UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2020 Pavel Křivda

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Multihostingový systém na platformě Docker

Pavel Křivda

Bakalářská práce

2020

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2018/2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Pavel Křivda**
Osobní číslo:	**I16316**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Multihostingový systém na platformě Docker**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem bakalářské práce je vytvořit multihostingový systém na platformě Docker. V rámci hostingu by měly být dostupné minimálně PHP aplikace s využitím MySQL (Maria) a MongoDB databází a aplikace typu Java EE. V teoretické části práce bude provedeno základní seznámení s platformou Docker (Compose/Swarm) a dále bude popsán navrhovaný systém a jednotlivé komponenty, které budou využity. Multihosting bude obsahovat minimálně http(s) server (uživatelé rozlišeni podle subdomén), podporu PHP skriptů a propojení s databázemi MySQL (Maria) a MongoDB. V rámci hostingu bude možné rovněž provozovat aplikace typu Java EE. Správa uživatelského dat bude realizována pomocí ftp(s). V praktické části budou vytvořeny konfigurační soubory pro vytvoření multihostingového prostředí. Dále bude vytvořena administrační aplikace (PHP, Java EE nebo jiný podporovaný jazyk v hostingu), ze které bude možné spravovat uživatelský účet (administrace hesel do databází, deploy JEE aplikací, ...).

Rozsah pracovní zprávy: **40 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

MCKENDRICK, Russ a Scott GALLAGHER Mastering Docker 2nd. Birmingham, UK: Packt Publishing, 2017. ISBN 9781787286207
SILVA, Steve Web Server Administration Course Technology, 2007. ISBN 9781423903239
BRITTAIN, Jason a Ian F. DARWIN Tomcat: The Definitive Guide 2nd Edition. O'Reilly Media, 2007. ISBN 9780596554941

Vedoucí bakalářské práce: **Ing. Roman Diviš**
Katedra softwarových technologií

Datum zadání bakalářské práce: **31. října 2018**
Termín odevzdání bakalářské práce: **12. května 2019**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Lukáš Čegan, Ph.D.
pověřený vedením katedry

V Pardubicích dne 14. prosince 2018

Prohlašuji:

Tuto práci jsem 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.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 5. 8. 2020

Pavel Křivda

PODĚKOVÁNÍ

Chtěl bych poděkovat panu Ing. Romanu Divišovi, za odborné vedení práce a cenné rady, které mi pomohly tuto práci zkompletovat. Dále bych rád poděkoval mojí rodině, která mě neúnavně podporuje po celou dobu mých studií.

ANOTACE

Práce je věnována konfiguraci multihostingového serveru na platformě Docker. Pro každého zákazníka je vytvořena subdoména s podporou nasazení Java EE a PHP aplikací které lze připojit k databázi. Z databází je na výběr relační databáze MySQL a noSQL databáze MongoDB. Pro správu uživatelských dat je dostupný FTP server.

V práci je popsána platforma Docker, na které je vytvořen multihostingový systém. Ten se skládá z technologií Apache, PHP, MongoDB a MySQL. Tyto služby jsou dále teoreticky popsány. V praktické části je implementována konfigurace pro Docker Compose a administrační aplikace pro multihosting.

KLÍČOVÁ SLOVA

Webhosting, Hosting, Multihosting, Docker

TITLE

Multihosting system based on Docker platform

ANNOTATION

Thesis is focused on creation and configuration of multihosting server on Docker platform. For every custumer there will be created one subdomain with support for running Java EE and PHP application, which are able to connect to database. From databases there is possibility to choose relational database MySQL and NoSQL database MongoDB. For user data management there is available FTP server.

Thesis describes Docker platform on top of which is multihosting system created. System consists of Apache, PHP, MongoDB and MySQL technologies. These services are then described theoretically. Practical part implements configurations for Docker Compose and administration application for multihosting.

KEYWORDS

Webhosting, Hosting, Multihosting, Docker

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK

API	Application Programming Interface
ASP	Active Server Pages
BSON	Binary JSON
CSS	Cascading Style Sheets
FTP	File Transport Protocol
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
IoT	Internet of Things
JDBC	Java Database Connectivity
JEE	Java Enterprise Edition
JSON	JavaScript Object Notation
JSP	JavaServer Pages
LXC	Linux Containers
MVC	Model View Controller
MySQL	My Structured Query Language
PHP	Hypertext Preprocessor
SQL	Structured Query Language
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
WWW	World Wide Web

ÚVOD

Počet uživatelů internetu neustále roste a jejich počet se nyní pohybuje až okolo 3,5 miliardy [1]. V souvislosti s tím bylo na internetu vytvořeno 1,3 miliardy webových stránek zřízené samotnými uživateli, nebo podniky [2]. Aby byli webové stránky dostupné na internetu, je nutné je vystavit na webovém serveru. Tyto servery nabízejí poskytovatelé hostingu, kteří webové stránky následně provozují.

Pro poskytovatele hostingu bývají klientské výpočetní nároky velmi malé, ale požadují vysokou spolehlivost a dostupnost webového obsahu [3]. Z tohoto důvodu je pronájem jednoho výkonného serveru několika klientům ideálním řešením, kdy se tento výkon mezi klienty rozdělí. Tento typ webhostingu je nazýván multihosting.

Většina webových stránek potřebuje pro plnohodnotný provoz několik služeb. Například provoz e-shopu vyžaduje kromě samotného úložného prostoru nutnost zajistit nepřetržitý provoz databází a webového serveru. Všechny tyto služby nabízí poskytovatelé webhostingu svým klientům v podobě řešení jednoduchého na správu.

Poskytovatel hostingu neustále řeší požadavky jako je správa dat a uživatelů a k tomu musí zajistit bezpečnou izolaci mezi jednotlivými klienty, přidělovat jim hardware a rozlišovat kde jsou uložena data jednotlivých uživatelů.

Tato bakalářská práce se zabývá vytvořením komplexního systému pro správu multihostingového serveru, který usnadňuje práci poskytovatele. Pro vytvoření multihostingového systému je nutné si nejdříve nastudovat danou problematiku, vybrat vhodné aplikace (webový, databázový, FTP server) a navrhnout daný systém tak, aby jej bylo možné spravovat pomocí webové aplikace.

1 TEORETICKÝ ÚVOD

V této kapitole jsou popsané základní technologie, které jsou nezbytné pro prohlížení internetových stránek.

1.1 WWW

Základem WWW je hypertext, což je systém pro vzájemné propojení dokumentů, které jsou přístupné pomocí internetu. Díky tomu je možné pomocí webového prohlížeče zobrazit jednotlivé webové stránky, text, obrázky nebo jiná multimédia na které webová stránka odkazuje. K navigaci ve službě WWW slouží hypertextové odkazy, které přesměrují uživatele na jiný text nebo obecně na jiný obsah. Hypertextové odkazy mohou mít libovolnou grafickou podobu, může se jednat např. o tlačítka, obrázky nebo jiné interaktivní prvky. [4]

1.2 HTTP

HTTP je zkratka od Hypertext Transfer Protocol, který byl původně určen k výměně hypertextových dokumentů mezi serverem a prohlížečem. Současná verze HTTP obvykle využívá protokol TCP na portu 80 a dokáže přenášet soubory.

HTTP je bezstavový protokol fungující na principu dotazu a odpovědi. To je důvodem proč kupříkladu nelze uložit obsah košíku v internetovém obchodě. Tento problém je možné vyřešit různými postupy, např. využitím cookies. [5]

1.3 HTTPS

HTTPS je zkratkou Hyper Text Transfer Protocol Secure, který používá protokol TCP na portu 443 pro komunikaci. HTTPS umožňuje vytvořit zabezpečené obousměrně šifrované spojení mezi serverem a prohlížečem uživatele. Toto opatření pomáhá chránit citlivé informace před odcizením nebo změnou. Šifrování komunikace se provedeno pomocí kombinace protokolu TLS (dříve SSL) a protokolu HTTP. [6]

1.4 Webhosting

Webhosting se někdy zkráceně nazývá hosting a znamená pronájem uložiště pro webové stránky. Na ně lze umístit např.: e-shop, diskuzní fórum či redakční systém, ve kterém se webové stránky vytvářejí a spravují. Na webhostingu jsou nahrané veškeré obrázky a data webové stránky ke které dostane nájemce webhostingu neomezený přístup a může je libovolně přidávat, mazat nebo upravovat. Dostupná kapacita pro každého zákazníka je buď neomezená nebo omezena v řádu gigabajtů. [7]

1.5 Multihosting

Multihosting je spojení více webhostingů dohromady, které umožňují nahrát více webových stránek pod různými doménami druhého řádu. Proto lze mít pohromadě např. prezentaci firmy a e-shop. Navíc zákazník u multihostingu neplatí za každý web zvlášť, ale jsou všechny sdružené pod jedním uživatelským účtem. Díky tomu získáváme jednodušší správu jednotlivých webů a v budoucnu snadnější propojení více webů dohromady. [8]

2 PROGRAMOVÉ VYBAVENÍ

V této kapitole je uveden popis jednotlivých služeb nezbytných pro fungování multihostingového serveru.

2.1 Operační systém

Pro multihostingový server je využit Linuxový operační systém Ubuntu, který je poskytován zdarma k použití s možností upravovat některé jeho zdrojové soubory. Ubuntu je založeno na distribuci Debian, ze které převzal velké množství softwaru. Výhodou Ubuntu je vydávání verzí s dlouhodobou podporou s označením LTS (Long Term Support) každé dva roky. Podpora každé verze činí 5 let po kterou uživatel dostává jak bezpečnostní, tak systémové aktualizace. [9]

2.2 Adminer

Adminer (dříve phpMinAdmin) je plnohodnotný nástroj pro správu databází napsaný v jazyce PHP a šířený pod licencí Apache License. Skládá z jediného souboru připraveného k nasazení na cílový server. Přitom velikost souboru činí necelých 500 kB a umožňuje připojení k mnoha databázovým systémům mezi které patří: MySQL, PostgreSQL, SQLite, MS SQL, Oracle, Firebird, SimpleDB, Elasticsearch a MongoDB. [10]

Vývojáři Admineru staví bezpečnost na první místo, a proto se nelze připojit k databázím, které nemají nastavené heslo. Dále je omezen počet pokusů při přihlášení tak, aby byla zajištěna ochrana před útoky hrubou silou. [10]

2.3 FTP Server

FTP je zkratka File Transport Protocol používaný pro přenos souborů mezi počítači pomocí počítačové sítě. Pro přenos dat využívá protokol TCP (port 20 pro přenos dat a 21 pro přenos příkazů) a je nezávislý na operačním systému. K práci s FTP je třeba využít nějaký program např. webový prohlížeč, či speciální FTP klient. FTP klient umožňuje provádět operace se soubory a adresáři, např.: kopírování, mazání, editace, přejmenování, nahrání dat na server a jejich stažení ze serveru do počítače. [11]

2.4 MySQL Server

MySQL server je open source databázový systém relačního typu, který umožňuje pomocí SQL příkazů provádět: ukládání, zpracování, získávání a správu dat v databázi. Každá databáze obsahuje tabulky, kde se každá tabulka skládá ze sloupců a řádků, kde v každém řádku jsou záznamy předem určeného datového typu. Pro sítovou komunikaci využívá protokol TCP a standardně port 3306. [12]

2.5 MongoDB

MongoDB je v současnosti nejpoužívanější open source multiplatformní dokumentovou databází, která patří mezi NoSQL databáze. K ukládání dat používá dokumenty podobné formátu JSON (MongoDB formát se nazývá BSON) a dynamické databázové schéma, které umožnuje aplikacím jednodušší a rychlejší správu dat. Pro komunikaci s klienty využívá protokol TCP na portu 27017. [13]

2.6 PHP

PHP je programovací jazyk, který se vykonává na straně serveru (kde jsou uloženy veškeré zdrojové kódy webových stránek), kde se nejprve provede PHP script na serveru a následně odešle prohlížeči pouze výsledek, server např. nejprve zpracuje matematický výraz, který posléze předá prohlížeči s výslednou hodnotou (to je rozdíl oproti JavaScriptu, který hodnotu vypočítá přímo v prohlížeči). Soubory s PHP skripty mají nejčastěji příponu .php. Syntaxe jazyka je podobná programovacím jazykům z rodiny C, C++. [14]

2.7 Apache HTTP Server

Apache HTTP Server je multiplatformní webový server, který je dostupný zdarma pod licencí The Apache License Version 2.0 [15]. Navíc společně s PHP a MySQL patří Apache k trojici nejčastěji užívaných programů k vytváření dynamických internetových stránek a současně je nepoužívanějším webovým serverem na světě [16]. Apache je navíc modulární, proto jej lze snadno rozšiřovat zaváděním různých modulů.

