řádek oznamuje klientovi, kde se nachází požadovaný soubor. Je vyprázdněn výstupní proud klient socketu a zavřen vstupní, výstupní proud. Prohlížeč následně změní adresu na *localhost:8080/mujweb/index.html* a požaduje novou odpověď od serveru. Pro další příklady stavových kódů jsou použity další stavové kódy. Jedním z nich je 505 HTTP Version Not Supported. Tento stavový kód klientovi říká, že není podporovaná verze protokolu HTTP 2.0 a následně je vypsána v okně prohlížeče chybová stránka s číslem a popisem chyby. Pokud přijde požadavek, ve kterém bude název webu shodný s názvem webu, který je uzamčen, posílá webový server klientovi stavový kód 423 Locked a následně posílá chybovou stránku s číslem stavového kódu a popiskem chyby.

Následující kód znázorňuje posílání stavového kódu 404 Not found klientovi společně s error page. Na tomto principu funguje stavový kód 423 a 505.

```
try {
    BufferedWriter out = new BufferedWriter(new OutputStreamWriter(clientSocket.getOutputStream(), "UTF-8"));
    //otevření výstupního proudu klient socketu
    out.write("HTTP/1.1 404 Not Found\r\n"); //zápis stavového kódu do klient socketu
    out.write("Content-Type: text/html\r\n\r\n"); //zápis, o jaký jde soubor

    File file = new File(config.getError404()); //uložení html chybové stránky do proměnné
    BufferedReader fw = new BufferedReader(new FileReader(file)); //vytvoření proměnné pro zápis klient socketu
    String line;
    while ((line = fw.readLine()) != null) {
        out.write(line); //postupné zapisování chybové stránky do klient socketu
        out.newLine(); //nový řádek
    }
    out.flush(); //vyprázdnění proudu
    out.close(); //zavření výstupního proudu
    in.close(); //zavření vstupního proudu
    server.numOfRunningThreads.decrementAndGet(); //počítadlo - snížení běžících vláken
    server.numOfThreads.incrementAndGet(); //počítadlo - zvýšení počtu vyřízených požadavků
    server.vytizeni(); //grafické znázornění využití serveru

} catch (IOException ex) {
    System.out.println("error page dont exists");
}
```

4.3 Testování serveru

Byly provedeny testy, na jaké úrovní je webový server funkční. Webserver dokáže vyřizovat požadavky: hmtl, css, js, png, jpg, gif, bmp, mp4. Dále je možné přidávat libovolný počet webů do *C:/www/nazevWebu*. Navíc při nevyplnění názvu požadovaného webu v adresní řádce webového prohlížeče je uživatel přesměrován na domovský web *home*.

Testování na webhosting

Bylo vytvořeno 5 webových stránek, na které bylo z webového prohlížeče přistupováno, veškerý přístup a zobrazení webových stránek funguje.

Testování více přenosů

Byly vytvořeny stránky, na kterých je několik obrázků a Javascript, úspěšně bylo vše z webového serveru přenášeno a funkční.

Low orbital ion cannon

Byl použit software LOIC a puštěn s různým počtem vláken, který posílal požadavky na vlastní webserver s maximálním použitím dvaceti vláken. Následující tabulka zobrazuje počet nevyřízených požadavků tohoto testu.