2.8 Apache Tomcat Server

Apache Tomcat slouží k nasazení webových aplikací založených na JEE (Java Enterprise Edition). Apache Tomcat webový kontejner, který je dostupný zdarma pod licencí The Apache Licence 2.0 a je vyvíjen a udržován komunitou vývojářů pod záštitou Apache Software Foundation. [17]

2.8.1 Java EE (Java Enterprise Edition)

Java je objektově orientovaný programovací jazyk, který vyvinula firma Sun Microsystems, a nyní je spravována společností Oracle. V současnosti je Java jeden z nejpoužívanějších programovacích jazyků na světě. Java EE je součást platformy Java, která je určená pro vývoj a provoz podnikových aplikací a informačních systémů. [18]

Java EE zjednodušuje vývoj aplikací vytvořením standardizovaných, opakovaně použitelných modulárních komponent. Přístupem k vrstvě API zvládá mnoho aspektů programování automaticky, a tím umožňuje znovupoužití značného množství kódu pro další aplikace. [18]

Zatímco jsou aplikace Java EE hostovány na straně serveru, příklady klientů Java EE jsou: zařízení internet věcí (IoT), operační systém pro chytrý telefon, RESTful webová služba, standardní webová aplikace nebo microservices spuštěné v kontejneru Docker. [18]

2.8.2 Servlet

Servlet je software, který umožňuje webovému serveru zpracovávat dynamický webový obsah vyvinutý v programovacím jazyce Java pomocí protokolu HTTP. Servlety jsou alternativou k jiným technologiím pro tvorbu dynamického webového obsahu jako jsou např. PHP nebo ASP.NET. [19]

2.8.3 JSP (JavaServer Pages)

JavaServer Pages umožňuje programování na straně serveru a vytváření dynamických metod nezávislých na platformě. Tyto metody jsou využity při vývoji webových aplikací. Navíc má JSP přístup k celé řadě Java API, včetně JDBC API pro přístup k podnikovým databázím. [19]

2.9 Řídící aplikace

2.9.1 Volba Jazyka

Řídící aplikace je dle zadání webovou aplikací, pro kterou bylo nutné si zvolit vhodný programovací jazyk. Jako programovací jazyk je zvolen PHP, který byl probrán v rámci studia. Součástí práce je taktéž Apache HTTP server, na kterém je možné PHP provozovat.

2.9.2 Volba frameworků

Boostrap

Pro rychlejší vývoj webu je použit framework Bootstrap4, jedná se o volně stažitelnou sadu nástrojů pro tvorbu webu a webových aplikací. Obsahuje mnoho šablon založených na HTML a CSS, sloužící pro úpravu např. formulářů, tlačítek, a dalších komponent.

Twig

Twig je šablonový modul pro PHP, který používá mnoho open-source projektů jako Drupal8, Symfony, phpBB. Navíc je technologie Twig přívětivá jak k návrhářům, tak i vývojářům aplikací tím, že se drží zásad PHP a přidává funkce užitečné pro prostředí šablon. Šablony sestavuje do prostého optimalizovaného kódu PHP. Tím je režie ve srovnání s běžným kódem PHP snížena na minimum. Dále využívá režim karantény pro vyhodnocení nedůvěryhodného kódu šablony, a tím zvyšuje bezpečnost aplikace. [20]

Návrhový vzor MVC

MVC je oblíbený architektonický vzor původně určený pro vývoj desktopových aplikací, i když se dnes využívá převážně při vývoji webových aplikací. Proto se stal součástí mnoha populárních webových frameworků, mezi které patří např. Zend nebo Nette pro PHP a Ruby On Rails pro Ruby. [21]

Základní myšlenkou MVC architektury je oddělení logiky aplikace od grafického výstupu. Tímto oddělením je odstraněn problém tzv. "špagetového kódu", kdy se v jedné třídě (souboru) nachází logické operace a zároveň formátování výstupu. Příkladem může být získání dat z databáze a následné formátování získaných dat v podobě tabulky, kdy se veškerý kód nachází v jednom souboru. Tím se kód stane špatně udržovatelným a obtížně rozšiřitelným. [21]

Komponenty

Aplikace je rozdělena na tři komponenty, které se nazývají Model (modely), View (pohledy) a Controller (kontrolery), proto MVC [21]. Následuje popis jednotlivých komponent:

Model

Model obsahuje logiku celé aplikace a patří sem např.: výpočty, databázové dotazy, validace vstupů od uživatele a podobně. Model pouze získává a zpracovává data nikdy je nezobrazuje. [21]

View

View zprostředkovává zobrazení dat získaných z modelu. Jde o HTML šablonu, obsahující HTML stránku s tagy jazyka, umožňující do šablony vkládat např.: proměnné, provádět cykly a podmínky. Jednotlivé šablony lze vkládat do sebe, aby nedocházelo k duplicitě kódu. [21]

View tedy obsahuje minimální množství logiky, která je pro výpis nutná (např. kontrola, zda si uživatel vyplnil přihlašovací údaje před jejím odesláním na server). View se stará pouze o zobrazení dat uživateli a stejně jako Model vůbec neví, odkud zobrazovaná data obdržel. [21]

Controller

Controller odpovídá za řízení aplikační logiky a jedná jako koordinátor mezi pohledem a modelem. Controller přijímá vstup od uživatelů prostřednictvím pohledu, pak zpracovává data uživatele pomocí modelu a předává výsledky zpět pomocí pohledu. [21]

3 KONTEJNEROVÁ VIRTUALIZACE DOCKER

V této kapitole je popsáno, co je to virtualizace a jaké existují druhy virtualizací. Poté následuje základní seznámení s technologií Docker.

3.1 Virtualizace

Virtualizace umožňuje běh jednoho nebo více virtuálních strojů na fyzickém počítači. Proto se jeden počítač může zvenčí jevit jako několik navzájem nezávislých počítačů, které mohou mít rozdílné architektury, operační systémy nebo různý připojený virtuální či reálný hardware.

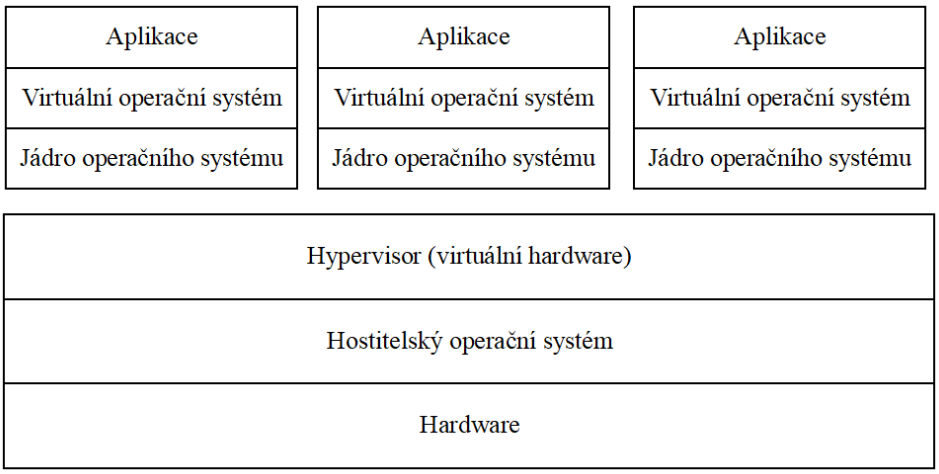
3.2 Druhy Virtualizace

3.2.1 Emulace

Emulátor provádí překlad strojových instrukcí emulovaného systému na strojové instrukce hostitelského stroje. Pro správný běh emulovaného systému je nutné emulovat procesor včetně registrů a veškerého hardware. Což umožní spustit operační systémy, které nejsou určeny pro architekturu hostitelského systému, čehož se využívá např. pro hraní starších herních titulů. [22]

3.2.2 Úplná virtualizace

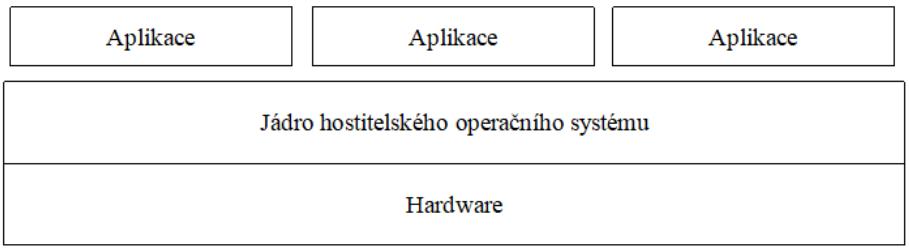
Úplná virtualizace vytváří kompletní virtuální hardware, pomocí kterého lze spouštět neupravené operační systémy izolovaně od hostitelského operačního systému. Nutnou podmínkou je, aby operační systémy byli stejné architektury, na které je spuštěn hostitelský počítač. Při úplné virtualizaci jednotlivé virtuální stroje nekomunikují přímo s fyzickým hardwarem, ale jednotlivé požadavky jsou předány virtuálním zařízením, které je následně předají těm fyzickým. [22]



Obrázek 1 – Schéma úplné virtualizace (zdroj: vlastní)

3.2.3 Virtualizace na úrovni jádra operačního systému, kontejnerová virtualizace

Virtualizované systémy běží na společném jádru hostitelského operačního systému. Tím dojde ke snížení velikosti virtuálních strojů, jelikož neobsahují jádro vlastní. Sdílené jádro umožní spouštět několik na sobě nezávislých a vzájemně izolovaných virtuálních systémů, pokud jsou stejné architektury jako hostitelský systém. Tato virtualizace ve srovnání s hostitelským systémem dosahuje téměř stejné efektivity v běhu systému. [22]



Obrázek 2 – Schéma virtualizace na úrovni jádra operačního systému (zdroj: vlastní)

3.2.4 Paravirtualizace

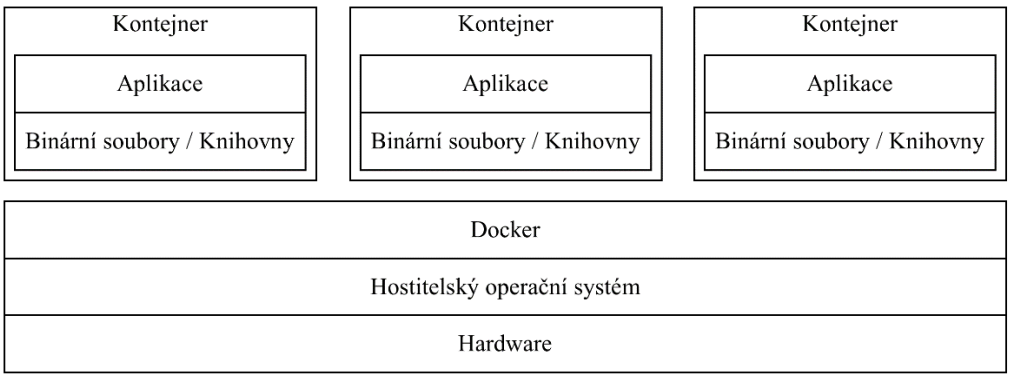
Virtuální stroj nevytváří kompletní virtuální hardware, ale nabízí speciální API upraveného jádra hostitelského systému, které umožní komunikaci virtualizovaného operačního systému s fyzickým hardware. Toto řešení sníží režii spojenou se vstupně-výstupními operacemi a způsobí, že aplikace i samotný operační systém běží nativně na procesoru a tím dosáhne lepšího výkonu oproti plné virtualizaci. [22]

3.2.5 Aplikační virtualizace

Aplikační virtualizace izoluje jednotlivé aplikace od hostitelského systému, tudíž dochází k téměř žádným ztrátám na výkonu. Izolace procesu umožňuje jednotlivé procesy přemisťovat mezi systémy, aniž by se měnilo jejich chování. [23]

3.3 Docker

Docker je nástroj určený k usnadnění vytváření, nasazení, správy a spouštění aplikací pomocí tzv. kontejnerů. Ty jsou vytvářeny pomocí kontejnerové virtualizace, která umožňuje snížit velikost kontejnerů o jádro systému. Pro nasazení aplikací v kontejnerech je vhodné, aby byly dodávány pouze se službami, které se buď nenacházejí na hostitelském počítači nebo nejsou spuštěny. Toto opatření významně zvyšuje výkon a snižuje velikost kontejneru. [24]



Obrázek 3 – Schema Docker kontejnerů (zdroj: vlastní)

Proto se nástroj Docker využívá od vývojářských stanic přes testovací a integrační prostředí až po prostředí určené pro produkci. Navíc lze Docker nasadit buď jako manuálně řízené řešení, nebo s využitím orchestračních technologií např.: Docker Swarm, Kubernetes nebo Apache Mesos. Podobně jako v případě linuxových RPM nebo DEB balíčků se aplikační Docker images ukládají do veřejného repozitáře (např. docker hub) nebo soukromého repozitáře. [24]

Navíc je Docker dostupný zdarma pod licencí The Apache 2.0, díky které může kdokoli přispět k vývoji projektu nebo si lze vytvořit vlastní verzi obohacenou o nové funkce [25]. Proto mohou vývojáři připravit optimalizované prostředí vyvíjené aplikace, a tak zajistit její větší spolehlivost a zjednodušit její nasazení. [24]

3.3.1 Historie Docker

Za touto technologií stojí společnost Docker Inc., která byla založena roku 2010. Roku 2013 prezentuje první verzi Dockeru na konferenci PyCon po které následovalo vydání pod licencí Apache Licence 2.0. V tomto roce Docker využíval technologii LXC (LinuX Container) pro běh kontejnerů. Tato technologie byla nahrazena o rok později s příchodem verze 0.9 která využívala API poskytované linuxovým jádrem a Docker byl přepsán pomocí programovacího jazyka Go. [24]

Po vydání první verze Dockeru došlo ke spolupráci společnosti Docker Inc. s dalšími technologickými společnostmi, mezi které patří např.: Red Hat, Amazon a IBM. Tyto společnosti následně provedli implementaci Dockeru do vlastních produktů. [24]

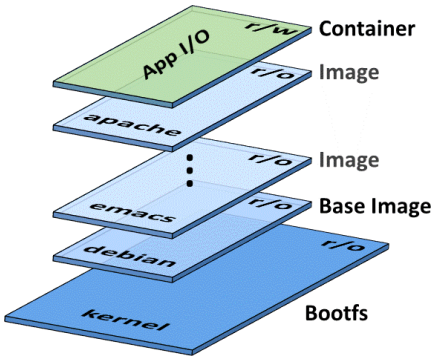
Docker Inc. následně oznámil práci na standardu nezávislém na operačním systému, a proto je dnes možné provozovat docker kontejnery jak na systémech založených na platformě Linux, tak i Windows. [24]

3.3.2 Docker kontejner

Kontejnery umožňují vývojářům zabalit aplikaci se všemi částmi, které jsou nezbytné k běhu aplikace, jako jsou např.: knihovny, konfigurační soubory a další závislosti. Následně lze z takto sestaveného kontejneru vytvořit Docker image, který lze odeslat jako jeden balíček. Proto si mohou být vývojáři jistí, že aplikace bude spuštěna na jakémkoli jiném počítači se stejným nastavením. [26]

3.3.3 Docker image

Docker image je vytvořen z řady vrstev, kde každá vrstva představuje instrukci v souboru Dockerfile. Jednotlivé vrstvy obsahují pouze sadu rozdílů od vrstvy před ní a jsou naskládány na sebe. Pak při vytvoření nového kontejneru z Docker image, dojde k přidání nové zapisovatelné vrstvy, která překryje nejvyšší z podkladových vrstev. Tato vrstva se často nazývá „container layer“ a zapisují do ní veškeré změny provedené v běžícím kontejneru, např.: zápis nových souborů, úprava existujících souborů a odstranění souborů. Ostatní vrstvy kromě té poslední jsou pouze pro čtení. [27]



Obrázek 4 – Schéma Docker image (zdroj: [28])

3.3.4 Dockerfile

Jedná se o textový soubor s instrukcemi pro vytvoření Docker image. Mezi základní instrukce patří: specifikace kontejneru (base image) obsahující operační systém, který bude sloužit jako základ výsledného kontejneru, příkazy pro instalaci nezbytného softwaru a kopírování dat z lokálního počítače, vybrané síťové porty sloužící ke komunikaci kontejneru s dalšími zařízeními atd. [29]

Ukázka souboru *Dockerfile*:

```
1.  FROM ubuntu:19.10
2.  COPY . /data_aplikace
3.  RUN make /data_aplikace
4.  CMD python / data_aplikace/aplikace.py
```

Tento soubor Dockerfile obsahuje čtyři příkazy, z nichž každý vytváří další vrstvu. Příkaz FROM vytvoří první vrstvu z Docker image s názvem ubuntu ve verzi 19.10. Následující příkaz COPY přidá všechny soubory z aktuálního adresáře, ve kterém se nachází soubor Dockerfile. Příkaz RUN spustí příkaz v příkazovém řádku a sestaví aplikaci pomocí programu make. Nakonec poslední vrstva určí, jaký příkaz se spustí při spuštění kontejneru. [29]

3.3.5 Docker compose

Docker compose je jednoduchý, ale výkonný nástroj, který je schopný spustit všechny potřebné kontejnery tak aby byl zajištěn bezproblémový chod aplikace. Nezbytná konfigurace probíhá pomocí mnoha pravidel deklarovaných v jednom konfiguračním souboru s názvem *docker-compose.yml* ve kterém lze specifikovat jednotlivé služby nezbytné pro chod aplikace. Tento soubor využívá formátování YAML, které je snadno čitelné a strojově optimalizované. Navíc musí každý konfigurační soubor specifikovat verzi formátu souboru a alespoň jednu službu. Dále zde mohou být specifikovány např. svazky pro permanentní uchování dat nebo nastavení síťování mezi službami. [30]

Jednotlivé fáze Docker Compose:

1. Definice prostředí aplikace pomocí *Dockerfile*.
2. Definice všech nezbytných služeb pro nasazení aplikace, tak aby byla zajištěna vzájemná komunikace služeb v izolovaném prostředí. Nezbytná konfigurace se provádí pomocí souboru docker-compose.yml.
3. Spuštění aplikace pomocí příkazu: *docker-compose up*

3.3.6 Docker Swarm

Docker Swarm je orchestrační nástroj pro kontejnery. Pomocí kterého mohou administrátoři spolu s vývojáři vytvořit a následně spravovat cluster skládající se z mnoha uzlů tak, jako by se jednalo o jediný virtuální systém. [31]

Proto se clustering stal důležitou funkcí pro kontejnerové technologie, jelikož vytváří kooperativní skupinu systémů (uzlů), které poskytují redundanci, díky které nedojde k výpadku služby při selhání jednoho nebo více uzlů. [31]

Navíc Docker Swarm využívá funkce k plánování, tak aby zajistil dostatek zdrojů pro distribuované kontejnery. Při spuštění technologie Swarm dojde k přiřazení kontejnerů do jednotlivých uzlů, po kterém dojde k automatické optimalizaci zdrojů tak, aby se kontejnery spouštěly na nejvhodnějších uzlech s dostatečnými zdroji. Pomocí plánování lze udržet nezbytný výkonu pro nasazenou aplikaci. [31]

4 MULTIHOSTING NA DOCKERU

V této kapitole jsou popsány jednotlivé výhody a nevýhody spojené s využitím nástroje Docker pro multihostingový systém.

4.1 Výhody Docker

4.1.1 Bezpečnost

Nástroj Docker zajistí spouštění aplikací v jednotlivých kontejnerech, tak aby byly od sebe z bezpečnostního hlediska odděleny. Proto žádný kontejner nemůže ovlivnit procesy probíhající uvnitř jiného kontejneru. [32]

4.1.2 Rozšiřitelnost

Nástroj Docker nabízí flexibilitu správcům systému, kteří si mohou snadno a rychle rozšířit stávající infrastrukturu služeb např. o další databázový nebo webový server. Toho lze snadno docílit pomocí serveru Docker Hub, na kterém je uloženo mnoho již vytvořených a přednastavených obrazů Docker obsahující širokou škálu služeb. Pak správci systému stačí vybrat potřebný image, který je třeba před jeho spuštěním nakonfigurovat tak, aby byl výsledný systém co nejbezpečnější a nejspolehlivější. [32]

4.1.3 Udržovatelnost

Docker je navržen tak, aby umožnil jednoduchou správu všech kontejnerů skrze jedno rozhraní. To je přístupné pomocí příkazového řádku, skrze který lze přistupovat k jednotlivým službám a v případě nutnosti se mezi nimi rychle přepínat. Navíc tento přístup umožňuje tvorbu skriptů pro automatizaci některých procesů. [32]

4.1.4 Testování

Pomocí technologie Docker mohou programátoři testovat vyvíjenou aplikaci ve stejném prostředí ve kterém bude aplikace provozována. Tím dojde k odstranění problémů souvisejících s rozdílným nastavením při vývoji a následným nasazení aplikací na produkčních prostředí. [32]

4.1.5 Přenositelnost

Nástroj Docker nabízí přenést konfigurační soubory, pomocí kterých je možné replikovat jednotlivé instance na dalších systémech, což umožní snazší škálování na další zařízení. [29]

4.2 Nevýhody Docker

4.2.1 Rychlost spouštění aplikací

Ve srovnání s plnou virtualizací mají Docker kontejnery nižší režii. Ale pokud dojde ke spuštění aplikace přímo na serveru, poběží tato aplikace rychleji, než kdyby byla spuštěna v kontejneru. [32]

4.2.2 Přenositelnost mezi platformou Windows a Linux

Jedním z hlavních problémů je, že pokud bude aplikace určená k nasazení v Docker kontejneru pro platformu Windows, pak ji nelze spustit na platformě Linux a naopak. [32]

4.2.3 Spouštět aplikace s grafickým rozhraním

Docker je obecně určen pro hostování aplikací, které běží v prostředí příkazového řádku. Ačkoliv existuje několik způsobů (např. X11 forwarding), které umožňují provozovat grafického rozhraní uvnitř kontejneru, nejsou pro aplikace vyžadující bohatá grafická rozhraní příliš vhodné. [32]

5 IMPLEMENTACE

V této kapitole jsou popsány jednotlivé konfigurační soubory jednotlivých služeb nezbytných pro multihostingový systém.

5.1 Analýza problému

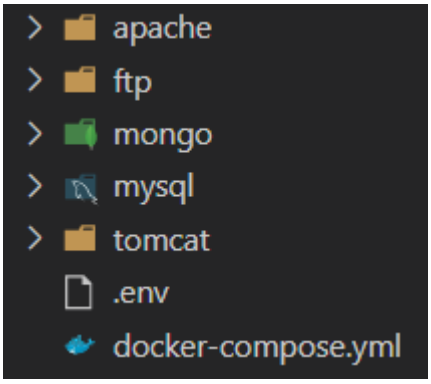
Cílem této práce je vytvořit multihostingový systém na platformě Docker. Tak aby byla zajištěna podpora PHP skriptů a možnost jejich propojení s databázemi MySQL a MongoDB, provoz aplikací typu Java EE a FTP pro přístup k webovým stránkám. Dále bude vytvořena aplikace pro administraci multihostingového serveru, která bude umožňovat vytváření a odstranění domén včetně správy přihlašovacích údajů jednotlivých služeb. Administrace bude umožňovat nasazení JEE a správu již nahraných aplikací.