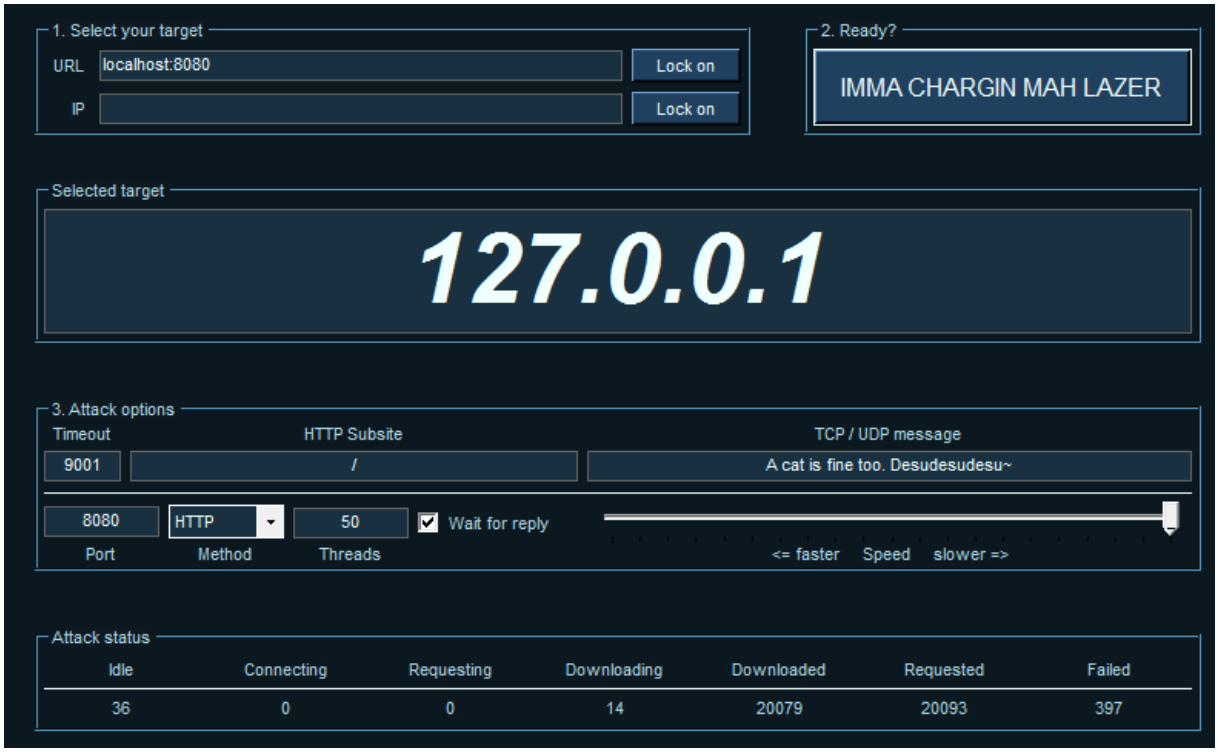
Tabulka 2 - Počet nevyřízených požadavků při 20 vláknech webového serveru a *n* vláknech softwaru LOIC

Počet zaslaných požadavků	Počet vláken softwaru LOIC	Počet nevyřízených požadavků
20046	10	10
20099	20	40
20073	21	146
20044	30	223
20034	50	394
20001	100	1133
20136	200	2256
-	250	Pád aplikace LOIC

Dále byl testován webový server při spuštění LOIC při 50 vláknech a různým maximálním počtem vláken webového serveru.

Tabulka 3 - Počet nevyřízených požadavků při *n* vláknech webového serveru a 50 vláknech softwaru LOIC

Počet zaslaných požadavků	Maximální počet vláken webového serveru	Počet nevyřízených požadavků
20018	1	960
20028	5	758
20051	10	692
20034	20	394



Obrázek 15 - Menu softwaru LOIC, nastaveno pro 20 vláken webového serveru a 50 vláken softwaru LOIC[21]

[21] Zdroj: autor

5 ZÁVĚR

Pro zpracování své bakalářské práce jsem použil znalosti z předmětů: Správa webserveru, Návrh a tvorba webových stránek, Datové struktury, Pokročilé programování v Javě, Algoritmizace a Počítačové sítě. Díky spojení znalostí ze všech těchto předmětů a patřičné píli se mi podařilo implementovat vlastní funkční webový server, který korektně pracuje s protokolem http, poskytuje webové stránky a je relativně odolný proti DoS útokům. Cíl bakalářské práce jsem tím splnil.

Problematika serveru, zejména webserveru, byla popsána a vysvětlena s důrazem na klíčové technologie a protokoly. Věřím, že popsání serveru, různých typů serverů, principů a využití webového serveru jsem dostatečně představil. Obohatil jsem se o mnoho informací a technik, které jsou každodenní záležitostí moderního IT světa. Doufám, že mé poznatky a informace obohatí nejen mě, ale také další čtenáře.

Vytvořený software, tedy aplikace (webserver), může posloužit jako dobré znázornění principů komunikace mezi uživatelem (klientem) a webovým serverem, který poskytuje webové stránky uživateli. Uživatel si může zkusit vylepšit, přepsat zdrojový kód, změnit konfiguraci souborů, aby si lépe vyjasnil funkčnost programu.

Ručně psaný webový server je funkční a velice dobře dokáže vyřizovat požadavky od klienta (webového prohlížeče). Nicméně dalo by se ještě o něco více zdokonalit aplikaci například lepším grafickým uživatelským rozhraním, možností přehlednějšího výpisů logů, lépe zapsaným konfiguračním souborem nebo jinou syntaxí kódu. Ovšem představu o těchto funkcionalitách má každý rozdílnou.