5.2 Návrh multihostingu na dockeru

Při budování multihostingu je postupováno od méně komplexních k těm komplexnějším. Nejdříve bylo nezbytné nalézt příslušné docker images, které budou využity jako základ pro jednotlivé služby. Následně bylo nezbytné vytvořit Dockerfile soubory, které slouží k dodatečné konfiguraci výsledných docker images, vystavěné na těch námi nalezenými. Dalším krokem bylo vytvoření docker-compose.yml souboru pro konfiguraci kontejnerů, který uchovává konfiguraci a pořadí jednotlivých služeb.

5.3 Základní struktura projektu

Projekt se skládá ze dvou konfiguračních souborů kontejnerů a z několika adresářů, ve kterých se nachází veškeré konfigurační soubory pro vytvoření kontejnerů jednotlivých služeb tak, aby byl zajištěn správný běh multihostingového serveru. Ukázka struktury projektu je uvedena níže:



Obrázek 5 – Struktura hlavního adresáře projektu (zdroj: vlastní)

Podrobnější seznámení s jednotlivými soubory a adresáři je k dispozici v následujících kapitolách.

5.4 Globální konfigurace docker kontejnerů

Pro snazší správu obrazů Docker je vhodné vytvořit globální konfiguraci. Tato konfigurace slouží k jednodušší obsluze některých proměnných, které jsou využity skrze projekt, jako například informace o síti, či jména jednotlivých služeb.

5.4.1 Soubor .env

Slouží pro specifikaci všech proměnných pro daná prostředí tak, aby při spuštění kontejnerů byli nastaveny např. účty pro přihlášení k databázím a k FTP serveru. Jednotlivé systémové proměnné jsou popsány níže:

```
1.  DOMAIN_NAME=multihosting.com
```

Pomocí proměnné `DOMAIN_NAME` lze specifikovat doménové jméno pro multihosting. Na této adrese bude dostupná aplikace pro vytváření a správu uživatelských domén.

```
2.  MYSQL_ROOT_PASSWORD=z7LbcUKgzH#t2AoBf@@L
3.  MYSQL_DATABASE=databasedomain
4.  MYSQL_USER=domainadmin
5.  MYSQL_PASSWORD=sFGp8SN#VK0rZXc9XWVR
6.  MYSQL_FTP_USER=ftpdatabaseuser
7.  MYSQL_FTP_PASSWORD=se06!u)$;7d6#h?tqs@e
```

Proměnné sloužící ke konfiguraci služby MySQL ve které se pomocí následujících proměnných specifikuje:

- `MYSQL_ROOT_PASSWORD` – nastavení přístupového hesla uživatele root, který je automaticky vytvořen při instalaci služby MySQL. Tento uživatel má přístup ke všem databázím a uživatelům včetně jejich přístupových práv, které může libovolně měnit.
- `MYSQL_DATABASE` – název databáze jenž bude obsahovat veškeré tabulky nezbytné pro správné fungování multihostingového systému.
- `MYSQL_USER` a `MYSQL_PASSWORD` – definují přihlašovací údaje uživatele, pomocí kterého bude řídící aplikace vytvářet: klientské databáze, uživatelské účty a nastavovat oprávnění.
- `MYSQL_FTP_USER` a `MYSQL_FTP_PASSWORD` – definují přihlašovací údaje uživatele, který bude mít přístup pouze k tabulkám uchovávající přihlašovací údaje ke službě FTP.

```
8.  FTP_ADMIN_NAME=ftpadmin
9.  FTP_ADMIN_PASSWORD=&n/vtt;2(a0~\[<%aqg]
```

Tyto proměnné specifikují uživatelské jméno a heslo pro uživatele služby FTP, který bude mít přístup ke všem uživatelským datům a zdrojovým kódům řídící aplikaci.

```
10. MONGODB_ADMIN_USER=superuzivatel
11. MONGODB_ADMIN_PASS=SKzsvp1Xmb5CZVL3NmKe
```

Proměnné `MONGODB_ADMIN_USER` a `MONGODB_ADMIN_PASS` specifikují uživatelské jméno a heslo uživatele pro MongoDB, který má přístup ke všem databázím, ve kterých může provádět jakékoliv databázové operace. Pokud jsou tyto proměnné nastaveny MongoDB bude vyžadovat autentizaci jednotlivých uživatelů při připojení k databázím.

```
12. TOMCAT_ADMIN_USER=administrator
13. TOMCAT_ADMIN_PASSWORD=eQ.rqDNJScwOcd
```

Tyto proměnné specifikují uživatelské jméno a heslo pro uživatele služby Tomcat, který bude mít přístup k administraci aplikací a virtuálních hostů.

```
14. CONTAINER_MYSQL=bak-mysql
15. CONTAINER_MONGODB=bak-mongodb
16. CONTAINER_FTP=bak-ftp
17. CONTAINER_APACHE=bak-apache
18. CONTAINER_TOMCAT=bak-tomcat
```

Pomocí těchto proměnných jsou specifikovány názvy kontejnerů. Navíc tyto názvy slouží k identifikaci kontejner v počítačové síti (hostname), proto jsou nezbytné pro komunikaci všech služeb v rámci multihostingového systému.

5.4.2 Soubor docker-compose.yml

Specifikující veškeré kontejnery včetně nastavení proměnných prostředí načtených ze souboru .env, portů sloužících ke komunikaci kontejneru s okolním světem, virtuálních sítí, pomocí kterých mohou jednotlivé kontejnery spolu komunikovat a volumes pro trvale uložení některých dat v kontejneru tak, aby byla data zachována i po jeho vymazání.

```
1.  version: '3'
```

Direktiva `version` specifikuje verzi pro formátování souboru docker-compose.yml. Jednotlivé verze se liší v možnostech konfigurace kontejnerů, jelikož mezi různými verzemi docházelo jak k přidávání, tak i odebírání možností konfigurace. Třetí verze byla vytvořena se záměrem zachovat kompatibilitu mezi projekty vytvořenými pomocí Docker Compose a Docker Swarm.

Ukázka konfigurace kontejneru obsahujícího službu MySQL v souboru docker-compose.yml:

```
services:
  mysql:
    container_name: ${CONTAINER_MYSQL}
    build: ./mysql/
    image: ${CONTAINER_MYSQL}
    restart: always
    volumes:
      - mysql_data:/var/lib/mysql
    environment:
      - MYSQL_DATABASE=${MYSQL_DATABASE}
      - MYSQL_USER=${MYSQL_USER}
      - MYSQL_PASSWORD=${MYSQL_PASSWORD}
      - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
      - DOMAIN_NAME=${DOMAIN_NAME}
      - FTP_ADMIN_NAME=${FTP_ADMIN_NAME}
      - FTP_ADMIN_PASSWORD=${FTP_ADMIN_PASSWORD}
      - MYSQL_FTP_USER=${MYSQL_FTP_USER}
      - MYSQL_FTP_PASSWORD=${MYSQL_FTP_PASSWORD}
    networks:
      - network_mysql_ftp
      - network_mysql_apache
      - network_mysql_tomcat
```

Za direktivou `services` jsou definovány jednotlivé kontejnery, u kterých je nejdříve uveden název služby následovaný dalšími direktivy pro konfiguraci kontejneru.

- `container_name` – specifikuje název pro docker images, tento název je uložen v proměnné MYSQL_DATABASE definované souboru *.env*.
- `build` – specifikuje cestu k souboru Dockerfile na základě kterého bude sestaven docker image.
- `image` – specifikace názvu docker image použitého pro kontejner, tento název je uložen v proměnné MYSQL_DATABASE definované souboru .env.
- `restart` – určí, zda se má automaticky restartovat kontejner. Lze nastavit jeden z parametrů:
  - *no* – nerestartovat automaticky,
  - *on-failure* – restartovat při chybě kontejneru,
  - *always* – restartovat vždy kromě situace kdy uživatel vypne kontejner. Při této situaci se kontejnery opět automaticky zpustí po restartování nástroje docker.
  - *unless-stopped* – podobná hodnotě always ale nedojde k automatickému spuštění kontejnerů při restartování nástroje docker.

- `volume` – specifikuje adresář který bude uložen na trvalé uložiště tak, aby po vymazání kontejneru nedošlo k vymazání databází.
- `environment` – specifikace proměnných prostředí, které jsou nezbytné pro nastavení uživatelských účtu v databázi.
- `networks` – specifikace virtuálních sítí pro jednotlivé služby tak, aby jednotlivé služby komunikovali jen s nezbytnými kontejnery.

Ukázka konfigurace virtuálních sítí pro kontejnery v souboru docker-compose.yml:

```
24.  networks:
25.    network_web_apps:
26.      driver: bridge
27.    network_mysql_ftp:
28.      driver: bridge
29.    network_mysql_apache:
30.      driver: bridge
31.    network_mysql_tomcat:
32.      driver: bridge
33.    network_monogdb_apache:
34.      driver: bridge
35.    network_monogdb_tomcat:
36.      driver: bridge
```

V této ukázce jsou specifikovány virtuální sítě umožňující komunikaci mezi jednotlivými kontejnery. V názvu každé virtuální sítě jsou vždy uvedeny služby, které spolu po této sítí komunikují. Z důvodu bezpečnosti jsou vytvořeny virtuální sítě propojující pouze dva kontejnery mezi sebou.

Ukázka konfigurace trvalých uložišť pro kontejnery v souboru docker-compose.yml:

```
37.  volumes:
38.    mysql_data:
39.      driver: local
40.    mongo_data:
41.      driver: local
42.    apache_data:
43.      driver: local
44.    tomcat_data:
45.      driver: local
```

Pro uložení dat služeb uvnitř kontejnerů jsou vytvořeny uložiště, které umožnují trvalé uložení dat. Díky tomu nedojde např. ke ztrátě databází při odstranění kontejneru obsahující službu MySQL. Všechny služby uchovávající data uživatelů a řídící aplikace jsou uložena v těchto trvalých uložištích. Podle názvu uložiště lze snadno poznat pro kterou službu je dané uložiště určeno.

5.5 Konfigurace jednotlivých kontejnerů

V této kapitole se nachází popis konfigurace, sloužící k vytvoření nezbytných kontejnerů pro multihostingový server.

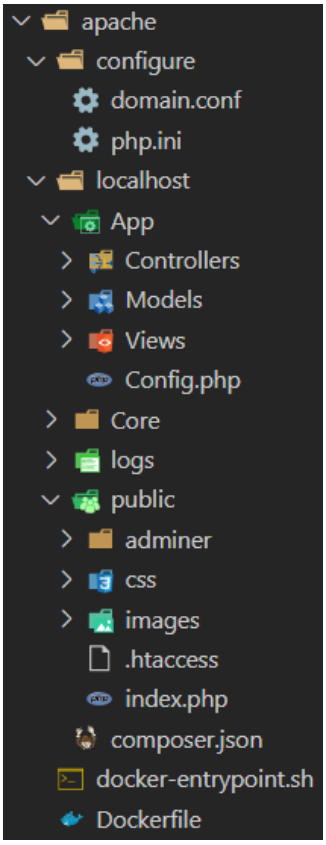
5.5.1 Soubory docker-entrypoint.sh

Tyto soubory slouží k nastavení proměnných prostředí definovaných v souboru .env nacházejícím se v hlavním adresáři projektu. Tyto proměnné jsou přiděleny k jednotlivým kontejnerům v souboru docker-compose.yml. Pomocí souborů docker-entrypoint.sh dochází v konfiguračních souborech k nahrazení předdefinovaných názvů jednotlivými proměnnými. Tím je například zajištěno, že při vytváření kontejneru se službou MySQL dojde k vytvoření uživatelských účtů se jmény a hesly definovanými v souboru .env. Na konci souboru docker-entrypoint.sh se nachází příkaz pro spuštění dané služby.

5.5.2 Apache

Konfigurace Apache se skládá z několika souborů konfigurace nacházející se v adresáři configure. V tomto adresáři je možné konfigurovat jak službu Apache, tak virtuální hosty, kteří slouží ke snadnější konfiguraci domén tím, že odstraňují potřebu vyhradit pro každou doménu jednu IP adresu. V konfiguraci se nastavují např.: moduly, které se mají zavést při spuštění služby, IP adresy a porty na kterých bude Apache naslouchat atd. V souboru domain.conf se poté ukrývá konfigurace jednotlivých virtuálních hostů.

Adresářová struktura



Obrázek 6 – Obsah adresáře apache (zdroj: vlastní)

- `configure` – *adresář* obsahující konfigurační soubory nezbytné pro zajištění správného fungovaní serveru Apache. Jednotlivé soubory jsou probrány v následujících kapitolách.
- `localhost` – adresář obsahující řídicí aplikaci vytvořenu pro multihostigový systém.
- `App` – adresář obsahující strukturu pro řídicí aplikaci využívající návrhová vzor MVC. Pomocí toho vzoru lze aplikaci rozdělit na datový model (adresář Models), uživatelské rozhraní (adresář Views) a řídicí logiku (adresář Controllers) do tří nezávislých komponent tak, aby při modifikaci jedné z částí byl minimální vliv na ostatní.
- `Core` – adresář obsahuje základní zdrojové kódy aplikace, ty jsou dále využívány v datových modelech.
- `logs` – v tomto adersáři se automaticky vytvářejí soubory obsahující chybové zprávy aplikace.
- `public` – adresář obsahující: kaskádové styly, obrázky, soubor index.php zpracovává požadavky na řídicí aplikaci, .htaccess slouží k dodatečné úpravě služby apache pomocí úpravy URL adresy tak, aby bylo možné využívat návrhový vzor MVC.

- `adminer` – adresář obsahující soubor .htaccess pro specifikaci verze PHP 5.6 a soubor adminer.php využívající tuto verzi. Soubor adminer.php slouží k administraci databází skrze webové rozhraní.
- `composer.json` – slouží ke specifikaci tříd které se mají automaticky zavést do projektu pomocí nástroje composer.
- Soubor `Dockerfile` je popsán později v této kapitole.

Soubor Dockerfile

```
1.  FROM ubuntu:19.10
```

Direktiva `FROM` určí základní kontejner pro docker image obsahující Ubuntu ve verzi 19.10.

```
2.  WORKDIR /var/www/html
```

Direktiva `WORDKDIR` nastaví pracovní adresář na umístění */var/www/html*. V tomto umístění se nacházejí adresáře jednotlivých domén a složka localhost obsahující řídící aplikaci.

```
3.  EXPOSE 80
```

Direktiva `EXPOSE` zpřístupní port 80, nezbytný pro komunikaci se službou Apache.

```
4.  RUN ln -sf /usr/share/zoneinfo/Europe/Prague /etc/localtime && \
5.      apt-get update && \
6.      apt-get install -y \
7.          software-properties-common && \
8.      add-apt-repository ppa:ondrej/php -y && \
9.      apt-get update && \
10.     apt-get dist-upgrade -y && \
11.     apt-get install -y \
12.         apache2 \
13.         php5.6 \
14.         php5.6-fpm \
15.         …
16.         php7.3 \
17.         php7.3-fpm \
18.         …
19.         libapache2-mod-fcgid \
20.         php-pear \
21.         autoconf \
22.         wget \
23.         composer && \
24.     apt-get autoremove -y && \
25.     apt-get autoclean && \
26.     rm -rf /var/lib/apt/list/*
```

Direktiva RUN provede skript, který nejdříve vytvoří symbolický odkaz, zajišťující nastavení časového pásma a lokace do české republiky. Tento krok je nezbytný k instalaci balíčku apache2 tak, aby při instalaci nevyžadoval nastavení časového pásma. Dalším krokem

je instalace balíčku *software-properties-common* umožňující přidat nový repozitář, uchovávající balíčky nezbytné k instalaci PHP 5.6. Následně se provede aktualizace všech balíčků, po které dojde k instalaci těch potřebných pro provozování službu Apache s podporou jazyka PHP. Bohužel nástroj Adminer nedokáže komunikovat s MongoDB databází při využití PHP 7.3. Proto bylo nezbytné nainstalovat PHP ve verzi 5.6 i 7.3 a následně specifikovat využití PHP 5.6 pro nástroj Admirer. Po instalaci balíčků dojde k odstranění instalačních dat.

```
27.  ADD ./configure/domain.conf /etc/apache2/sites-available/
28.  ADD ./configure/php.ini /etc/php/7.3/fpm/
29.  ADD ./configure/php.ini /etc/php/5.6/fpm/
30.  COPY ./localhost /var/www/html/localhost
```

Kopírování konfiguračních souborů do příslušných umístění tak, aby byl zajištěn bezproblémový chod Apache.

```
31.  RUN pecl install mongodb && \
32.      echo "extension=mongodb.so" >> /etc/php/5.6/fpm/php.ini && \
33.      echo "extension=mongodb.so" >> /etc/php/7.3/fpm/php.ini && \
34.      a2enmod vhost_alias \
35.              rewrite \
36.              proxy \
37.              proxy_http \
38.              actions \
39.              alias \
40.              proxy_fcgi \
41.              fcgid && \
42.      a2ensite domain.conf && \
43.      rm index.html /etc/apache2/sites-enabled/000-default.conf && \
44.      mkdir /var/log/apache && \
45.      composer install -d ./localhost/ && \
46.      chmod 744 -R /var/www && \
47.      usermod -u 5500 www-data && \
48.      groupmod -g 5500 www-data && \
49.      chown 5500:5500 -R /var/www && \
50.      wget "http://www.adminer.org/latest.php" \
51.            -O localhost/public/adminer/index.php
```

Provede instalaci ovladače pro MongoDB, který je poté nezbytné zavést v konfiguračním souboru *php.ini* pro příslušné verze PHP. Po instalaci následuje načtení několika modulů rozšiřující službu Apache tak, aby bylo možné např. využít virtuální hosty, či rewrite pro přepis URL adresy.

Načtení modulů umožní zavést konfigurační soubor obsahující virtuální hosty pro multihostingový systém. Ten bude nadále používán při vytváření a odebírání nových domén jednotlivými uživateli.

Následně dojde k odstranění souborů automaticky vygenerovaných během instalace služby Apache, aby nedošlo k uložení nevyužitých souborů z důvodu šetření úložného prostoru a následné vytvoření adresáře pro ukládání logů služby Apache.

Poté *composer* zpracuje konfigurační soubor *composer.json*, ve kterém jsou definovány třídy které se mají stáhnout a adresáře které se mají automaticky načítat. Po tomto načtení dojde k vytvoření symbolických odkazů.

Následuje změna oprávnění adresářů a uživatele tak, aby bylo možné se do adresářů připojit pomocí FTP klienta. Poslední příkaz zajistí stažení nástroje Adminer do adresáře s řídící aplikací.

```
52.  ADD docker-entrypoint.sh /
53.  RUN chmod a+x /docker-entrypoint.sh
54.  ENTRYPOINT ["/docker-entrypoint.sh"]
```

Direktiva `ADD` přidá soubor *docker-entrypoint.sh* do kořenového adresáře. Po přidání následuje příkaz pro přidání práv sloužících ke spuštění přidaného souboru. Posledním příkazem je `ENTRYPOINT` pro specifikaci souboru se skriptem, který se provede při spuštění kontejneru.

Soubor domain.conf

Tento soubor obsahuje nastavení pro virtuální hosty, každý virtuální host začíná tagem <*VirtualHost*> ve kterém lze specifikovat konkrétní IP adresu / adresy a číslo portu na kterém bude virtuální host naslouchat. Pokud požadujeme, aby daný virtuální host obsluhoval všechny IP adresy na standardním portu pro HTTP stačí specifikovat <*VirtualHost *:80*>. Virtuální host končí tagem </ *VirtualHost*>. Mezi těmito tagy mohou být např. uvedeny následující direktivy:

- `ServerAdmin` – specifikuje e-mailovou adresu, pod kterou se odesílají všechny chybové zprávy klientovi,
- `ServerName` – specifikuje název domény, který bude tímto virtuálním hostem obsloužen. Název domény se většinou uvádí ve tvaru www.název-domény.cz.
- `ServerAlias` – pomocí kterého můžeme uvést více názvů domény, které bude obsluhovat stejný virtuální host,
- `VirtualDocumentRoot` – umožňuje určit, kde Apache najde příslušné soubory na základě názvu serveru,
- `ErrorLog` – specifikuje soubor do kterého Apache odešle diagnostické informace a zaznamenané chyby, se kterými se setká při zpracování požadavků. Při problému s Apachem je tento soubor velmi důležitý k identifikaci a odstranění chyb.

- `Directory` – umožňuje specifikovat pro adresář např.: z kterých IP adres lze vstoupit do adresáře a zda je povolené indexování souborů.

Ukázka části souboru domain.conf:

```
1.  <VirtualHost *:80>
2.          ServerAdmin webmaster@domain
3.          ServerName www.domain
4.
5.          SetEnv DOMAIN_NAME "${DOMAIN_NAME}"
6.          …
7.          SetEnv TOMCAT_ADMIN_PASSWORD "${TOMCAT_ADMIN_PASSWORD}"
8.
9.          <Directory /var/www/html/localhost/*>
10.                 AllowOverride All
11.                 Order allow,deny
12.                 allow from all
13.         </Directory>
14.
15.         <FilesMatch \.php$>
16.         SetHandler "proxy:unix:/var/run/php/php7.3-fpm.sock|fcgi://localhost/"
17.         </FilesMatch>
18.
19.         DocumentRoot "/var/www/html/localhost/public"
20.
21.         CustomLog /var/log/apache/www.domain-access.log combined
22.         ErrorLog /var/log/apache/www.domain-error.log
23. </VirtualHost>
```

Virtuální host, jenž obslouží všechny dotazy směrovány na doménu multihosting.com. Uvádí také systémové proměnné, které budou v tomto virtuálním hostu dostupné. Všichni uživatelé mají přístup do adresáře /var/www/html/localhost, kde je uložena řídící aplikace. Tento adresář je nastaven jako kořenový adresář pro doménu multihosting.com. Proto se v tomto adresáři budou hledat soubory s názvem index.html nebo index.php. Direktivy `FileMatch` a `SetHandler` specifikují pro provádění skriptů PHP ve verzi 7.3. Dále budu logovány chyby způsobeny při komunikaci s tímto hostem, tyto logy budou uloženy do adresáře nacházející se v */var/log/apache/*.

```
<VirtualHost *:80>
        ServerAdmin webmaster@domain
        ServerAlias www.*.domain

        <Directory /var/www/html/*>
                AllowOverride All
                Order allow,deny
                allow from all
        </Directory>

         <FilesMatch \.php$>
        SetHandler "proxy:unix:/var/run/php/php7.3-fpm.sock|fcgi://localhost/"
        </FilesMatch>

        UseCanonicalName Off
        VirtualDocumentRoot "/var/www/html/%-3"

        CustomLog /var/log/apache/subdomain-access.log combined
        ErrorLog /var/log/apache/subdomain-error.log
</VirtualHost>
```

Virtuální host obsluhující veškeré subdomény služby Apache. Tento host má přístup ke všem adresářům v umístění */var/www/html* a pro spuštění PHP skriptů využívá PHP ve verzi 7.3.

`UseCanonicalName Off`

V Apache httpd vytvoří vlastní referenční adresy URL pomocí hostitele a portu dodaného klientem, jsou-li dodány (jinak použije kanonické jméno, jak je definováno výše).

`VirtualDocumentRoot`

Nabízí možnost specifikovat podadresář, který je následně vložen do adresáře vhosts na základě jména domény.

Soubor php.ini

Tento soubor je jedním ze základních konfiguračních souborů v PHP, který je načten po spuštění Apache HTTP. Mezi základní nastavení patří např. logování chyb nebo seznam povolených rozšíření mezi které patří mongo-driver pro komunikaci s databází MongoDB. Také se zde specifikují funkce, které mají být zakázané, jelikož si může jakýkoliv zákazník založit vlastní hosting je nezbytné zakázat všechny funkce které umožnují ovlivnit konfiguraci serveru. Mezi tyto funkce patří např: exec, passthru, shell_exec nebo system.