Nějaký čas mi trvalo zhotovit takto psaný webový server. Zdrojů ohledně napsání webového serveru jsem mnoho nenašel. Ovšem z určitých kousků vzorových příkladů se webový server napsat dal. Jsem přesvědčen, že jsem zadání z velké části splnil, jediné, v čem bych viděl problém, by bylo to, že jsem nedokázal použít více stavových kódů. Občas jsem měl problém s rozdělováním řetězců a prací s cestami k webovým stránkám. Dlouhou dobu mi nešel odeslat obrázek klientovi ve formátu png, poté jsem zjistil, že mám opět pouze chybu v rozdělení řetězce. Tyto nepřesnosti jsem časem vyladil a zprovoznil dobře fungující webový server.

6 POUŽITÁ LITERATURA

[1] ALBERT, Saul. How to share large amounts of research video data with Syncthing. *Saul Albert* [online]. 2016-07-21 [cit. 2017-03-06]. Dostupné z: http://saulalbert.net/blog/how-to-share-large-amounts-of-research-video-data-with-syncthing/

[2] Anonymní proxy server. *Správa sítě* [online]. c2016 [cit. 2017-03-06]. Dostupné z: http://www.sprava-site.eu/anonymni-proxy-server/

[3] Aplikační server. *Management Mania* [online]. c2011-2016 [cit. 2017-03-06]. ISSN 2327-3658. Dostupné z: https://managementmania.com/cs/aplikacni-server-aps

[4] BUCHTA, Martin. K čemu je DLNA a jak sdílet video a foto v celé domácnosti. *Buchtič blog* [online]. c2005-2017 [cit. 2017-03-06]. Dostupné z: https://blog.buchtic.net/jak-na-dlna-server-ve-vasem-obyvaku-a-k-cemu-vlastne-je/#.WLUyyTs1-70

[5] Co je cloud computing? Průvodce pro začátečníky. *Microsoft Azure* [online]. c2017 [cit. 2017-03-20]. Dostupné z: https://azure.microsoft.com/cs-cz/overview/what-is-cloud-computing/

[6] Co je dedikovaný server? *Hosting WEDOS* [online]. c2017 [cit. 2017-03-06]. Dostupné z: https://hosting.wedos.com/cs/dedicated/co-je.html

[7] Co je proxy server? *Správa sítě* [online]. c2016 [cit. 2017-03-06]. Dostupné z: http://www.sprava-site.eu/proxy-server/

[8] Co je to server? *BEST hosting* [online]. c2017 [cit. 2017-03-06]. Dostupné z: https://best-hosting.cz/cs/napoveda/co-je-to-server

[9] Co jsou redakční systémy? *MIREC* [online]. c2004-2016 [cit. 2017-03-06]. Dostupné z: http://www.mirec.cz/cs/web-redakcnisystemy/

[10] ČÍŽEK, Jakub. HTTP/2 je hotový. Nahradí současný protokol WWW z konce 20. století. *Živě* [online]. 2015-02-18 [cit. 2017-03-06]. ISSN 1213-8991. Dostupné z: http://www.zive.cz/bleskovky/http2-je-hotovy-nahradi-soucasny-protokol-www-z-konce-20-stoleti/sc-4-a-177215/default.aspx

[11] Databázový server. *Správa sítě* [online]. c2016 [cit. 2017-03-06]. Dostupné z: http://www.sprava-site.eu/databazovy-server/

[12] Elektronická pošta. *Za školou se školou* [online]. 2004-12-02 [cit. 2017-03-06]. Dostupné z: http://skola.amoskadan.cz/s_pp/s_pp_pi/pi6.htm

[13] EVANS, Keith. How Does a File Server Work? *EHow* [online]. c1999-2017 [cit. 2017-03-06]. Dostupné z: http://www.ehow.com/how-does_4761418_file-server-work.html

[14] FARANA, Radim. *Databázové systémy* [online]. Ostrava, 1995 [cit. 2017-03-06]. Dostupné z: http://books.fs.vsb.cz/dbacc20/dbacc01.htm. Technická univerzita Ostrava.