5.5.3 FTP

Soubor Dockerfile

```
1.  FROM debian:stretch
```

Pomocí direktivy `FROM` se určí základní kontejner pro docker image obsahující Debian ve verzi Stretch.

```
2.  RUN apt-get update && \
3.      apt-get full-upgrade && \
4.      apt-get install -y perl \
5.      proftpd \
6.      proftpd-mod-mysql \
7.      mysql-client \
8.      mysql-common && \
9.      apt-get autoremove -y && \
10.     apt-get autoclean && \
11.     rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
```

Provede aktualizaci operačního systému, po které následuje instalace potřebných balíčků pro službu FTP. Následuje vymazání všech nepotřebných dat tak, aby byl výsledný docker image co nejmenší.

```
12.  EXPOSE 20 21
```

Direktiva `EXPOSE` zpřístupňuje port 20 a 21. Tyto porty jsou nezbytné ke komunikaci s FTP serverem. Na portu 20 se přenášejí pouze data, pro komunikaci se serverem slouží port 21, na kterém server naslouchá příchozímu spojení FTP klienta.

```
13.  COPY ./configure/ /etc/proftpd
```

Direktiva `COPY` zkopíruje veškeré konfigurační soubory služby FTP.

```
14.  RUN mkdir -p ftp /var/
```

Tento příkaz vytvoří adresář FTP, ve kterém se budou nacházet data uživatelů v jednotlivých adresářích. Zároveň slouží jako kořenové adresáře při přihlášení k FTP serveru.

```
15.  WORKDIR /var/ftp
```

Nastaví pracovní adresář na umístění */var/ftp*. V tomto umístění se následně provádějí veškeré příkazy.

```
16.  ADD docker-entrypoint.sh /
17.  RUN chmod a+x /docker-entrypoint.sh
18.  ENTRYPOINT ["/docker-entrypoint.sh"]
```

Příkaz `ADD` přidá soubor docker-entrypoint.sh do kořenového adresáře. Po přidání následuje příkaz pro přidání práv sloužících ke spuštění přidaného souboru. Posledním příkazem je `ENTRYPOINT` pro specifikaci souboru se skriptem, který se provede při spuštění kontejneru.

Soubor modules.conf

V tomto souboru jsou uvedeny moduly, které se načítají spolu se spuštěním ProFTPD serveru. Pro povolení autentizace uživatelů pomocí MySQL databáze je třeba načíst modul `mod_sql.c` a `mod_sql_mysql.c`.

Soubor proftpd.conf

Tento soubor obsahuje většinu konfigurace pro ProFTPD server, zde lze specifikovat např.: číslo portu na kterém bude server naslouchat, maximální povolený počet připojení na server. Pro správné fungování FTP serveru bylo nezbytné provést následující konfiguraci:

```
DefaultRoot             ~
```

Direktiva *DefaultRoot* řídí výchozí kořenový adresář přiřazený danému uživateli při přihlášení tak, aby došlo k izolaci dat uživatele od zbytku souborového systému tím, že zabráníme uživateli opustit daný kořenový adresář.

```
RequireValidShell       off
```

Direktiva *RequireValidShell* je nastavena na hodnotu off aby nevyžadovala použití jednoho z shellů uvedených v */etc/shells*. Díky tomu se mohou na FTP server připojit i uživatelé operačního systému Windows.

```
PassivePorts            49152 49500
```

Direktiva *PassivePorts* omezuje rozsah portů, ze kterých server vybere při odeslání příkazu `PASV` z klienta. Následně server náhodně vybere číslo ze zadaného rozsahu, dokud nenajde otevřený port. Pokud nebudou v daném rozsahu nalezeny žádné otevřené porty, bude server standardně nastaven na standardní port přiřazený jádru. PassivePorts spolu se specifikací portů je nutnou podmínkou pro připojení z operačního systému Windows.

```
Include /etc/proftpd/sql.conf
```

Direktiva *Include* zajistí, aby došlo k začlenění nastavení v souboru /etc/proftpd/sql.conf do konfigurace služby Proftpd.

Soubor sql.conf

Tento soubor obsahuje konfiguraci umožňující autentizaci uživatelů proti databázi, ve které se nacházejí přihlašovací jména a hashovaná hesla uživatelů. Jednotlivé direktivy budou následně popsány v této kapitole.

Ukázka souboru sql.conf:

```
1.  SQLBackend              mysql
2.
3.  SQLAuthTypes            OpenSSL
4.  SQLAuthenticate         users groups
5.
6.  SQLConnectInfo  ftp-database@bak-mysql ftp-user ftp-password
7.
8.  SQLUserInfo     ftpuser userid passwd uid gid homedir shell
9.
10. SQLGroupInfo    ftpgroup groupname gid members
11.
12. SQLMinID        500
13.
14. SQLLog PASS updatecount
15. SQLNamedQuery updatecount UPDATE "count=count+1, accessed=now()
16.               WHERE userid='%u'" ftpuser
17.
18. SQLLog  STOR,DELE modified
19. SQLNamedQuery modified UPDATE "modified=now() WHERE userid='%u'" ftpuser
20.
21. SqlLogFile /var/log/proftpd/sql.log
```

Popis jednotlivých direktiv použitých v souboru sql.conf:

SQLBackend

Specifikuje typ použitého modulu pro komunikaci s databází.

SQLAuthTypes

Slouží ke specifikování tvaru ukládání uživatelských hesel do databáze.

OpenSSL

Umožňuje, aby hesla v databázi byla uložena ve tvaru '`{digest-name} hashed-value`', kde `digest-name` specifikuje použitou hash funkci a `hashed-value` (někdy také nazývaná Hashes nebo Checksum) je řetězcová hodnota (specifické délky), která je výsledkem výpočtu hash funkce.

SQLAuthenticate

Specifikuje údaje, které je nezbytné ověřit pro přihlášení uživatele k FTP serveru.

SQLConnectInfo

Specifikuje nezbytné údaje pro připojení k databázi. Mezi nezbytné údaje patří: název databáze, jméno hostitele nebo IP adresa databázového serveru a uživatelského jméno a heslo uživatele, které se použije při připojení k databázi.

SQLUserInfo

Specifikuje jednotlivé sloupce nacházející se v tabulce určené pro uložení informací o uživatelích. Nejdůležitějších sloupce:

- *usertable* – specifikuje název tabulky obsahující informace o uživatelích,
- *username* – specifikuje sloupec v tabulce uživatelů, který obsahuje uživatelské jméno,
- *passwd* – specifikuje sloupec v tabulce uživatelů, který obsahuje heslo uživatele,
- *uid* – specifikuje sloupec v tabulce uživatelů, který obsahuje UID (User Identifier) uživatele,
- *gid* – specifikuje sloupec v tabulce uživatelů, který obsahuje GID (Group Identifier) uživatele,
- *homedir* – specifikuje sloupec v tabulce uživatelů, který obsahuje domovský adresář uživatele,
- *shell* – specifikuje sloupec v tabulce uživatelů, který určuje shell uživatele.

SQLGroupInfo

Specifikuje jednotlivé sloupce nacházející se v tabulce určené pro uložení informací o skupinách. Popis jednotlivých sloupců:

- *grouptable* – specifikuje název tabulky obsahující informace o skupinách,
- *groupname* – specifikuje sloupec v tabulce skupin, který obsahuje název skupiny,
- *gid* – specifikuje sloupec v tabulce skupin, který obsahuje název GID skupiny,
- *members* – specifikuje sloupec v tabulce skupin, který obsahuje seznam členů skupiny.

SQLMinID

Nabízí rychlý způsob nastavení SQLMinUserGID a SQLMinUserUID. Tyto hodnoty jsou kontrolovány vždy, když se načte GID nebo UID uživatele.

```
14.  SQLLog PASS updatecount
15.  SQLNamedQuery updatecount UPDATE "count=count+1, accessed=now()
16.                WHERE userid='%u'" ftpuser
```

Při úspěšném přihlášení uživatele aktualizuje počet přihlášení a nastaví aktuální datum a čas posledního přihlášení.

```
18.  SQLLog  STOR,DELE modified
19.  SQLNamedQuery modified UPDATE "modified=now() WHERE userid='%u'" ftpuser
```

Při změně dat na FTP serveru dojde k aktualizaci datumu a času poslední změny dat.

SqlLogFile

Specifikuje soubor k ukládání hlášení o provedení SQL dotazů.

5.5.4 Mysql

Soubor Dockerfile

```
1.  FROM mysql:5.7
```

Pomocí direktivy `FROM` se určí základní kontejner pro docker image obsahující MySQL ve verzi 5.7.

```
2.  RUN apt-get update && \
3.      apt-get install -y procps && \
4.      apt-get full-upgrade -y && \
5.      apt-get autoremove -y && \
6.      apt-get autoclean && \
7.      rm -rf /var/lib/apt/list/* &&
```

Direktiva `RUN` provede instalaci balíčku *procps*, nezbytného k úspěšné aktualizaci všech nainstalovaných balíčků. Po této aktualizaci balíčků následuje odstranění všech instalačních souborů.

```
8.  sed -i "2 i\sed -i 's/ftp-user/'\$FTP_ADMIN_NAME'/g' /docker-entrypoint-
    initdb.d/setup.sql" /usr/local/bin/docker-entrypoint.sh && \
9.  sed -i "3 i\sed -i 's/ftp-password/'\$(echo \"{md5}\"\$(echo -n \$FTP_AD-
    MIN_PASSWORD | openssl dgst -binary -md5 | openssl enc -base64))'/g' /docker-
    entrypoint-initdb.d/setup.sql" /usr/local/bin/docker-entrypoint.sh && \
10. sed -i "4 i\sed -i 's/domain-name/'\$DOMAIN_NAME'/g' /docker-entrypoint-
    initdb.d/setup.sql" /usr/local/bin/docker-entrypoint.sh && \
11. sed -i "5 i\sed -i 's/database-user/'\$MYSQL_USER'/g' /docker-entrypoint-
    initdb.d/setup.sql" /usr/local/bin/docker-entrypoint.sh && \
12. sed -i "6 i\sed -i 's/ftp-mysql-user/'\$MYSQL_FTP_USER'/g' /docker-entrypoint-
    initdb.d/setup.sql" /usr/local/bin/docker-entrypoint.sh && \
13. sed -i "7 i\sed -i 's/ftp-mysql-password/'\$MYSQL_FTP_PASSWORD'/g' /docker-
    entrypoint-initdb.d/setup.sql" /usr/local/bin/docker-entrypoint.sh && \
14. sed -i "8 i\sed -i 's/database-name/'\$MYSQL_DATABASE'/g' /docker-entrypoint-
    initdb.d/setup.sql" /usr/local/bin/docker-entrypoint.sh
```

Vloží příkazy provádějící dodatečnou konfiguraci po zpuštění kontejneru do souboru /usr/local/bin/docker-entrypoint.sh. Tento krok je nezbytný, aby se za textové řetězce dosadily správné názvy pro databázi, jména a hesla uživatelů v souboru `setup.sql`.

```
15. ADD setup.sql /docker-entrypoint-initdb.d
```

Přidá SQL skript do adresáře */docker-entrypoint-initdb.d*. Následně při zpuštění kontejneru dojde k provedení toho skriptu.

Soubor Setup.sql

V tomto souboru se nachází SQL script, který vytvoří databází včetně všech nezbytných tabulek a nastavení oprávnění uživatelů, aby byl zajištěn bezproblémový běh multihostingového serveru.

```
1.  SET NAMES 'utf8';
```

Nastaví kódování databáze na UTF8 aby bylo zajištěno správné uložení znaků české abecedy do databáze.

```
2.  CREATE TABLE IF NOT EXISTS domain (
3.      id              int NOT NULL AUTO_INCREMENT,
4.      name            varchar(100) NOT NULL,
5.      registered      datetime NOT NULL DEFAULT '2000-01-01 00:00:00',
6.      PRIMARY KEY (id)
7.  );
8.
9.  CREATE TABLE  IF NOT EXISTS user (
10.     id              int NOT NULL AUTO_INCREMENT,
11.     login           varchar(50) NOT NULL,
12.     password        varchar(50) NOT NULL,
13.     created         datetime NOT NULL DEFAULT '2000-01-01 00:00:00',
14.     updated         datetime NOT NULL DEFAULT '2000-01-01 00:00:00',
15.     last_login      datetime NOT NULL DEFAULT '2000-01-01 00:00:00',
16.     domain_id       int NOT NULL,
17.     PRIMARY KEY (id)
18. );
```

Tato část skriptu vytváří nezbytné tabulky pro chod řídící aplikace. Tabulku pro ukládání vytvořených subdomén, do které se spolu s názvem ukládá i čas vytvoření subdomény. Druhá tabulka slouží k ukládání uživatelských účtů v administrační části řídící aplikace. Do této tabulky se ukládá jak uživatelské jméno a hashované heslo tak datumy: vytvoření uživatelského účtu, poslední aktualizace a posledního úspěšného přihlášení.

```
19.  INSERT INTO domain (name, registered) VALUES ('domain-name', NOW());
```

Tato část skriptu vkládá do databáze název domény multihostingového systému spolu s časem vložení do databáze.

```
20.  CREATE TABLE IF NOT EXISTS ftpgroup (
21.    groupname        varchar(16) NOT NULL,
22.    gid              smallint(6) NOT NULL DEFAULT 5500,
23.    members          varchar(50) NOT NULL,
24.    KEY groupname (groupname)
25.  );
26.
27.  CREATE TABLE  IF NOT EXISTS ftpuser (
28.    id                 int unsigned NOT NULL AUTO_INCREMENT,
29.    userid             varchar(32) NOT NULL DEFAULT '',
30.    passwd             varchar(32) NOT NULL DEFAULT '',
31.    uid                smallint(6) NOT NULL DEFAULT 5500,
32.    gid                smallint(6) NOT NULL DEFAULT 5500,
33.    homedir            varchar(255) NOT NULL DEFAULT '',
34.    shell              varchar(16) NOT NULL DEFAULT '/sbin/nologin',
35.    count              int(11) NOT NULL DEFAULT '0',
36.    accessed           datetime NOT NULL DEFAULT '2000-01-01 00:00:00',
37.    modified           datetime NOT NULL DEFAULT '2000-01-01 00:00:00',
38.    domain_id          int NOT NULL,
39.    PRIMARY KEY (id),
40.    UNIQUE KEY userid (userid)
41.  );
```

Tato část skriptu vytváří tabulky nezbytné pro ukládání FTP uživatelů a skupin. Bez těchto tabulek by nebylo možné ověření přihlašovacích údajů.

```
42.  INSERT INTO ftpuser (userid, passwd, homedir, domain_id) VALUES ('ftp-user',
     'ftp-password', '/var/ftp', 1);
43.  INSERT INTO ftpgroup (groupname, gid, members) VALUES ('ftpgroup', 5500,
     'ftpuser');
```

Tato část skriptu vytváří účet administrátora FTP, který bude mít přístup ke všem uživatelským datům a řídící aplikaci. Poslední příkaz vytváří skupinu pro FTP uživatele.

```
44.  UPDATE mysql.user SET Host = '%' WHERE  User = 'domain';
45.  GRANT GRANT OPTION ON *.* TO 'database-user';
46.  GRANT ALL PRIVILEGES ON *.* TO 'database-user';
47.
48.  CREATE USER 'ftp-mysql-user'@'%' IDENTIFIED BY 'ftp-mysql-password';
49.  GRANT ALL PRIVILEGES ON database-name.ftpgroup TO 'ftp-mysql-user';
50.  GRANT ALL PRIVILEGES ON database-name.ftpuser TO 'ftp-mysql-user';
51.
52.  flush privileges;
```

Tento kousek skriptu nastavuje oprávněním uživatelů jak k jednotlivým tabulkám, tak k jednotlivým databázím. Jeden uživatel, pod kterým se vytvářejí a spravují jednotlivé databáze z řídící aplikace. Druhý uživatel pro komunikaci služby FTP s MySQL databází. Poslední příkaz všechny nastavená oprávnění aktualizuje.

5.5.5 MongoDB

Soubor Dockerfile

```
1.  FROM mongo:4.2
```

Pomocí direktivy `FROM` se určí základní kontejner pro docker image obsahující službu MongoDB ve verzi 4.2.

```
2.  RUN apt-get update && \
3.      apt-get dist-upgrade -y --force-yes && \
4.      apt-get autoremove -y && \
5.      apt-get autoclean && \
6.      rm -rf /var/lib/apt/list/*
```

Direktiva `RUN` provede skript sloužící k aktualizaci všech nainstalovaných balíčků a následně vymaže všechny instalační soubory.

5.5.6 Tomcat

Dockerfile

```
1.  FROM tomcat:9.0-jdk11-openjdk
```

Pomocí direktivy `FROM` se určí základní kontejner pro docker image obsahující službu Tomcat ve verzi 9.0 s přeinstalovanou verzí balíčku OpenJDK11.

```
2.  WORKDIR /usr/local/tomcat
```

Nastaví pracovní adresář na umístění /usr/local/tomcat. V tomto umístění se následně provádějí veškeré příkazy.

```
3.  RUN apt-get update && \
4.      apt-get dist-upgrade -y && \
5.      apt-get install -y dnsutils && \
6.      apt-get autoremove -y && \
7.      apt-get autoclean && \
8.      rm -rf /var/lib/apt/list/* && \
9.      cp -r webapps.dist/manager webapps.dist/host-manager webapps/
```

Direktiva `RUN` spustí skript sloužící k aktualizaci všech nainstalovaných balíčků a následně vymaže všechny instalační soubory. Poslední příkaz skriptu nakopíruje aplikaci Manager a Host-Manager do adresáře /usr/local/tomcat/webapps tak, aby byli dostupné skrze službu Tomcat.

```
10.  ADD server.xml /usr/local/tomcat/conf
11.  ADD tomcat-users.xml /usr/local/tomcat/conf
12.  COPY context.xml /usr/local/tomcat/webapps/manager/META-INF
```

Kopírování konfiguračních souborů do příslušných umístění tak, aby byl zajištěn bezproblémový chod služby Tomcat.

```
13.  ADD docker-entrypoint.sh /
14.  RUN chmod a+x /docker-entrypoint.sh
15.  ENTRYPOINT ["/docker-entrypoint.sh"]
```

Příkaz `ADD` přidá soubor *docker-entrypoint.sh* do kořenového adresáře. Po přidání následuje příkaz pro přidání práv sloužících ke spuštění přidaného souboru. Posledním příkazem je `ENTRYPOINT` pro specifikaci souboru se skriptem, který se provede při spuštění kontejneru.

Soubor context.xml

Kvůli zabezpečení Tomcat serveru je nezbytné omezit přístup k aplikacím sloužícím ke správě serveru. Mezi tyto aplikace patří Host Manger sloužícím ke správě virtuálních hostů tak, aby nebylo třeba pro každou aplikaci definovat unikátní IP adresu. Dále obsahuje Manager sloužící k nasazení nových aplikaci a administraci již nasazených aplikací. Jedním z nejjednodušších způsobů, jak povolit nebo zakázat přístup k Tomcat, je přes filtr vzdálených adres. Toho lze dosáhnout přidáním následujícího řádku do souboru context.xml nacházející se v adresáři `META-INF` příslušné aplikace.

```
1.  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
2.            allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1|ip-address" />
```

Filtr porovnává IP adresu vzdáleného klienta se seznamem v atributu `allow`. Pokud se adresa vyskytuje v seznamu, je jeho požadavek přijat v opačném případě, dojde k odmítnutí. Není-li atribut `allow` specifikován budou všechny žádosti přijaty, pokud nebyla nalezena shoda IP adresy klienta v seznamu `deny`.

Soubor server.xml

Soubor server.xml je hlavním konfiguračním souborem služby Tomcat a odpovídá za určení počáteční konfigurace Tomcat při spuštění a za definování způsobu a pořadí, v jakém systém Tomcat zavádí a sestavuje. Prvky souboru server.xml patří do pěti základních kategorií: prvky nejvyšší úrovně, konektory, kontejnery, vnořené komponenty a globální nastavení. Všechny prvky v těchto kategoriích mají mnoho atributů, které lze využít k jejich podrobnější konfiguraci. Mezi tyto úpravy nejčastěji patří například změna portu aplikace.

Soubor tomcat-users.xml

Tento soubor slouží k vytváření uživatelských účtů sloužících ke správě serveru Tomcat. K vytvoření účtu stačí definovat uživatelské jméno, heslo a jednotlivé role specifikující možnosti konfigurace serveru pro daný účet. Mezi základní role patří:

- manager-gui – umožňující přístup do aplikace Manager pomocí webového rozhraní,
- manager-status – umožňující přístup k získání informací o stavu nasazených aplikací na serveru,
- manager-script – umožňující přístup k nástrojům, které zajišťují uživatelsky přívětivý přístup k nástrojům skrze rozhraní ve formátu plain-text a také k přístupu pro kontrolu stavu serveru,
- manager-jmx – umožňující přístup k rozhraní JMX proxy a stavu serveru,
- admin-gui – umožňující přístup do aplikace Host Manager pomocí webového rozhraní,
- admin-script – umožňující role sloužící pro skriptování na webovém rozhraní.

Ukázka souboru tomcat-users.yml:

```
1.  <role rolename="manager-gui"/>
2.  <role rolename="manager-script"/>
3.  <role rolename="admin-gui"/>
4.  <role rolename="admin-script"/>
5.  <user username="user-name" password="user-password" roles="admin-gui,manager-
    gui,admin-script,manager-script"/>
```

5.6 Řídící aplikace

Tato část se zaměřuje na návrh a implementaci řídící webové aplikace. V předchozích kapitolách byly popsány jednotlivé služby dostupné na serveru a jejich konfigurace spolu jejími možnostmi. Následuje popis řídící aplikace včetně všech částí tak, jak se zobrazují v uživatelském rozhraní.

5.6.1 Registrace

Registrace domény je nutnou podmínkou k tomu, aby uživatel získal vstup do řídící aplikace. Při procesu registrace se vytvářejí databáze pro registrovanou doménu, dále se vytvoří uživatelský účtu do administrace a všech ostatních služeb.

Pro registraci účtu je nutné:

1. zadat název subdomény, který se stane přihlašovacím jménem (loginem) do webové administrace pro danou subdoménu
2. zadat heslo do webové administrace pro danou subdoménu

5.6.2 Nabídka Doména

Slouží ke změně přístupového hesla do administrace, kde stačí zadat původní heslo a nové heslo. Nabídka také umožňuje zrušit hosting subdomény. Pro tento krok je třeba zadat heslo do administrativy po kterém dojde k odstranění veškerých uživatelských dat včetně všech uživatelských účtů a databází spjatých s danou subdoménou.

5.6.3 Nabídky FTP, MySQL, MongoDB

Slouží ke změně přístupových súdajů k FTP, resp. MySQL a MongoDB serverů. Pro změnu je nutné zadat nový název uživatele a přístupové heslo.

5.6.4 Nabídka Tomcat

Obsahuje přehled všech nahraných aplikací na serveru Tomcat. Umožnuje správu jednotlivých aplikací, kde každá aplikace může být: spuštěna, zastavena, obnovena nebo odstraněna ze serveru. Pro přidání nové aplikace stačí vybrat soubor s aplikací, který musí mít příponu *.war*.

5.6.5 Nabídka Logins

Obsahuje přehled všech přihlašovacích jmen k jednotlivým službám, včetně návodu a ukázek, jak se ke službám připojit. V případě registrace nové domény jsou zde též dostupná automaticky vygenerovaná hesla jednotlivých služeb.

5.6.6 Nabídka Adminer

Souží k přesměrování uživatele do nástroje Adminer, kde je možné obsluhovat databázi MySQL, včetně vytváření, mazání, zobrazování i editace jednotlivých uživatelských tabulek.