[15] HEROUT, Tomáš. Co jsou to dynamické webové stránky. *Help Mark* [online]. 2012-01-06 [cit. 2017-03-06]. Dostupné z: http://www.helpmark.cz/slovnikpojmu/32-dynamicke-webove-stranky

[16] Hyper Text Transfer Protocol. *Katedra Informatiky* [online]. 2003? [cit. 2017-03-17]. Dostupné z: http://www.cs.vsb.cz/grygarek/kotasek/htp04.htm

[17] Hyper Text Transfer Protocol. *Katedra Informatiky* [online]. 2003? [cit. 2017-03-17]. Dostupné z: http://www.cs.vsb.cz/grygarek/kotasek/htp05.htm

[18] Hyper Text Transfer Protocol. *Katedra Informatiky* [online]. 2003? [cit. 2017-03-17]. Dostupné z: http://www.cs.vsb.cz/grygarek/kotasek/indexhtp.htm

[19] Hypertext Transfer Protocol. *Wikipedie* [online]. 2016-11-28 [cit. 2017-03-06]. Dostupné z: https://cs.wikipedia.org/wiki/Hypertext_Transfer_Protocol

[20] Jak funguje DNS. *Cz.nic* [online]. c2015 [cit. 2017-03-06]. Dostupné z: http://www.jakfungujedns.cz/

[21] Jak funguje FTP. *Československá poradna* [online]. 2006-09-30 [cit. 2017-03-06]. Dostupné z: http://pc.poradna.net/a/view/307878-jak-funguje-ftp

[22] Jak se tiskový server funguje? *Myartve.net* [online]. c2017 [cit. 2017-03-06]. Dostupné z: http://www.myartve.net/jak-se-tiskovy-server-funguje/

[23] Jaký je rozdíl mezi POP3 a IMAP? *ONEbit hosting* [online]. c2017 [cit. 2017-03-06]. Dostupné z: https://www.onehelp.cz/onebit/kb/jaky-je-rozdil-mezi-pop3-a-imap

[24] Java Server-Side Programming: Java Servlets. *Programming notes* [online]. 2012 [cit. 2017-03-06]. Dostupné z: http://www.ntu.edu.sg/home/ehchua/programming/java/javaservlets.html

[25] JENKOV, Jakob. Multithreaded Server in Java. *Jenkov* [online]. 2014-10-31 [cit. 2017-03-06]. Dostupné z: http://tutorials.jenkov.com/java-multithreaded-servers/multithreaded-server.html

[26] K čemu slouží Aplikační Servery? *Servery* [online]. c2017 [cit. 2017-03-20]. Dostupné z: https://servery.cz/jak-na-to/k-cemu-slouzi-aplikacni-servery-66/

[27] Klient-server. *Počítačové sítě* [online]. c2010 [cit. 2017-03-06]. Dostupné z: http://site-pc.webnode.cz/klient-server/

[28] KOSEK, Jiří. Webový server. *HTML guru* [online]. c1997-2014 [cit. 2017-03-17]. Dostupné z: http://htmlguru.cz/vystaveni-webovy-server.html

[29] LISKA, Richard. S*íťové služby* [online]. 2016-03-27 [cit. 2017-03-06]. Dostupné z: http://www-troja.fjfi.cvut.cz/~liska/unix/node17.html

[30] Mailserver - Jak to celé funguje. *SPM's blog* [online]. c2008-2016 [cit. 2017-03-06]. Dostupné z: http://www.spamik.cz/howto/mailserver/info

[31] MAXON License Server. *Cinema4D* [online]. c2001-2017 [cit. 2017-03-06]. Dostupné z: http://www.cinema4d.cz/produkty/maxon/license-server/

[32] Návod: NAS server – šikovná hračka, kterou zvládne zapojit a nastavit i začátečník. *CZC.CZ* [online]. 2014-09-08 [cit. 2017-03-06]. Dostupné z: https://www.czc.cz/navod-nas-server-sikovna-hracka-kterou-zvladne-zapojit-a-nastavit-i-zacatecnik/clanek

[33] NECKÁŘ, Jan. Java (22) - Vlákna. *Algoritmy* [online]. c2016 [cit. 2017-03-06]. Dostupné z: https://www.algoritmy.net/article/39287/Vlakna-22

[34] NOVÁK, Petr. *Mapové servery* [online]. Ústí nad Labem, 2010 [cit. 2017-03-06]. Dostupné z: http://gis.fzp.ujep.cz/files/Prednaska11_1GIS2-Mapove_servery.pdf. Univerzita J. E. Purkyně.