5.7 Instalace

Na oficiálních stránkách projektu Docker se nachází sekce “Get Started”, ve které se nacházejí návody k instalaci nástroje Docker pro jednotlivé platformy. Následně stačí vybrat platformu a postupovat dle připraveného návodu.

5.7.1 Linux

Pro platformu Linux existují tři hlavní způsoby instalace Dockeru, jejichž výběr závisí na situaci a prostředí.

- Použití repozitářů Docker – nejčastěji využívaná možnost instalace, jelikož nabízí snadnou instalaci i následné aktualizace,
- Stahování balíčku – vhodné pro počítače offline, které nemají přístup k internetu,
- Používání automatizovaných skriptů – vhodné pro vývojová a testovací prostředí.

5.7.2 Windows

Před instalací nástroje Docker na operační systém Windows je nezbytné aktivovat službu Hyper-V, která je dostupná pouze ve verzích Pro, Enterprise a Education. Následně stačí provést instalaci připraveného instalátoru obsahující nástroj Docker. Během instalace je nezbytné specifikovat, zda bude požadováno virtualizovat Linuxové nebo Windows kontejnery.

ZÁVĚR

Cílem této bakalářské práce bylo vytvořit multihostingový systém na platformě Docker. Teoretická část měla za úkol popsat proces vývoje řídící aplikace pro multihostingový systém včetně konfigurace kontejnerů pro platformu Docker.

První kapitola bakalářské práce se věnuje teoretickému úvodu. Je zde popsáno, co je to World Wide Web a pomocí kterých komunikačních protokolů dochází k výměně dat mezi uživatelem a serverem. Dále je zde také popsán rozdíl mezi webhostingem a multihostingem.

Druhá kapitola se zaměřuje na popis nezbytného software vybavení pro multihostingový systém. V této části je uveden popis využitého operačního systému, včetně všech nezbytných služeb tvořících výsledný systém. Následuje odůvodnění použitého programovacího jazyka a všech využitých frameworků pro řídící aplikaci.

Třetí kapitola se věnuje pojmu virtualizace a také základnímu seznámení s platformou Docker. V této kapitole jsou uvedeny jednotlivé typy virtualizací včetně jejich specifických použití, po kterých následuje stručná historie, včetně popisu základních stavebních kamenů platformy Docker, mezi které patří Docker Kontejnery a Docker Images. Dále je zde popsáno k čemu slouží Docker Compose a Docker Swam.

Čtvrtá kapitola se zaměřuje na jednotlivé výhody/nevýhody při budování multihostingového systému pomocí platformy Docker. Je zde například popsáno, proč je vybudovaný systém bezpečný, snadno rozšiřitelný a udržovatelný.

Pátá kapitola popisuje celou implementaci praktické části bakalářské práce. Kde lze nalézt návrh multihostingového systému pomocí platformy Docker, například konfigurační soubory pro jednotlivé kontejnery (služby), řídící aplikaci. Je zde vyobrazena struktura celého projektu a rozebrány jednotlivé služby včetně jejich konfiguračních souborů. Dále je zde popis řídící aplikace a způsob instalace nástroje Docker na operační systémem Windows a Linux.

Výsledná praktická část je plně fungujícím multihostingovým systémem, sloužící k registraci vlastních subdomén, na kterých lze provozovat vlastní webové stránky nebo aplikace naprogramované v jazyce Java. Protože se jedná o komplexní řešení pro hostování tak je možné jej nasadit na samostatný server. V současnosti řídící aplikace umožňuje šetřit čas díky automatizaci vybraných procesů např. vytvoření databází, založení FTP účtu pro manipulaci s daty na serveru. Tímto snižuje náklady pro správce webhostingu a s dalšími přidanými moduly může tato úspora dále růst. Další úspora vzniká při využití kontejnerů, které je možné

nasadit na jednom fyzickém serveru a provozovat tak na něm několik služeb zároveň s nižší režií. Tato úspora může být opět rozšířena při využití Docker Swarm k orchestraci kontejnerů při nasazení na více serverech.

Jelikož byl při tvorbě systému kladen důraz na jeho modularitu, je možné ho v budoucnu snadno rozšířit o případné další funkcionality, jako například hosting více domén prvního řádu.

Toto téma bylo mnou zvoleno zejména kvůli mé touze se podrobněji seznámit s moderními technologiemi, které jsou nutné pro provoz multihostigového systému a s platformou Docker. Všechny tyto znalosti mohou znamenat velký přínos pro mou budoucí práci se zaměřením na konfiguraci serverů.

POUŽITÁ LITERATURA

[1] Roser, Max, Hannah Ritchie a Esteban Ortiz-Ospina. Internet. *Published online at OurWorldInData.org* [online]. 2015 [cit. 2020-08-01]. Dostupné z: https://ourworldindata.org/internet

[2] Huss, Nick. How Many Websites Are There Around the World? [2020]. *Siteefy* [online]. 26. 5. 2020 [cit. 2020-08-01]. Dostupné z: https://siteefy.com/how-many-websites-are-there/

[3] Hardin, Edward. Web server requirements (hardware). *Web-Site Scripts* [online]. 4. 11. 2017 [cit. 2020-08-01]. Dostupné z: http://www.web-site-scripts.com/knowledge-base/article/AA-00505/0/Web-server-requirements-hardware.html

[4] Rouse, Margaret. HTTP (Hypertext Transfer Protocol). *TechTarget* [online]. červenec 2020 [cit. 2020-08-01]. Dostupné z: https://whatis.techtarget.com/definition/HTTP-Hypertext-Transfer-Protocol

[5] HTTP. *Adaptic* [online]. ©2020 [cit. 2020-08-01]. Dostupné z: https://www.adaptic.cz/znalosti/slovnicek/http/

[6] HTTP vs HTTPS: What's the Difference?. *Guru99* [online]. ©2020 [cit. 2020-08-01]. Dostupné z: https://www.guru99.com/difference-http-vs-https.html

[7] Nejlevnější (nejlepší) webhosting. *Nejlepší webhostingy* [online]. 2. 3. 2020 [cit. 2020-08-01]. Dostupné z: https://www.nejlepsi-webhostingy.cz

[8] Bílek, Pavel. Hosting a doména. *Tvorba webových stránek | Pavel Bíl*ek [online]. ©2020 [cit. 2020-08-01]. Dostupné z: https://webstrankylevne.cz/hosting-a-domena.html

[9] Sneddon, Joey. Ubuntu 18.04 LTS: What's New and Where to Download?. *Omg!ubuntu!* [online]. 9. 6. 2020 [cit. 2020-08-01]. Dostupné z: https://www.omgubuntu.co.uk/2018/04/ubuntu-18-04-download-release-features

[10] Wallen, Jack. How to make MySQL administration simple with Adminer. *TechRepublic* [online]. 13. 2. 2018 [cit. 2020-08-01]. Dostupné z: https://www.techrepublic.com/article/how-to-make-mysql-administration-simple-with-adminer/

[11] Roeder, Linda. What Is FTP and How Do I Use It?. *Lifewire* [online]. 3. 7. 2020 [cit. 2020-08-01]. Dostupné z: https://www.lifewire.com/ftp-defined-2654479

[12] MySQL Documentation. *MySQL* [online]. ©2020 [cit. 2020-08-01]. Dostupné z: https://dev.mysql.com/doc/

[13] Rouse, Margaret. MongoDB. *TechTarget* [online]. srpen 2018 [cit. 2020-08-01]. Dostupné z: https://searchdatamanagement.techtarget.com/definition/MongoDB

[14] Morris, Scott. Everything you need to know about PHP. *Skillcrush* [online]. ©2020 [cit. 2020-08-01]. Dostupné z: https://skillcrush.com/blog/php/

[15] The Apache License, Version 2.0. *Apache* [online]. ©2020 [cit. 2020-08-01]. Dostupné z: https://httpd.apache.org/docs/2.4/license.html

[16] Thakur, Ankush. 7 Open Source Web Servers for Small to Large Sites. G*eekflare* [online]. 10. 3. 2020 [cit. 2020-08-01]. Dostupné z: https://geekflare.com/open-source-web-servers/

[17] 3.1 Apache Software Foundation Licenses. *Oracle* [online]. ©2015 [cit. 2020-08-01]. Dostupné z: https://docs.oracle.com/cd/E51728_01/E61820/html/license-apache.html

[18] Rouse, Margaret. Java Platform, Enterprise Edition (Java EE). *TheServerSide Your Enterprise Java Community*. [online]. březen 2017 [cit. 2020-08-01]. Dostupné z: https://www.theserverside.com/definition/J2EE-Java-2-Platform-Enterprise-Edition

[19] Vaidya, Neha. Servlet and JSP Tutorial - How to Build Web Applications in Java?. *Edureka!* [online]. 21. 7. 2020 [cit. 2020-08-01]. Dostupné z: https://www.edureka.co/blog/servlet-and-jsp-tutorial/

[20] Twig - The flexible, fast, and secure PHP template engine. *Twig* [online]. ©2020 [cit. 2020-08-01]. Dostupné z: https://twig.symfony.com

[21] Čápka, David. Lekce 1 - Popis MVC architektury. *ITnetwork.cz* [online]. 2012 [cit. 2020-08-01]. Dostupné z: https://www.itnetwork.cz/php/mvc/objektovy-mvc-redakcni-system-v-php-popis-architektury

[22] Tomeček, Jaroslav. Virtualizace na úrovni jádra operačního systému. *AbcLinuxu.cz* [online]. 31. 7. 2007 [cit. 2020-08-01]. ISSN 1214-1267. Dostupné z: https://www.abclinuxu.cz/clanky/system/virtualizace-na-urovni-jadra-operacniho-systemu

[23] Bonuccelli, Giorgio. Application Virtualization | What Is It and Why Your Organization Needs It?. *Parallels* [online]. 21. 7. 2020 [cit. 2020-08-01]. Dostupné z: https://www.parallels.com/blogs/ras/application-virtualization/

[24] Stopka, Mark. Úvod do Dockeru (1). *AbcLinuxu.cz* [online]. 18. 9. 2019 [cit. 2020-08-01]. ISSN 1214-1267. Dostupné z: https://www.abclinuxu.cz/clanky/uvod-do-dockeru-1

[25] Components & Licenses. *Docker* [online]. ©2020 [cit. 2020-08-01]. Dostupné z: https://www.docker.com/legal/components-licenses

[26] SDxCentral Staff. What is a Docker Container? Definition. *SdxCentral* [online]. 7. 4. 2015 [cit. 2020-08-01]. Dostupné z: https://www.sdxcentral.com/containers/definitions/what-is-docker-container/

[27] Leandrogoe. About storage drivers. *Docker docs* [online]. 29. 6. 2020 [cit. 2020-08-01]. Dostupné z: https://docs.docker.com/storage/storagedriver/

[28] Neo, Kobo. Neo Kobo: Docker Layered Environment. *Neo Kobo* [online]. 12. 3. 2017 [cit. 2020-08-01]. Dostupné z: http://neokobo.blogspot.com/2017/03/docker-layered-environment.html

[29] Niedringhaus, Paige. Docker 101: Fundamentals & The Dockerfile. *ITNEXT by LINKIT* [online]. 23. 6. 2018 [cit. 2020-08-01]. Dostupné z: https://itnext.io/docker-101-fundamentals-the-dockerfile-b33b59d0f14b

[30] Ligios, Andrea. Introduction to Docker Compose. *Baeldung* [online]. 6. 9. 2019 [cit. 2020-08-01]. Dostupné z: https://www.baeldung.com/docker-compose

[31] Rouse, Margaret. Docker Swarm. *TechTarget* [online]. srpen 2016 [cit. 2020-08-01]. Dostupné z: https://searchitoperations.techtarget.com/definition/Docker-Swarm

[32] Tým Dataflair. Advantages and Disadvantages of Docker - Learn Docker. *DataFlair* [online]. 27. 11. 2018 [cit. 2020-08-01]. Dostupné z: https://data-flair.training/blogs/advantages-and-disadvantages-of-docker/