[35] Oficiální server. *Wikipedie* [online]. 2017-01-20 [cit. 2017-03-06]. Dostupné z: https://cs.wikipedia.org/wiki/Ofici%C3%A1ln%C3%AD_server

[36] Peer-to-peer sítě: Jak fungují a kde je problém. *Bezpečně online.cz* [online]. 2006? [cit. 2017-03-06]. Dostupné z: http://www.bezpecne-online.cz/surfuj-bezpecne/sosani-a-sdileni-dat/peer-to-peer-site-jak-funguji-a-kde-je-problem.html

[37] Počítačové sítě - Protokoly TCP/IP. *Sítě* [online]. 2001? [cit. 2017-03-17]. Dostupné z: http://site.the.cz/index.php?id=3

[38] Port. *Správa sítě* [online]. c2016 [cit. 2017-03-06]. Dostupné z: http://www.sprava-site.eu/port/

[39] Red Hat Enterprise Linux. *Wikipedie* [online]. 2016-12-23 [cit. 2017-03-06]. Dostupné z: https://cs.wikipedia.org/wiki/Red_Hat_Enterprise_Linux

[40] Server. *Wikipedie* [online]. 2016-12-06 [cit. 2017-03-06]. Dostupné z: https://cs.wikipedia.org/wiki/Server

[41] Statická webová stránka. *Wikipedie* [online]. 2013-04-05 [cit. 2017-03-06]. Dostupné z: https://cs.wikipedia.org/wiki/Statick%C3%A1_webov%C3%A1_str%C3%A1nka

[42] Stavíme server. *WSEOHW.NET* [online]. c2006 [cit. 2017-03-06]. Dostupné z: http://vseohw.net/clanky/slozeni/stavime-server-cast1-preview

[43] Systém řízení báze dat. *Wikipedie* [online]. 2016-10-03 [cit. 2017-03-06]. Dostupné z: https://cs.wikipedia.org/wiki/Syst%C3%A9m_%C5%99%C3%ADzen%C3%AD_b%C3%A1ze_dat

[44] TCP/IP. *Wikipedie* [online]. 2017-03-04 [cit. 2017-03-06]. Dostupné z: https://cs.wikipedia.org/wiki/TCP/IP

[45] TP-LINK TL-PS110U. *MICRONET COMPUTERS* [online]. c2017 [cit. 2017-03-20]. Dostupné z: https://www.mironet.cz/tplink-tlps110u-tiskovy-server-1x-rj45-1x-usb-20+dp131514/?gclid=CKv4k7305dICFYk_Gwod5GoLkg#4910471

[46] VALÁŠEK, Michal. Pohled do hlubin webserverovy duše (aneb jak fungují HTTP moduly a handlery). *Aspnet* [online]. 2005-01-10 [cit. 2017-03-06]. ISSN 1801-9447. Dostupné z: http://www.aspnet.cz/articles/10-pohled-do-hlubin-webserverovy-duse-aneb-jak-funguji-http-moduly-a-handlery

[47] Webhosting, VPS nebo dedikovaný server? *Hosting WEDOS* [online]. c2017 [cit. 2017-03-06]. Dostupné z: https://hosting.wedos.com/cs/srovnani-hosting.html

[48] Webový server. *Wikipedie* [online]. 2016-11-25 [cit. 2017-03-06]. Dostupné z: https://cs.wikipedia.org/wiki/Webov%C3%BD_server

[49] What are all the possible values for HTTP “Content-Type” header? *Stack Overflow* [online]. c2017 [cit. 2017-03-17]. Dostupné z: http://stackoverflow.com/questions/23714383/what-are-all-the-possible-values-for-http-content-type-header

[50] ZAKHOUR, Sharon. *Java 6: výukový kurz.* Brno: Computer Press, 2007. ISBN 978-80-251-1575-6.

[51] Základní klasifikace serveru. *SlidePlayer* [online]. c2017 [cit. 2017-03-06]. Dostupné z: http://slideplayer.cz/slide/2487400/

7 PŘÍLOHY

Příloha A – *KorecekM_Webserver_JB_2cast_2017.zip* 58

Příloha B – *KorecekM_Webserver_JB_3cast_2017.zip* 59

Příloha C – *Návod k používání webového serveru* 60

Příloha A – *KorecekM_Webserver_JB_2cast_2017.zip*

Na přiloženém disku se nachází soubor *KorecekM_Webserver_JB_2cast_2017.zip*, kde je ručně psaný webový server v jazyce Java.

Příloha B – *KorecekM_Webserver_JB_3cast_2017.zip*

Na přiloženém disku se nachází soubor *KorecekM_Webserver_JB_3cast_2017.zip*. Po rozbalení souboru nakopírujte složku www na disk C. Z této složky načítá webový server statickou webovou stránku a další podporované webové stránky.

Příloha C – *Návod k používání webového serveru*

Po spuštění aplikace se otevře hlavní okno. Toto okno obsahuje několik výpisů: stav serveru (vypnuto, zapnuto), počet běžících vláken, počet vyřízených požadavků. Aplikace obsahuje tlačítka pro zapnutí serveru, vypnutí a restart serveru. Dalšími tlačítky jsou konfigurační tlačítka pro zobrazení konfiguračního souboru v textovém poli, zálohování konfiguračního souboru a smazání konfiguračního souboru. Textové pole slouží k vypisování logů. Dále aplikace disponuje tlačítky pro zobrazení logových hlášek, zálohování logových hlášek a smazání logových hlášek. Je možné pomocí comboboxu nastavit úroveň logování.

Při spuštění aplikace je server defaultně vypnutý, všechna tlačítka jsou zamrzlá s výjimkou tlačítka Start, které slouží ke spuštění serveru. Po kliknutí na tlačítko Spustit se načte konfigurace ze souboru. Pokud není konfigurační soubor nalezen v defaultním umístění (složka s projektem), tak se aplikace dotáže, zda se má pokusit obnovit konfigurační soubor ze zálohy. Pokud je kliknuto na tlačítko Yes, otevře se dialog pro výběr souboru, zde je možné dohledat konfigurační zálohu a aplikace se následně pokusí obnovit konfigurační soubor ze zálohy. Pokud se konfigurace povedla, server zahlásí, že byla úspěšně načtena konfigurace ze zálohy, a následně se spustí server. Pokud se obnova nepovede, server zahlásí chybu a nespustí se. Pokud stiskneme tlačítko No, spuštění serveru se zastaví.

Poté co se server úspěšně spustí, tlačítko Spustit se zamrazí a ostatní tlačítka začnou být funkční. Také se v textovém poli objeví první logy: server byl spuštěn, na jakém portu byl spuštěn a že byla nastavena hladina logování (načte se z konfiguračního souboru při inicializaci serveru).

Nyní je možné volat server z prohlížeče, pokud nedojde k problému, tak bude server odpovídat na požadavky z webového prohlížeče (klient).

Logování funguje tak, že každá interakce uživatele (stisk tlačítka apod.) se zapíše do logu. Dále se do logu zapisují přístupy k souborům. Například prohlížeč požádá o *index.html*, do logu se zapíše, že bylo požádáno o daný soubor s hladinou *WARN*. Logy mají několik hladin, kterými jsou: *INFO* (informace), *WARN* (varování), *ERROR* (chyba), *SYSTEM* (systém), *ALL* (všechny hladiny), *NONE* (žádná hladina). Defaultně se používá hladina uvedená v konfiguračním souboru, ale lze ji změnit pomocí checkboxu. Pokud je nastavena hladina *SYSTEM*, tak jsou vypisovány pouze hladiny *WARN* a *INFO*, hladina *ERROR* vypisována nebude. Logy se ukládají do souboru pouze v případě vypnutí serveru, nebo při kliknutí na tlačítko záloha. Soubor s logy se ukládá do složky, kterou určuje cesta v konfiguračním souboru. Záloha logů

se ukládá do kořenového adresáře projektu, jinak se veškeré logy drží jen v paměti. Po kliknutí na tlačítko Zobrazit log se otevře textový soubor (pokud existuje). Otevírá se v editoru, který je nastaven jako výchozí pro otevírání txt souborů v operačním systému.

Konfigurační soubor je možno zobrazit kliknutím na tlačítko Zobrazit konfiguraci. Otevírá se stejně jako logy v defaultně nastaveném editoru operačního systému. Zde je možno editovat konfigurační soubor. Změny se projeví při následujícím startu systému. Konfigurační soubor je možno smazat. Smaže se konfigurační soubor, který se nachází ve složce s projektem webserveru. Nedoporučuje se mazat, pokud není vytvořena záloha. Zálohování se provede kliknutím na tlačítko Záloha conf. Vytvoří se kopie konfiguračního souboru. Cesta, kam se ukládá kopie, je uvedena v konfiguračním souboru.